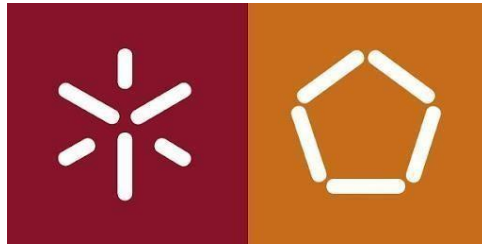


Universidade do Minho
Licenciatura em Engenharia Informática



Redes de Computadores
Trabalho prático 2

Braga, Maio 2023

Trabalho realizado por:

João Ricardo Ribeiro Rodrigues – a100598

Rafael Lima Mesquita - a95097

Sandra Fabiana Pires Cerqueira – a100681

Índice

| | |
|---------------------------|-----------|
| Trabalho realizado por: | 1 |
| 1 - Primeira parte | 4 |
| 1.1 Exercício 1 | 4 |
| 1.1.a Alínea a) | 5 |
| 1.1.b Alínea b) | 7 |
| 1.1.c Alínea c) | 7 |
| 1.1.d Alínea d) | 8 |
| 1.2 Exercício 2 | 9 |
| 1.2.a Alínea a) | 9 |
| 1.2.b Alínea b) | 10 |
| 1.2.c Alínea c) | 10 |
| 1.2.d Alínea d) | 10 |
| 1.2.e Alínea e) | 11 |
| 1.2.f Alínea f) | 11 |
| 1.2.g Alínea g) | 11 |
| 1.2.h Alínea h) | 12 |
| 1.3 Exercício 3 | 13 |
| 1.3.a Alínea a) | 13 |
| 1.3.b Alínea b) | 13 |
| 1.3.c Alínea c) | 14 |
| 1.3.d Alínea d) | 14 |
| 1.3.e Alínea e) | 15 |
| 1.3.f Alínea f) | 16 |
| 1.3.g Alínea g) | 17 |
| 1.3.h Alínea h) | 18 |
| 1.3.i Alínea i) | 18 |
| 1.3.j Alínea j) | 19 |
| 2 - Segunda parte | 19 |
| 2.1 Exercício 1 | 19 |
| 2.1.a Alínea a) | 20 |
| 2.1.b Alínea b) | 21 |
| 2.1.c Alínea c) | 22 |
| 2.1.d Alínea d) | 27 |
| 2.1.e Alínea e) | 30 |
| 2.1.f Alínea f) | 30 |
| 2.1.g Alínea g) | 30 |
| 2.2 Exercício 2 | 32 |
| 2.2.a Alínea a) | 32 |

| | |
|-----------------------|-----------|
| 2.2.b Alínea b) | 33 |
| 2.2.c Alínea c) | 33 |
| 2.3 Exercício 3 | 35 |
| 2.3.1 Alínea 1) | 35 |
| 2.3.2 Alínea 2) | 35 |
| 2.3.3 Alínea 3) | 36 |
| 3 - Conclusões | 38 |

1 - Primeira parte

1.1 Exercício 1

Questão: 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.

A seguinte topologia CORE foi a que criamos para verificar o comportamento do *traceroute*. Na topologia existe: 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*. Nas ligações (links) da rede de core, foi estabelecido um tempo de propagação de 15 ms.

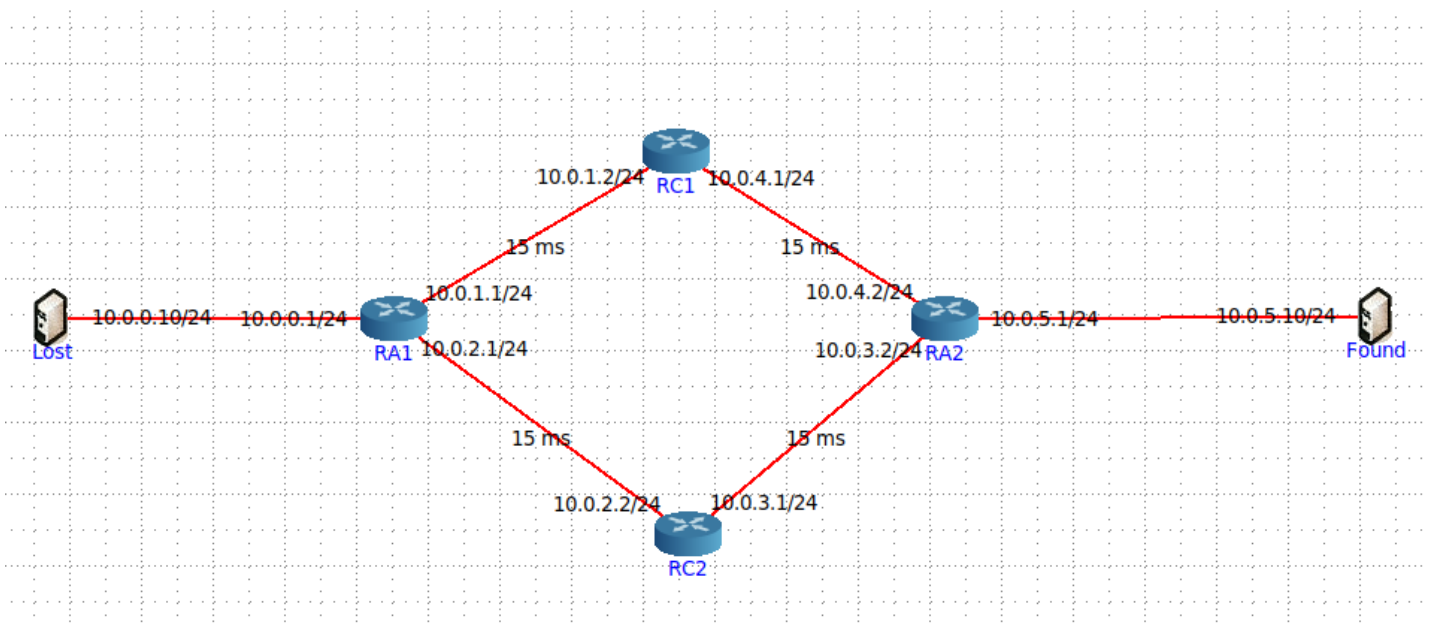


Figura 1-Topologia Core utilizada no exercício 1

1.1.a Alínea a)

Questão: Active o Wireshark no host Lost. Numa shell de Lost execute o comando `tracert -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 `tracert`.

Após a ativação do Wireshark no host Lost e executarmos numa Shell o comando `tracert -I 10.0.5.10` para o endereço IP do Found e obtivemos os seguintes outputs:

```

vcmd
tracert to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.172 ms 0.011 ms 0.005 ms
 2 10.0.1.2 (10.0.1.2) 34.131 ms 34.121 ms 34.117 ms
 3 10.0.3.2 (10.0.3.2) 137.613 ms 137.613 ms 137.611 ms
 4 10.0.5.10 (10.0.5.10) 137.609 ms 137.608 ms 137.607 ms
root@Lost:/tmp/pycore.40279/Lost.conf# tracert -I 10.0.5.10
tracert to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.036 ms 0.006 ms 0.004 ms
 2 10.0.1.2 (10.0.1.2) 32.647 ms 32.639 ms 32.635 ms
 3 10.0.3.2 (10.0.3.2) 65.534 ms 65.532 ms 65.530 ms
 4 10.0.5.10 (10.0.5.10) 65.527 ms 65.525 ms 65.522 ms
root@Lost:/tmp/pycore.40279/Lost.conf# tracert -I 10.0.5.10
tracert to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.034 ms 0.006 ms 0.005 ms
 2 10.0.1.2 (10.0.1.2) 31.463 ms 31.456 ms 31.452 ms
 3 10.0.3.2 (10.0.3.2) 62.640 ms 62.638 ms 62.635 ms
 4 10.0.5.10 (10.0.5.10) 62.632 ms 62.628 ms 62.626 ms
root@Lost:/tmp/pycore.40279/Lost.conf# tracert -I 10.0.5.10
tracert to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.035 ms 0.006 ms 0.004 ms
 2 10.0.1.2 (10.0.1.2) 33.311 ms 33.303 ms 33.300 ms
 3 10.0.3.2 (10.0.3.2) 64.524 ms 64.521 ms 64.518 ms
 4 10.0.5.10 (10.0.5.10) 64.515 ms 64.512 ms 64.510 ms
root@Lost:/tmp/pycore.40279/Lost.conf#

```

Figura 2-Resultado da execução do comando `tracert -I`

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|-----------|-------------|----------|--------|---|
| 21 | 20.697139662 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=1/256, ttl=1 (no response..) |
| 22 | 20.697248353 | 10.0.0.1 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 23 | 20.697261037 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=2/512, ttl=1 (no response..) |
| 24 | 20.697267895 | 10.0.0.1 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 25 | 20.697271841 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=3/768, ttl=1 (no response..) |
| 26 | 20.697275592 | 10.0.0.1 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 27 | 20.697279640 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=4/1024, ttl=2 (no respons..) |
| 28 | 20.697291384 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=5/1280, ttl=2 (no respons..) |
| 29 | 20.697295185 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=6/1536, ttl=2 (no respons..) |
| 30 | 20.697299073 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=7/1792, ttl=3 (no respons..) |
| 31 | 20.697302471 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=8/2048, ttl=3 (no respons..) |
| 32 | 20.697306138 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=9/2304, ttl=3 (no respons..) |
| 33 | 20.697309628 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=10/2560, ttl=4 (reply in ..) |
| 34 | 20.697312978 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=11/2816, ttl=4 (reply in ..) |
| 35 | 20.697316219 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=12/3072, ttl=4 (reply in ..) |
| 36 | 20.697320198 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=13/3328, ttl=5 (reply in ..) |
| 37 | 20.697324245 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=14/3584, ttl=5 (reply in ..) |
| 38 | 20.697327540 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=15/3840, ttl=5 (reply in ..) |
| 39 | 20.697334063 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=16/4096, ttl=6 (reply in ..) |
| 40 | 20.698277289 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=17/4352, ttl=6 (reply in ..) |
| 41 | 20.698294825 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=18/4608, ttl=6 (reply in ..) |
| 42 | 20.698299560 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=19/4864, ttl=7 (reply in ..) |
| 43 | 20.731406197 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 44 | 20.731410460 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 45 | 20.731411167 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 46 | 20.731913170 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=20/5120, ttl=7 (reply in ..) |
| 47 | 20.731932783 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=21/5376, ttl=7 (reply in ..) |
| 48 | 20.731938166 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=22/5632, ttl=8 (reply in ..) |
| 49 | 20.834904128 | 10.0.3.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 50 | 20.834913021 | 10.0.3.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 51 | 20.834915223 | 10.0.3.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 52 | 20.834917394 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=10/2560, ttl=61 (request ..) |
| 53 | 20.834919273 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=11/2816, ttl=61 (request ..) |
| 54 | 20.834921259 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=12/3072, ttl=61 (request ..) |
| 55 | 20.834923245 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=13/3328, ttl=61 (request ..) |
| 56 | 20.834925328 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=14/3584, ttl=61 (request ..) |
| 57 | 20.834927315 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=15/3840, ttl=61 (request ..) |
| 58 | 20.834929317 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=16/4096, ttl=61 (request ..) |
| 59 | 20.834931383 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=17/4352, ttl=61 (request ..) |
| 60 | 20.834933369 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=18/4608, ttl=61 (request ..) |

Figura 3- Resultado da execução do Wireshark

Para cada erro que ocorre, o ICMP fornece uma mensagem indicando qual foi o problema.

Estas mensagens são geralmente enviadas imediatamente após a ocorrência do erro. No entanto, devido ao *delay* que atribuímos às ligações da rede de core (15ms), após algum tempo as mensagens de erro começam a aparecer desfasadas, não estando logo a seguir ao local onde ocorreu o erro.

Como é possível verificar, as mensagens de erro geradas pelo ICMP quando o TTL = 1 estão imediatamente após ter ocorrido o erro, enquanto que no caso do TTL = 2 e do TTL = 3 estas aparecem separadas das linhas onde estão os pacotes.

Quando o TTL atinge o valor de 4, deixam de ocorrer erros.

```

Frame 25: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface veth1.0.ca, id 0
Ethernet II, Src: 00:00:00:aa:00:00 (00:00:00:aa:00:00), Dst: 00:00:00:aa:00:01 (00:00:00:aa:00:01)
Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.5.10
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 60
  Identification: 0x9bed (39917)
  Flags: 0x0000
  Fragment offset: 0
  Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x04c1 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.0.0.10
  Destination: 10.0.5.10
Internet Control Message Protocol
  Code: 0
  Checksum: 0x824c [correct]
  [Checksum Status: Good]
  Identifier (BE): 43 (0x002b)
  Identifier (LE): 11008 (0x2b00)
  Sequence number (BE): 3 (0x0003)
  Sequence number (LE): 768 (0x0300)
  [No response seen]
  Data (32 bytes)
  
```

Figura 4-Output do Wireshark

```

Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.5.10
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 60
  Identification: 0x9beb (39915)
  Flags: 0x0000
  Fragment offset: 0
  Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x04c3 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.0.0.10
  Destination: 10.0.5.10
Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x824e [unverified] [in ICMP error packet]
  [Checksum Status: Unverified]
  Identifier (BE): 43 (0x002b)
  Identifier (LE): 11008 (0x2b00)
  Sequence number (BE): 1 (0x0001)
  
```

Figura 5-Detalhe do output do Wireshark

1.1.b Alínea b)

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

O valor inicial mínimo do campo TTL para alcançar o servidor *Found* é 4. Como se pode verificar pela imagem acima, quando o TTL tem os valores 1, 2 e 3 não é possível alcançar o servidor, resultando em erros.

Independentemente do caminho escolhido até ao servidor *Found*, este terá sempre tamanho 4. Como em cada *router* o TTL é decrementado uma unidade, será necessário o TTL ter um valor igual ou superior a 4 para poder atingir o servidor sem dar erro.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|-----------|-------------|----------|--------|--|
| 21 | 20.697139662 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=1/256, ttl=1 (no response.. |
| 22 | 20.697248355 | 10.0.0.10 | 10.0.5.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 23 | 20.697261037 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=2/512, ttl=1 (no response.. |
| 24 | 20.697267895 | 10.0.0.10 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 25 | 20.697271841 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=3/768, ttl=1 (no response.. |
| 26 | 20.697275592 | 10.0.0.10 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 27 | 20.697279640 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=4/1024, ttl=2 (no respons.. |
| 28 | 20.697291384 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=5/1280, ttl=2 (no respons.. |
| 29 | 20.697295185 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=6/1536, ttl=2 (no respons.. |
| 30 | 20.697299073 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=7/1792, ttl=3 (no respons.. |
| 31 | 20.697302471 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=8/2048, ttl=3 (no respons.. |
| 32 | 20.697306138 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=9/2304, ttl=3 (no respons.. |
| 33 | 20.697309628 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=10/2560, ttl=4 (reply in .. |
| 34 | 20.697312978 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=11/2816, ttl=4 (reply in .. |
| 35 | 20.697316219 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=12/3072, ttl=4 (reply in .. |
| 36 | 20.697320109 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=13/3328, ttl=5 (reply in .. |
| 37 | 20.697324245 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=14/3584, ttl=5 (reply in .. |
| 38 | 20.697327540 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=15/3840, ttl=5 (reply in .. |
| 39 | 20.697334063 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=16/4096, ttl=6 (reply in .. |
| 40 | 20.698277289 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=17/4352, ttl=6 (reply in .. |
| 41 | 20.698294825 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=18/4608, ttl=6 (reply in .. |
| 42 | 20.698299568 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=19/4864, ttl=7 (reply in .. |
| 43 | 20.731400617 | 10.0.0.10 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 44 | 20.731410460 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 45 | 20.731411167 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 46 | 20.731913170 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=20/5120, ttl=7 (reply in .. |
| 47 | 20.731932783 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=21/5376, ttl=7 (reply in .. |
| 48 | 20.731938166 | 10.0.0.10 | 10.0.5.10 | ICMP | 74 | Echo (ping) request id=0x002b, seq=22/5632, ttl=8 (reply in .. |
| 49 | 20.834904128 | 10.0.0.10 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 50 | 20.834913021 | 10.0.0.10 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 51 | 20.834915223 | 10.0.0.10 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 52 | 20.834917304 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=10/2560, ttl=61 (request .. |
| 53 | 20.834919273 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=11/2816, ttl=61 (request .. |
| 54 | 20.834921259 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=12/3072, ttl=61 (request .. |
| 55 | 20.834923245 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=13/3328, ttl=61 (request .. |
| 56 | 20.834925328 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=14/3584, ttl=61 (request .. |
| 57 | 20.834927315 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=15/3840, ttl=61 (request .. |
| 58 | 20.834929317 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=16/4096, ttl=61 (request .. |
| 59 | 20.834931383 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=17/4352, ttl=61 (request .. |
| 60 | 20.834933369 | 10.0.5.10 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x002b, seq=18/4608, ttl=61 (request .. |

Figura 6-Resultado da execução do Wireshark

1.1.c Alínea c)

Questão: 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.

O valor médio do tempo de ida-e-volta (RTT - Round-Trip Time) obtido no acesso ao servidor é 68,9173 usando 10 pacotes de teste.

Este resultado foi obtido utilizando os valores resultantes da execução do comando *tracert* para 10 pacotes de teste, *tracert -l 10.0.5.10 -q 10*, como se verifica na figura abaixo.

```

vcmd
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.035 ms 0.006 ms 0.004 ms
 2 10.0.1.2 (10.0.1.2) 33.311 ms 33.303 ms 33.300 ms
 3 10.0.3.2 (10.0.3.2) 64.524 ms 64.521 ms 64.518 ms
 4 10.0.5.10 (10.0.5.10) 64.515 ms 64.512 ms 64.510 ms
root@Lost:/tmp/pycore.40279/Lost.conf# traceroute -I 10.0.5.10 -q 5
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.044 ms 0.005 ms 0.005 ms 0.005 ms 0.004 ms
 2 10.0.1.2 (10.0.1.2) 35.508 ms 35.499 ms 35.496 ms 35.493 ms 35.489 ms
 3 10.0.3.2 (10.0.3.2) 68.814 ms 68.811 ms 68.808 ms 68.805 ms 68.803 ms
 4 10.0.5.10 (10.0.5.10) 68.801 ms 68.274 ms 68.264 ms 68.261 ms 68.258 ms
root@Lost:/tmp/pycore.40279/Lost.conf# traceroute -I 10.0.5.10 -q 500
no more than 10 probes per hop
root@Lost:/tmp/pycore.40279/Lost.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 10.0.0.1 (10.0.0.1) 0.033 ms 0.005 ms 0.005 ms 0.004 ms 0.004 ms
 2 10.0.1.2 (10.0.1.2) 32.319 ms 32.312 ms 32.309 ms 32.306 ms 32.303 ms
 3 10.0.3.2 (10.0.3.2) 63.307 ms 63.304 ms 72.185 ms 71.985 ms 71.968 ms
 4 10.0.5.10 (10.0.5.10) 72.561 ms 72.559 ms 67.595 ms 67.567 ms 68.396 ms

```

Figura 7-Resultado das execuções dos comandos `traceroute -I 10.0.5.10 -q 5` e `traceroute -I 10.0.5.10 -q 10`

1.1.d Alínea d)

Questão: 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, o valor médio do atraso em um sentido (*One-Way Delay*) não pode ser calculado com precisão simplesmente dividindo o RTT por dois.

O RTT (*Round Trip Time*) é a medida do tempo que leva para um pacote de dados ser enviado de um ponto a outro e a resposta ser recebida de volta. Dividir o RTT por dois não fornece o valor médio do atraso em um sentido porque não leva em consideração atrasos adicionais que podem ocorrer durante o processamento de pacotes de dados nos roteadores intermediários ou em outros dispositivos de rede.

O cálculo do valor médio do atraso em um sentido em uma rede real pode ser difícil devido a vários fatores. Por exemplo: Atrasos variáveis: os atrasos na rede podem variar dependendo do congestionamento da rede, do número de dispositivos conectados e da qualidade da conexão. Isso pode tornar difícil a determinação do valor médio do atraso em um sentido. Diferentes rotas: em uma rede real, os pacotes podem seguir diferentes rotas para chegar ao destino, o que pode levar a diferentes valores de atraso em um sentido. Isso torna difícil a obtenção de um valor preciso e representativo do valor médio do atraso em um sentido. Problemas de sincronização: a sincronização precisa dos relógios em dispositivos de rede diferentes é necessária para medir com precisão o atraso em um sentido.

No entanto, isso pode ser difícil de alcançar em uma rede real. Problemas de medição: as ferramentas de medição utilizadas para medir o atraso em um sentido podem não ser precisas o suficiente para fornecer um valor médio confiável.

1.2 Exercício 2

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

1.2.a Alínea a)

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

O endereço IP da interface ativa do computador utilizado pelo grupo, bem como os respetivos dados do Wireshark do mesmo encontram-se apresentados abaixo:

O endereço IP é : 172.26.126.110

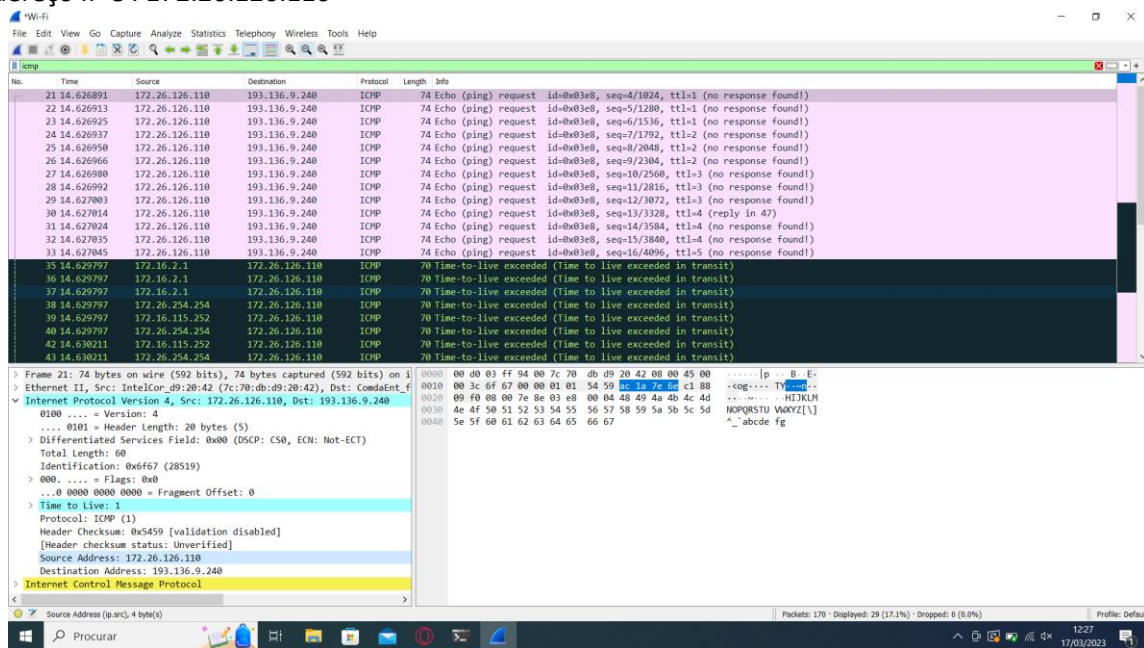


Figura 8-Output Wireshark

```
> Frame 21: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on i
> Ethernet II, Src: IntelCor_d9:20:42 (7c:70:db:d9:20:42), Dst: ComdaEnt_f
> Internet Protocol Version 4, Src: 172.26.126.110, 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: 0x6f67 (28519)
> 000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
> Time to Live: 1
  Protocol: ICMP (1)
  Header Checksum: 0x5459 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.126.110
  Destination Address: 193.136.9.240
> Internet Control Message Protocol
```

Figura 9-Detalhe do output do Wireshark

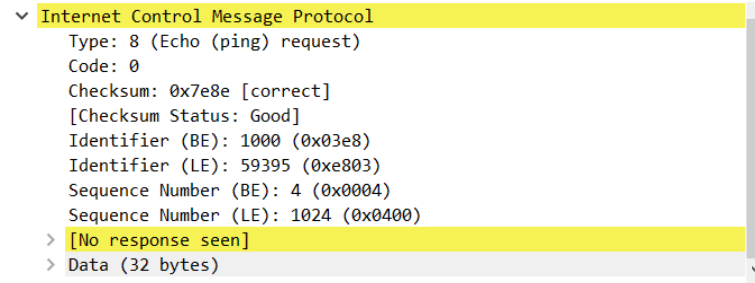


Figura 10-Detalhe do output do Wireshark

1.2.b Alínea b)

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

O valor do campo 'Protocol' é ICMP (1), como se pode ver na figura 11.

Este permite identificar que foi o ICMP que enviou a mensagem de erro, pois é um protocolo de comunicação utilizado em redes de computadores para enviar mensagens de erro e informações de controlo de rede. As mensagens do ICMP são utilizadas para relatar problemas na transmissão de dados e para auxiliar no diagnóstico de problemas de rede.

Protocol: ICMP (1)

Figura 11-Valor do campo Protocol

1.2.c Alínea c)

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

Tal como se pode verificar pelas imagens apresentadas na alínea a) deste exercício, o cabeçalho IPv4 tem 20 bytes e o campo de dados (payload) do datagrama tem 40 bytes, valor este que é calculado subtraindo o *total length*, que no nosso caso é 60, pelo *header length*, que é 20.

1.2.d Alínea d)

Pergunta: O datagrama IP foi fragmentado? Justifique.

O datagrama IP não foi fragmentado, visto que *More Fragments: Not Set* e o *Fragment Offset: 0*.

Como se vê na fig.12 o *Fragment Offset* é zero, a posição onde começa é zero, ou seja, começa do início, e como "*More Fragments: Not Set*" sabemos que não existem mais fragmentos. Logo, o datagrama IP não foi fragmentado.

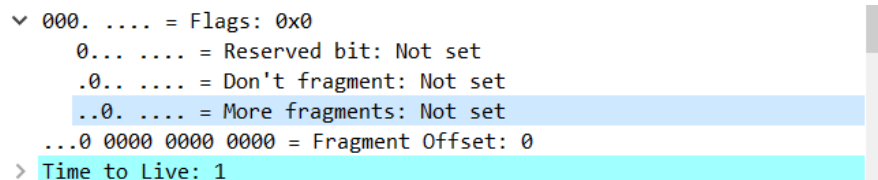


Figura 12-Detalhe do output do Wireshark

1.2.e Alínea e)

Pergunta: Indique que campos do cabeçalho IP variam de pacote para pacote.

Os campos que variam de pacote para pacote são o 'Identification', 'Time to Live' e 'Header checksum', tal como se pode ver nas figuras abaixo.

```
> Frame 23: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on i
> Ethernet II, Src: IntelCor_d9:20:42 (7c:70:db:d9:20:42), Dst: ComdaEnt_f
v Internet Protocol Version 4, Src: 172.26.126.110, 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: 0xcc34 (52276)
  > 000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
  > Time to Live: 1
  Protocol: ICMP (1)
  Header Checksum: 0xf78b [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.126.110
  Destination Address: 193.136.9.240
> Internet Control Message Protocol
```

Figura 13-Pacote x

```
> Frame 24: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on i
> Ethernet II, Src: IntelCor_d9:20:42 (7c:70:db:d9:20:42), Dst: ComdaEnt_f
v Internet Protocol Version 4, Src: 172.26.126.110, 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: 0xcc35 (52277)
  > 000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
  > Time to Live: 1
  Protocol: ICMP (1)
  Header Checksum: 0xf78a [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.126.110
  Destination Address: 193.136.9.240
> Internet Control Message Protocol
```

Figura 14-Pacote n

1.2.f Alínea f)

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

Ao analisar o tráfego, conseguimos verificar que o campo 'Identification' aumenta de pacote para pacote, de 1 em 1, e o campo 'Time to Live' também aumenta de 1 em 1 mas, de três em três pacotes. Isto deve-se ao facto do traceroute enviar três pacotes por predefinição em que todos têm o mesmo TTL.

1.2.g Alínea g)

G1) Pergunta: 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 do campo TTL recebido no computador do grupo é 255, 254 e 253, tal como se pode verificar nas imagens abaixo.

```
> Frame 37: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on i
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: IntelCor_d
v Internet Protocol Version 4, Src: 172.26.254.254, Dst: 172.26.126.110
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x6c08 (27656)
  > 000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 255
  Protocol: ICMP (1)
  Header Checksum: 0x795a [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.254.254
  Destination Address: 172.26.126.110
> Internet Control Message Protocol
```

Figura 15-Valor 255

```
> Frame 40: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on i
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: IntelCor_d
v Internet Protocol Version 4, Src: 172.16.2.1, Dst: 172.26.126.110
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x5d1f (23839)
  > 000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 254
  Protocol: ICMP (1)
  Header Checksum: 0x870b [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.16.2.1
  Destination Address: 172.26.126.110
> Internet Control Message Protocol
```

Figura 16- Valor 254

```

> Frame 44: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on i
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: IntelCor_d
v Internet Protocol Version 4, Src: 172.16.115.252, Dst: 172.26.126.110
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0xd4e6 (54502)
> 000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 253
  Protocol: ICMP (1)
  Header Checksum: 0x9e48 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.16.115.252
  Destination Address: 172.26.126.110
> Internet Control Message Protocol

```

Figura 17-Valor 253

Este valor não permanece constante para todas as mensagens de resposta ICMP TTL *Exceeded* recebidas no meu computador, uma vez que, cada vez que o TTL é excedido num *router* é mais uma ligação que ele percorre que decresce TTL enviado.

G2) Pergunta: *Porque razão as mensagens de resposta ICMP TTL Exceeded são sempre enviadas na origem com um valor TTL relativamente alto?*

Como ICMP TTL Exceeded indica que o pacote foi descartado devido ao TTL baixo ter atingido zero, então o router manda a resposta com um TTL alto já predefinido para garantir que este chega ao destino e não é descartado como aconteceu previamente.

1.2.h Alínea h)

Pergunta: *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?*

Ainda que seja teoricamente possível incluir as informações do ICMP no cabeçalho do IPv4, esta inclusão poderia resultar em algumas desvantagens tais como:

- O aumento do tamanho do cabeçalho do IPv4 pois, o cabeçalho do IPv4 tem um tamanho fixo de 20 bytes, e incluir as informações do cabeçalho ICMP poderia aumentar o tamanho do cabeçalho, resultando em pacotes maiores e menos eficientes em termos de utilização de largura de banda;
- Maior processamento nos roteadores, uma vez que, estes teriam que realizar mais processamento para analisar o cabeçalho do IPv4 e extrair as informações do ICMP, o que poderia resultar em atrasos na transmissão de pacotes;
- Maior complexidade no tratamento de erros, porque incluir as informações do ICMP no cabeçalho do IPv4 poderia tornar o tratamento de erros mais complexo e menos eficiente.

Por outro lado, existem algumas vantagens em incluir as informações do ICMP no cabeçalho do IPv4, tais como:

- Redução do número de pacotes na rede pois, ao incluir as informações do ICMP no cabeçalho do IPv4, seria possível reduzir o número de pacotes na rede, uma vez que não seria necessário enviar pacotes separados para as mensagens de controlo e de erro.
- Maior confiabilidade, uma vez que o pacote completo seria protegido pelo checksum do IPv4.
- Maior eficiência no tratamento de erros: uma vez que todas as informações necessárias estariam disponíveis num único pacote.

1.3 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). De modo a poder visualizar os fragmentos, aceda a Edit -> Preferences -> Protocols e em IPv4 desative a opção "Reassemble fragmented IPv4 datagrams".

1.3.a Alínea a)

Pergunta: *Porque é que houve necessidade de fragmentar o pacote inicial?*

Houve a necessidade de fragmentar o pacote uma vez que o seu tamanho 3536 ($3500 + 36$) excede o MTU (Max Transfer Unit) de 1500 bytes, valor máximo que se pode mandar de cada vez, logo foi necessário fragmentar o pacote.

1.3.b Alínea b)

Pergunta: *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?*

Na figura seguinte, apresentámos o primeiro fragmento da primeira mensagem ICMP captada no contexto dos pacotes com tamanho de 3536 bytes.

```
Internet Protocol Version 4, Src: 172.26.126.110, 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: 0xe447 (58439)
  001. .... = Flags: 0x1, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 128
  Protocol: ICMP (1)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.126.110
  Destination Address: 193.136.9.240
```

Figura 18-Primeiro fragmento IP

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

1.3.c Alínea c)

Pergunta: *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?*

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

```

v Internet Protocol Version 4, Src: 172.26.126.110, 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: 0xe447 (58439)
  v 001. .... = Flags: 0x1, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    ...0 0000 1011 1001 = Fragment Offset: 1480
  Time to Live: 128
  Protocol: ICMP (1)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.126.110
  Destination Address: 193.136.9.240
    
```

Figura 19-Segundo fragmento IP

Como o *fragment offset* tem o valor 1480, podemos concluir que este não é o primeiro fragmento do datagrama original, pois fazendo uma simples subtração do *total length* pelo *header length*, $1500 - 20 = 1480$, concluímos que o primeiro fragmento acaba no 1480 e o segundo começa onde o primeiro acaba. Existem mais fragmentos a seguir a este, pois a flag *More fragments* está em “Set” (1) e se não houvesse estaria “Not set” (0).

1.3.d Alínea d)

Pergunta: *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 *datagrama* original foi dividido em três fragmentos e o último deles transporta 536 bytes.

Cálculos auxiliares:

- *Número de fragmentos gerados a partir do datagrama IP original :*

$$3536 : 1500 = 3$$

- *Nº bytes transportados no último fragmento:*

$$3536 - 2 * 1500 = 536$$

Apresentámos agora abaixo os resultados obtidos através do *Wireshark*.

Tal como se pode ver está a ser fragmentado de 3 em 3 o que prova o nosso valor de *Número de fragmentos gerados a partir do datagrama IP original*.

| | | | | | |
|----|----------|----------------|----------------|------|--|
| 12 | 2.034589 | 172.26.126.110 | 193.136.9.240 | ICMP | 1514 Echo (ping) request id=0x0001, seq=9/2304, ttl=128 (reply in 15) |
| 13 | 2.034589 | 172.26.126.110 | 193.136.9.240 | IPv4 | 1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=e447) |
| 14 | 2.034589 | 172.26.126.110 | 193.136.9.240 | IPv4 | 618 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=e447) |
| 15 | 2.039216 | 193.136.9.240 | 172.26.126.110 | ICMP | 1514 Echo (ping) reply id=0x0001, seq=9/2304, ttl=61 (request in 12) |
| 16 | 2.039216 | 193.136.9.240 | 172.26.126.110 | IPv4 | 618 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=1c1f) |
| 17 | 2.039216 | 193.136.9.240 | 172.26.126.110 | IPv4 | 1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1c1f) |
| 18 | 3.048811 | 172.26.126.110 | 193.136.9.240 | ICMP | 1514 Echo (ping) request id=0x0001, seq=10/2560, ttl=128 (reply in 21) |
| 19 | 3.048811 | 172.26.126.110 | 193.136.9.240 | IPv4 | 1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=e448) |
| 20 | 3.048811 | 172.26.126.110 | 193.136.9.240 | IPv4 | 618 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=e448) |

Figura 20- Output Wireshark

De seguida, pela figura abaixo, verificamos que não chegou a 1500, logo não há suficiente para exceder o MTU logo é o fim da informação.

```

v Internet Protocol Version 4, Src: 172.26.126.110, 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: 604
    Identification: 0xe447 (58439)
  v 000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0001 0111 0010 = Fragment Offset: 2960
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.126.110
    Destination Address: 193.136.9.240
  
```

Figura 21- Detalhe output do Wireshark

Nº bytes transportados no último fragmento:

Deu 604 como se vê na imagem abaixo que não foi o que nos tinha dado pois ele recebe bytes extra de “lixo” para a fragmentação e reassembly.

Total Length: 604

Figura 22- bytes transportados no último fragmento

1.3.e Alínea e)

Pergunta: 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.

A deteção do último fragmento correspondente ao datagrama original é feita através do campo *More Fragments* no cabeçalho do datagrama IP, quando este campo passar a ter “Not Set” sabemos que estamos perante este último fragmento.

..0. = More fragments: Not set

Figura 23- Detalhe campo More fragments

De seguida, encontra-se o filtro estabelecido no Wireshark que permite listar o último fragmento do primeiro datagrama IP segmentado.

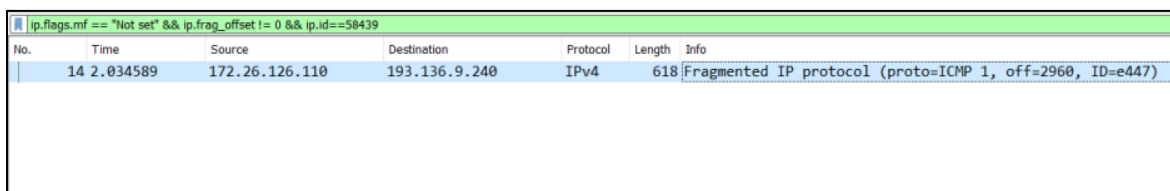
- `ip.flags.mf == "Not set"` -> prova que já não há mais fragmentos para além deste, porém isto aconteceria mesmo que não tivesse sido fragmentado em primeiro lugar, portanto é necessário provar que foi fragmentado inicialmente e que não há mais pedaços de fragmento;
- `&& ip.frag_offset != 0` -> Significa que foi fragmentado;
- `&& ip.id==58439` -> id do primeiro datagrama.

Identification: 0xe447 (58439)

Figura 24

Filtro Completo: `ip.flags.mf == "Not set" && ip.frag_offset != 0 && ip.id == 58439`

Após aplicarmos este filtro no Wireshark iremos obter exatamente o último fragmento correspondente ao datagrama original.



The image shows a Wireshark packet list with a single entry. The filter bar at the top contains the expression: `ip.flags.mf == "Not set" && ip.frag_offset != 0 && ip.id == 58439`. The packet list table has columns: No., Time, Source, Destination, Protocol, Length, and Info. The single entry is:

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------|----------------|---------------|----------|--------|--|
| 14 | 2.034589 | 172.26.126.110 | 193.136.9.240 | IPv4 | 618 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=e447) |

Figura 25- Output obtido após utilização do filtro completo

1.3.f Alínea f)

Pergunta: *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 equipamento destino, neste caso seria marco.uminho.pt.

A reconstrução não poderia ter ocorrido noutro equipamento, pois se os *routers* tivessem ocupados a fazer o *reassembly* ia ficar muito lento.

1.3.g Alínea g)

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

Os campos do cabeçalho IP que mudam entre os fragmentos, como se verifica nas imagens abaixo, são: o *fragment offset*, a *total length* e a *flag More fragments*.

```

v Internet Protocol Version 4, Src: 172.26.126.110, 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: 0xe447 (58439)
v 001. .... = Flags: 0x1, More fragments
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..1. .... = More fragments: Set
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 128
  Protocol: ICMP (1)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.126.110
  Destination Address: 193.136.9.240

```

Figura 26

```

v Internet Protocol Version 4, Src: 172.26.126.110, 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: 0xe447 (58439)
v 001. .... = Flags: 0x1, More fragments
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..1. .... = More fragments: Set
  ...0 0000 1011 1001 = Fragment Offset: 1480
  Time to Live: 128
  Protocol: ICMP (1)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.126.110
  Destination Address: 193.136.9.240

```

Figura 27

```

v Internet Protocol Version 4, Src: 172.26.126.110, 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: 604
  Identification: 0xe447 (58439)
v 000. .... = Flags: 0x0
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..0. .... = More fragments: Not set
  ...0 0001 0111 0010 = Fragment Offset: 2960
  Time to Live: 128
  Protocol: ICMP (1)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.126.110
  Destination Address: 193.136.9.240

```

Figura 28

A medida que os fragmentos vão chegando ao host de destino, este sabe que recebeu o primeiro fragmento quando a *flag More fragments* está a 1 e o *fragment offset* a 0. Posteriormente, ele vai recebendo os seguintes fragmentos cujos valores da *flag -More fragments* e do *fragment offset* são diferentes de 0. Ao receber o último fragmento, o *fragment offset* será diferente de 0 e a *flag More fragments* terá o valor 0, para além disso, terá uma *total length* menor do MTU o que indica que seria possível transferir mais, mas não há mais informação. No fim deste processo, ele faz o *reassembly* destes fragmentos cuja ordem é dada pelo *fragment offset*.

1.3.h Alínea h)

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

O primeiro fragmento de cada pacote é identificado como sendo um ICMP, uma vez que este contém o cabeçalho do pacote original, incluindo o campo “Protocolo”. Este campo indica o tipo de protocolo usado no pacote original. Os restantes fragmentos de um pacote IP são idênticos ao primeiro fragmento, exceto no campo do *offset* do fragmento, que indica a posição do fragmento no pacote original, eles não podem ser identificados como pacotes ICMP, pois eles contêm apenas parte do *payload* original e não têm informações suficientes para serem identificados como pacotes ICMP, em vez disso eles são simplesmente partes do pacote original, que podem ser reagrupados pelo destinatário para reconstruir o pacote original.

1.3.i Alínea i)

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

O tamanho do datagrama é comparado com o MTU (*Maximum Transmission Unit*) do link de rede para determinar se ele deve ser fragmentado ou não. O MTU é o tamanho máximo do pacote que pode ser transmitido por um link de rede. Se o tamanho do datagrama exceder o MTU do link, o datagrama é fragmentado em vários pacotes menores que se encaixam no MTU. Os efeitos na rede ao aumentar ou diminuir o valor do MTU podem afetar a eficiência e a qualidade da transmissão de dados.

Se o MTU for aumentado os pacotes de dados poderão ser transmitidos com menor fragmentação, o que pode melhorar a eficiência da transmissão de dados, uma vez que, uma menor fragmentação significa menos pacotes e menos sobrecarga no processamento de fragmentação e retransmissão de pacotes. Porém, um MTU maior também pode aumentar possibilidade de perda de pacotes em redes com alta taxa de erros de transmissão, o que pode afetar pela negativa a qualidade da transmissão.

Por outro lado, se diminuíssemos o MTU poderia levar ao aumento da fragmentação de pacotes e diminuir a eficiência da transmissão dos dados. Mais fragmentação significa mais pacotes, o que aumenta a sobrecarga no processamento de fragmentação e retransmissão de pacotes. Porém um MTU menor também pode reduzir a perda de pacotes em redes com alta taxa de erros de transmissão, melhorando a qualidade da transmissão.

1.3.j Alínea j)

Pergunta: Determine o valor máximo de SIZE sem que ocorra fragmentação do pacote? Justifique o valor obtido.
O valor máximo de SIZE sem que ocorra fragmentação do pacote é de 1472 porque apesar de 1500 ser o MTU 20 são para o cabeçalho e sobra 1480 mas, como há lixo tem de haver de haver espaço o mesmo.

2 - Segunda parte

2.1 Exercício 1

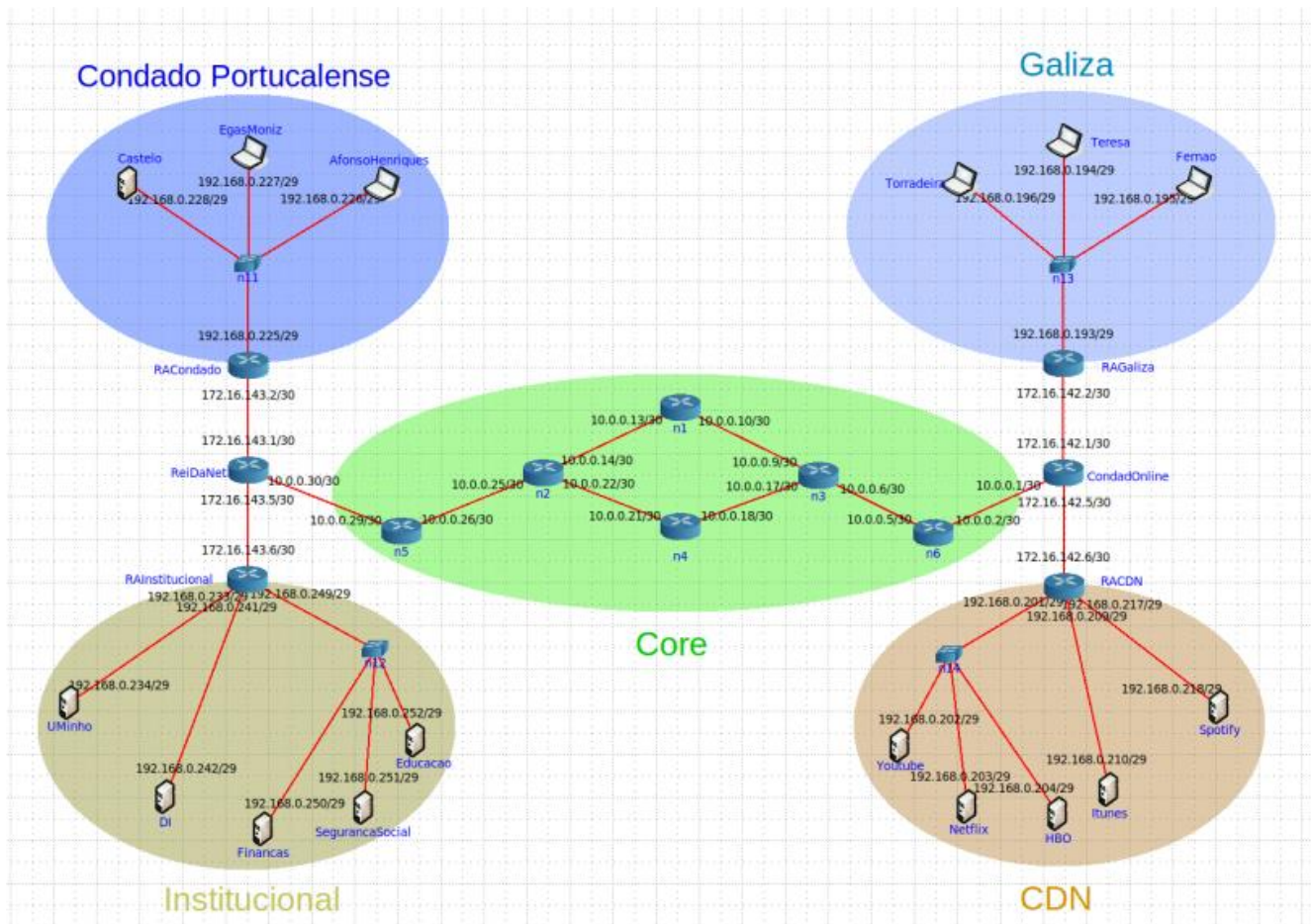


Figura 29-Topologia da Rede

2.1.a Alínea a)

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

Através do comando “ping <<IP>>” podemos confirmar que o Host “AfonsoHenriques” não tem conexão com o Host “Teresa”, mas tem, de facto, conexão com os Hosts “Finanças”, “Spotify”, “Itunes”, “Youtube”, “Netflix”, “HBO”.

Podemos provar isto com os comandos cuja execução é visível nas figuras a seguir.

```
--- 192.168.0.250 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10226ms
rtt min/avg/max/mdev = 0.059/0.109/0.203/0.036 ms
<ycore.41135/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=0.217 ms
64 bytes from 192.168.0.203: icmp_seq=2 ttl=55 time=0.148 ms
64 bytes from 192.168.0.203: icmp_seq=3 ttl=55 time=0.098 ms
64 bytes from 192.168.0.203: icmp_seq=4 ttl=55 time=0.103 ms
64 bytes from 192.168.0.203: icmp_seq=5 ttl=55 time=0.095 ms
64 bytes from 192.168.0.203: icmp_seq=6 ttl=55 time=0.194 ms
64 bytes from 192.168.0.203: icmp_seq=7 ttl=55 time=0.167 ms
64 bytes from 192.168.0.203: icmp_seq=8 ttl=55 time=0.987 ms
^C
--- 192.168.0.203 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7172ms
rtt min/avg/max/mdev = 0.095/0.251/0.987/0.281 ms
root@AfonsoHenriques:/tmp/pycore.41135/AfonsoHenriques.conf#
```

Figura 30 - Resultado do comando ping para o endereço Netflix

```
--- 192.168.0.218 ping statistics ---
22 packets transmitted, 22 received, 0% packet loss, time 21577ms
rtt min/avg/max/mdev = 0.102/0.192/0.374/0.063 ms
<core.41135/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=0.100 ms
64 bytes from 192.168.0.218: icmp_seq=2 ttl=55 time=0.196 ms
64 bytes from 192.168.0.218: icmp_seq=3 ttl=55 time=0.165 ms
64 bytes from 192.168.0.218: icmp_seq=4 ttl=55 time=0.182 ms
64 bytes from 192.168.0.218: icmp_seq=5 ttl=55 time=0.107 ms
64 bytes from 192.168.0.218: icmp_seq=6 ttl=55 time=0.264 ms
^C
--- 192.168.0.218 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5121ms
rtt min/avg/max/mdev = 0.100/0.169/0.264/0.055 ms
root@AfonsoHenriques:/tmp/pycore.41135/AfonsoHenriques.conf#
```

Figura 31 - Resultado do comando ping para o endereço Spotify

```
<ycore.37825/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=0.434 ms
64 bytes from 192.168.0.202: icmp_seq=2 ttl=55 time=0.095 ms
64 bytes from 192.168.0.202: icmp_seq=3 ttl=55 time=0.330 ms
64 bytes from 192.168.0.202: icmp_seq=4 ttl=55 time=0.103 ms
64 bytes from 192.168.0.202: icmp_seq=5 ttl=55 time=0.387 ms
64 bytes from 192.168.0.202: icmp_seq=6 ttl=55 time=0.319 ms
^C
--- 192.168.0.202 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5103ms
rtt min/avg/max/mdev = 0.095/0.278/0.434/0.132 ms
root@AfonsoHenriques:/tmp/pycore.37825/AfonsoHenriques.conf#
```

Figura 32 - Resultado do comando ping para o endereço Youtube

```

<core.37825/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=0.173 ms
64 bytes from 192.168.0.204: icmp_seq=2 ttl=55 time=0.142 ms
64 bytes from 192.168.0.204: icmp_seq=3 ttl=55 time=0.100 ms
64 bytes from 192.168.0.204: icmp_seq=4 ttl=55 time=0.101 ms
64 bytes from 192.168.0.204: icmp_seq=5 ttl=55 time=0.395 ms
^C
--- 192.168.0.204 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4079ms
rtt min/avg/max/mdev = 0.100/0.182/0.395/0.109 ms
root@AfonsoHenriques:/tmp/pycore.37825/AfonsoHenriques.conf#

```

Figura 33 - Resultado do comando ping para o endereço HBO

```

<core.37825/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=0.498 ms
64 bytes from 192.168.0.210: icmp_seq=2 ttl=55 time=0.086 ms
64 bytes from 192.168.0.210: icmp_seq=3 ttl=55 time=0.090 ms
64 bytes from 192.168.0.210: icmp_seq=4 ttl=55 time=0.166 ms
64 bytes from 192.168.0.210: icmp_seq=5 ttl=55 time=0.208 ms
^C
--- 192.168.0.210 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4081ms
rtt min/avg/max/mdev = 0.086/0.209/0.498/0.151 ms
root@AfonsoHenriques:/tmp/pycore.37825/AfonsoHenriques.conf#

```

Figura 34 - Resultado do comando ping para o endereço Itunes

2.1.b Alínea b)

Pergunta: *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.*

Através do comando “`netstat -rn`” em cada um dos terminais de “AfonsoHenriques” e “Teresa” podemos verificar que não há nenhum erro com as suas tabelas de encaminhamento, pois ambas as entidades correspondem ao IP da interface do router a que estão conectados.

“AfonsoHenriques” está ligado à interface de IP 192.168.0.228 assim como confirmado pela tabela de encaminhamento presente na Figura 36.

“Teresa” está ligado à interface de IP 192.168.0.193 assim como confirmado pela tabela de encaminhamento presente na Figura 37.

As entradas de cada uma das tabelas são muito importantes para identificar os possíveis saltos e informações sobre a rede/conexão, permitindo uma melhor e mais eficiente gestão de rotas.

```

vcmd
root@AfonsoHenriques:/tmp/pycore.37825/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
root@AfonsoHenriques:/tmp/pycore.37825/AfonsoHenriques.conf#

```

Figura 35 - Resultado da execução do comando `netstat` para AfonsoHenriques


```

vcmd
root@Teresa:/tmp/pycore.37825/Teresa.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt  Iface
0.0.0.0          192.168.0.193  0.0.0.0         UG        0 0        0    eth0
192.168.0.192    0.0.0.0        255.255.255.248 U        0 0        0    eth0
root@Teresa:/tmp/pycore.37825/Teresa.conf#

```

Figura 36 - Resultado da execução do comando netstat para Teresa

2.1.c Alínea c)

Pergunta: 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.

Utilizando o Wireshark para analisar o tráfego entre “AfonsoHenriques” e “Teresa”, Figura 37, verificamos que é enviada uma mensagem de erro no router da interface com IP 10.0.0.29, ou seja, o router “n5”, ao analisar a sua tabela de encaminhamento verificamos que não existe rota com o destino “Teresa”, Figura 39, logo devemos adicionar uma rota com o comando “`ip route add 192.168.0.192/29 via 10.0.0.25`”, Figura 40.

Novamente, com o Wireshark vemos que falha no router da interface de IP 10.0.0.13, ou seja, o router “n1”, Figura 48. Desta vez, o router “n1” contém uma rota para “Teresa”, porém o IP para o próximo salto (Gateway) está errado. Devemos então apagar essa rota e adicionar a rota correta, Figuras 49 e 50.

A partir daí, o tráfego chega a “CondadOnline”, logo o problema estava em “n5” e “n1” e a alínea está resolvida.

Nota:

Para corrigir “n1” – `ip route del 192.168.0.192/29 via 10.0.0.14` e `ip route add 192.168.0.192/29 via 10.0.0.9`
Devemos também apagar a rota que leva diretamente a Teresa no “n2”.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|----------------------|---------------|----------|--------|---|
| 1 | 0.000000000 | fe80::200:ff:feaa:18 | ff02::5 | OSPF | 90 | Hello Packet |
| 2 | 1.144944902 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet |
| 3 | 2.721273062 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=1/256, ttl=1 (no response.. |
| 4 | 2.721333337 | 192.168.0.225 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (time to live exceeded in transit) |
| 5 | 2.721359360 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=2/512, ttl=1 (no response.. |
| 6 | 2.721359724 | 192.168.0.225 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (time to live exceeded in transit) |
| 7 | 2.721360890 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=3/768, ttl=1 (no response.. |
| 8 | 2.721364840 | 192.168.0.225 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (time to live exceeded in transit) |
| 9 | 2.721368781 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=4/1024, ttl=1 (no respons.. |
| 10 | 2.721368508 | 172.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (time to live exceeded in transit) |
| 11 | 2.721705218 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=5/1280, ttl=2 (no respons.. |
| 12 | 2.721712630 | 172.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (time to live exceeded in transit) |
| 13 | 2.721726193 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=6/1536, ttl=2 (no respons.. |
| 14 | 2.721726302 | 172.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (time to live exceeded in transit) |
| 15 | 2.721730534 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=7/1792, ttl=3 (no respons.. |
| 16 | 2.721751205 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Destination unreachable (Network unreachable) |
| 17 | 2.721755158 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=8/2048, ttl=3 (no respons.. |
| 18 | 2.721760990 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Destination unreachable (Network unreachable) |
| 19 | 2.721767424 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=9/2304, ttl=3 (no respons.. |
| 20 | 2.721777338 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=10/2560, ttl=4 (no respons.. |
| 21 | 2.721783569 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=11/2816, ttl=4 (no respons.. |
| 22 | 2.721789066 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=12/3072, ttl=4 (no respons.. |
| 23 | 2.721794639 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=13/3328, ttl=5 (no respons.. |
| 24 | 2.721799682 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=14/3584, ttl=5 (no respons.. |
| 25 | 2.721804719 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=15/3840, ttl=5 (no respons.. |
| 26 | 2.721810467 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0031, seq=16/4096, ttl=6 (no respons.. |
| 27 | 3.146919964 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet |
| 28 | 5.147491899 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet |

Figura 37 - Resultado da execução do Wireshark

```

pycore.37825/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225) 0.081 ms 0.011 ms 0.006 ms
 2 172.16.143.1 (172.16.143.1) 0.320 ms 0.017 ms 0.008 ms
 3 10.0.0.29 (10.0.0.29) 0.023 ms !N 0.010 ms !N *
root@AfonsoHenriques:/tmp/pycore.37825/AfonsoHenriques.conf#

```

Figura 38 - Output AfonsoHenriques (comprova que não passou no n5)

```

vcmd
root@n5:/tmp/pycore.37825/n5.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0         10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.4         10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.8         10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.12        10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.16        10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.20        10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.24        0.0.0.0        255.255.255.252 U        0 0        0 eth1
10.0.0.28        0.0.0.0        255.255.255.252 U        0 0        0 eth0
172.0.0.0        10.0.0.30      255.0.0.0       UG      0 0        0 eth0
172.16.142.0     10.0.0.25      255.255.255.248 UG      0 0        0 eth1
172.16.143.0     10.0.0.30      255.255.255.252 UG      0 0        0 eth0
172.16.143.0     10.0.0.30      255.255.255.248 UG      0 0        0 eth0
172.16.143.4     10.0.0.30      255.255.255.252 UG      0 0        0 eth0
192.142.0.4      10.0.0.25      255.255.255.252 UG      0 0        0 eth1
192.168.0.200    10.0.0.25      255.255.255.248 UG      0 0        0 eth1
192.168.0.208    10.0.0.25      255.255.255.248 UG      0 0        0 eth1
192.168.0.216    10.0.0.25      255.255.255.248 UG      0 0        0 eth1
192.168.0.224    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
192.168.0.232    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
192.168.0.240    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
192.168.0.248    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
root@n5:/tmp/pycore.37825/n5.conf#

```

Figura 39 – Tabela de encaminhamento de n5 (nenhuma rota leva a Teresa)

```

root@n5:/tmp/pycore.37825/n5.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0         10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.4         10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.8         10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.12        10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.16        10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.20        10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.24        0.0.0.0        255.255.255.252 U        0 0        0 eth1
10.0.0.28        0.0.0.0        255.255.255.252 U        0 0        0 eth0
172.0.0.0        10.0.0.30      255.0.0.0       UG      0 0        0 eth0
172.16.142.0     10.0.0.25      255.255.255.248 UG      0 0        0 eth1
172.16.143.0     10.0.0.30      255.255.255.252 UG      0 0        0 eth0
172.16.143.0     10.0.0.30      255.255.255.248 UG      0 0        0 eth0
172.16.143.4     10.0.0.30      255.255.255.252 UG      0 0        0 eth0
192.142.0.4      10.0.0.25      255.255.255.252 UG      0 0        0 eth1
192.168.0.200    10.0.0.25      255.255.255.248 UG      0 0        0 eth1
192.168.0.208    10.0.0.25      255.255.255.248 UG      0 0        0 eth1
192.168.0.216    10.0.0.25      255.255.255.248 UG      0 0        0 eth1
192.168.0.224    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
192.168.0.232    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
192.168.0.240    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
192.168.0.248    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
root@n5:/tmp/pycore.37825/n5.conf# ip route add 192.168.0.192/29 via 10.0.0.25
root@n5:/tmp/pycore.37825/n5.conf#

```

Figura 40 – Acrescentada à tabela de encaminhamento anterior a rota necessária

```

vcmd
root@n5:/tmp/pycore.37825/n5.conf# ip route add 192.168.0.192/29 via 10.0.0.25
root@n5:/tmp/pycore.37825/n5.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0         10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.4         10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.8         10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.12        10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.16        10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.20        10.0.0.25      255.255.255.252 UG      0 0        0 eth1
10.0.0.24        0.0.0.0        255.255.255.252 U        0 0        0 eth1
10.0.0.28        0.0.0.0        255.255.255.252 U        0 0        0 eth0
172.0.0.0        10.0.0.30      255.0.0.0       UG      0 0        0 eth0
172.16.142.0     10.0.0.25      255.255.255.248 UG      0 0        0 eth1
172.16.143.0     10.0.0.30      255.255.255.252 UG      0 0        0 eth0
172.16.143.0     10.0.0.30      255.255.255.248 UG      0 0        0 eth0
172.16.143.4     10.0.0.30      255.255.255.252 UG      0 0        0 eth0
192.142.0.4      10.0.0.25      255.255.255.252 UG      0 0        0 eth1
192.168.0.192    10.0.0.25      255.255.255.248 UG      0 0        0 eth1
192.168.0.200    10.0.0.25      255.255.255.248 UG      0 0        0 eth1
192.168.0.208    10.0.0.25      255.255.255.248 UG      0 0        0 eth1
192.168.0.216    10.0.0.25      255.255.255.248 UG      0 0        0 eth1
192.168.0.224    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
192.168.0.232    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
192.168.0.240    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
192.168.0.248    10.0.0.30      255.255.255.248 UG      0 0        0 eth0
root@n5:/tmp/pycore.37825/n5.conf#

```

Figura 41 - Execução do comando traceroute para provar que a rota foi corretamente adicionada

```

vcmd
root@AfonsoHenriques:/tmp/pycore.37825/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
pycore.37825/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225) 0.081 ms 0.011 ms 0.006 ms
 2 172.16.143.1 (172.16.143.1) 0.320 ms 0.017 ms 0.008 ms
 3 10.0.0.29 (10.0.0.29) 0.023 ms !N 0.010 ms !N *
pycore.37825/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225) 0.042 ms 0.008 ms 0.006 ms
 2 172.16.143.1 (172.16.143.1) 0.019 ms 0.008 ms 0.007 ms
 3 10.0.0.29 (10.0.0.29) 0.022 ms 0.010 ms 0.009 ms
 4 10.0.0.25 (10.0.0.25) 0.127 ms 0.019 ms 0.013 ms
 5 10.0.0.25 (10.0.0.25) 3068.431 ms !H 3068.398 ms !H 3068.386 ms !H
root@AfonsoHenriques:/tmp/pycore.37825/AfonsoHenriques.conf#

```

Figura 42 – Resultado da execução do traceroute (verificamos que agora o problema está em n2)

| | | | | | | | |
|----|-------------|----------------------|---------------|------|-----|--|--|
| 1 | 0.000000000 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet | |
| 2 | 0.943289905 | fe80::200:ff:feaa:18 | ff02::5 | OSPF | 90 | Hello Packet | |
| 3 | 1.771222417 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=1/256, ttl=1 (no response...) | |
| 4 | 1.771249998 | 192.168.0.226 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 5 | 1.771259098 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=2/512, ttl=1 (no response...) | |
| 6 | 1.771265002 | 192.168.0.226 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 7 | 1.771269086 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=3/768, ttl=1 (no response...) | |
| 8 | 1.771273195 | 192.168.0.226 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 9 | 1.771277159 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=4/1024, ttl=2 (no respons...) | |
| 10 | 1.771295284 | 172.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 11 | 1.771298991 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=5/1280, ttl=2 (no respons...) | |
| 12 | 1.771305053 | 172.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 13 | 1.771313900 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=6/1536, ttl=2 (no respons...) | |
| 14 | 1.771313960 | 172.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 15 | 1.771317491 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=7/1792, ttl=3 (no respons...) | |
| 16 | 1.771337893 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 17 | 1.771341765 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=8/2048, ttl=3 (no respons...) | |
| 18 | 1.771350125 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 19 | 1.771353686 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=9/2304, ttl=3 (no respons...) | |
| 20 | 1.771361547 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 21 | 1.771365442 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=10/2560, ttl=4 (no respon...) | |
| 22 | 1.771490686 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 23 | 1.771499544 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=11/2816, ttl=4 (no respon...) | |
| 24 | 1.771513518 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 25 | 1.771517741 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=12/3072, ttl=4 (no respon...) | |
| 26 | 1.771528789 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 27 | 1.771533115 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=13/3328, ttl=5 (no respon...) | |
| 28 | 1.771569128 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=14/3584, ttl=5 (no respon...) | |
| 29 | 1.771579604 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=15/3840, ttl=5 (no respon...) | |
| 30 | 1.771587802 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=16/4096, ttl=6 (no respon...) | |
| 31 | 1.772422159 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=17/4352, ttl=6 (no respon...) | |
| 32 | 1.772439810 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0032, seq=18/4608, ttl=6 (no respon...) | |

Figura 43 - Output Wireshark

```

root@n2:/tmp/pycore.37825/n2.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt  Iface
10.0.0.0          10.0.0.13       255.255.255.252 UG      0 0        0     eth1
10.0.0.4          10.0.0.21       255.255.255.252 UG      0 0        0     eth0
10.0.0.8          10.0.0.13       255.255.255.252 UG      0 0        0     eth1
10.0.0.12         0.0.0.0         255.255.255.252 U        0 0        0     eth1
10.0.0.16         10.0.0.13       255.255.255.252 UG      0 0        0     eth1
10.0.0.20         0.0.0.0         255.255.255.252 U        0 0        0     eth0
10.0.0.24         0.0.0.0         255.255.255.252 U        0 0        0     eth2
10.0.0.28         10.0.0.26       255.255.255.252 UG      0 0        0     eth2
172.0.0.0         10.0.0.26       255.0.0.0       UG      0 0        0     eth2
172.16.142.0      10.0.0.13       255.255.255.252 UG      0 0        0     eth1
172.16.142.4      10.0.0.21       255.255.255.252 UG      0 0        0     eth0
172.16.143.0      10.0.0.26       255.255.255.252 UG      0 0        0     eth2
172.16.143.4      10.0.0.26       255.255.255.252 UG      0 0        0     eth2
192.168.0.192     10.0.0.13       255.255.255.248 UG      0 0        0     eth1
192.168.0.194     10.0.0.25       255.255.255.254 UG      0 0        0     eth2
192.168.0.200     10.0.0.21       255.255.255.248 UG      0 0        0     eth0
192.168.0.208     10.0.0.21       255.255.255.248 UG      0 0        0     eth0
192.168.0.216     10.0.0.21       255.255.255.248 UG      0 0        0     eth0
192.168.0.224     10.0.0.26       255.255.255.248 UG      0 0        0     eth2
192.168.0.232     10.0.0.26       255.255.255.248 UG      0 0        0     eth2
192.168.0.240     10.0.0.26       255.255.255.248 UG      0 0        0     eth2
192.168.0.248     10.0.0.26       255.255.255.248 UG      0 0        0     eth2

```

Figura 44 - Resultado da execução do comando netstat


```
<2.conf# ip route del 192.168.0.194/31 via 10.0.0.25
root@n2:/tmp/pycore.37825/n2.conf# netstat -rn
Kernel IP routing table
Destination        Gateway           Genmask          Flags        MSS Window  irtt  Iface
10.0.0.0           10.0.0.13        255.255.255.252 UG           0 0        0     eth1
10.0.0.4           10.0.0.21        255.255.255.252 UG           0 0        0     eth0
10.0.0.8           10.0.0.13        255.255.255.252 UG           0 0        0     eth1
10.0.0.12          0.0.0.0          255.255.255.252 U            0 0        0     eth1
10.0.0.16          10.0.0.13        255.255.255.252 UG           0 0        0     eth1
10.0.0.20          0.0.0.0          255.255.255.252 U            0 0        0     eth0
10.0.0.24          0.0.0.0          255.255.255.252 U            0 0        0     eth2
10.0.0.28          10.0.0.26        255.255.255.252 UG           0 0        0     eth2
172.0.0.0          10.0.0.26        255.0.0.0        UG           0 0        0     eth2
172.16.142.0       10.0.0.13        255.255.255.252 UG           0 0        0     eth1
172.16.142.4       10.0.0.21        255.255.255.252 UG           0 0        0     eth0
172.16.143.0       10.0.0.26        255.255.255.252 UG           0 0        0     eth2
172.16.143.4       10.0.0.26        255.255.255.252 UG           0 0        0     eth2
192.168.0.192      10.0.0.13        255.255.255.248 UG           0 0        0     eth1
192.168.0.200      10.0.0.21        255.255.255.248 UG           0 0        0     eth0
192.168.0.208      10.0.0.21        255.255.255.248 UG           0 0        0     eth0
192.168.0.216      10.0.0.21        255.255.255.248 UG           0 0        0     eth0
192.168.0.224      10.0.0.26        255.255.255.248 UG           0 0        0     eth2
192.168.0.232      10.0.0.26        255.255.255.248 UG           0 0        0     eth2
192.168.0.240      10.0.0.26        255.255.255.248 UG           0 0        0     eth2
192.168.0.248      10.0.0.26        255.255.255.248 UG           0 0        0     eth2
root@n2:/tmp/pycore.37825/n2.conf#
```

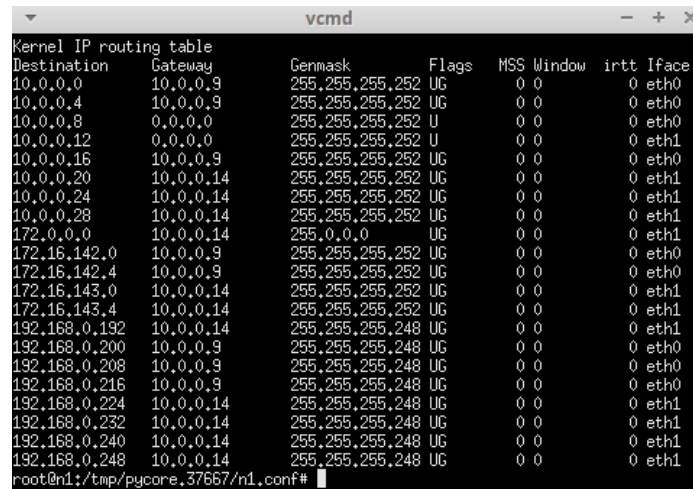
Figura 45 - Tabela de encaminhamento após a remoção da rota

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|---------------|---------------|----------|--------|--|
| 1 | 0.000000000 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet |
| 2 | 1.756320893 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=1/256, ttl=1 (no response...) |
| 3 | 1.756359244 | 192.168.0.225 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 4 | 1.756368161 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=2/512, ttl=1 (no response...) |
| 5 | 1.756373658 | 192.168.0.225 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 6 | 1.756376807 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=3/768, ttl=1 (no response...) |
| 7 | 1.756380752 | 192.168.0.225 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 8 | 1.756384318 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=4/1024, ttl=2 (no response...) |
| 9 | 1.756392009 | 192.168.0.225 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 10 | 1.756406491 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=5/1280, ttl=2 (no response...) |
| 11 | 1.756413978 | 192.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 12 | 1.756416515 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=6/1536, ttl=2 (no response...) |
| 13 | 1.756422360 | 192.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 14 | 1.756425901 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=7/1792, ttl=3 (no response...) |
| 15 | 1.756446722 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 16 | 1.756450611 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=8/2048, ttl=3 (no response...) |
| 17 | 1.756459208 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 18 | 1.756462517 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=9/2304, ttl=3 (no response...) |
| 19 | 1.756470742 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 20 | 1.756474474 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=10/2560, ttl=4 (no response...) |
| 21 | 1.756500740 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 22 | 1.756504656 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=11/2816, ttl=4 (no response...) |
| 23 | 1.756510992 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 24 | 1.756520547 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=12/3072, ttl=4 (no response...) |
| 25 | 1.756531631 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 26 | 1.756535092 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=13/3328, ttl=5 (no response...) |
| 27 | 1.756677287 | 10.0.0.13 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 28 | 1.756686952 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=14/3584, ttl=5 (no response...) |
| 29 | 1.756703231 | 10.0.0.13 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 30 | 1.756707395 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=15/3840, ttl=5 (no response...) |
| 31 | 1.756720253 | 10.0.0.13 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 32 | 1.756724737 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=16/4096, ttl=6 (no response...) |
| 33 | 1.756739515 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 34 | 1.757537664 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=17/4352, ttl=6 (no response...) |
| 35 | 1.757564681 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 36 | 1.757570180 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=18/4608, ttl=6 (no response...) |
| 37 | 1.757583452 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 38 | 1.757587708 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=19/4864, ttl=7 (no response...) |
| 39 | 1.757603140 | 10.0.0.13 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 40 | 1.757606582 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=20/5120, ttl=7 (no response...) |
| 41 | 1.757620761 | 10.0.0.13 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 42 | 1.757624401 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=21/5376, ttl=7 (no response...) |
| 43 | 1.757630992 | 10.0.0.13 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 44 | 1.757643638 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=22/5632, ttl=8 (no response...) |
| 45 | 1.757654857 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=23/5888, ttl=8 (no response...) |
| 46 | 1.757665677 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0033, seq=24/6144, ttl=8 (no response...) |

Figura 46 - Output Wireshark

```
<5/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225)  0.051 ms  0.007 ms  0.006 ms
 2 172.16.143.1 (172.16.143.1)  0.020 ms  0.008 ms  0.008 ms
 3 10.0.0.29 (10.0.0.29)  0.022 ms  0.010 ms  0.010 ms
 4 10.0.0.25 (10.0.0.25)  0.028 ms  0.014 ms  0.013 ms
 5 10.0.0.13 (10.0.0.13)  0.144 ms  0.021 ms  0.015 ms
 6 10.0.0.25 (10.0.0.25)  0.017 ms  0.032 ms  0.016 ms
 7 10.0.0.13 (10.0.0.13)  0.017 ms  0.016 ms  0.017 ms
 8 * * *
 9 * * *
10 * * *
11 * * *
12 * * *
```

Figura 47 - Resultado da execução do comando traceroute

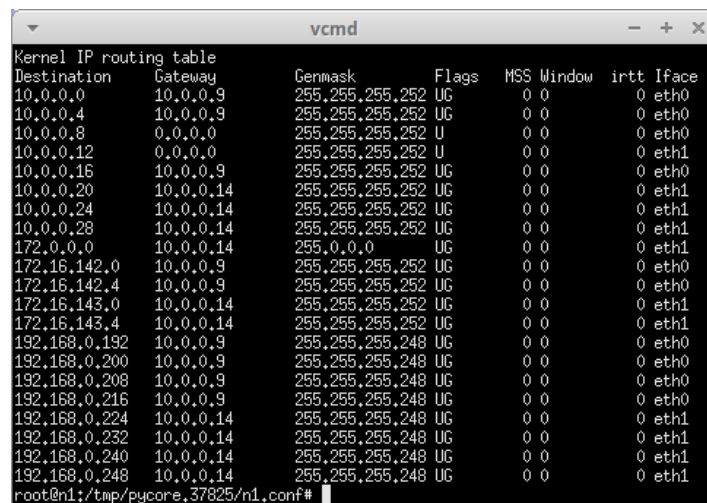


```

vcmd
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
10.0.0.4 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
10.0.0.8 0.0.0.0 255.255.255.252 U 0 0 0 eth0
10.0.0.12 0.0.0.0 255.255.255.252 U 0 0 0 eth1
10.0.0.16 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
10.0.0.20 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
10.0.0.24 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
10.0.0.28 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
172.0.0.0 10.0.0.14 255.0.0.0 UG 0 0 0 eth1
172.16.142.0 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
172.16.142.4 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
172.16.143.0 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
172.16.143.4 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
192.168.0.192 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
192.168.0.200 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.208 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.216 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.224 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
192.168.0.232 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
192.168.0.240 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
192.168.0.248 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
root@n1:/tmp/pycore.37667/n1.conf#

```

Figura 48 - Prova de que falha

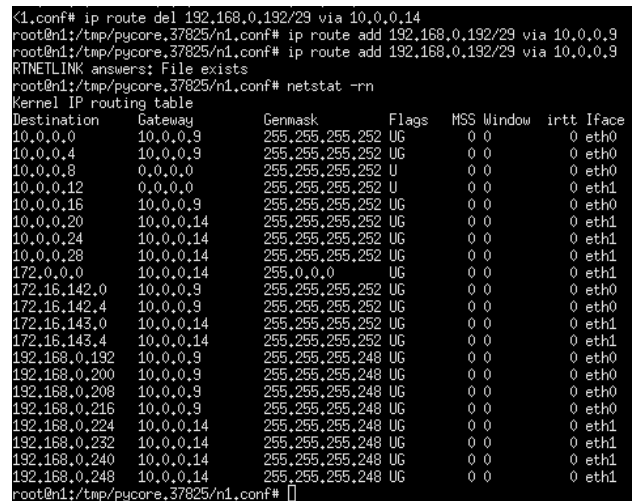


```

vcmd
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
10.0.0.4 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
10.0.0.8 0.0.0.0 255.255.255.252 U 0 0 0 eth0
10.0.0.12 0.0.0.0 255.255.255.252 U 0 0 0 eth1
10.0.0.16 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
10.0.0.20 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
10.0.0.24 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
10.0.0.28 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
172.0.0.0 10.0.0.14 255.0.0.0 UG 0 0 0 eth1
172.16.142.0 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
172.16.142.4 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
172.16.143.0 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
172.16.143.4 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
192.168.0.192 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.200 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.208 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.216 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.224 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
192.168.0.232 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
192.168.0.240 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
192.168.0.248 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
root@n1:/tmp/pycore.37825/n1.conf#

```

Figura 49 - Correção do problema anterior



```

1.conf# ip route del 192.168.0.192/29 via 10.0.0.14
root@n1:/tmp/pycore.37825/n1.conf# ip route add 192.168.0.192/29 via 10.0.0.9
root@n1:/tmp/pycore.37825/n1.conf# ip route add 192.168.0.192/29 via 10.0.0.9
RINETLINK answers: File exists
root@n1:/tmp/pycore.37825/n1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
10.0.0.4 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
10.0.0.8 0.0.0.0 255.255.255.252 U 0 0 0 eth0
10.0.0.12 0.0.0.0 255.255.255.252 U 0 0 0 eth1
10.0.0.16 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
10.0.0.20 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
10.0.0.24 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
10.0.0.28 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
172.0.0.0 10.0.0.14 255.0.0.0 UG 0 0 0 eth1
172.16.142.0 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
172.16.142.4 10.0.0.9 255.255.255.252 UG 0 0 0 eth0
172.16.143.0 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
172.16.143.4 10.0.0.14 255.255.255.252 UG 0 0 0 eth1
192.168.0.192 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.200 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.208 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.216 10.0.0.9 255.255.255.248 UG 0 0 0 eth0
192.168.0.224 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
192.168.0.232 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
192.168.0.240 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
192.168.0.248 10.0.0.14 255.255.255.248 UG 0 0 0 eth1
root@n1:/tmp/pycore.37825/n1.conf#

```

Figura 50 - Tabela de encaminhamento após a correção do erro

```

<5/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225)  0.055 ms  0.007 ms  0.005 ms
 2 172.16.143.1 (172.16.143.1)  0.023 ms  0.008 ms  0.008 ms
 3 10.0.0.29 (10.0.0.29)  0.024 ms  0.011 ms  0.010 ms
 4 10.0.0.25 (10.0.0.25)  0.045 ms  0.017 ms  0.013 ms
 5 10.0.0.13 (10.0.0.13)  0.031 ms  0.016 ms  0.014 ms
 6 10.0.0.17 (10.0.0.17)  0.170 ms  0.086 ms  0.026 ms
 7 10.0.0.5 (10.0.0.5)  0.133 ms  0.030 ms  0.023 ms
 8 10.0.0.1 (10.0.0.1)  0.121 ms  0.031 ms  0.025 ms
 9 * * *
10 * * *
11 * * *
12 * * *
13 * * *
14 * * C
root@AfonsoHenriques:/tmp/pycore.37825/AfonsoHenriques.conf#

```

Figura 51 - Resultado da execução do comando traceroute

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|---------------|---------------|----------|--------|--|
| 8 | 2.564401494 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=3/768, ttl=1 (no response...) |
| 9 | 2.564405013 | 192.168.0.226 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 10 | 2.564408806 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=4/1024, ttl=2 (no response...) |
| 11 | 2.564427093 | 172.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 12 | 2.564430971 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=5/1280, ttl=2 (no response...) |
| 13 | 2.564437634 | 172.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 14 | 2.564440921 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=6/1536, ttl=2 (no response...) |
| 15 | 2.564447057 | 172.16.143.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 16 | 2.564450736 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=7/1792, ttl=3 (no response...) |
| 17 | 2.564470902 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 18 | 2.564474793 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=8/2048, ttl=3 (no response...) |
| 19 | 2.564484129 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 20 | 2.564487763 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=9/2304, ttl=3 (no response...) |
| 21 | 2.564496118 | 10.0.0.29 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 22 | 2.564499895 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=10/2560, ttl=4 (no response...) |
| 23 | 2.564502745 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 24 | 2.564505905 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=11/2816, ttl=4 (no response...) |
| 25 | 2.564509348 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 26 | 2.564539427 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=12/3072, ttl=4 (no response...) |
| 27 | 2.564550818 | 10.0.0.25 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 28 | 2.564553819 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=13/3328, ttl=5 (no response...) |
| 29 | 2.564558201 | 10.0.0.13 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 30 | 2.564558882 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=14/3584, ttl=5 (no response...) |
| 31 | 2.564599822 | 10.0.0.13 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 32 | 2.564603415 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=15/3840, ttl=5 (no response...) |
| 33 | 2.564616611 | 10.0.0.13 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 34 | 2.564620523 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=16/4096, ttl=6 (no response...) |
| 35 | 2.564608909 | 10.0.0.17 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 36 | 2.565806613 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=17/4352, ttl=6 (no response...) |
| 37 | 2.565858798 | 10.0.0.17 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 38 | 2.5658585237 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=18/4608, ttl=6 (no response...) |
| 39 | 2.565882951 | 10.0.0.17 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 40 | 2.565887584 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=19/4864, ttl=7 (no response...) |
| 41 | 2.565918130 | 10.0.0.5 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 42 | 2.565922228 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=20/5120, ttl=7 (no response...) |
| 43 | 2.565941765 | 10.0.0.5 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 44 | 2.565945622 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=21/5376, ttl=7 (no response...) |
| 45 | 2.565994141 | 10.0.0.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 46 | 2.565968185 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=22/5632, ttl=8 (no response...) |
| 47 | 2.565999464 | 10.0.0.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 48 | 2.566003478 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=23/5888, ttl=8 (no response...) |
| 49 | 2.566025995 | 10.0.0.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 50 | 2.566049961 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=24/6144, ttl=8 (no response...) |
| 51 | 2.566080493 | 10.0.0.1 | 192.168.0.226 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 52 | 2.566088321 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=25/6400, ttl=9 (no response...) |
| 53 | 2.566110518 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 | Echo (ping) request id=0x0035, seq=26/6656, ttl=9 (no response...) |

Figura 52 - Output do Wireshark

2.1.d Alínea d)

Pergunta: 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.

D1) Pergunta: 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.

Através da análise do Wireshark (Figura 54) concluímos que os pacotes são efetivamente recebidos pela “Teresa”, porém ela consegue mandar a resposta, porque o router da “RAGaliza” não tem, na sua tabela de encaminhamento uma entrada com destino igual á sub-rede “AfonsoHenriques”, havendo, portanto, a necessidade de a adicionar com o comando “ip route add 192.168.0.224/29 via 172.16.142.1”.

```
<pycore.37825/AfonsoHenriques.conf# ping 192.168.0.194
PING 192.168.0.194 (192.168.0.194) 56(84) bytes of data.
^C
--- 192.168.0.194 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9236ms

root@AfonsoHenriques:/tmp/pycore.37825/AfonsoHenriques.conf#
```

Figura 53 - Resultado da execução do comando ping para Teresa

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|----------------------|-------------------|----------|--------|--|
| 1 | 0.000000000 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet |
| 2 | 1.107324516 | 192.168.0.226 | 192.168.0.194 | ICMP | 98 | Echo (ping) request id=0x0037, seq=1/256, ttl=64 (no response found!) |
| 3 | 2.001304270 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet |
| 4 | 2.138443005 | 192.168.0.226 | 192.168.0.194 | ICMP | 98 | Echo (ping) request id=0x0037, seq=2/512, ttl=64 (no response found!) |
| 5 | 3.164838586 | 192.168.0.226 | 192.168.0.194 | ICMP | 98 | Echo (ping) request id=0x0037, seq=3/768, ttl=64 (no response found!) |
| 6 | 4.006661679 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet |
| 7 | 4.184059954 | 192.168.0.226 | 192.168.0.194 | ICMP | 98 | Echo (ping) request id=0x0037, seq=4/1024, ttl=64 (no response found!) |
| 8 | 5.207303861 | 192.168.0.226 | 192.168.0.194 | ICMP | 98 | Echo (ping) request id=0x0037, seq=5/1280, ttl=64 (no response found!) |
| 9 | 6.006642388 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet |
| 10 | 6.231271350 | 00:00:00:aa:00:19 | 00:00:00:aa:00:18 | ARP | 42 | Who has 192.168.0.225? Tell 192.168.0.226 |
| 11 | 6.231694252 | 00:00:00:aa:00:18 | 00:00:00:aa:00:19 | ARP | 42 | 192.168.0.225 is at 00:00:00:aa:00:18 |
| 12 | 6.231727028 | 192.168.0.226 | 192.168.0.194 | ICMP | 98 | Echo (ping) request id=0x0037, seq=6/1536, ttl=64 (no response found!) |
| 13 | 7.255200346 | 192.168.0.226 | 192.168.0.194 | ICMP | 98 | Echo (ping) request id=0x0037, seq=7/1792, ttl=64 (no response found!) |
| 14 | 8.007898876 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet |
| 15 | 8.278034990 | 192.168.0.226 | 192.168.0.194 | ICMP | 98 | Echo (ping) request id=0x0037, seq=8/2048, ttl=64 (no response found!) |
| 16 | 8.732188774 | fe80::200:ff:feaa:18 | ff02::5 | OSPF | 90 | Hello Packet |
| 17 | 9.302376168 | 192.168.0.226 | 192.168.0.194 | ICMP | 98 | Echo (ping) request id=0x0037, seq=9/2304, ttl=64 (no response found!) |
| 18 | 10.008712745 | 192.168.0.225 | 224.0.0.5 | OSPF | 78 | Hello Packet |

Figura 54 - Output do Wireshark (comprova que a Teresa recebe mas não responde)

```
vcmd
root@RAGaliza:/tmp/pycore.37825/RAGaliza.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.4 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.8 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.12 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.16 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.20 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.24 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.28 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
172.0.0.0 172.16.142.1 255.0.0.0 UG 0 0 0 eth0
172.16.142.0 0.0.0.0 255.255.255.252 U 0 0 0 eth0
172.16.142.4 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
172.16.143.0 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
172.16.143.4 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
192.168.0.192 0.0.0.0 255.255.255.248 U 0 0 0 eth1
192.168.0.200 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.208 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.216 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.232 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.240 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.248 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
root@RAGaliza:/tmp/pycore.37825/RAGaliza.conf#
```

Figura 55 - Tabela de encaminhamento do router Galiza

```
<7825/RAGaliza.conf# ip route add 192.168.0.224/29 via 172.16.142.1
root@RAGaliza:/tmp/pycore.37825/RAGaliza.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.4 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.8 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.12 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.16 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.20 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.24 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
10.0.0.28 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
172.0.0.0 172.16.142.1 255.0.0.0 UG 0 0 0 eth0
172.16.142.0 0.0.0.0 255.255.255.252 U 0 0 0 eth0
172.16.142.4 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
172.16.143.0 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
172.16.143.4 172.16.142.1 255.255.255.252 UG 0 0 0 eth0
192.168.0.192 0.0.0.0 255.255.255.248 U 0 0 0 eth1
192.168.0.200 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.208 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.216 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.224 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.232 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.240 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
192.168.0.248 172.16.142.1 255.255.255.248 UG 0 0 0 eth0
root@RAGaliza:/tmp/pycore.37825/RAGaliza.conf#
```

Figura 56 - Tabela de encaminhamento após a adição da rota


```

Core,37825/AfonsoHenriques.conf# ping 192.168.0.194
PING 192.168.0.194 (192.168.0.194) 56(84) bytes of data.
64 bytes from 192.168.0.194: icmp_seq=1 ttl=55 time=0.140 ms
64 bytes from 192.168.0.194: icmp_seq=2 ttl=55 time=0.388 ms
64 bytes from 192.168.0.194: icmp_seq=3 ttl=55 time=1.14 ms
64 bytes from 192.168.0.194: icmp_seq=4 ttl=55 time=0.290 ms
64 bytes from 192.168.0.194: icmp_seq=5 ttl=55 time=0.367 ms
64 bytes from 192.168.0.194: icmp_seq=6 ttl=55 time=0.356 ms
64 bytes from 192.168.0.194: icmp_seq=7 ttl=55 time=0.337 ms
64 bytes from 192.168.0.194: icmp_seq=8 ttl=55 time=0.294 ms
64 bytes from 192.168.0.194: icmp_seq=9 ttl=55 time=0.348 ms
64 bytes from 192.168.0.194: icmp_seq=10 ttl=55 time=0.376 ms
64 bytes from 192.168.0.194: icmp_seq=11 ttl=55 time=0.344 ms
64 bytes from 192.168.0.194: icmp_seq=12 ttl=55 time=0.151 ms
64 bytes from 192.168.0.194: icmp_seq=13 ttl=55 time=0.290 ms
64 bytes from 192.168.0.194: icmp_seq=14 ttl=55 time=0.332 ms
64 bytes from 192.168.0.194: icmp_seq=15 ttl=55 time=0.346 ms
64 bytes from 192.168.0.194: icmp_seq=16 ttl=55 time=0.280 ms
64 bytes from 192.168.0.194: icmp_seq=17 ttl=55 time=0.344 ms
64 bytes from 192.168.0.194: icmp_seq=18 ttl=55 time=0.298 ms
64 bytes from 192.168.0.194: icmp_seq=19 ttl=55 time=0.287 ms
64 bytes from 192.168.0.194: icmp_seq=20 ttl=55 time=0.347 ms
64 bytes from 192.168.0.194: icmp_seq=21 ttl=55 time=0.296 ms
64 bytes from 192.168.0.194: icmp_seq=22 ttl=55 time=0.343 ms

```

Figura 57 - Resultado da execução do comando ping

D2) Pergunta: 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.

Sim, as rotas dos pacotes ICMP *echo reply* são, de facto, as mesmas, mas em sentido inverso, que as rotas dos pacotes ICMP *echo request* enviados entre “AfonsoHenriques” e “Teresa” tal como se pode verificar nas Figuras 58 e 60, e os gráficos da rota seguida nos dois sentidos pelos pacotes ICMP encontram-se nas imagens 59 e 61.

| IP | Port | Time to Live | Time to Live Exceeded | Time to Live Exceeded (in Transit) |
|----|-----------|---------------|-----------------------|------------------------------------|
| 56 | 646269783 | 192.168.0.226 | 192.168.0.194 | ICMP |
| 57 | 646395264 | 192.168.0.194 | 192.168.0.226 | ICMP |
| 58 | 646310701 | 192.168.0.226 | 192.168.0.194 | ICMP |
| 59 | 646338615 | 192.168.0.194 | 192.168.0.226 | ICMP |
| 60 | 646342429 | 192.168.0.226 | 192.168.0.194 | ICMP |
| 61 | 646368955 | 192.168.0.194 | 192.168.0.226 | ICMP |
| 62 | 646373323 | 192.168.0.226 | 192.168.0.194 | ICMP |
| 63 | 646399416 | 192.168.0.194 | 192.168.0.226 | ICMP |
| 64 | 646403266 | 192.168.0.226 | 192.168.0.194 | ICMP |
| 65 | 646429184 | 192.168.0.194 | 192.168.0.226 | ICMP |

Figura 58 - Rotas pacotes ICMP entre AfonsoHenriques e Teresa

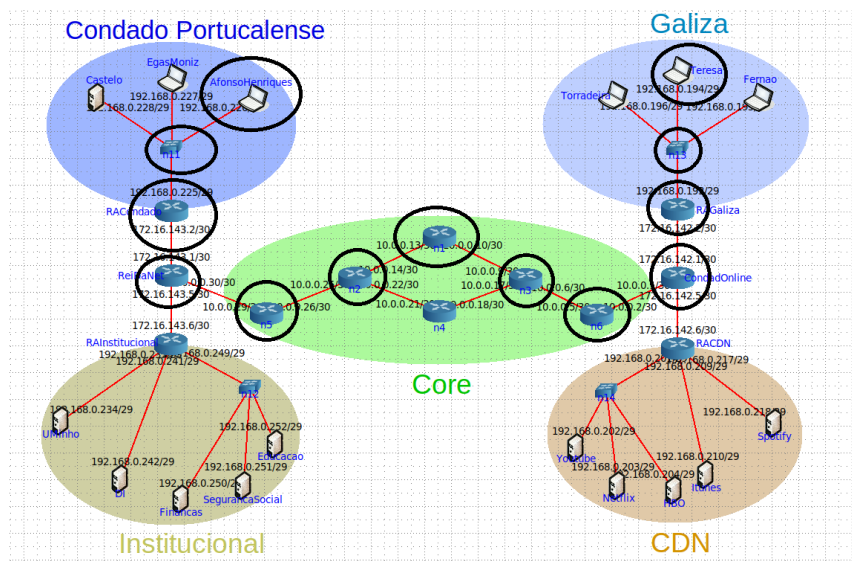


Figura 59 - Rota seguida no sentido AfonsoHenriques -> Teresa

| | | | | | | |
|----|-------------|---------------|---------------|------|------------------------|---|
| 57 | 3.142127441 | 192.168.0.194 | 192.168.0.226 | ICMP | 74 Echo (ping) request | id=0x0024, seq=28/7168, ttl=10 (reply in... |
| 58 | 3.142161051 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 Echo (ping) reply | id=0x0024, seq=28/7168, ttl=55 (request ... |
| 59 | 3.142166468 | 192.168.0.194 | 192.168.0.226 | ICMP | 74 Echo (ping) request | id=0x0024, seq=29/7424, ttl=10 (reply in... |
| 60 | 3.142192480 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 Echo (ping) reply | id=0x0024, seq=29/7424, ttl=55 (request ... |
| 61 | 3.142196362 | 192.168.0.194 | 192.168.0.226 | ICMP | 74 Echo (ping) request | id=0x0024, seq=30/7680, ttl=10 (reply in... |
| 62 | 3.142221736 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 Echo (ping) reply | id=0x0024, seq=30/7680, ttl=55 (request ... |
| 63 | 3.142225852 | 192.168.0.194 | 192.168.0.226 | ICMP | 74 Echo (ping) request | id=0x0024, seq=31/7936, ttl=11 (reply in... |
| 64 | 3.142251180 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 Echo (ping) reply | id=0x0024, seq=31/7936, ttl=55 (request ... |
| 65 | 3.142255019 | 192.168.0.194 | 192.168.0.226 | ICMP | 74 Echo (ping) request | id=0x0024, seq=32/8192, ttl=11 (reply in... |
| 66 | 3.142280274 | 192.168.0.226 | 192.168.0.194 | ICMP | 74 Echo (ping) reply | id=0x0024, seq=32/8192, ttl=55 (request ... |

Figura 60- Rotas pacotes ICMP entre Teresa e AfonsoHenriques

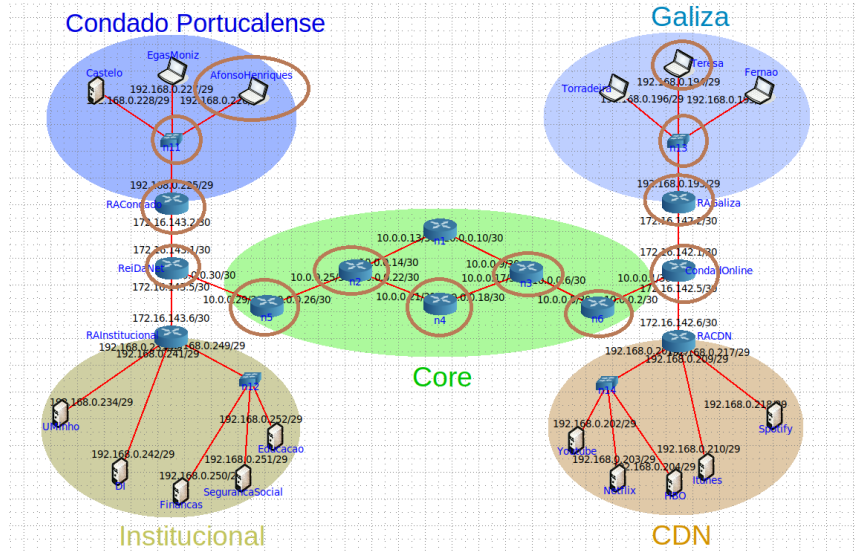


Figura 61 - Rota seguida no sentido Teresa -> AfonsoHenriques

2.1.e Alínea e)

Pergunta: 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.

Para o polo “Galiza” essa entrada não é utilizada pois o pacote estaria a andar para trás e nunca chegaria ao destino. Para o polo “CDN” a situação é a mesma. Logo, caso esta seja a entrada utilizada para o encaminhamento ela não permitirá o funcionamento esperado do dispositivo.

2.1.f Alínea f)

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

Todos os endereços usados são privados pois pertencem ao bloco de endereços privados e não se conectam à Internet.

IP addressing: reserved/private addr

Endereços reservados:

- os primeiros 4 bits não podem ser 1 (classe E)
- 127.x.x.x é o endereço reservado para *loopback*
- bits de host a 0s ou 1s (qualquer host, todos os hosts)
- bits de rede / subrede a 0s ou 1s (qualquer rede, todas as redes)

Endereços privados: atribuídos para internets privadas (sem conectividade IP global, não devem ser visíveis, nem são encaminhados na Internet) (ver RFC1918):

- bloco 192.168.0.0 - 192.168.255.255 / 16
- bloco 172.16.0.0 - 172.31.255.255 / 12
- bloco 10.0.0.0 - 10.255.255.255 / 8

Host com várias interfaces é designado de *multihomed*

Figura 62- IP addressing: reserved/private addr

Podemos ver que todos os endereços quer do core da rede como ISPs fazem parte do bloco de endereços privados.

2.1.g Alínea g)

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

Um *switch* geralmente tem um IP atribuído para permitir que estes possam ser gerenciados remotamente e de forma mais eficiente e possam funcionar como dispositivo intermediário para o *routing* do tráfego entre redes diferentes. O *switch* precisará de um IP para permitir a comunicação entre essas duas redes.

O endereço IP atribuído a um *switch* pode ajudar na gestão de rotas, especialmente em redes maiores e mais complexas. Quando um *switch* é configurado com um endereço IP, ele pode ser adicionado a uma tabela de roteamento, juntamente com outros dispositivos de rede, como roteadores, *firewalls*, servidores e outros *switches*. Com a tabela de roteamento atualizada, o *switch* pode determinar a melhor rota para encaminhar pacotes de dados de um dispositivo para outro, com base no endereço IP de destino. Se um pacote é destinado a outro dispositivo na mesma rede local, o *switch* pode encaminhá-lo diretamente para o destino, sem passar pelo roteador ou *gateway* padrão. Se o destino está em outra rede, o *switch* pode encaminhar o pacote para o roteador ou *gateway* padrão, que por sua vez encaminhará o pacote para a rede de destino.

2.2 Exercício 2

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

2.2.a Alínea a)

Pergunta: *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.*

Através do comando “`ip route del default`” conseguimos apagar a rota *default* que permite que “Castelo” conecte às outras redes.

Devemos depois adicionar rotas para substituir a função da rota *default*, que tem como utilidade evitar o uso de diversas rotas para cada lugar quando podemos apenas escrever a *default* e evitar encher a tabela de encaminhamento com dezenas de rotas desnecessárias.

```
root@Castelo:/tmp/pycore.37825/Castelo.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
root@Castelo:/tmp/pycore.37825/Castelo.conf# ip route del default
root@Castelo:/tmp/pycore.37825/Castelo.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
192.168.0.224 0.0.0.0 255.255.255.248 U 0 0 0 eth0
root@Castelo:/tmp/pycore.37825/Castelo.conf#
```

Figura 63 - Remoção da rota default

```
root@Castelo:/tmp/pycore.43109/Castelo.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
192.168.0.192 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.200 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.208 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.216 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.224 0.0.0.0 255.255.255.248 U 0 0 0 eth0
192.168.0.232 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.240 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.248 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
root@Castelo:/tmp/pycore.43109/Castelo.conf#
```

Figura 64-Tabela de encaminhamento do Castelo


```

192.168.0.248 192.168.0.229 255.255.255.248 0 0 0 0 etno
root@Castelo:/tmp/pycore.43109/Castelo.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=0.053 ms
64 bytes from 192.168.0.250: icmp_seq=2 ttl=61 time=0.050 ms
64 bytes from 192.168.0.250: icmp_seq=3 ttl=61 time=0.051 ms
^C
--- 192.168.0.250 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.050/0.051/0.053/0.001 ms
root@Castelo:/tmp/pycore.43109/Castelo.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=61 time=0.074 ms
64 bytes from 192.168.0.242: icmp_seq=2 ttl=61 time=0.067 ms
64 bytes from 192.168.0.242: icmp_seq=3 ttl=61 time=0.050 ms
^C
--- 192.168.0.242 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2058ms
rtt min/avg/max/mdev = 0.050/0.063/0.074/0.010 ms
root@Castelo:/tmp/pycore.43109/Castelo.conf#

```

Figura 65-Prova que já tem ligação

2.2.b Alínea b)

Pergunta: 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.

Como o nosso grupo é o PL36 então foi nos atribuído o IP 172.16.36.128/26.

Se queremos que as nossas redes possam ter 10 ou mais *hosts* temos de conceder pelo menos 4 bits do IP para o *host* pois se fosse 3 (ou menos) haveria apenas $2^3 - 2$ *hosts* disponíveis, ou seja 6 *hosts*, enquanto com 4 bits temos $2^4 - 2$, ou seja 14 *hosts*, logo temos de alterar a máscara para /28 (28 bits para a Rede e 4 bits para o *Host*).

Iremos então conectar o Castelo de Braga ao ISP ReiDaNet, logo a interface do ISP terá como IP 172.16.36.129/28 e o Castelo de Braga pode ter por exemplo 172.16.36.130/28.

2.2.c Alínea c)

Pergunta: 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?

Conectamos o Host Castelo de Braga ao router ReiDaNet e atribuímos ao Host o IP 172.16.36.130/28 e ao ReiDaNet o IP 172.16.36.129/28 pois este tem de ser imediatamente a seguir a 172.16.36.128/28.

Conseguimos verificar que o Host Castelo de Braga tem conexão com todos os *Hosts* exceto o “Castelo” pois o Castelo não tem rota *default* e não foi criada uma rota para o novo Host Castelo de Braga.

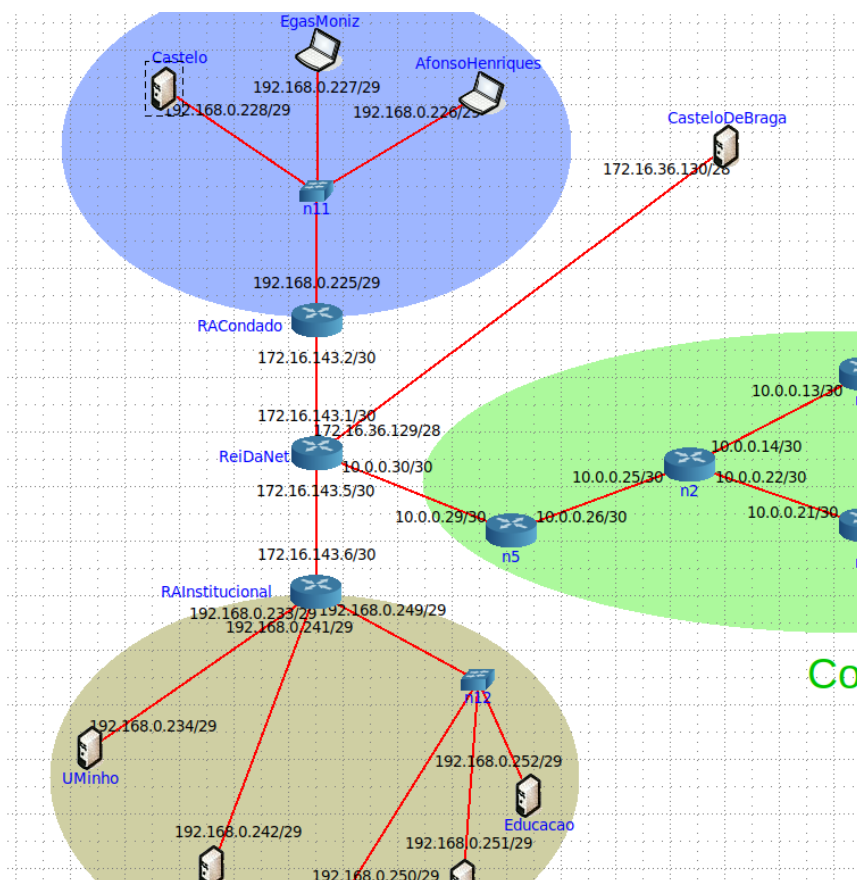


Figura 66- Castelo de Braga com Ip atribuido

```
root@CasteloDeBraga:/tmp/pycore.43109/CasteloDeBraga.conf# ping 192.168.0.228
PING 192.168.0.228 (192.168.0.228) 56(84) bytes of data.
^C
--- 192.168.0.228 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4099ms
root@CasteloDeBraga:/tmp/pycore.43109/CasteloDeBraga.conf#
```

Figura 67- Ping para o Castelo

```
root@CasteloDeBraga:/tmp/pycore.43109/CasteloDeBraga.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=62 time=0.064 ms
64 bytes from 192.168.0.250: icmp_seq=2 ttl=62 time=0.042 ms
64 bytes from 192.168.0.250: icmp_seq=3 ttl=62 time=0.143 ms
^C
--- 192.168.0.250 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.042/0.083/0.143/0.043 ms
root@CasteloDeBraga:/tmp/pycore.43109/CasteloDeBraga.conf#
```

Figura 68- Ping para outro (prova de que funciona)

2.3 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

2.3.1 Alínea 1 e Alínea 2

Pergunta 1: 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.

Após eliminar as rotas com destino a qualquer IP na “Galiza” ou “CDN”, aplicamos o comando “*ip route add 192.168.0.192/27 via 10.0.0.1*” e conseguimos assim criar um *Supernetting* que conecta os dois polos.

Pergunta2: Repita o processo descrito na alínea anterior para CondadoPortucalense e Institucional, também no dispositivo n6.

Apagamos todas as rotas com destino no “Condado Portucalense” e “Institucional”. Através do comando “*ip route add 192.168.0.224/27 via 10.0.0.6*” criamos outras *Supernet* agora para estes polos

Como ambas as alíneas pedem algo semelhante juntamos a resposta numa só.

Para fazermos o *supernetting* das rotas devemos apagar as entradas individuais para cada uma das redes e posteriormente adicionar uma única entrada que agrupe as entradas originais. Para criar uma entrada que consiga agrupar as entradas originais devemos começar por passar os IPs de cada entrada para binário e verificar qual o maior prefixo de bits comuns.

No caso das entradas tanto para a alínea a) e para a alínea b) vemos que os primeiros 27 bits são comuns, logo a máscara de rede será /27.

```
192.168.0.192 -> 11000000.10101000.00000000.11000000
192.168.0.200 -> 11000000.10101000.00000000.11001000
192.168.0.208 -> 11000000.10101000.00000000.11010000
192.168.0.216 -> 11000000.10101000.00000000.11011000
```

De seguida devemos verificar qual o IP dessa *supernetting* que receberá a máscara /27, novamente analisando os valores em binário. Como os 3 primeiros octetos são iguais e só os 3 primeiros bits do último octeto são comuns, então os últimos 5 bits são reduzidos a 0s, logo o IP que agrupa estas entradas é 192.168.0.192/27 (Alínea a) e 192.168.0.224/27 (alínea b)). De seguida, escolhemos a *Gateway* pela qual chegarão a essas redes que é a *Gateway* original de cada uma das entradas (10.0.0.1 para a alínea a) e 10.0.0.6 para alínea b)) logo através dos comandos – “*ip route add 192.168.0.192/27 via 10.0.0.1*” e “*ip route add 192.168.0.224/27 via 10.0.0.6*” adicionamos as entradas para as *supernets* da alínea a) e b) respetivamente. (Figura 66)

```

10.0.0.16      10.0.0.6      255.255.255.252 UG      0 0      0 eth1
10.0.0.20      10.0.0.6      255.255.255.252 UG      0 0      0 eth1
10.0.0.24      10.0.0.6      255.255.255.252 UG      0 0      0 eth1
10.0.0.28      10.0.0.6      255.255.255.252 UG      0 0      0 eth1
172.0.0.0      10.0.0.6      255.0.0.0      UG      0 0      0 eth1
172.16.142.0   10.0.0.1      255.255.255.252 UG      0 0      0 eth0
172.16.142.4   10.0.0.1      255.255.255.252 UG      0 0      0 eth0
172.16.143.0   10.0.0.6      255.255.255.252 UG      0 0      0 eth1
172.16.143.4   10.0.0.6      255.255.255.252 UG      0 0      0 eth1
192.168.0.192  10.0.0.1      255.255.255.248 UG      0 0      0 eth0
192.168.0.200  10.0.0.1      255.255.255.248 UG      0 0      0 eth0
192.168.0.208  10.0.0.1      255.255.255.248 UG      0 0      0 eth0
192.168.0.216  10.0.0.1      255.255.255.248 UG      0 0      0 eth0
192.168.0.224  10.0.0.6      255.255.255.248 UG      0 0      0 eth1
192.168.0.232  10.0.0.6      255.255.255.248 UG      0 0      0 eth1
192.168.0.240  10.0.0.6      255.255.255.248 UG      0 0      0 eth1
192.168.0.248  10.0.0.6      255.255.255.248 UG      0 0      0 eth1
root@n6:/tmp/pycore.37667/n6.conf# ip route del 192.168.0.192/29 via 10.0.0.1
root@n6:/tmp/pycore.37667/n6.conf# ip route del 192.168.0.200/29 via 10.0.0.1
root@n6:/tmp/pycore.37667/n6.conf# ip route del 192.168.0.208/29 via 10.0.0.1
root@n6:/tmp/pycore.37667/n6.conf# ip route del 192.168.0.216/29 via 10.0.0.1
root@n6:/tmp/pycore.37667/n6.conf# ip route del 192.168.0.224/29 via 10.0.0.6
root@n6:/tmp/pycore.37667/n6.conf# ip route del 192.168.0.232/29 via 10.0.0.6
root@n6:/tmp/pycore.37667/n6.conf# ip route del 192.168.0.240/29 via 10.0.0.6
root@n6:/tmp/pycore.37667/n6.conf# ip route del 192.168.0.248/29 via 10.0.0.6
root@n6:/tmp/pycore.37667/n6.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask         Flags    MSS Window  irtt Iface
10.0.0.0         0.0.0.0         255.255.255.252 U        0 0         0 eth0
10.0.0.4         0.0.0.0         255.255.255.252 U        0 0         0 eth1
10.0.0.8         10.0.0.6        255.255.255.252 UG       0 0         0 eth1
10.0.0.12        10.0.0.6        255.255.255.252 UG       0 0         0 eth1
10.0.0.16        10.0.0.6        255.255.255.252 UG       0 0         0 eth1
10.0.0.20        10.0.0.6        255.255.255.252 UG       0 0         0 eth1
10.0.0.24        10.0.0.6        255.255.255.252 UG       0 0         0 eth1
10.0.0.28        10.0.0.6        255.255.255.252 UG       0 0         0 eth1
172.0.0.0        10.0.0.6        255.0.0.0      UG       0 0         0 eth1
172.16.142.0     10.0.0.1        255.255.255.252 UG       0 0         0 eth0
172.16.142.4     10.0.0.1        255.255.255.252 UG       0 0         0 eth0
172.16.143.0     10.0.0.6        255.255.255.252 UG       0 0         0 eth1
172.16.143.4     10.0.0.6        255.255.255.252 UG       0 0         0 eth1
root@n6:/tmp/pycore.37667/n6.conf# ip route add 192.168.0.110/27 via 10.0.0.1
Error: Invalid prefix for given prefix length.
root@n6:/tmp/pycore.37667/n6.conf# ip route add 192.168.0.192/27 via 10.0.0.1
root@n6:/tmp/pycore.37667/n6.conf# ip route add 192.168.0.224/27 via 10.0.0.6
root@n6:/tmp/pycore.37667/n6.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask         Flags    MSS Window  irtt Iface
10.0.0.0         0.0.0.0         255.255.255.252 U        0 0         0 eth0
10.0.0.4         0.0.0.0         255.255.255.252 U        0 0         0 eth1
10.0.0.8         10.0.0.6        255.255.255.252 UG       0 0         0 eth1
10.0.0.12        10.0.0.6        255.255.255.252 UG       0 0         0 eth1
10.0.0.16        10.0.0.6        255.255.255.252 UG       0 0         0 eth1
10.0.0.20        10.0.0.6        255.255.255.252 UG       0 0         0 eth1
10.0.0.24        10.0.0.6        255.255.255.252 UG       0 0         0 eth1
10.0.0.28        10.0.0.6        255.255.255.252 UG       0 0         0 eth1
172.0.0.0        10.0.0.6        255.0.0.0      UG       0 0         0 eth1
172.16.142.0     10.0.0.1        255.255.255.252 UG       0 0         0 eth0
172.16.142.4     10.0.0.1        255.255.255.252 UG       0 0         0 eth0
172.16.143.0     10.0.0.6        255.255.255.252 UG       0 0         0 eth1
172.16.143.4     10.0.0.6        255.255.255.252 UG       0 0         0 eth1
192.168.0.192    10.0.0.1        255.255.255.224 UG       0 0         0 eth0
192.168.0.224    10.0.0.6        255.255.255.224 UG       0 0         0 eth1
root@n6:/tmp/pycore.37667/n6.conf#

```

Figura 69- Rede supernetting

2.3.2 Alínea 3)

Pergunta: Comente os aspetos positivos e negativos do uso do Supernetting.

O *Supernetting* é uma técnica usada para reduzir a quantidade de endereços IP necessários numa rede, permitindo que várias sub-redes sejam combinadas numa única rede maior. Embora o Supernetting possa trazer benefícios no que diz respeito à eficiência e gestão de recursos, também apresenta desvantagens significativas.

No que diz respeito a aspetos positivos do uso do *Supernetting* destacámos 3:

- Economia de endereços IP: o Supernetting permite que várias sub-redes sejam combinadas numa única rede maior, o que pode economizar endereços IP, pois são necessários menos endereços IP para endereçar uma rede maior.

- Melhor eficiência: ao usar *Supernetting*, é ainda possível reduzir a quantidade de informações de roteamento na tabela de roteamento, o que pode melhorar a eficiência da rede e diminuir o tempo de resposta.
- Facilidade de gerenciamento: o *Supernetting* permite que várias sub-redes sejam gerenciadas como uma única rede, o que pode simplificar a administração da rede.

No que diz respeito a aspetos negativos do uso do *Supernetting*:

- Falta de flexibilidade: ao usar *Supernetting*, as sub-redes não podem ser facilmente separadas ou divididas em sub-redes menores, o que pode limitar a flexibilidade da rede.
- Complexidade do roteamento: o *Supernetting* pode aumentar a complexidade do roteamento, especialmente se as sub-redes incluídas no supernet não estiverem fisicamente próximas umas das outras.
- Maior risco de falha: ao usar *Supernetting*, a falha de um único roteador pode afetar toda a rede supernetada, o que aumenta o risco de falha da rede como um todo.

Em suma, o *Supernetting* pode ser uma técnica útil para economizar endereços IP e simplificar a administração da rede. No entanto, é também importante ter em consideração os possíveis riscos e desvantagens associados ao *Supernetting* antes de o implementar numa rede.

3 - Conclusões

Realizar este trabalho possibilitou um aprofundamento considerável dos conhecimentos adquiridos nas aulas teóricas. Foi-nos possível colocar em prática o funcionamento do comando traceroute, realizar análises de pacotes IP e compreender o processo de fragmentação dos mesmos. Além disso, aprendeu-se a identificar os procedimentos necessários para verificar o funcionamento adequado de uma rede local, consultar tabelas de encaminhamento, as implicações de remover rotas (como a rota *default*) e, ainda, como corrigir possíveis erros no encaminhamento.