

Redes de Computadores

Trabalho Prático 2

Rodrigo Monteiro, Diogo Abreu, e Gustavo Barros

Universidade do Minho, Departamento de Informática, 4710-057 Braga, Portugal

e-mail: {a100706,a100646,a100656}@alunos.uminho.pt

Abstract

A realização deste trabalho prático tem como objetivo aprofundar os nossos conhecimentos no âmbito de Internet Protocol. Assim, estudamos as principais vertentes do IP:

1. estudo do formato de um pacote ou datagrama IP;
2. fragmentação de pacotes IP;
3. endereçamento IP;
4. encaminhamento IP.

Parte 1

Na primeira parte do trabalho, é realizado o registo de datagramas enviados e recebidos através da utilização do Wireshark, de uma topologia CORE e do traceroute.

Exercício 1.

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

icmp					
No.	Time	Source	Destination	Protocol	Length Info
9	12.650693507	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=1/256, ttl=1 (no response..)
10	12.650714873	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
11	12.650721226	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=2/512, ttl=1 (no response..)
12	12.650723511	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
13	12.650725395	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=3/768, ttl=1 (no response..)
14	12.650727259	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
15	12.650729364	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=4/1024, ttl=2 (no respons..)
16	12.650736669	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=5/1280, ttl=2 (no respons..)
17	12.650739496	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=6/1536, ttl=2 (no respons..)
18	12.650742081	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=7/1792, ttl=3 (no respons..)
19	12.650744426	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=8/2048, ttl=3 (no respons..)
20	12.650750870	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=9/2304, ttl=3 (no respons..)
21	12.650753385	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=10/2560, ttl=4 (reply in ..)
22	12.650756381	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=11/2816, ttl=4 (reply in ..)
23	12.650758626	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=12/3072, ttl=4 (reply in ..)
24	12.650761232	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=13/3328, ttl=5 (reply in ..)
25	12.650763767	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=14/3584, ttl=5 (reply in ..)
26	12.650765862	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=15/3840, ttl=5 (reply in ..)
27	12.650768477	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=16/4096, ttl=6 (reply in ..)
28	12.651084794	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=17/4352, ttl=6 (reply in ..)
29	12.651091378	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=18/4608, ttl=6 (reply in ..)
30	12.651094134	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=19/4864, ttl=7 (reply in ..)
31	12.680904051	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
32	12.680976480	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
33	12.680978162	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
34	12.681488422	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=20/5120, ttl=7 (reply in ..)
35	12.681583324	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=21/5376, ttl=7 (reply in ..)
36	12.681511211	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=8x083e, seq=22/5632, ttl=8 (reply in ..)
37	12.711108515	10.0.2.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
38	12.711114067	10.0.2.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
39	12.711115229	10.0.2.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
40	12.711176621	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=10/2560, ttl=61 (request ..)
41	12.711178964	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=11/2816, ttl=61 (request ..)
42	12.711179156	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=12/3072, ttl=61 (request ..)
43	12.711180228	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=13/3328, ttl=61 (request ..)
44	12.711181250	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=14/3584, ttl=61 (request ..)
45	12.711182273	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=15/3840, ttl=61 (request ..)
46	12.711183285	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=16/4096, ttl=61 (request ..)
47	12.711308445	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=17/4352, ttl=61 (request ..)
48	12.711309279	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=18/4608, ttl=61 (request ..)
49	12.711309221	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=19/4864, ttl=61 (request ..)
50	12.741751926	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=20/5120, ttl=61 (request ..)
51	12.741759051	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=21/5376, ttl=61 (request ..)
52	12.741759452	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=8x083e, seq=22/5632, ttl=61 (request ..)

4	10.0.5.10 (10.0.5.10)	60.247 ms	60.246 ms	60.245 ms
---	-----------------------	-----------	-----------	-----------

```
root@Lost:/tmp/pycore.36601/Lost.conf#
```

- a) Active o Wireshark no host Lost. Numa shell de Lost execute o comando `traceroute -I` para o endereço IP do Found. Registe e analise o tráfego ICMP enviado pelo sistema Lost e o tráfego ICMP recebido como resposta. Explique os resultados obtidos tendo em conta o princípio de funcionamento do traceroute.

Ao executar o comando, podemos observar que o "Lost" envia pacotes ICMP Echo Request com diferentes valores de TTL. Quando um pacote chega a um router, o TTL é decrementado em 1 e o router envia um pacote ICMP Time Exceeded de volta para o "Lost" caso chegue a 0. No Wireshark, podemos analisar o tráfego ICMP enviado pelo "Lost" e o tráfego ICMP recebido como resposta para determinar cada salto na rota. Para cada pacote ICMP Echo Request, podemos ver a fonte, o destino, o TTL e o tempo de resposta para cada salto. Assim, com pacotes com TTL igual a 1, não é possível alcançar o "Found", e por isso são recebidos os três pacotes "Time-to-live exceeded" (no. 10, 12, e 14). O mesmo acontece para os três pacotes com TTL igual a 2 (no. 31, 32 e 33) e para os três pacotes com TTL igual a 3 (no. 37, 38 e 39).

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

O valor inicial de TTL para alcançar é 4 visto que é número de "saltos" necessários para alcançar o "Found" na topologia utilizada. Por exemplo, fazendo um ping com parâmetro 3, não é possível alcançar o "Found", enquanto que com parâmetro 4 já é possível.

```

[Terminal - core@xubunc... Terminal - core@xubunc... 15 mar, 17:47]
vcmid
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 60,256/61,090/63,502/1,392 ms
root@Lost:/tmp/pycore.36601/Lost.conf# ping -t 3 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data:
From 10.0.2.2 icmp_seq=1 Time to live exceeded
From 10.0.2.2 icmp_seq=2 Time to live exceeded
From 10.0.2.2 icmp_seq=3 Time to live exceeded
From 10.0.2.2 icmp_seq=4 Time to live exceeded
From 10.0.2.2 icmp_seq=5 Time to live exceeded
^C
--- 10.0.5.10 ping statistics ---
5 packets transmitted, 0 received, +5 errors, 100% packet loss, time 4005ms

root@Lost:/tmp/pycore.36601/Lost.conf# ping -t 3 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data:
From 10.0.2.2 icmp_seq=1 Time to live exceeded
From 10.0.2.2 icmp_seq=2 Time to live exceeded
From 10.0.2.2 icmp_seq=3 Time to live exceeded
From 10.0.2.2 icmp_seq=4 Time to live exceeded
^C
--- 10.0.5.10 ping statistics ---
5 packets transmitted, 0 received, +4 errors, 100% packet loss, time 4004ms

root@Lost:/tmp/pycore.36601/Lost.conf# S

```

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

```

root@Lost:/tmp/pycore.36601/Lost.conf# traceroute -I -q 6 10.0.5.10
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.051 ms 0.010 ms 0.016 ms 0.006 ms 0.006 ms 0.007 ms
 2 10.0.1.2 (10.0.1.2) 30.182 ms 30.171 ms 30.166 ms 30.158 ms 30.152 ms 30.148 ms
 3 10.0.2.2 (10.0.2.2) 60.257 ms 60.252 ms 60.248 ms 60.243 ms 60.228 ms 60.213 ms
 4 10.0.5.10 (10.0.5.10) 60.196 ms 60.191 ms 60.186 ms 60.182 ms 60.280 ms 60.259 ms
root@Lost:/tmp/pycore.36601/Lost.conf# 

```

Cálculo do RTT médio:

1. $\frac{0.051+0.010+0.016+0.006+0.006+0.007}{6} = 0.016 \text{ ms (1 salto)}$
2. $\frac{30.182+30.171+30.166+30.158+30.152+30.148}{6} = 30.1628(3) \text{ ms (2 saltos)}$
3. $\frac{60.257+60.252+60.248+60.243+60.228+60.213}{6} = 60.2401(6) \text{ ms (3 saltos)}$
4. $\frac{60.196+60.191+60.186+60.182+60.280+60.259}{6} = 60.215(6) \text{ ms (4 saltos : tempo médio de ida e volta ao host)}$

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

Dividir o RTT por dois não dá o atraso médio num único sentido com precisão, visto que o RTT inclui o tempo que leva para um pacote viajar do "Lost" para o "Found" e também para voltar. Numa rede real não se pode assumir que o tempo de ida é igual ao tempo de volta devido a vários fatores como congestionamento de rede, mudanças de rota e tempos de processamento dos dispositivos (por exemplo, routers).

Exercício 2.

Pretende-se agora usar o traceroute na sua máquina nativa e gerar datagramas IP de diferentes tamanhos. Documente e justifique todas as respostas às seguintes alíneas:

2

4

5

*wlan0

</

O datagrama IP não foi fragmentado, visto que o valor de `more fragments` bit nas flags é 0, tal como o `fragments offset`.

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

13	1.852178972	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=1/256, ttl=1 (no response found!)
14	1.852212646	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=2/512, ttl=1 (no response found!)
15	1.852222054	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=3/768, ttl=1 (no response found!)
16	1.852231151	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=4/1024, ttl=2 (no response found!)
17	1.852238886	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=5/1280, ttl=2 (no response found!)
18	1.852246831	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=6/1536, ttl=2 (no response found!)
19	1.852256459	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=7/1792, ttl=3 (no response found!)
20	1.852263883	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=8/2048, ttl=3 (no response found!)
21	1.852271377	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=9/2304, ttl=3 (no response found!)
22	1.852279733	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=10/2560, ttl=4 (reply in 32)
23	1.852287388	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=11/2816, ttl=4 (no response found!)
24	1.852295683	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=12/3072, ttl=4 (no response found!)
25	1.852304360	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=13/3328, ttl=5 (no response found!)
26	1.852311774	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=14/3584, ttl=5 (no response found!)
27	1.852319168	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=15/3840, ttl=5 (no response found!)
28	1.852327363	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=16/4096, ttl=6 (no response found!)
51	1.875033868	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=22/5632, ttl=8 (reply in 53)
65	1.894692960	172.26.53.255	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0002, seq=25/6400, ttl=9 (reply in 67)
32	1.854978095	193.136.9.240	172.26.53.255	ICMP	74 Echo (ping) reply	id=0x0002, seq=10/2560, ttl=61 (request in 22)
53	1.876818980	193.136.9.240	172.26.53.255	ICMP	74 Echo (ping) reply	id=0x0002, seq=22/5632, ttl=61 (request in 51)
67	1.896703067	193.136.9.240	172.26.53.255	ICMP	74 Echo (ping) reply	id=0x0002, seq=25/6400, ttl=61 (request in 65)

As diferenças são o `seq` (número na sequência de pacotes enviados), o `TTL`, Time To Live - número máximo de saltos possíveis) e caso uma resposta foi encontrada ou não (`no response found` ou `reply in xyz`). A cada três pacotes enviados, os próximos têm um `TTL` incrementado em 1.

- f) Observa algum padrão nos valores do campo de Identificação do datagrama IP e `TTL`?

O campo de identificação aumenta a cada pacote que é enviado, e o `TTL` aumenta a cada três pacotes, ambos em 1 valor.

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

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

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

O valor do campo `TTL` não permanece constante, pois varia entre: 253 3, 254 3, 255 3. As mensagens ICMP `TTL Exceeded` são enviadas na origem com um valor alto, de modo a que consiga chegar ao destino, neste caso à interface 192.168.1.1, independentemente da rota.

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

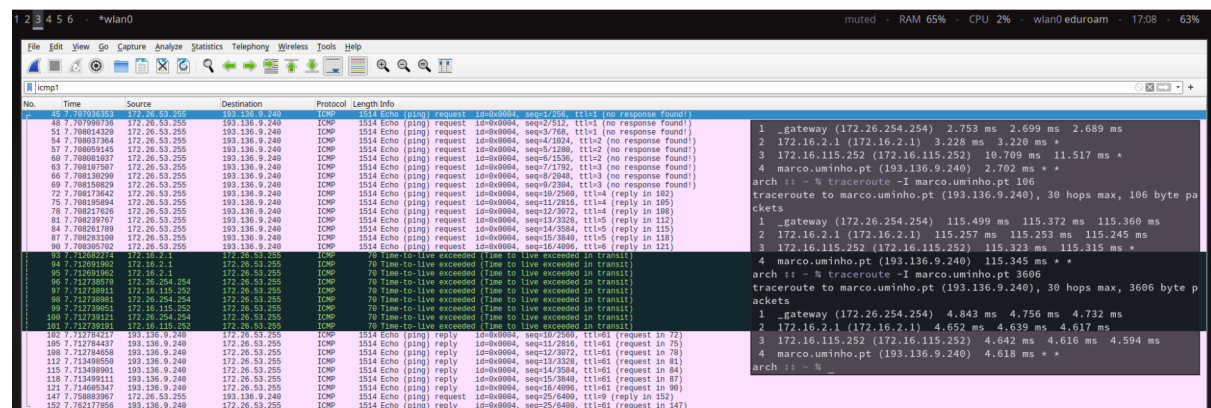
Poderia ser possível incluir o cabeçalho ICMP no cabeçalho IPv4, as vantagens seriam a diminuição de overhead e facilitaria o roteamento do pacote, já que a informação de erro ou controle seria imediatamente identificada no cabeçalho. No entanto, teria desvantagens: aumentaria a complexidade do cabeçalho, dificultando a interpretação do cabeçalho pelos dispositivos, ficaria mais difícil de processar; dificultar a implementação de futuras atualizações nos protocolos.

Exercício 3.

Pretende-se agora analisar a fragmentação de pacotes IP. Documente e justifique todas as respostas às seguintes alíneas:

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

Houve necessidade de fragmentar o pacote, visto que 3606 excede o MTU (Maximum Transmission Unit) da rede “eduroam”.



No.	Time	Source	Destination	Protocol	Length	Info
48	7.707990736	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=27512, ttl=1 (no response found)
51	7.708014238	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=37160, ttl=1 (no response found)
54	7.708037364	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=47162, ttl=1 (no response found)
57	7.708059145	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=57160, ttl=1 (no response found)
60	7.708081037	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=67158, ttl=1 (no response found)
63	7.708107507	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=77192, ttl=1 (no response found)
66	7.708130298	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=87204, ttl=1 (no response found)
69	7.708150829	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=97394, ttl=1 (no response found)
72	7.708173642	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=107250, ttl=1 (reply in 105)
75	7.708195084	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=117192, ttl=1 (reply in 112)
78	7.708217626	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=127072, ttl=1 (reply in 115)
81	7.708239767	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=137328, ttl=1 (reply in 118)
84	7.708261709	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=147354, ttl=1 (reply in 121)
87	7.708283198	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=157384, ttl=1 (reply in 121)
90	7.708305702	172.16.2.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0x0004, seq=167406, ttl=1 (reply in 121)
93	7.712019822	172.16.2.1	193.136.9.240	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
94	7.712019822	172.16.2.1	193.136.9.240	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
95	7.712019822	172.16.2.1	193.136.9.240	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
96	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
97	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
98	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
99	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
100	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
101	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
102	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
103	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
104	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
105	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
106	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
107	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
108	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
109	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
110	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
111	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
112	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
113	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
114	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
115	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
116	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
117	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
118	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
119	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
120	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
121	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
122	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
123	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
124	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
125	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
126	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
127	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
128	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
129	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
130	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
131	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
132	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
133	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
134	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
135	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
136	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
137	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
138	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
139	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
140	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
141	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
142	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
143	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
144	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
145	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
146	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
147	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
148	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
149	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
150	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
151	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
152	7.712738570	172.16.254.254	172.16.2.1	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)

- b)** Imprima o primeiro fragmento do datagrama IP original. Que informação no cabeçalho indica que o datagrama foi fragmentado? Que informação no cabeçalho IP indica que se trata do primeiro fragmento? Qual é o tamanho deste datagrama IP?

Nota: Os fragmentos IP não fazem match com o filtro "icmp", portanto tem de se clicar num dos fragmentos que aparecem, clicar em *conversation filter* e depois em IPv4.

ip.addr eq 172.26.53.255 and ip.addr eq 193.136.9.254					
No.	Time	Source	Destination	Protocol	Length Info
11	1.554526095	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=1/256, ttl=1 (no response found!)
12	1.554545632	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca52)
13	1.554549970	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca52)
14	1.554563545	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=2/512, ttl=1 (no response found!)
15	1.554567843	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca53)
16	1.554572871	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca53)
17	1.554583833	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=3/768, ttl=1 (no response found!)
18	1.554588492	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca54)
19	1.554592479	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca54)
20	1.554603860	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=4/1024, ttl=2 (no response found!)
21	1.5546097928	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca55)
22	1.554611715	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca55)
23	1.554623187	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=5/1280, ttl=2 (no response found!)
24	1.554627455	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca56)
25	1.554631831	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca56)
26	1.554641721	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=6/1536, ttl=2 (no response found!)
27	1.554645629	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca57)
28	1.554649105	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca57)
29	1.554660015	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=7/1792, ttl=3 (reply in 74)
30	1.554663863	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca58)
31	1.554667249	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca58)
32	1.554677168	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=8/2048, ttl=3 (reply in 79)
33	1.554681245	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca59)
34	1.554684782	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca59)
35	1.554694891	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=9/2304, ttl=3 (reply in 86)
36	1.554697996	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca5a)
37	1.554708571	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca5a)
38	1.554707594	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=10/2560, ttl=4 (reply in 92)
39	1.554709708	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca5b)
40	1.554712003	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca5b)
41	1.554716311	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=11/2816, ttl=4 (reply in 100)
42	1.554719034	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca5c)
43	1.554719677	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca5c)
44	1.554730551	172.26.53.255	193.136.9.254	ICMP	1514 Echo (ping) request id=0x0002, seq=12/3072, ttl=4 (reply in 103)
45	1.554733857	172.26.53.255	193.136.9.254	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ca5d)
46	1.554785420	172.26.53.255	193.136.9.254	IPv4	660 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ca5d)

```

Total Length: 1500
Identification: 0xca52 (51794)
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: 1
Protocol: ICMP (1)
Header Checksum: 0x1c2f [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.26.53.255
Destination Address: 193.136.9.254

```

A flag "More fragments" está a 1, o que indica que o datagrama foi fragmentado. Este trata-se do primeiro fragmento, visto que o fragment offset (campo de 13 bits) é igual a 0. O tamanho do datagrama IP é 1500 bytes.

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

Este trata-se do segundo fragmento, visto que o fragment offset (campo de 13 bits) é igual a 1480, que é o primeiro diferente de 0. Existem mais fragmentos porque a flag de MoreFragments está a 1

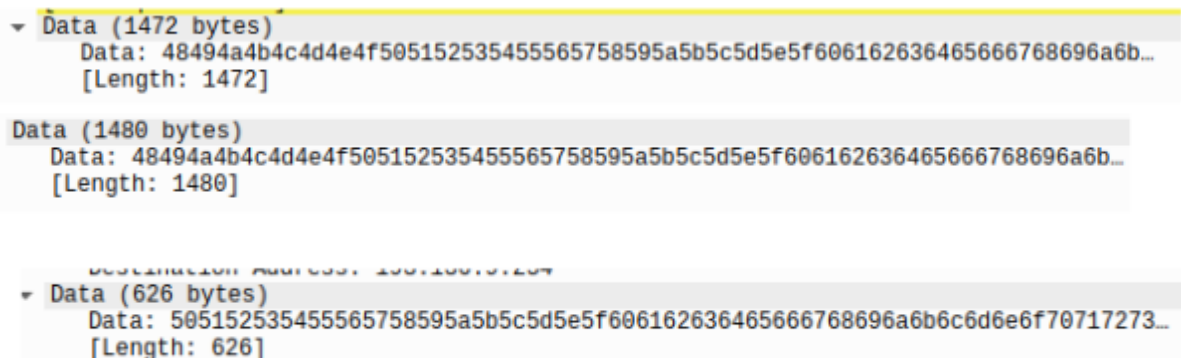
```

0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentially Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1500
Identification: 0xf578 (62840)
Flags: 0x20b9, More fragments
  0... .... = Reserved bit: Not set
  .0... .... = Don't fragment: Not set
  ..1. .... = More fragments: Set
Fragment offset: 1480
Time to live: 1
Protocol: ICMP (1)

```

- d) *Estime teoricamente o número de fragmentos gerados a partir do datagrama IP original e o número de bytes transportados no último fragmento desse datagrama. Compare os dois valores estimados com os obtidos através do wireshark.*

Como o valor do MTU (maximum transmission unit) é 1500 bytes, são gerados 3 fragmentos, dois com 1470 ~ 1500 bytes e o último com os restantes bytes (600 ~ 660. No wireshark foi obtido um valor de 1472 bytes de data length, sendo os restantes bytes utilizados para headers. O segundo fragmento tem um valor maior, 1480 bytes, uma vez que não possui o ICMP header como o primeiro. O último fragmento possui os restantes bytes (626) - de acordo com o total size indicado no comando traceroute (3606).



; 1500+1500+646
3646

Quanto ao tamanho total: são 1500 bytes para o primeiro pacote e para o segundo, e 646 para o último ⇒ Maior do que 3606 devido a alinhamento, padding, limites específicos/quantizados de bytes e overhead (devido aos headers).

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

A flag "More fragments" aparece com valor 0.

▼ 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

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

Os fragmentos são enviados através da rede e, eventualmente, recebidos pelo destino final (neste caso, google.com), onde são reconstituídos para formar o pacote IP original. A reconstrução do pacote ocorre no destino final e não em nenhum dos routers intermediários, pois apenas o destino final possui todas as informações necessárias para recriar o pacote original a partir dos fragmentos recebidos - o destino final tem acesso a todos os fragmentos do pacote original e, portanto, pode recriar o pacote original com base nas informações de cabeçalho contidas em cada fragmento; utiliza a identificação do pacote original e o valor do deslocamento para reordenar os fragmentos na ordem correta e recriar o pacote original com os dados corretos.

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

Entre os diferentes fragmentos, os campos que mudam no cabeçalho IP são a flag "more fragments", a "total length", e claro o "header checksum". O identificador permanece igual, e assim, o equipamento que recebe o pacote utiliza os valores de offset para colocar os fragmentos na ordem correta e o bit "more fragments" para saber qual é o último fragmento. Para além disso, o checksum de cada fragmento é recalculado e comparado ao valor original para verificar a integridade do pacote / se houve erros na transmissão.

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

Apenas o primeiro fragmento precisa de um ICMP header, os restantes fragmentos incluem apenas informação sobre fonte, destino, fragment offset, identificação, etc. Portanto, reconstrução do pacote, este será identificado como um ICMP devido ao primeiro fragmento, os outros fragmentos não precisam de transportar mais essas informações, visto que poderia ser redundante e desnecessário.

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

O valor do tamanho do datagrama é comparado com o MTU (maximum transmission unit) - neste caso é 1500 bytes. Se o tamanho exceder esse MTU, o pacote é fragmentado em fragmentos menores antes de ser transmitido na rede. Caso o MTU seja aumentado, pode haver um aumento de eficiência na transmissão de dados, visto que implica menos sobrecarga de cabeçalhos e overhead, e portanto menos processamento nos routers. No entanto, se o MTU for muito grande, há a possibilidade de ocorrer mais erros de integridade, taxas mais altas de erros, pacotes danificados e perdidos, principalmente em redes com muita interferência, o que pode levar a uma diminuição no desempenho da rede. E vice versa caso o MTU seja diminuído.

- j) Sabendo que no comando ping a opção *-f* (Windows), *-M* do (Linux) ou *-D* (Mac) ativa a flag “Don’t Fragment” (DF) no cabeçalho do IPv4, determine o valor máximo de SIZE sem que ocorra fragmentação do pacote? Justifique o valor obtido.

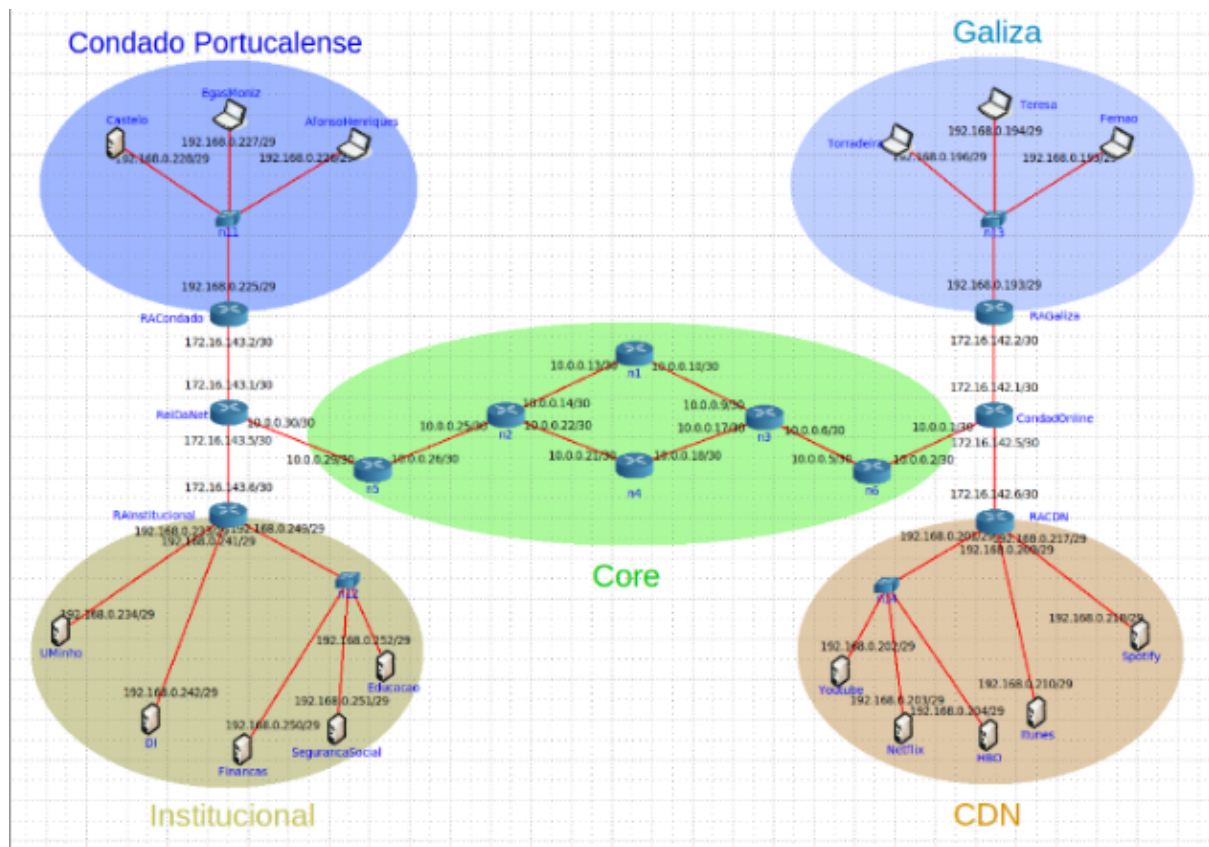
```
arch :: ~ % ping -s 2000 -M do marco.uminho.pt
PING marco.uminho.pt (193.136.9.240) 2000(2028) bytes of data.
ping: local error: message too long, mtu=1500
ping: local error: message too long, mtu=1500
ping: local error: message too long, mtu=1500
^C
--- marco.uminho.pt ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2015ms
```

O valor máximo é 1472, visto que os restantes bits são necessários para os headers IP e ICMP. (sendo que o header ICMP existe apenas no primeiro fragmento).

```
arch :: ~ % ping -s 1473 -M do marco.uminho.pt
PING marco.uminho.pt (193.136.9.240) 1473(1501) bytes of data.
ping: local error: message too long, mtu=1500
^C
--- marco.uminho.pt ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

arch :: ~ % ping -s 1472 -M do marco.uminho.pt
PING marco.uminho.pt (193.136.9.240) 1472(1500) bytes of data.
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=1 ttl=61 time=3.09 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=2 ttl=61 time=5.80 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=3 ttl=61 time=4.91 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=4 ttl=61 time=6.58 ms
^C
--- marco.uminho.pt ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 3.094/5.094/6.582/1.298 ms
```

PARTE 2



Neste trabalho continua-se o estudo do protocolo IPv4 com ênfase no endereçamento e encaminhamento IP.

Exercício 1.

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

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

```
<ycore.36847/AfonsoHenriques.conf# ping -c 5 192.168.0.250
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=0.106 ms
64 bytes from 192.168.0.250: icmp_seq=2 ttl=61 time=0.112 ms
64 bytes from 192.168.0.250: icmp_seq=3 ttl=61 time=0.068 ms
64 bytes from 192.168.0.250: icmp_seq=4 ttl=61 time=0.058 ms
64 bytes from 192.168.0.250: icmp_seq=5 ttl=61 time=0.125 ms

--- 192.168.0.250 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4096ms
rtt min/avg/max/mdev = 0.058/0.093/0.125/0.026 ms
root@AfonsoHenriques:/tmp/pycore.36847/AfonsoHenriques.conf#
```

Esta primeira alínea tinha como objetivo analisar se D.Afonso Henriques tinha conectividade com D.Teresa. Através do comando ping foram transmitidos 5 pacotes. Pela imagem, observamos que os pacotes chegaram ao destinatário com TTL de valor 61 e em curto espaço de tempo. Logo a afirmação na questão foi confirmada.

A mesma coisa aconteceu para a conectividade com os servidores da CDN com TTL de 55.

```
<core.36847/AfonsoHenriques.conf# ping -c 5 192.168.0.203
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=0.177 ms
64 bytes from 192.168.0.203: icmp_seq=2 ttl=55 time=0.108 ms
64 bytes from 192.168.0.203: icmp_seq=3 ttl=55 time=0.106 ms
64 bytes from 192.168.0.203: icmp_seq=4 ttl=55 time=0.131 ms
64 bytes from 192.168.0.203: icmp_seq=5 ttl=55 time=0.116 ms

--- 192.168.0.203 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4077ms
rtt min/avg/max/mdev = 0.106/0.127/0.177/0.026 ms
root@AfonsoHenriques:/tmp/pycore.36847/AfonsoHenriques.conf#
```

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

```
root@AfonsoHenriques:/tmp/pycore.36847/AfonsoHenriques.conf# netstat -r
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
default 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.36847/AfonsoHenriques.conf#
```

Tendo como destino 192.168.0.224, ou seja, o router RACondado, o gateway é 0.0.0.0 (default), ou seja, significa que não existem saltos intermediários necessários devido ao sistema estar diretamente ligado ao destino. O destino default define a rota para os pacotes cujo destino não correspondeu a nenhum dos destinos presentes na tabela de encaminhamento. O mesmo se verifica para a tabela de encaminhamento da D.Teresa como é possível ver na imagem abaixo.

```
root@Teresa:/tmp/pycore.36847/Teresa.conf# netstat -r
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
default 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.36847/Teresa.conf#
```

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

O host AfonsoHenriques não consegue estabelecer ligação com o host Teresa devido ao router n5 como é possível observar pela imagem seguinte e pela tabela de endereçamento

o mesmo em que o core n2 não recebe os pacotes ICMP por 10.0.0.25.

1	0.000000000	10.0.0.30	224.0.0.5	OSPF	82	Hello Packet			
2	0.184722916	192.168.0.226	192.168.0.194	ICMP	98	Echo (ping) request	id=0x0020, seq=18/4608, ttl=62		
3	1.208712395	192.168.0.226	192.168.0.194	ICMP	98	Echo (ping) request	id=0x0020, seq=19/4864, ttl=62		
4	1.546949294	10.0.0.29	224.0.0.5	OSPF	82	Hello Packet			
5	2.000517853	10.0.0.30	224.0.0.5	OSPF	82	Hello Packet			
6	2.232749503	192.168.0.226	192.168.0.194	ICMP	98	Echo (ping) request	id=0x0020, seq=20/5120, ttl=62		
7	3.256730827	192.168.0.226	192.168.0.194	ICMP	98	Echo (ping) request	id=0x0020, seq=21/5376, ttl=62		
8	3.256748808	10.0.0.29	192.168.0.226	ICMP	126	Destination unreachable (Network unreachable)			

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

Logo, é necessário adicionar o seguinte comando para direcionar tráfego para IPs 192.168.0.192.

```
/n5.conf# ip route add 192.168.0.192/29 via 10.0.0.25 dev eth1
```

De seguida, o problema aparece com o router n2 que possui uma entrada errada com uma ligação mais longa a D.Teresa. Logo, é necessário retirar essa rota incorreta com o seguinte comando

```
root@n2:/tmp/pycore.39153/n2.conf# ip route del 192.168.0.194/31 via 10.0.0.25
```

Os problemas de conexão continuam, visto que o router n1 envia os pacotes novamente para n2, o que provoca um loop até ao fim do TTL dos pacotes. É necessário então aplicar os dois comandos seguintes

```
<n1.conf# ip route del 192.168.0.192/29 via 10.0.0.14 dev eth1
root@n1:/tmp/pycore.45741/n1.conf#
<n1.conf# ip route add 192.168.0.192/29 via 10.0.0.9 dev eth0
```

Depois de efetuadas todas estas alterações, já é possível a conexão entre AfonsoHenriques e Teresa como é possível verificar na imagem abaixo


```

<46011/AfonsoHenriques.conf# ping -c 5 192.168.0.194
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=0.136 ms
64 bytes from 192.168.0.194: icmp_seq=2 ttl=55 time=0.208 ms
64 bytes from 192.168.0.194: icmp_seq=3 ttl=55 time=0.196 ms
64 bytes from 192.168.0.194: icmp_seq=4 ttl=55 time=0.181 ms
64 bytes from 192.168.0.194: icmp_seq=5 ttl=55 time=0.184 ms

--- 192.168.0.194 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4086ms
rtt min/avg/max/mdev = 0.136/0.181/0.208/0.024 ms
root@AfonsoHenriques:/tmp/pycore.46011/AfonsoHenriques.conf#

```

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

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

Os pacotes são, de facto, recebidos por D.Teresa. No entanto, a mensagem de volta não é enviada pois os dados não passam do router RAGaliza.

```

root@Teresa:/tmp/pycore.36317/Teresa.conf# traceroute 192.168.0.226
traceroute to 192.168.0.226 (192.168.0.226), 30 hops max, 60 byte packets
 1 192.168.0.193 (192.168.0.193) 0.066 ms !N 0.008 ms !N *
root@Teresa:/tmp/pycore.36317/Teresa.conf#

```

Isto deve-se ao facto de o router não ter registado caminho para os IPs 192.168.0.224/29. Utilizando então o seguinte comando, estabelecemos então a conexão, visível na imagem seguinte ao comando add

```

iza.conf# ip route add 192.168.0.224/29 via 172.16.142.1 dev eth0

```

```

<46011/AfonsoHenriques.conf# ping -c 5 192.168.0.194
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=0.136 ms
64 bytes from 192.168.0.194: icmp_seq=2 ttl=55 time=0.208 ms
64 bytes from 192.168.0.194: icmp_seq=3 ttl=55 time=0.196 ms
64 bytes from 192.168.0.194: icmp_seq=4 ttl=55 time=0.181 ms
64 bytes from 192.168.0.194: icmp_seq=5 ttl=55 time=0.184 ms

--- 192.168.0.194 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4086ms
rtt min/avg/max/mdev = 0.136/0.181/0.208/0.024 ms
root@AfonsoHenriques:/tmp/pycore.46011/AfonsoHenriques.conf#

```

ii) As rotas dos pacotes ICMP echo reply são as mesmas, mas em sentido inverso, que as rotas dos pacotes ICMP echo request enviados entre AfonsoHenriques e

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

As rotas são diferentes. No traceroute de Teresa para AfonsoHenriques, no router n3 o destino 192.168.224/29 tem como gateway 10.0.0.18, seguindo portanto pelo router n4.

No traceroute de AfonsoHenriques para Teresa, no router n2, o destino 192.168.192/29 tem como gateway 10.0.0.13 seguindo pelo router n1.

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

192.168.0.192	20.0.0.18	255.255.255.240	UG	0 0	0 eth1
---------------	-----------	-----------------	----	-----	--------

Existe uma correspondência (match) nesta entrada para pacotes enviados para o polo Galiza? E para CDN? Caso seja essa a entrada utilizada para o encaminhamento, permitirá o funcionamento esperado do dispositivo? Ofereça uma explicação pela qual essa entrada é ou não utilizada.

Sim, existem tal como é possível observar na imagem seguinte.

192.168.0.192	10.0.0.5	255.255.255.248	UG	0 0	0 eth2
192.168.0.192	10.0.0.18	255.255.255.240	UG	0 0	0 eth1
192.168.0.200	10.0.0.5	255.255.255.248	UG	0 0	0 eth2
192.168.0.208	10.0.0.5	255.255.255.248	UG	0 0	0 eth2
192.168.0.216	10.0.0.5	255.255.255.248	UG	0 0	0 eth2
192.168.0.224	10.0.0.18	255.255.255.248	UG	0 0	0 eth1
192.168.0.232	10.0.0.10	255.255.255.248	UG	0 0	0 eth0
192.168.0.240	10.0.0.10	255.255.255.248	UG	0 0	0 eth0
192.168.0.248	10.0.0.10	255.255.255.248	UG	0 0	0 eth0

192.168.0.192 tem duas entradas na tabela. Deste modo, os pacotes com esse destino podem continuar tanto por 10.0.0.5 como por 10.0.0.18. A rota usada depende do protocolo usado (OSPF, BGP).

A rota de envio para o polo galiza é a 192.168.0.192 com gateway 10.0.0.5, caso seja usada a 10.0.0.18 os pacotes vão “para trás”, isto é, para n4, e n4 mandará os pacotes novamente para n3, dando origem a um loop.

No entanto, a entrada escolhida é a 10.0.0.5. Essa situação verifica-se visto que é uma rota mais específica, abrange um menor intervalo de endereços (8 endereços devido à máscara de rede /29, em comparação aos 16 endereços do /28). (Em regra geral, quando há múltiplas entradas com o mesmo endereço de destino, o protocolo escolhe a entrada mais específica com base na máscara de sub-rede - garante que o tráfego seja encaminhado para a sub-rede correta e não seja entregue ao gateway errado ou perdido na rede).

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

Todos os endereços utilizados são privados, pois são todos partições de 192.168.0.0/24, 172.16.0.0/12 ou de 10.0.0.0/8, os endereços privados reservados.

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

Os switches não possuem endereços IP pois não funcionam ao nível da rede, mas sim ao nível da ligação de dados. Funciona como uma ponte e conecta todos os elementos da rede. Apenas opera sobre endereços físicos e um endereço IP é um endereço lógico.

Exercício 2.

Tendo feito as pazes com a mãe, D. Afonso Henriques vê-se com algum tempo livre e decide fazer remodelações no condado: a. Não estando satisfeito com a decoração do Castelo, opta por eliminar a sua rota default. Adicione as rotas necessárias para que o Castelo continue a ter acesso a cada um dos três polos. Mostre que a conectividade é restabelecida, assim como a tabela de encaminhamento resultante. Explícite ainda a utilidade de uma rota default.

Foi eliminada a rota default via o comando seguinte e adicionadas novas rotas visíveis na segunda e terceira imagens

```
./Castelo.conf# ip route del 0.0.0.0/0 via 192.168.0.225 dev eth0
1/Castelo.conf# ip route add 192.168.0.232/29 via 192.168.0.225 dev eth0

root@Castelo:/tmp/pycore.35741/Castelo.conf# ip route add 192.168.0.240/29 via 192.168.0.225 dev eth0
root@Castelo:/tmp/pycore.35741/Castelo.conf# ip route add 192.168.0.248/29 via 192.168.0.225 dev eth0
root@Castelo:/tmp/pycore.35741/Castelo.conf# ip route add 192.168.0.192/29 via 192.168.0.225 dev eth0
root@Castelo:/tmp/pycore.35741/Castelo.conf# ip route add 192.168.0.200/29 via 192.168.0.225 dev eth0
root@Castelo:/tmp/pycore.35741/Castelo.conf# ip route add 192.168.0.208/29 via 192.168.0.225 dev eth0
root@Castelo:/tmp/pycore.35741/Castelo.conf# ip route add 192.168.0.216/29 via 192.168.0.225 dev eth0
root@Castelo:/tmp/pycore.35741/Castelo.conf# netstat -r
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
```

O destino 0.0.0.0 (default) com gateway 192.168.0.225 é a rota usada quando não existe match para mais nenhum destino na tabela. Depois de removida é necessário adicionar os destinos acima, as diversas sub-redes dos polos, à tabela.

Como qualquer dos destinos adicionados tem como gateway 192.168.0.255, uma rota default com essa gateway seria mais útil de modo a reduzir o número de entradas da tabela.

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

Foi nos atribuído o endereço de rede IP é 172.16.106.128/26 tendo assim um total de 64 endereços disponíveis. Podemos distribuir esses endereços por 4 redes iguais. Como temos 4 redes precisamos de mais 2 dígitos para identificar cada subrede (utilizamos assim o /28). Temos assim $64/4 = 16$ endereços em cada subrede distribuindo então pelos seguintes:

172.16.106.128/28
172.16.106.144/28
172.16.106.160/28
172.16.106.176/28

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

Pelas imagens seguintes é possível verificar que o host tem conexão com o polo Institucional, o Condado Portucalense e o polo Galiza.

```
root@n33:/tmp/pycore.35741/n33.conf# ping -c 2 192.168.0.234
PING 192.168.0.234 (192.168.0.234) 56(84) bytes of data.
64 bytes from 192.168.0.234: icmp_seq=1 ttl=62 time=0.132 ms
64 bytes from 192.168.0.234: icmp_seq=2 ttl=62 time=0.088 ms

--- 192.168.0.234 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1017ms
rtt min/avg/max/mdev = 0.088/0.110/0.132/0.022 ms
root@n33:/tmp/pycore.35741/n33.conf#
```

```
root@n33:/tmp/pycore.35741/n33.conf# ping -c 2 192.168.0.226
PING 192.168.0.226 (192.168.0.226) 56(84) bytes of data.
64 bytes from 192.168.0.226: icmp_seq=1 ttl=62 time=0.071 ms
64 bytes from 192.168.0.226: icmp_seq=2 ttl=62 time=0.117 ms

--- 192.168.0.226 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 0.071/0.094/0.117/0.023 ms
root@n33:/tmp/pycore.35741/n33.conf#
```

```
root@n33:/tmp/pycore.35741/n33.conf# ping -c 2 192.168.0.218
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=56 time=0.164 ms
64 bytes from 192.168.0.218: icmp_seq=2 ttl=56 time=0.174 ms

--- 192.168.0.218 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1021ms
rtt min/avg/max/mdev = 0.164/0.169/0.174/0.005 ms
```

Porém, não tem conexão com o host Castelo. Isto deve-se ao facto de a rota default do castelo ter sido apagada e substituída por 192.168.0.0/24.

Como não é possível fazer match para 172.16.106.130/28 na tabela de encaminhamento do Castelo, o pacote é descartado e por isso respostas não voltam para o n33 quando é feito um ping ou traceroute.

Tal pode ser resolvido adicionando o seguinte endereço à tabela de encaminhamento do castelo, ou, por exemplo, voltando a adicionar a rota default.

```
<741/Castelo.conf# ip route add 172.16.106.128/26 via 192.168.0.225 dev eth0
```

Exercício 3.

Ao planejar 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.

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

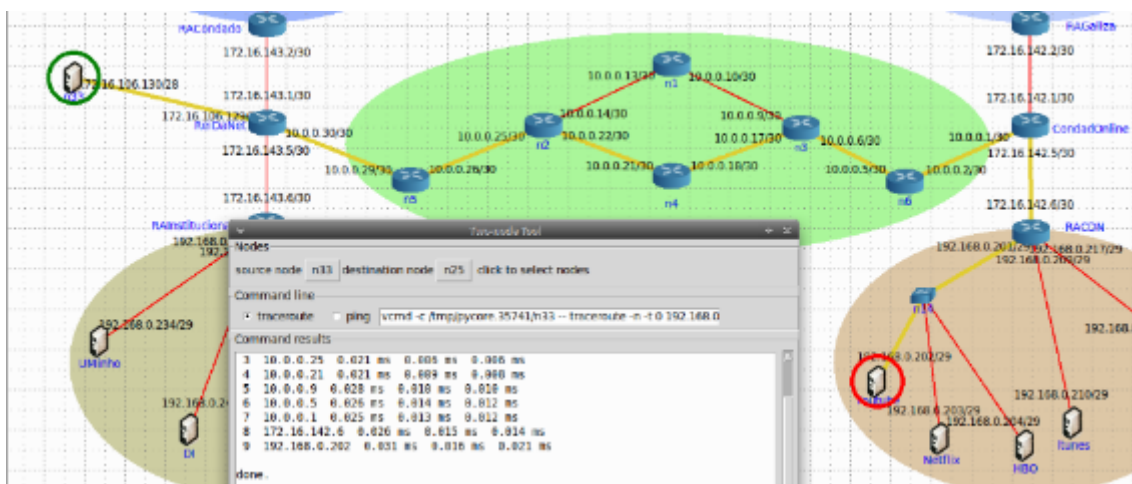
Foram eliminadas então as seguintes rotas sendo a primeira relativa a Galiza e as restantes a CDN.

```
<6.conf# ip route del 192.168.0.192/29 via 10.0.0.1 dev eth0
root@n6:/tmp/pycore.35741/n6.conf# ip route del 192.168.0.200/29 via 10.0.0.1 dev eth0
root@n6:/tmp/pycore.35741/n6.conf# ip route del 192.168.0.208/29 via 10.0.0.1 dev eth0
root@n6:/tmp/pycore.35741/n6.conf# ip route del 192.168.0.216/29 via 10.0.0.1 dev eth0
```

Entre 192 e 224 temos 32 bits, logo 5 bits para hosts e 27 para network. Adicionamos assim a seguinte rota

```
root@n6:/tmp/pycore.35741/n6.conf# ip route add 192.168.0.192/27 via 10.0.0.1 dev eth0
```

A conectividade é mantida como é visível abaixo



b) Repita o processo descrito na alínea anterior para CondadoPortugalense e Institucional, também no dispositivo n6.

A situação é mais ou menos idêntica à anterior pois entre 224 e 256 temos 32 bits, logo 5 para hosts e 27 para network, e, por isso, aplicam-se os seguintes comandos


```

root@n6:/tmp/pycore.35741/n6.conf# ip route del 192.168.0.224/29 via 10.0.0.6 dev eth1
root@n6:/tmp/pycore.35741/n6.conf# ip route del 192.168.0.232/29 via 10.0.0.6 dev eth1
root@n6:/tmp/pycore.35741/n6.conf# ip route del 192.168.0.240/29 via 10.0.0.6 dev eth1
root@n6:/tmp/pycore.35741/n6.conf# ip route del 192.168.0.248/29 via 10.0.0.6 dev eth1
ip route add 192.168.0.224/27 via 10.0.0.6 dev eth1

```

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

Aspetos positivos:

1. Redução do número de endereços IP: O supernetting permite a agregação de vários endereços IP em um único bloco, reduzindo o número de endereços IP necessários para uma determinada rede.
2. Maior eficiência de encaminhamento: com menos endereços IP sendo usados, o encaminhamento torna-se mais eficiente, reduzindo o tráfego na rede e melhorando a velocidade de comunicação.
3. Melhor segurança: o uso de supernetting pode ajudar a evitar ataques (DDoS) ao reduzir o número de rotas necessárias para serem mantidas e atualizadas nos routers.

Aspetos negativos:

1. Problemas de compatibilidade: supernetting pode não ser compatível com todos os tipos de routers ou software de rede, o que pode tornar difícil a implantação.
2. Dificuldade em gerir redes grandes: supernetting pode tornar a gerência de redes grandes mais complexa, uma vez que é necessário garantir que todos os endereços IP agregados sejam alocados corretamente.
3. Maior risco de conflitos de endereço IP: como os endereços IP são agregados num único bloco, há um risco maior de conflitos de endereço IP, o que pode levar a problemas de conectividade e segurança.