



Universidade do Minho

Braga, Portugal

TRABALHO PRÁTICO 2 - RELATÓRIO

GRUPO 82

Redes de Computadores

Engenharia Informática 2024/25

Equipa de Trabalho:

A106932 - Luís António Peixoto Soares

A104438 - Gonçalo Filipe Duarte Barbosa

A104619 - Gonçalo da Silva Carmo

2 de Abril

Índice

1. Objetivos	1
2. 1º Parte	2
2.1. Exercício 1	2
2.2. Exercício 2	5
2.3. Exercício 3	8
3. 2º Parte	14
3.1. Exercício 1	14
3.2. Exercício 2	20
3.3. Exercício 3	36
4. Conclusão	43

1. Objetivos

Este relatório tem como objetivo analisar e compreender o funcionamento do Internet Protocol(IP), nas suas principais vertentes, sendo elas a estrutura e formato dos datagramas ou pacotes IP, o encaminhamento e endereçamento de pacotes IP e a fragmentação de pacotes.

Além disso, este trabalho foi realizado através da execução de experimentos práticos utilizando ferramentas como o traceroute e Wireshark, que possibilitaram a captura e análise do tráfego de rede.

2. 1º Parte

2.1. 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 15ms. Após ativar a topologia, note que pode não existir conectividade IP imediata entre Lost e Found até que o anúncio de rotas entre routers estabilize.

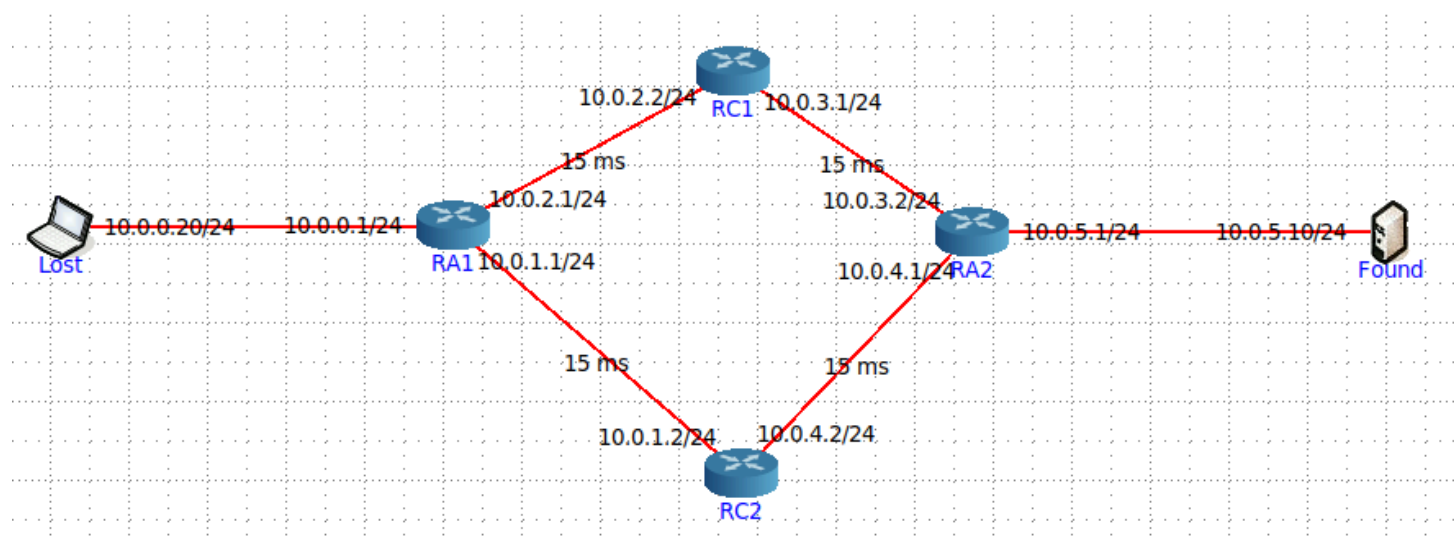


Figura 1: Topologia core(exercício 1)

a) Ative 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 traceroute.

13	15.209016832	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=1/256, ttl=1 (no response...
14	15.209076590	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded	(Time to live exceeded in transit)
15	15.209109962	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=2/512, ttl=1 (no response...
16	15.209131220	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded	(Time to live exceeded in transit)
17	15.209148673	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=3/768, ttl=1 (no response...
18	15.209168221	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded	(Time to live exceeded in transit)
19	15.209231436	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=4/1024, ttl=2 (no respons...
20	15.209290923	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=5/1280, ttl=2 (no respons...
21	15.209309836	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=6/1536, ttl=2 (no respons...
22	15.209331205	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=7/1792, ttl=3 (no respons...
23	15.209348080	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=8/2048, ttl=3 (no respons...
24	15.209365309	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=9/2304, ttl=3 (no respons...
25	15.209386285	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=10/2560, ttl=4 (reply in ...
26	15.209403630	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=11/2816, ttl=4 (reply in ...
27	15.209420828	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=12/3072, ttl=4 (reply in ...
28	15.209441531	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=13/3328, ttl=5 (reply in ...
29	15.209458310	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=14/3584, ttl=5 (reply in ...
30	15.209476790	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=15/3840, ttl=5 (reply in ...
31	15.209496672	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=16/4096, ttl=6 (reply in ...
32	15.217722108	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=17/4352, ttl=6 (reply in ...
33	15.217756925	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=18/4608, ttl=6 (reply in ...
34	15.217776826	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=19/4864, ttl=7 (reply in ...
35	15.276227008	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded	(Time to live exceeded in transit)
36	15.276234073	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded	(Time to live exceeded in transit)
37	15.276236966	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded	(Time to live exceeded in transit)
38	15.278779641	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=20/5120, ttl=7 (reply in ...
39	15.279813860	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=21/5376, ttl=7 (reply in ...
40	15.279834889	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request	id=0x001b, seq=22/5632, ttl=8 (reply in ...
41	15.332077906	10.0.3.2	10.0.0.20	ICMP	102 Time-to-live exceeded	(Time to live exceeded in transit)
42	15.332089821	10.0.3.2	10.0.0.20	ICMP	102 Time-to-live exceeded	(Time to live exceeded in transit)
43	15.332093955	10.0.3.2	10.0.0.20	ICMP	102 Time-to-live exceeded	(Time to live exceeded in transit)
44	15.332097812	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=10/2560, ttl=61 (request ...
45	15.332101974	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=11/2816, ttl=61 (request ...
46	15.332106304	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=12/3072, ttl=61 (request ...
47	15.332109892	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=13/3328, ttl=61 (request ...
48	15.332113114	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=14/3584, ttl=61 (request ...
49	15.332115924	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=15/3840, ttl=61 (request ...
50	15.332118727	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=16/4096, ttl=61 (request ...
51	15.332121538	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=17/4352, ttl=61 (request ...
52	15.332124331	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=18/4608, ttl=61 (request ...
53	15.332127116	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=19/4864, ttl=61 (request ...
54	15.332129920	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=20/5120, ttl=61 (request ...
55	15.332132726	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=21/5376, ttl=61 (request ...
56	15.332135518	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x001b, seq=22/5632, ttl=61 (request ...

Figura 2: Output Wireshark(exercicio 1.a)

```

vcmd
root@Lost:/tmp/pycore.33577/Lost.conf# traceroute -I 10.0.5.10
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.166 ms 0.033 ms 0.029 ms
 2 10.0.1.2 (10.0.1.2) 61.011 ms 60.953 ms 60.936 ms
 3 10.0.3.2 (10.0.3.2) 122.762 ms 122.750 ms 122.737 ms
 4 10.0.5.10 (10.0.5.10) 122.721 ms 122.707 ms 122.694 ms
root@Lost:/tmp/pycore.33577/Lost.conf#

```

Figura 3: Output traceroute(exercicio 1.a)

O traceroute é uma ferramenta de diagnóstico de rede que utiliza pacotes ICMP com TTL (Time To Live) cada vez maior para determinar a rota que os pacotes estão a seguir de um dispositivo de origem para um determinado destino. Como podemos observar na figura 2, os pacotes vão sendo enviados com TTL cada vez maior do endereço de ip do host Lost e as respostas a esses pacotes são registadas. Quando o TTL é igual a 4 uma resposta é obtida como podemos, mais uma vez, ver na figura 2.

b) Qual deve ser o valor inicial mínimo do campo TTL para alcançar o servidor Found? Esboce um esquema com o valor do campo TTL à chegada a cada um dos routers percorridos até ao servidor Found. Verifique na prática que a sua resposta está correta.

O valor inicial mínimo do campo TTL para alcançar o servidor Found deve ser 4 uma vez que é o número mínimo de ligações que o pacote tem de passar entre o host Lost e o servidor Found. Podemos ver na figura 2 que quando o TTL é igual a 4 o pacote chega ao destino sem retornar erros.

Host(PC) Lost → TTL 4

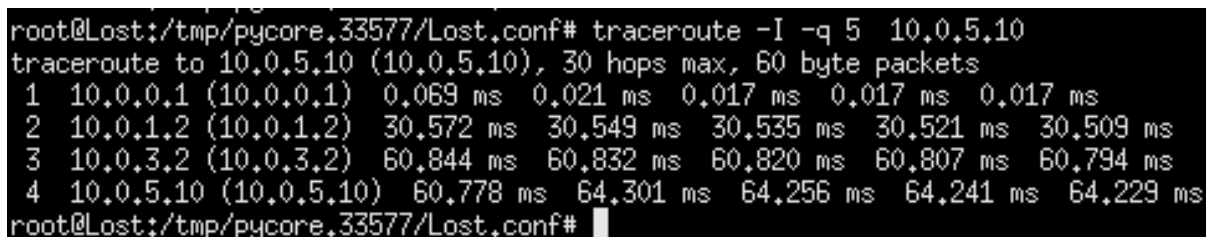
Router RA1 → TTL 3

Router RC1 ou RC2 → TTL 2

Router RA2 → TTL 1

Host(Servidor) Found → Chegou ao destino

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



```
root@Lost:/tmp/pycore.33577/Lost.conf# traceroute -I -q 5 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) 0.069 ms 0.021 ms 0.017 ms 0.017 ms 0.017 ms
 2 10.0.1.2 (10.0.1.2) 30.572 ms 30.549 ms 30.535 ms 30.521 ms 30.509 ms
 3 10.0.3.2 (10.0.3.2) 60.844 ms 60.832 ms 60.820 ms 60.807 ms 60.794 ms
 4 10.0.5.10 (10.0.5.10) 60.778 ms 64.301 ms 64.256 ms 64.241 ms 64.229 ms
root@Lost:/tmp/pycore.33577/Lost.conf#
```

Figura 4: Output traceroute(exercicio 1.c)

$(60.778+64.301+64.256+64.241+64.229)/5=63.561$ ms.

Como podemos ver pela figura 4, o RTT médio é 63.561 ms.

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

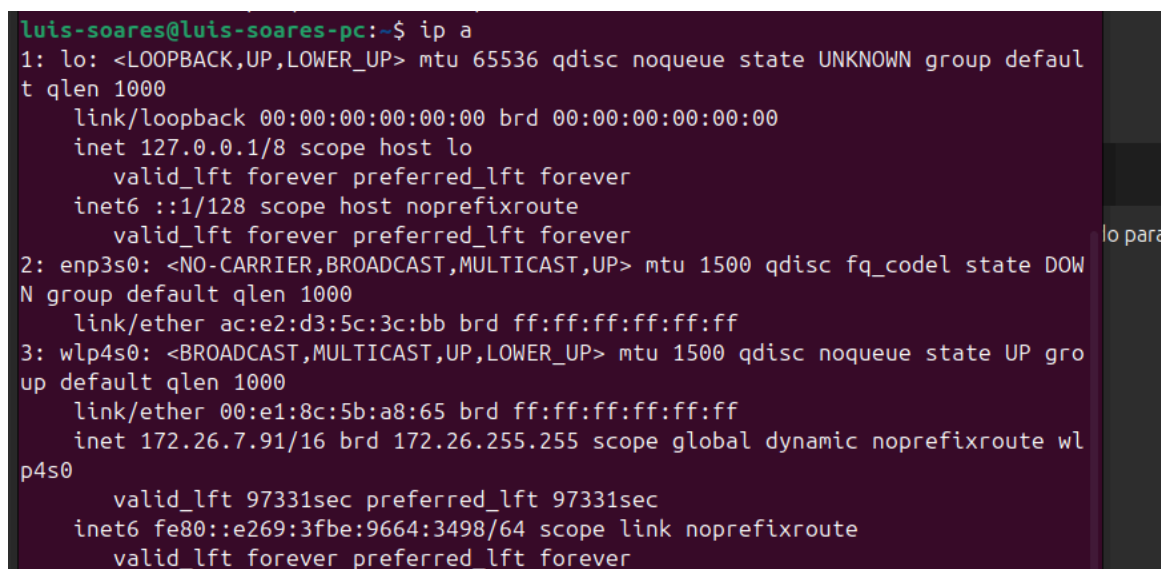
O valor médio do atraso num sentido não pode ser calculado ao dividir o RTT por 2, pois as rotas de ida e de volta podem ser diferentes, os dispositivos de rede como routers introduzem latência o que afeta o cálculo do atraso num sentido. Numa rede real o cálculo torna-se ainda mais difícil devido ao congestionamento da rede que varia com o tempo, routing dinâmico e o jitter.

2.2. Exercício 2

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

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

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



```
luis-soares@luis-soares-pc:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp3s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether ac:e2:d3:5c:3c:bb brd ff:ff:ff:ff:ff:ff
3: wlp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:e1:8c:5b:a8:65 brd ff:ff:ff:ff:ff:ff
    inet 172.26.7.91/16 brd 172.26.255.255 scope global dynamic noprefixroute wlp4s0
        valid_lft 97331sec preferred_lft 97331sec
    inet6 fe80::e269:3fbe:9664:3498/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Figura 5: Output do comando 'ip a'

Como podemos ver na figura 5, ao usar o comando 'ip a' no terminal do linux somos capazes de obter o endereço IP da interface ativa do computador que estamos a utilizar, onde, neste caso, conseguimos ver que, neste momento, o endereço IP da máquina usada como exemplo é 172.26.7.91.

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

No.	Time	Source	Destination	Protocol
45	9.297793554	172.26.7.91	193.136.9.240	ICMP
46	9.297840372	172.26.7.91	193.136.9.240	ICMP
47	9.297859806	172.26.7.91	193.136.9.240	ICMP
48	9.297882194	172.26.7.91	193.136.9.240	ICMP

▶ Frame 45: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on
 ▶ Linux cooked capture v1
 ▶ Internet Protocol Version 4, Src: 172.26.7.91, 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: 0x49d4 (18900)
 ▼ 000. = Flags: 0x0
 0... = Reserved bit: Not set
 .0.. = Don't fragment: Not set
 ..0. = More fragments: Not set
 ...0 0000 0000 0000 = Fragment Offset: 0
 ▶ Time to Live: 1
 Protocol: ICMP (1)
 Header Checksum: 0xf0ff [validation disabled]
 [Header checksum status: Unverified]
 Source Address: 172.26.7.91
 Destination Address: 193.136.9.240

Figura 6: Análise da primeira mensagem ICMP capturada

Como podemos ver na figura 6, o valor do campo protocol é ICMP (1) e ele permite-nos identificar o protocolo usado no envio da mensagem.

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

Como podemos ver na figura 6, o tamanho do cabeçalho IPv4 é 20 bytes, enquanto que o tamanho total é 60 bytes, logo o tamanho do payload é 60-20 bytes, ou seja, 40 bytes.

d) O datagrama IP foi fragmentado? Justifique.

O datagrama não foi fragmentado, pois como podemos ver na figura 6, o campo “Fragment Offset” é 0, e o campo “More fragments” tem valor 0 e está como “Not set”, o que nos leva a concluir que o pacote não foi fragmentado, pois no Wireshark, em pacotes fragmentados, o campo “More fragments” é igual a 1 para todos os fragmentos exceto o último.

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. Justifique estas mudanças.

No.	Time	Source	Destination	Protocol	Length	Info
45	9.297793554	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=1/256, ttl=1 (no response found!)
46	9.297840372	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=2/512, ttl=1 (no response found!)
47	9.297859806	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=3/768, ttl=1 (no response found!)
48	9.297882194	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=4/1024, ttl=2 (no response found!)
49	9.297906378	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=5/1280, ttl=2 (no response found!)
50	9.297931931	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=6/1536, ttl=2 (no response found!)
51	9.297964564	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=7/1792, ttl=3 (no response found!)
52	9.297991604	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=8/2048, ttl=3 (no response found!)
53	9.298016669	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=9/2304, ttl=3 (no response found!)
54	9.298041427	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=10/2560, ttl=4 (reply in 76)
55	9.298062440	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=11/2816, ttl=4 (reply in 77)
56	9.298086761	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=12/3072, ttl=4 (reply in 78)
57	9.298116616	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=13/3328, ttl=5 (reply in 79)
58	9.298141985	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=14/3584, ttl=5 (reply in 80)
59	9.298170027	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=15/3840, ttl=5 (reply in 81)
60	9.298203620	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=16/4096, ttl=6 (reply in 82)
61	9.301538809	172.26.254.254	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
64	9.305858498	172.26.254.254	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
65	9.305858807	172.26.254.254	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
68	9.307442627	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=17/4352, ttl=6 (reply in 95)
69	9.307523593	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=18/4608, ttl=6 (reply in 96)
70	9.307572973	172.26.7.91	193.136.9.240	ICMP	76	Echo (ping) request id=0x19a3, seq=19/4864, ttl=7 (reply in 97)
71	9.312211000	172.16.2.1	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
74	9.313084761	172.16.2.1	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
75	9.313085110	172.16.2.1	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
76	9.313085261	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=10/2560, ttl=61 (request in 54)
77	9.313085413	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=11/2816, ttl=61 (request in 55)
78	9.313085558	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=12/3072, ttl=61 (request in 56)
79	9.313085708	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=13/3328, ttl=61 (request in 57)
80	9.313085867	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=14/3584, ttl=61 (request in 58)
81	9.313086011	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=15/3840, ttl=61 (request in 59)
82	9.313154237	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=16/4096, ttl=61 (request in 60)
85	9.317719122	172.16.115.252	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
86	9.317719620	172.16.115.252	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
87	9.317719735	172.16.115.252	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)

Figura 7: Output wireshark (exercício 2)

Os campos que variam de pacote para pacote são os campos “Identification”, “TTL” e o “Header Checksum”. O campo “TTL” vai aumentando devido ao traceroute que envia pacotes ICMP com TTL cada vez maior, o campo “Identification” varia, pois cada pacote tem uma identificação única e o campo “Header Checksum”, representado por 16 bits, varia, pois ele é calculado como a soma complementar para 1 de todos os campos do cabeçalho, divididos em palavras de 16 bits, logo como de pacote para pacote, certos campos como o “TTL” e o “Identification” variam, o campo “Header Checksum” acaba também por variar.

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

Após analisar o tráfego ICMP fomos capazes de notar que a cada pacote recebido, o campo “Identification” incrementava de 1 em 1, enquanto que o campo “TTL” é incrementado a cada 3 pacotes enviados.

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

No.	Time	Source	Destination	Protocol	Length	Info
61	9.301538809	172.26.254.254	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
64	9.305858498	172.26.254.254	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
65	9.305858807	172.26.254.254	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
71	9.312211000	172.16.2.1	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
74	9.313084761	172.16.2.1	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
75	9.313085110	172.16.2.1	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
76	9.313085261	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=10/2560, ttl=61 (request in 54)
77	9.313085413	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=11/2816, ttl=61 (request in 55)
78	9.313085558	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=12/3072, ttl=61 (request in 56)
79	9.313085708	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=13/3328, ttl=61 (request in 57)
80	9.313085867	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=14/3584, ttl=61 (request in 58)
81	9.313086011	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=15/3840, ttl=61 (request in 59)
82	9.313154237	193.136.9.240	172.26.7.91	ICMP	76	Echo (ping) reply id=0x19a3, seq=16/4096, ttl=61 (request in 60)
85	9.317719122	172.16.115.252	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
86	9.317719620	172.16.115.252	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
87	9.317719735	172.16.115.252	172.26.7.91	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)

Figura 8: Respostas ICMP TTL Exceeded

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

O valor do TTL das mensagens de resposta ICMP TTL Exceed, varia entre 255 e 253. Isso acontece devido a estas mensagens serem respostas a um traceroute. Na prática, o TTL destas mensagens é igual a 256 menos o TTL da mensagem a que estão a responder, o que significa que estas mensagens de resposta ICMP TTL Exceeded estão a responder a mensagens que originalmente teriam entre 1 e 3 de TTL e não foram capazes de chegar ao seu destino, uma vez que, assim como podemos ver na figura 7, apenas existe uma resposta quando o TTL é igual a 4 ou superior.

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

Uma vez que o TTL da mensagem a que estão a responder não foi suficiente para chegar ao destino não é possível saber qual o TTL necessário entre a fonte e o destino. Assim, envia-se a resposta com um TTL bastante elevado para que tenhamos a certeza de que este chega ao seu destino.

h) A informação contida no cabeçalho ICMP poderia ser incluída no cabeçalho IPv4? Se sim, quais seriam as suas vantagens/desvantagens?

Seria teoricamente possível incluir a informação do protocolo ICMP no cabeçalho IPv4. A única vantagem seria a simplificação do processamento. As desvantagens seriam o aumento do tamanho do cabeçalho, complexidade de implementação e menor flexibilidade. Assim considera-se que existem mais desvantagens que vantagens e por isso a inclusão da informação do cabeçalho ICMP no cabeçalho IPv4 seria prejudicial e por isso não deve ser feita.

2.3. 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 (3800 + 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".

36	6.024087591	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=1/256, ttl=1 (no response found!)
37	6.024135528	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=271f)
38	6.024147347	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=271f)
39	6.024176798	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=2/512, ttl=1 (no response found!)
40	6.024188709	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2720)
41	6.024198918	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2720)
42	6.024224356	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=3/768, ttl=1 (no response found!)
43	6.024235600	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2721)
44	6.024245367	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2721)
45	6.024274070	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=4/1024, ttl=2 (no response found!)
46	6.024285234	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2722)
47	6.024295056	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2722)
48	6.024320801	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=5/1280, ttl=2 (no response found!)
49	6.024331831	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2723)
50	6.024344002	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2723)
51	6.024377365	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=6/1536, ttl=2 (no response found!)
52	6.024388560	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2724)
53	6.024398556	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2724)
54	6.024427232	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=7/1792, ttl=3 (no response found!)
55	6.024436899	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2725)
56	6.024448126	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2725)
57	6.024472800	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=8/2048, ttl=3 (no response found!)
58	6.024483768	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2726)
59	6.024493545	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2726)
60	6.024518384	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=9/2304, ttl=3 (no response found!)
61	6.024529810	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2727)
62	6.024539740	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2727)
63	6.024567243	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=10/2560, ttl=4 (reply in 94)
64	6.024578128	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2728)
65	6.024588569	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2728)
66	6.024621433	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=11/2816, ttl=4 (reply in 97)
67	6.024632534	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2729)
68	6.024642298	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2729)
69	6.024667195	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request id=0xb90, seq=12/3072, ttl=4 (reply in 101)
70	6.024678103	172.26.7.91	193.136.9.240	IPv4	1516 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=272a)
71	6.024687833	172.26.7.91	193.136.9.240	IPv4	938 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=272a)

Figura 9: Output wireshark(exercício 3)

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

O pacote inicial foi fragmentado porque o seu tamanho é maior que o MTU (Maximum transmission unit), ou seja, o tamanho que queremos transmitir é de 3882 bytes que é demasiado grande para ser transmitido, visto que o limite máximo é 1500 bytes(MTU), por isso o pacote inicial teve de ser dividido de maneira que nenhum dos fragmentos resultantes exceda o MTU.

b) Imprima o primeiro fragmento do datagrama IP segmentado. 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?

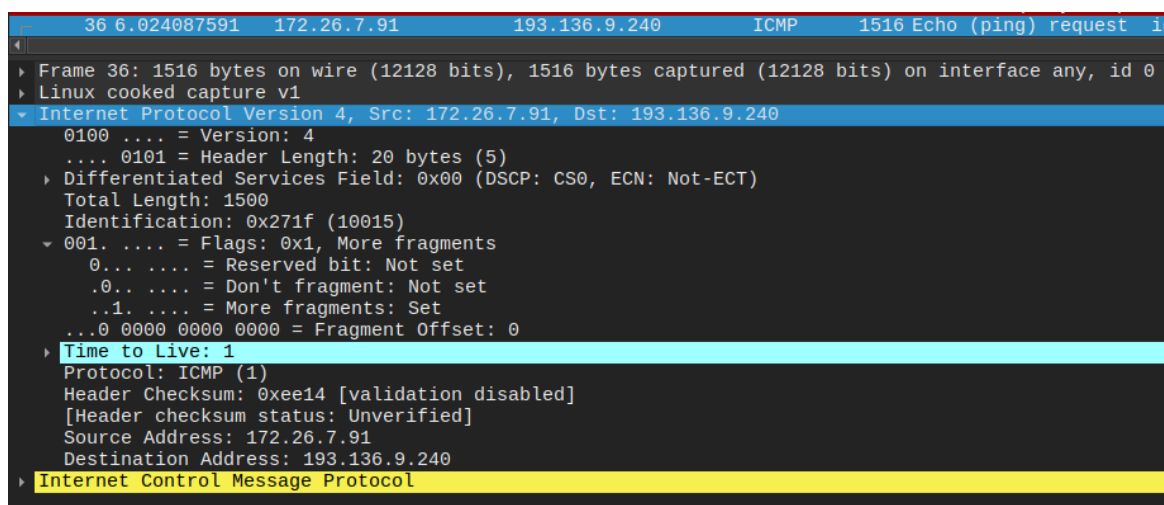


Figura 10: Primeiro fragmento do datagrama IP segmentado

A fragmentação é identificada por dois campos no cabeçalho IP. No campo Flags temos o campo “More fragments” com valor 1 e como “Set”, o que significa que o pacote ainda tem mais fragmentos a seguir e temos o “Fragment Offset” com valor 0, sendo 0 a posição do fragmento dentro do datagrama original, o que confirma que este se trata do primeiro fragmento. Por fim, na figura 10 também somos capazes de ver que o tamanho total deste datagrama IP é de 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?

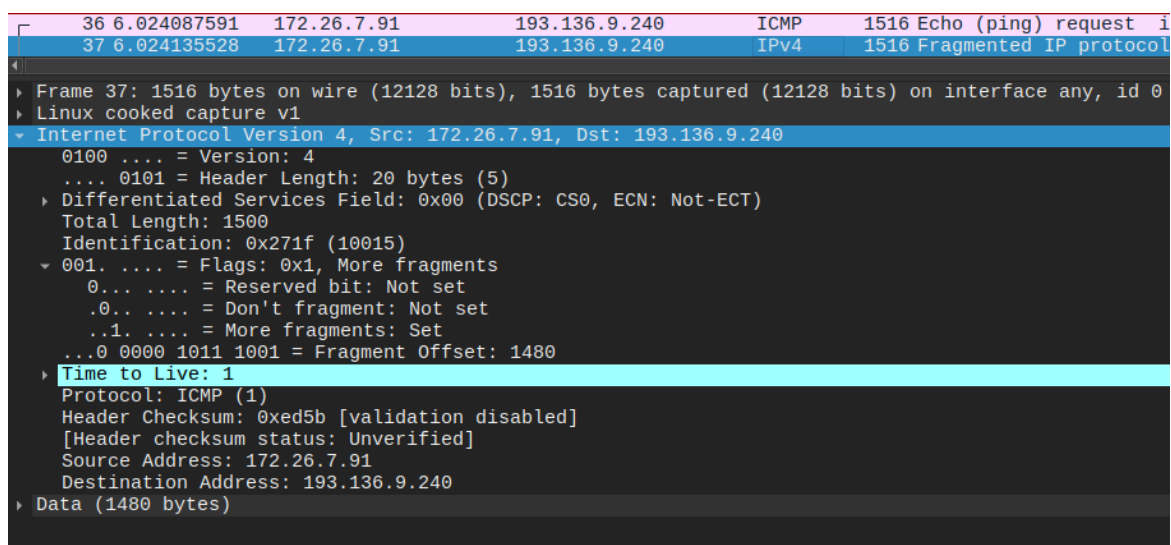


Figura 11: Segundo fragmento do datagrama IP segmentado

Como podemos ver na figura 11, o campo “Fragment Offset” tem valor igual a 1480, o que nos permite concluir que este não é o primeiro fragmento, além disso o campo “More fragments” tem valor 1 e está como “Set”, logo ainda existem mais fragmentos.

d) Quantos fragmentos foram criados a partir do datagrama original? Como se detecta o último fragmento correspondente ao datagrama original? Estabeleça um filtro no Wireshark que permita listar apenas o último fragmento do primeiro datagrama IP segmentado.

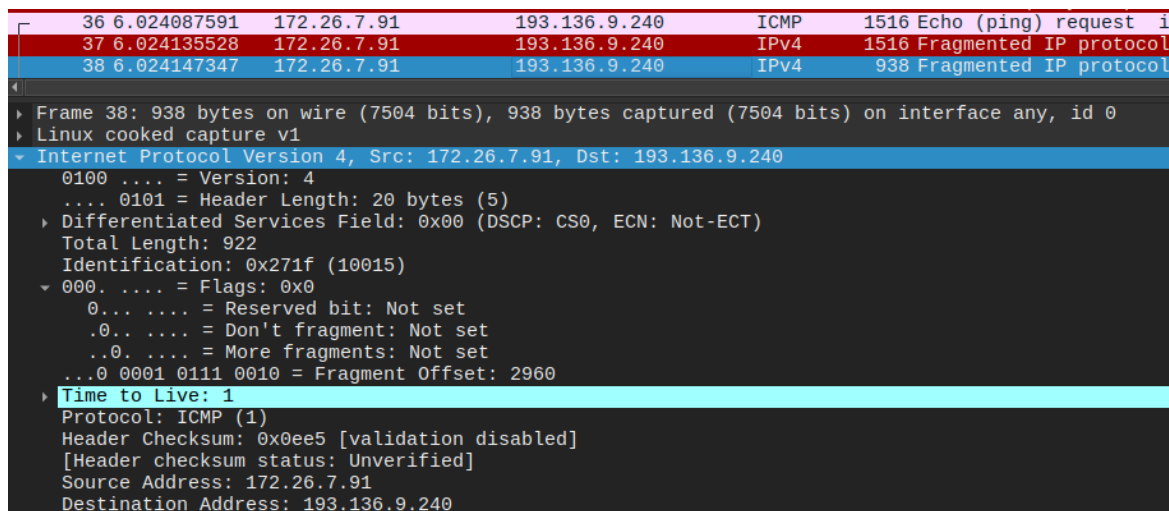


Figura 12: Terceiro fragmento do diagrama IP segmentado

Como podemos ver na figura 12, o terceiro fragmento da primeira mensagem ICMP tem o campo “More fragments” com valor 0, o que nos permite concluir que este fragmento se trata do último.

Se o datagrama tiver o campo “More fragments” com valor 0, mas tiver um “Fragment Offset” diferente de 0 este é o último datagrama fragmentado. O filtro no Wireshark que permite listar apenas o último fragmento do primeiro datagrama IP segmentado é `ip.flags.mf == 0 and ip.frag_offset != 0 and ip.id == 0x271f`.

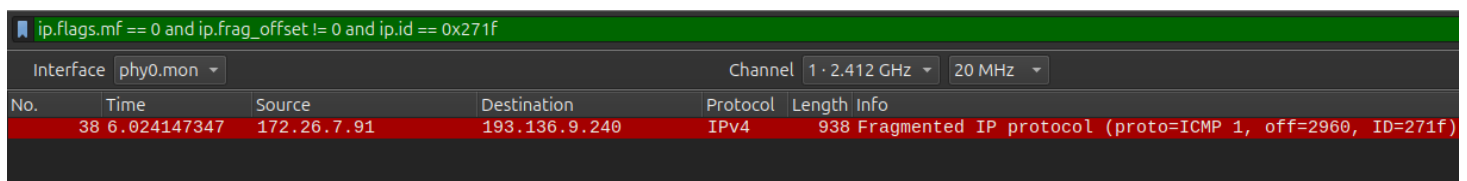


Figura 13: Utilização do filtro no wireshark

e) 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.

Nos diferentes fragmentos de um datagrama IP os campos que mudam num cabeçalho IP são o “Total Length”, o “Header Checksum”, os valores do campo “Flags” e o campo “Fragment Offset”. A reconstrução do datagrama original é possível porque cada fragmento contém informações suficientes para determinar sua posição relativa dentro do datagrama original. No destino são identificados todos os fragmentos pertencentes ao mesmo datagrama IP com base no valor do campo “Identification”, onde são ordenados com base no campo “Fragment Offset” até encontrar o fragmento com a flag “More fragments” igual a zero, que corresponde ao último fragmento.

f) Estime teoricamente o número de fragmentos gerados e o número de bytes transportados em cada um dos fragmentos. Apresente todos os cálculos

efetuados, incluindo os campos do cabeçalho IP relevantes para cada um dos fragmentos.

Como pretendemos enviar 3882 bytes, o pacote gera 3 fragmentos pois $3882/1480$ é aproximadamente igual a 2.623, onde os dois primeiros fragmentos irão transportar 1480 bytes dos 3882 mais 20 bytes do Cabeçalho IPv4 e o último fragmento irá transportar $3882-2960=922$ bytes mais os restantes 20 bytes. Através das figuras 10, 11 e 12, somos capazes de ver a partir do campo “Fragment Offset” quantos bytes dos 3882 foram transportados por cada fragmento.

Como podemos ver na figura 9, onde é apresentado uma parte do output do wireshark usado e analisado no exercício 3, o wireshark, assim como estimamos, gera 3 fragmentos por pacote.

g) Por que razão apenas o primeiro fragmento de cada pacote é identificado pelo Wireshark como sendo um pacote ICMP? Justifique a sua resposta com base no conceito de Fragmentação apresentado nas aulas teóricas.

Quando um pacote é fragmentado em vários fragmentos menores para serem transmitidos através de uma rede que não suporta o tamanho total do pacote, apenas o primeiro fragmento retém o cabeçalho original do protocolo de transporte (neste caso o protocolo ICMP). Os fragmentos seguintes têm apenas informações sobre a fragmentação e não tem cabeçalhos completos do protocolo transporte.

h) 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?

Para se determinar se um certo datagrama deve ser fragmentado, deve-se comparar o tamanho deste com o valor MTU (Maximum Transmission Unit) da interface de rede. O MTU representa o tamanho máximo do pacote que pode ser transmitido em uma interface de rede específica sem fragmentação. Aumentar o MTU iria diminuir a quantidade de fragmentação necessária para a transmissão de pacotes nessa rede e a diminuição vice-versa.

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

230	27.738008428	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request	id=0x1aa6, seq=14/3584, ttl=64 (reply in 231)
231	27.740624787	193.136.9.240	172.26.7.91	ICMP	1516 Echo (ping) reply	id=0x1aa6, seq=14/3584, ttl=61 (request in 230)
232	28.197852644	172.26.7.91	162.247.241.14	TLSv1.2	1003 Application Data	
233	28.199077739	172.26.7.91	162.247.243.29	TCP	76 57052 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=12269	
234	28.206248068	162.247.243.29	172.26.7.91	TCP	76 443 → 57052 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1250 SACK_PERM	
235	28.206301413	172.26.7.91	162.247.243.29	TCP	68 57052 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1226949985 TSecr=2	
236	28.206357399	162.247.241.14	172.26.7.91	TCP	68 443 → 38174 [ACK] Seq=1366 Ack=4742 Win=18 Len=0 TSval=3599010789 TSecr=2	
237	28.208527638	172.26.7.91	162.247.243.29	TCP	1306 57052 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=1238 TSval=1226949987 TSecr=2	
238	28.208593686	172.26.7.91	162.247.243.29	TLSv1.3	1173 Client Hello (SNI=bam.nr-data.net)	
239	28.217470704	162.247.243.29	172.26.7.91	TCP	68 443 → 57052 [ACK] Seq=1 Ack=1239 Win=147456 Len=0 TSval=2091857128 TSecr=2	
240	28.218164772	162.247.243.29	172.26.7.91	TCP	68 443 → 57052 [ACK] Seq=1 Ack=2344 Win=150016 Len=0 TSval=2091857128 TSecr=2	
241	28.218165093	162.247.243.29	172.26.7.91	TLSv1.3	527 Server Hello, Change Cipher Spec, Application Data, Application Data,	
242	28.218223949	172.26.7.91	162.247.243.29	TCP	68 57052 → 443 [ACK] Seq=2344 Ack=460 Win=63872 Len=0 TSval=1226949997 TSecr=2	
243	28.219967958	172.26.7.91	162.247.243.29	TLSv1.3	132 Change Cipher Spec, Application Data	
244	28.248721377	162.247.243.29	172.26.7.91	TCP	68 443 → 57052 [ACK] Seq=460 Ack=2408 Win=150016 Len=0 TSval=2091857160 TSecr=2	
245	28.338806416	162.247.241.14	172.26.7.91	TLSv1.2	523 Application Data	
246	28.338857851	172.26.7.91	162.247.241.14	TCP	68 38174 → 443 [ACK] Seq=4742 Ack=1821 Win=307 Len=0 TSval=1261318601 TSecr=2	
247	28.339868881	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request	id=0x1aa6, seq=15/3840, ttl=64 (reply in 248)
248	28.743029129	193.136.9.240	172.26.7.91	ICMP	1516 Echo (ping) reply	id=0x1aa6, seq=15/3840, ttl=61 (request in 247)
249	28.782812078	172.26.7.91	23.147.168.177	NTP	92 NTP Version 4, client	
250	29.069674219	23.147.168.177	172.26.7.91	NTP	92 NTP Version 4, server	
251	29.070741632	51.116.145.35	172.26.7.91	TLSv1.2	93 Application Data	
252	29.070807501	172.26.7.91	51.116.145.35	TCP	68 39998 → 443 [ACK] Seq=1035 Ack=123 Win=449 Len=0 TSval=3312552386 TSecr=2	
253	29.741339655	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request	id=0x1aa6, seq=16/4096, ttl=64 (reply in 254)
254	29.745312295	193.136.9.240	172.26.7.91	ICMP	1516 Echo (ping) reply	id=0x1aa6, seq=16/4096, ttl=61 (request in 253)
255	30.742615732	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request	id=0x1aa6, seq=17/4352, ttl=64 (reply in 256)
256	30.746613292	193.136.9.240	172.26.7.91	ICMP	1516 Echo (ping) reply	id=0x1aa6, seq=17/4352, ttl=61 (request in 255)
257	31.743751276	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request	id=0x1aa6, seq=18/4608, ttl=64 (reply in 258)
258	31.747158794	193.136.9.240	172.26.7.91	ICMP	1516 Echo (ping) reply	id=0x1aa6, seq=18/4608, ttl=61 (request in 257)
259	32.744790858	172.26.7.91	193.136.9.240	ICMP	1516 Echo (ping) request	id=0x1aa6, seq=19/4864, ttl=64 (reply in 260)
260	32.747952357	193.136.9.240	172.26.7.91	ICMP	1516 Echo (ping) reply	id=0x1aa6, seq=19/4864, ttl=61 (request in 259)

Figura 14: Output do comando ‘ping -M do -s 1472 marco.uminho.pt’ no wireshark

```

luis-soares@luis-soares-pc:~$ ping -M do -s 1473 marco.uminho.pt
PING marco.uminho.pt (193.136.9.240) 1473(1501) bytes of data.
ping: local error: message too long, mtu=1500
ping: local error: message too long, mtu=1500
ping: local error: message too long, mtu=1500
^Z
[1]+  Interrompido                  ping -M do -s 1473 marco.uminho.pt
luis-soares@luis-soares-pc:~$ ping -M do -s 1472 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=2.23 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=2 ttl=61 time=6.90 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=3 ttl=61 time=7.38 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=4 ttl=61 time=3.18 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=5 ttl=61 time=2.62 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=6 ttl=61 time=3.07 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=7 ttl=61 time=3.53 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=8 ttl=61 time=3.39 ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=9 ttl=61 time=4.18 ms

```

Figura 15: Outputs do comando ping com diferentes valores de SIZE

O valor máximo de SIZE foi de 1472 pois o MTU é igual a 1500, o cabeçalho IPv4 tem 20 bytes de tamanho e o cabeçalho ICMP tem tamanho igual a 8 bytes. Assim, $1500 - (20 + 8) = 1472$.

Além disso, como podemos ver na figura 15, ao usar o comando ping com um SIZE maior que 1472(1473 no exemplo usado), retorna erro, pois o tamanho total do pacote ultrapassa o MTU, que é 1500, mas também podemos ver que o mesmo não acontece com SIZE igual 1472, onde também na figura 15, somos capazes de ver que os pacotes são enviados. Assim, provamos que 1472 é o valor máximo de SIZE que se pode usar sem que ocorra fragmentação de pacote.

3. 2º Parte

3.1. Exercício 1

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

a) De modo a garantir uma posição estrategicamente mais vantajosa e ter casa de férias para relaxar entre batalhas, ordena a construção de um segundo Castelo, em Braga. Não tendo qualquer queixa do serviço prestado, recorre aos serviços do ISP ReiDaNet, que já utiliza no condado, para ter acesso à rede no segundo Castelo. O ISP atribuiu-lhe o endereço de rede IP 172.68.XX.192/26 em que XX corresponde ao seu número de grupo (PLXX).

Defina um esquema de endereçamento que permita o estabelecimento de pelo menos 6 redes e que garanta que cada uma destas possa ter 5 ou mais hosts. Assuma que todos os endereços de sub-redes são utilizáveis.

Como fazemos parte do grupo 82, significa que nos foi atribuída a rede 172.68.82.192/26, onde como estamos a utilizar uma máscara de 26 bits, significa que temos um espaço de endereçamento de $2^6=64$ endereços, onde 62 são utilizáveis devido a termos 2 endereços de broadcast.

Como na rede que nos foi atribuída originalmente tínhamos uma máscara de 26 bits (/26), significa que temos 6 bits que podemos manipular ($32-26=6$), onde desses 6, podemos usar 3 bits para criar até 8 sub-redes com máscaras de 29 bits, o que nos deixa com 3 bits livres para manipular em cada sub-rede, o que significa que cada sub-rede teria $2^3=8$ endereços, onde 6 seriam utilizáveis, o que atende aos nossos requisitos de ter pelo menos 6 redes e que cada uma destas tenha 5 ou mais hosts.

Rede atribuída: 172.68.82.192/26

Exemplos de 6 sub-redes com 6 endereços/hosts cada:

172.68.82.192/29 → Hosts: 193 a 198

172.68.82.200/29 → Hosts: 201 a 206

172.68.82.208/29 → Hosts: 209 a 214

172.68.82.216/29 → Hosts: 217 a 222

172.68.82.224/29 → Hosts: 225 a 230

172.68.82.248/29 → Hosts: 249 a 254

b) Ligue um novo host Castelo2 diretamente ao router ReiDaNet. Associe-lhe um endereço, à sua escolha, pertencente a uma sub-rede disponível das criadas na alínea anterior (garanta que a interface do router ReiDaNet

utiliza o primeiro endereço válido da sub-rede escolhida). Verifique que tem conectividade com os dispositivos do Condado Portucalense.

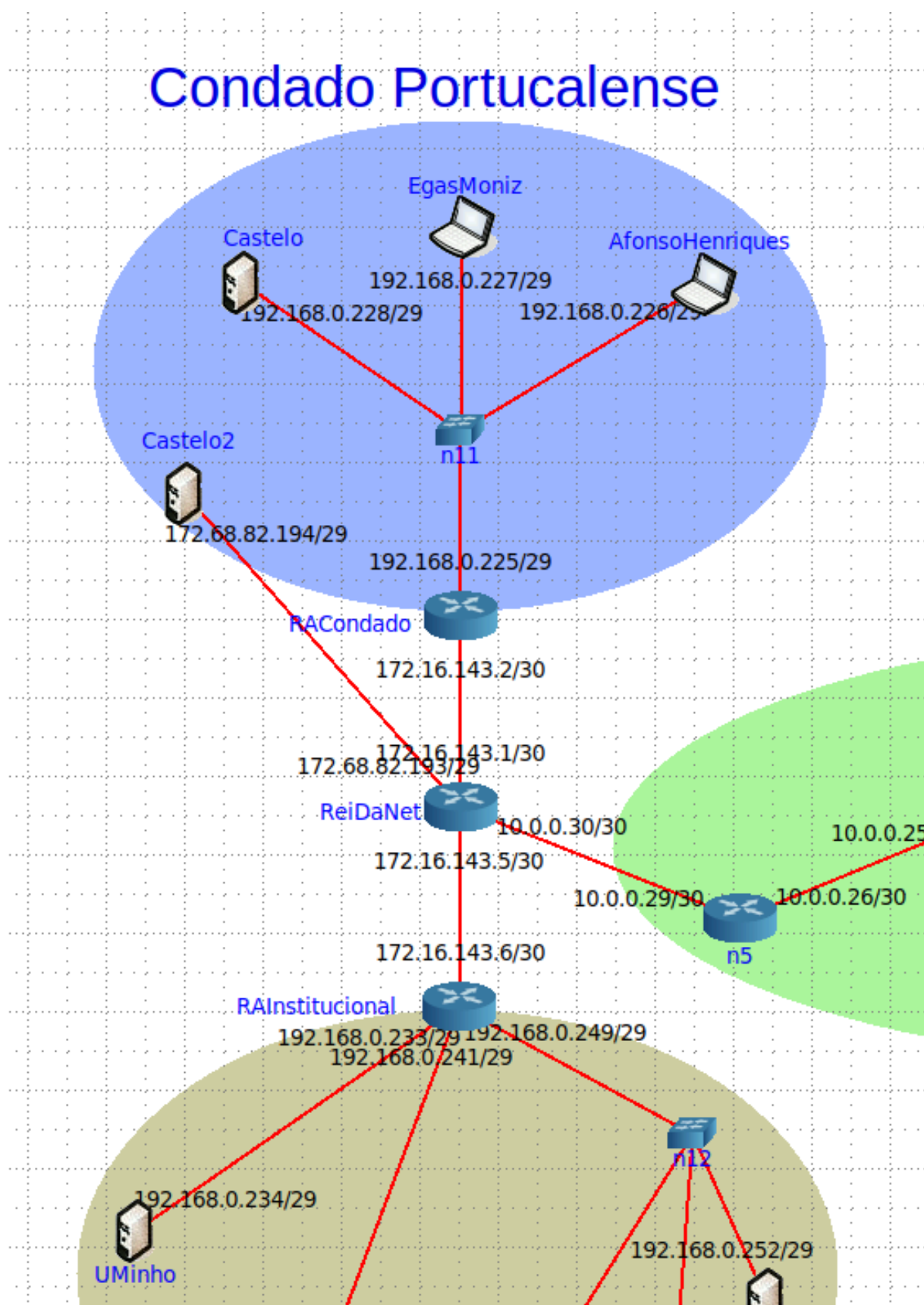


Figura 16: Topologia com o Castelo2

```
root@Castelo2:/tmp/pycore.46695/Castelo2.conf# ping 192.168.0.228
PING 192.168.0.228 (192.168.0.228) 56(84) bytes of data:
64 bytes from 192.168.0.228: icmp_seq=1 ttl=62 time=0.078 ms
64 bytes from 192.168.0.228: icmp_seq=2 ttl=62 time=0.066 ms
64 bytes from 192.168.0.228: icmp_seq=3 ttl=62 time=0.053 ms
64 bytes from 192.168.0.228: icmp_seq=4 ttl=62 time=0.040 ms
64 bytes from 192.168.0.228: icmp_seq=5 ttl=62 time=0.042 ms
64 bytes from 192.168.0.228: icmp_seq=6 ttl=62 time=0.032 ms
64 bytes from 192.168.0.228: icmp_seq=7 ttl=62 time=0.068 ms
64 bytes from 192.168.0.228: icmp_seq=8 ttl=62 time=0.101 ms
64 bytes from 192.168.0.228: icmp_seq=9 ttl=62 time=0.045 ms
64 bytes from 192.168.0.228: icmp_seq=10 ttl=62 time=0.044 ms
^C
--- 192.168.0.228 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9196ms
rtt min/avg/max/mdev = 0.032/0.056/0.101/0.020 ms
root@Castelo2:/tmp/pycore.46695/Castelo2.conf#
```

Figura 17: Prova da conectividade entre o Castelo e Castelo2

Como podemos ver na figura 16, adicionamos um novo host Castelo2 diretamente ao router ReiDaNet dando-lhe um endereço da sub-rede 172.68.82.192/29, nomeadamente o segundo utilizável, enquanto que na interface do router ReiDaNet utilizámos o primeiro endereço válido da sub-rede que escolhemos.

Por fim, na figura 17 somos capazes de ver que existe conectividade entre os hosts Castelo e Castelo2, pois ao utilizar o comando ping para enviar pacotes do host Castelo2 para o endereço do host Castelo, os pacotes foram recebidos, o que nos permite concluir que o host Castelo2 tem conectividade com os dispositivos do Condado Portucalense.

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

```
root@Castelo2:/tmp/pycore.46695/Castelo2.conf# ip route del default
<6695/Castelo2.conf# ip route add 192.168.0.224/29 via 172.68.82.193
<6695/Castelo2.conf# ip route add 192.168.0.240/29 via 172.68.82.193
<6695/Castelo2.conf# ip route add 192.168.0.232/29 via 172.68.82.193
<6695/Castelo2.conf# ip route add 192.168.0.248/29 via 172.68.82.193
root@Castelo2:/tmp/pycore.46695/Castelo2.conf#
```

Figura 18: Remoção da rota default e adição de novas rotas

```

root@Castelo2:/tmp/pycore.46695/Castelo2.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask          Flags      MSS Window  irtt  Iface
172.68.82.192    0.0.0.0          255.255.255.248 U          0 0        0     eth0
192.168.0.224    172.68.82.193    255.255.255.248 UG         0 0        0     eth0
192.168.0.232    172.68.82.193    255.255.255.248 UG         0 0        0     eth0
192.168.0.240    172.68.82.193    255.255.255.248 UG         0 0        0     eth0
192.168.0.248    172.68.82.193    255.255.255.248 UG         0 0        0     eth0
root@Castelo2:/tmp/pycore.46695/Castelo2.conf# █

```

Figura 19: Tabela de encaminhamento resultante

```

root@Castelo2:/tmp/pycore.46695/Castelo2.conf# ping 192.168.0.228
PING 192.168.0.228 (192.168.0.228) 56(84) bytes of data:
64 bytes from 192.168.0.228: icmp_seq=1 ttl=62 time=0.058 ms
64 bytes from 192.168.0.228: icmp_seq=2 ttl=62 time=0.175 ms
64 bytes from 192.168.0.228: icmp_seq=3 ttl=62 time=0.045 ms
64 bytes from 192.168.0.228: icmp_seq=4 ttl=62 time=0.067 ms
64 bytes from 192.168.0.228: icmp_seq=5 ttl=62 time=0.049 ms
^C
--- 192.168.0.228 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4091ms
rtt min/avg/max/mdev = 0.045/0.078/0.175/0.048 ms
root@Castelo2:/tmp/pycore.46695/Castelo2.conf# ping 192.168.0.225
PING 192.168.0.225 (192.168.0.225) 56(84) bytes of data:
64 bytes from 192.168.0.225: icmp_seq=1 ttl=63 time=0.216 ms
64 bytes from 192.168.0.225: icmp_seq=2 ttl=63 time=0.049 ms
64 bytes from 192.168.0.225: icmp_seq=3 ttl=63 time=0.057 ms
64 bytes from 192.168.0.225: icmp_seq=4 ttl=63 time=0.036 ms
^C
--- 192.168.0.225 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3062ms
rtt min/avg/max/mdev = 0.036/0.089/0.216/0.073 ms
root@Castelo2:/tmp/pycore.46695/Castelo2.conf# █

```

Figura 20: Prova de conectividade com o Condado Portu-
calense

```
root@Castelo2:/tmp/pycore.46695/Castelo2.conf# ping 192.168.0.234
PING 192.168.0.234 (192.168.0.234) 56(84) bytes of data.
64 bytes from 192.168.0.234: icmp_seq=1 ttl=62 time=0.070 ms
64 bytes from 192.168.0.234: icmp_seq=2 ttl=62 time=0.045 ms
64 bytes from 192.168.0.234: icmp_seq=3 ttl=62 time=0.063 ms
64 bytes from 192.168.0.234: icmp_seq=4 ttl=62 time=0.044 ms
^C
--- 192.168.0.234 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3053ms
rtt min/avg/max/mdev = 0.044/0.055/0.070/0.011 ms
root@Castelo2:/tmp/pycore.46695/Castelo2.conf# ping 192.168.0.242
PING 192.168.0.242 (192.168.0.242) 56(84) bytes of data.
64 bytes from 192.168.0.242: icmp_seq=1 ttl=62 time=0.064 ms
64 bytes from 192.168.0.242: icmp_seq=2 ttl=62 time=0.171 ms
64 bytes from 192.168.0.242: icmp_seq=3 ttl=62 time=0.160 ms
^C
--- 192.168.0.242 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.064/0.131/0.171/0.048 ms
root@Castelo2:/tmp/pycore.46695/Castelo2.conf# ping 192.168.0.252
PING 192.168.0.252 (192.168.0.252) 56(84) bytes of data.
64 bytes from 192.168.0.252: icmp_seq=1 ttl=62 time=0.088 ms
64 bytes from 192.168.0.252: icmp_seq=2 ttl=62 time=0.045 ms
64 bytes from 192.168.0.252: icmp_seq=3 ttl=62 time=0.215 ms
^C
--- 192.168.0.252 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2045ms
rtt min/avg/max/mdev = 0.045/0.116/0.215/0.072 ms
root@Castelo2:/tmp/pycore.46695/Castelo2.conf#
```

Figura 21: Prova de conectividade com a Rede Institucional

Como podemos ver nas figuras 20 e 21, existe conectividade entre o host Castelo2 e o Condado Portucalense e a rede Institucional, uma vez que os pacotes enviados do host Castelo2, com o comando ping, para certos endereços das respectivas zonas onde queremos provar a existência de conectividade, como os endereços dos hosts UMinho, DI, Bombeiros e Castelo, assim como o router RACondado, foram recebidos. Além disso, também somos capazes de ver a tabela de encaminhamento na figura 19, e na figura 18 somos capazes de ver os comandos usados para remover a rota default e adicionar novas rotas que conectassem o host Castelo2 às sub-redes Condado Portucalense e Rede Institucional.

Por fim, a rota default, é uma rota que especifica para onde enviar pacotes quando não há uma rota específica para o destino pretendida na tabela de encaminhamento. É bastante útil pois facilita o roteamento, já que em vez de adicionar rotas individuais para cada destino externo, todos os pacotes desconhecidos são enviados para um gateway

padrão, além disso, evita configuração manual excessiva, pois sem uma rota default, cada destino teria que ser especificado manualmente.

3.2. Exercício 2

D.Afonso Henriques quer enviar fotos do novo Castelo à sua mãe, D.Teresa, mas está a ter alguns problemas de comunicação. Este alega que o problema deverá estar no dispositivo de D.Teresa, uma vez que no dia anterior conseguiu fazer stream de Fortnite para todos os seus subscritores da Twitch, e acabou de sair de uma discussão política no Reddit.

a) Confirme, através do comando ping, que AfonsoHenriques tem efetivamente conectividade com os servidores Reddit e Twitch

```
root@AfonsoHenriques:/tmp/pycore.46695/AfonsoHenriques.conf# ping 192.168.0.155
PING 192.168.0.155 (192.168.0.155) 56(84) bytes of data.
64 bytes from 192.168.0.155: icmp_seq=1 ttl=55 time=0.311 ms
64 bytes from 192.168.0.155: icmp_seq=2 ttl=55 time=0.088 ms
64 bytes from 192.168.0.155: icmp_seq=3 ttl=55 time=0.141 ms
64 bytes from 192.168.0.155: icmp_seq=4 ttl=55 time=0.087 ms
64 bytes from 192.168.0.155: icmp_seq=5 ttl=55 time=0.167 ms
64 bytes from 192.168.0.155: icmp_seq=6 ttl=55 time=0.113 ms
^C
--- 192.168.0.155 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5100ms
rtt min/avg/max/mdev = 0.087/0.151/0.311/0.076 ms
```

Figura 22: Prova de conectividade com o servidor Reddit

```
root@AfonsoHenriques:/tmp/pycore.46695/AfonsoHenriques.conf# ping 192.168.0.146
PING 192.168.0.146 (192.168.0.146) 56(84) bytes of data.
64 bytes from 192.168.0.146: icmp_seq=1 ttl=55 time=0.217 ms
64 bytes from 192.168.0.146: icmp_seq=2 ttl=55 time=0.197 ms
64 bytes from 192.168.0.146: icmp_seq=3 ttl=55 time=0.088 ms
64 bytes from 192.168.0.146: icmp_seq=4 ttl=55 time=0.147 ms
64 bytes from 192.168.0.146: icmp_seq=5 ttl=55 time=0.199 ms
^C
--- 192.168.0.146 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4072ms
rtt min/avg/max/mdev = 0.088/0.169/0.217/0.046 ms
root@AfonsoHenriques:/tmp/pycore.46695/AfonsoHenriques.conf# █
```

Figura 23: Prova de conectividade com o servidor Twitch

Como podemos ver nas figuras 22 e 23, ao utilizar o comando ping para enviar pacotes do host AfonsoHenriques para os endereços dos servidores Reddit e Twitch, os pacotes foram recebidos, logo existe conectividade.

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.46695/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.46695/AfonsoHenriques.conf# █
```

Figura 24: Tabela de encaminhamento do host AfonsoHenriques

```
root@Teresa:/tmp/pycore.46695/Teresa.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask         Flags   MSS Window  irtt Iface
0.0.0.0          192.168.0.129   0.0.0.0         UG        0 0          0 eth0
192.168.0.128    0.0.0.0         255.255.255.248 U        0 0          0 eth0
root@Teresa:/tmp/pycore.46695/Teresa.conf# █
```

Figura 25: Tabela de encaminhamento do host Teresa

As tabelas de encaminhamento do host AfonsoHenriques e do host Teresa não têm erros nem problemas com as suas entradas. Cada dispositivo possui, na sua tabela de encaminhamento, uma rota default, neste caso na primeira entrada das tabelas, que permite encaminhar pacotes para redes externas através do roteador local conectado à sua sub-rede. Além disso, também neste caso, a segunda entrada indica que o tráfego para a rede local não precisa de encaminhamento, pois o dispositivo já está dentro da mesma sub-rede que os hosts, garantindo que pacotes destinados a IPs dentro da mesma sub-rede sejam entregues diretamente, sem precisar de roteamento adicional.

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

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

```
<ycore.46695/AfonsoHenriques.conf# ping 192.168.0.130
PING 192.168.0.130 (192.168.0.130) 56(84) bytes of data.
From 10.0.0.25 icmp_seq=1 Destination Host Unreachable
From 10.0.0.25 icmp_seq=2 Destination Host Unreachable
From 10.0.0.25 icmp_seq=3 Destination Host Unreachable
From 10.0.0.25 icmp_seq=4 Destination Host Unreachable
From 10.0.0.25 icmp_seq=5 Destination Host Unreachable
From 10.0.0.25 icmp_seq=6 Destination Host Unreachable
^C
--- 192.168.0.130 ping statistics ---
8 packets transmitted, 0 received, +6 errors, 100% packet loss, time 7143ms
pipe 4
<ycore.46695/AfonsoHenriques.conf# traceroute 192.168.0.130
traceroute to 192.168.0.130 (192.168.0.130), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225) 0.029 ms 0.006 ms 0.005 ms
 2 172.16.143.1 (172.16.143.1) 0.023 ms 0.010 ms 0.008 ms
 3 10.0.0.29 (10.0.0.29) 0.022 ms 0.011 ms 0.010 ms
 4 10.0.0.25 (10.0.0.25) 0.018 ms 0.013 ms 0.013 ms
 5 10.0.0.25 (10.0.0.25) 3052.087 ms !H 3052.060 ms !H 3052.040 ms !H
root@AfonsoHenriques:/tmp/pycore.46695/AfonsoHenriques.conf#
```

Figura 26: Envio de pacotes do host AfonsoHenriques para o host Teresa

```
root@Teresa:/tmp/pycore.46695/Teresa.conf# ping 192.168.0.226
PING 192.168.0.226 (192.168.0.226) 56(84) bytes of data.
From 192.168.0.129 icmp_seq=1 Destination Net Unreachable
From 192.168.0.129 icmp_seq=2 Destination Net Unreachable
From 192.168.0.129 icmp_seq=3 Destination Net Unreachable
From 192.168.0.129 icmp_seq=4 Destination Net Unreachable
^C
--- 192.168.0.226 ping statistics ---
6 packets transmitted, 0 received, +4 errors, 100% packet loss, time 5112ms

root@Teresa:/tmp/pycore.46695/Teresa.conf# traceroute 192.168.0.226
traceroute to 192.168.0.226 (192.168.0.226), 30 hops max, 60 byte packets
 1 192.168.0.129 (192.168.0.129) 0.054 ms !N 0.006 ms !N *
root@Teresa:/tmp/pycore.46695/Teresa.conf#
```

Figura 27: Envio de pacotes do host Teresa para o host AfonsoHenriques

Após usar o comando ping para enviar pacotes entre os hosts AfonsoHenriques e Teresa, como é possível ver nas figuras 26 e 27, verificamos que não foi possível enviar os pacotes, o que significa que não deverá existir conectividade entre os dois hosts.

Assim como é referido na alínea decidimos inspecionar os routers do core da rede(n1 a n6) para tentar encontrar possíveis problemas que nos estejam a impossibilitar de fazer o envio de pacotes entre os hosts.

Router n1


```

root@n1:/tmp/pycore.46695/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
192.168.0.128     10.0.0.14        255.255.255.248 UG           0 0         0 eth1
192.168.0.136     10.0.0.9         255.255.255.248 UG           0 0         0 eth0
192.168.0.144     10.0.0.9         255.255.255.248 UG           0 0         0 eth0
192.168.0.152     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

```

Figura 28: Tabela de encaminhamento do router n1(Antes)

```
<1.conf# ip route del 192.168.0.128/29 via 10.0.0.14
```

Figura 29: Remoção de rota incorreta

```
root@n1:/tmp/pycore.46695/n1.conf# ip route add 192.168.0.128/29 via 10.0.0.9
```

Figura 30: Adição da rota correta

```

root@n1:/tmp/pycore.46695/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
192.168.0.128     10.0.0.9         255.255.255.248 UG           0 0         0 eth0
192.168.0.136     10.0.0.9         255.255.255.248 UG           0 0         0 eth0
192.168.0.144     10.0.0.9         255.255.255.248 UG           0 0         0 eth0
192.168.0.152     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.46695/n1.conf#

```

Figura 31: Tabela de encaminhamento do router n1(Depois)

Ao analisar o router n1, fomos capazes de verificar que possuía uma rota incorreta devido a uma Gateway incorreto(10.0.0.14/30), o que impossibilitava a continuação do percurso. Para resolver este problema adicionamos outra rota com o GateWay correto(10.0.0.9/30) e removemos a incorreta.

Router n2

```
root@n2:/tmp/pycore.46695/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
192.168.0.128	10.0.0.13	255.255.255.248	UG	0	0	0	eth1
192.168.0.130	10.0.0.25	255.255.255.254	UG	0	0	0	eth2
192.168.0.136	10.0.0.21	255.255.255.248	UG	0	0	0	eth0
192.168.0.144	10.0.0.21	255.255.255.248	UG	0	0	0	eth0
192.168.0.152	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 32: Tabela de encaminhamento do router n2(Antes)

```
root@n2:/tmp/pycore.46695/n2.conf# ip route del 192.168.0.130/31
```

Figura 33: Remoção de rota incorreta

```

root@n2:/tmp/pycore.46695/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
192.168.0.128     10.0.0.13       255.255.255.248 UG           0 0        0 eth1
192.168.0.136     10.0.0.21       255.255.255.248 UG           0 0        0 eth0
192.168.0.144     10.0.0.21       255.255.255.248 UG           0 0        0 eth0
192.168.0.152     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.46695/n2.conf#

```

Figura 34: Tabela de encaminhamento do router n2(De-
pois)

Depois de analisar o router n2, verificamos que a sua tabela de encaminhamento possuía uma entrada incorreta, com o campo Destination(192.168.0.130/31) e o campo Gateway(10.0.0.25/30) errados, o que nos levou a remover a rota incorreta. Como a tabela já possuía uma entrada com uma rota com o destino correto e o campo Gateway também com o endereço certo, depois de remover-mos a rota incorreta fomos capazes de resolver o problema no router n2.

Router n3

```

root@n3:/tmp/pycore.46695/n3.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags        MSS Window  irtt Iface
10.0.0.0         10.0.0.5       255.255.255.252 UG           0 0        0 eth2
10.0.0.4         0.0.0.0        255.255.255.252 U            0 0        0 eth2
10.0.0.8         0.0.0.0        255.255.255.252 U            0 0        0 eth0
10.0.0.12        10.0.0.10      255.255.255.252 UG           0 0        0 eth0
10.0.0.16        0.0.0.0        255.255.255.252 U            0 0        0 eth1
10.0.0.20        10.0.0.18      255.255.255.252 UG           0 0        0 eth1
10.0.0.24        10.0.0.18      255.255.255.252 UG           0 0        0 eth1
10.0.0.28        10.0.0.10      255.255.255.252 UG           0 0        0 eth0
172.0.0.0        10.0.0.10      255.0.0.0       UG           0 0        0 eth0
172.16.142.0     10.0.0.5       255.255.255.252 UG           0 0        0 eth2
172.16.142.4     10.0.0.5       255.255.255.252 UG           0 0        0 eth2
172.16.143.0     10.0.0.18      255.255.255.252 UG           0 0        0 eth1
172.16.143.4     10.0.0.10      255.255.255.252 UG           0 0        0 eth0
192.168.0.136    10.0.0.5       255.255.255.248 UG           0 0        0 eth2
192.168.0.144    10.0.0.5       255.255.255.248 UG           0 0        0 eth2
192.168.0.152    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

```

Figura 35: Tabela de encaminhamento do router n3(Antes)

```

root@n3:/tmp/pycore.46695/n3.conf# ip route add 192.168.0.128/29 via 10.0.0.5

```

Figura 36: Adição da rota em falta

```

root@n3:/tmp/pycore.46695/n3.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags        MSS Window  irtt Iface
10.0.0.0         10.0.0.5       255.255.255.252 UG           0 0        0 eth2
10.0.0.4         0.0.0.0        255.255.255.252 U            0 0        0 eth2
10.0.0.8         0.0.0.0        255.255.255.252 U            0 0        0 eth0
10.0.0.12        10.0.0.10      255.255.255.252 UG           0 0        0 eth0
10.0.0.16        0.0.0.0        255.255.255.252 U            0 0        0 eth1
10.0.0.20        10.0.0.18      255.255.255.252 UG           0 0        0 eth1
10.0.0.24        10.0.0.18      255.255.255.252 UG           0 0        0 eth1
10.0.0.28        10.0.0.10      255.255.255.252 UG           0 0        0 eth0
172.0.0.0        10.0.0.10      255.0.0.0       UG           0 0        0 eth0
172.16.142.0     10.0.0.5       255.255.255.252 UG           0 0        0 eth2
172.16.142.4     10.0.0.5       255.255.255.252 UG           0 0        0 eth2
172.16.143.0     10.0.0.18      255.255.255.252 UG           0 0        0 eth1
172.16.143.4     10.0.0.10      255.255.255.252 UG           0 0        0 eth0
192.168.0.128    10.0.0.5       255.255.255.248 UG           0 0        0 eth2
192.168.0.136    10.0.0.5       255.255.255.248 UG           0 0        0 eth2
192.168.0.144    10.0.0.5       255.255.255.248 UG           0 0        0 eth2
192.168.0.152    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
root@n3:/tmp/pycore.46695/n3.conf#

```

Figura 37: Tabela de encaminhamento do router n3(De- pois)

Após analisar o router n3, fomos capazes de notar que faltava uma rota para a sub-rede do host Teresa, o que impossibilitava o envio de pacotes para o mesmo, para corrigir esse problema adicionamos uma rota para a sub-rede onde se encontra o host Teresa(192.168.0.128/29) com o Gateway correto(10.0.0.5).

Routers n4, n5 e n6

```

root@n4:/tmp/pycore.46695/n4.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags        MSS Window  irtt  Iface
10.0.0.0         10.0.0.17      255.255.255.252 UG           0 0        0     eth0
10.0.0.4         10.0.0.17      255.255.255.252 UG           0 0        0     eth0
10.0.0.8         10.0.0.17      255.255.255.252 UG           0 0        0     eth0
10.0.0.12        10.0.0.22      255.255.255.252 UG           0 0        0     eth1
10.0.0.16        0.0.0.0        255.255.255.252 U            0 0        0     eth0
10.0.0.20        0.0.0.0        255.255.255.252 U            0 0        0     eth1
10.0.0.24        10.0.0.22      255.255.255.252 UG           0 0        0     eth1
10.0.0.28        10.0.0.22      255.255.255.252 UG           0 0        0     eth1
172.0.0.0        10.0.0.22      255.0.0.0       UG           0 0        0     eth1
172.16.142.0     10.0.0.17      255.255.255.252 UG           0 0        0     eth0
172.16.142.4     10.0.0.17      255.255.255.252 UG           0 0        0     eth0
172.16.143.0     10.0.0.22      255.255.255.252 UG           0 0        0     eth1
172.16.143.4     10.0.0.22      255.255.255.252 UG           0 0        0     eth1
192.168.0.128    10.0.0.17      255.255.255.248 UG           0 0        0     eth0
192.168.0.136    10.0.0.17      255.255.255.248 UG           0 0        0     eth0
192.168.0.144    10.0.0.17      255.255.255.248 UG           0 0        0     eth0
192.168.0.152    10.0.0.17      255.255.255.248 UG           0 0        0     eth0
192.168.0.224    10.0.0.22      255.255.255.248 UG           0 0        0     eth1
192.168.0.232    10.0.0.22      255.255.255.248 UG           0 0        0     eth1
192.168.0.240    10.0.0.22      255.255.255.248 UG           0 0        0     eth1
192.168.0.248    10.0.0.22      255.255.255.248 UG           0 0        0     eth1
root@n4:/tmp/pycore.46695/n4.conf#

```

Figura 38: Tabela de encaminhamento do router n4

```

root@n5:/tmp/pycore.46695/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
192.168.0.0      10.0.0.30      255.255.255.0   UG           0 0        0     eth0
192.168.0.128    10.0.0.25      255.255.255.248 UG           0 0        0     eth1
192.168.0.136    10.0.0.25      255.255.255.248 UG           0 0        0     eth1
192.168.0.144    10.0.0.25      255.255.255.248 UG           0 0        0     eth1
192.168.0.152    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

```

Figura 39: Tabela de encaminhamento do router n5

```
root@n6:/tmp/pycore.46695/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
192.168.0.128	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.136	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.144	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.152	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

Figura 40: Tabela de encaminhamento do router n6

Depois de verificar os router n4, n5 e n6, concluímos que nenhum apresentava erros que impossibilitassem o envio de pacotes do host AfonsoHenriques para o host Teresa, visto que todos os três routers possuíam uma rota com o campo Destination com a sub-rede do host Teresa e o campo Gateway com o endereço do próximo dispositivo correto.

```
root@AfonsoHenriques:/tmp/pycore.46695/AfonsoHenriques.conf# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=57 time=0.248 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=57 time=0.316 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=57 time=0.212 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=57 time=0.177 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=57 time=0.190 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=57 time=0.172 ms
^C
--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5109ms
rtt min/avg/max/mdev = 0.172/0.219/0.316/0.050 ms
root@AfonsoHenriques:/tmp/pycore.46695/AfonsoHenriques.conf#
```

Figura 41: Envio de pacotes para o endereço do router
CondadoOnline

Como podemos ver na figura 41, ao usar o comando ping fomos capazes de enviar pacotes do host AfonsoHenriques para endereço do host CondadoOnline(10.0.0.1/30), o que nos permite concluir que os routers do core da rede não possuem mais qualquer problema ou erro.

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.

No.	Time	Source	Destination	Protocol	Length	Info
13	17.127814290	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=1/256, ttl=55 (reply in 1...
14	17.127837449	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=1/256, ttl=64 (request in...
15	17.127861069	192.168.0.129	192.168.0.130	ICMP	126	Destination unreachable (Network unreachable)
17	18.142498795	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=2/512, ttl=55 (reply in 1...
18	18.142527443	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=2/512, ttl=64 (request in...
19	18.142554106	192.168.0.129	192.168.0.130	ICMP	126	Destination unreachable (Network unreachable)
20	19.166435014	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=3/768, ttl=55 (reply in 2...
21	19.166462617	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=3/768, ttl=64 (request in...
22	19.166486739	192.168.0.129	192.168.0.130	ICMP	126	Destination unreachable (Network unreachable)
24	20.190473002	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=4/1024, ttl=55 (reply in ...
25	20.190566933	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=4/1024, ttl=64 (request i...
26	20.190626111	192.168.0.129	192.168.0.130	ICMP	126	Destination unreachable (Network unreachable)
27	21.214333636	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=5/1280, ttl=55 (reply in ...
28	21.214357799	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=5/1280, ttl=64 (request i...
30	22.238495753	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=6/1536, ttl=55 (reply in ...
31	22.238542116	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=6/1536, ttl=64 (request i...
36	23.263313422	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=7/1792, ttl=55 (reply in ...
37	23.263358088	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=7/1792, ttl=64 (request i...
40	24.288075014	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=8/2048, ttl=55 (reply in ...
41	24.288085744	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=8/2048, ttl=64 (request i...
42	25.310543344	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=9/2304, ttl=55 (reply in ...
43	25.310573353	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=9/2304, ttl=64 (request i...
45	26.334344573	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=10/2560, ttl=55 (reply in...
46	26.334369155	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=10/2560, ttl=64 (request ...
47	27.358395994	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=11/2816, ttl=55 (reply in...
48	27.358419932	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=11/2816, ttl=64 (request ...
50	28.393785110	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=12/3072, ttl=55 (reply in...
51	28.393810659	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=12/3072, ttl=64 (request ...
52	29.406374682	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=13/3328, ttl=55 (reply in...
53	29.406398543	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=13/3328, ttl=64 (request i...
55	30.430830487	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=14/3584, ttl=55 (reply in...
56	30.430863702	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=14/3584, ttl=64 (request ...
57	31.454381716	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=15/3840, ttl=55 (reply in...
58	31.454407552	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=15/3840, ttl=64 (request ...
60	32.478439121	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=16/4096, ttl=55 (reply in...
61	32.478462134	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=16/4096, ttl=64 (request ...
62	33.502453259	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=17/4352, ttl=55 (reply in...
63	33.502478735	192.168.0.130	192.168.0.226	ICMP	98	Echo (ping) reply id=0x001d, seq=17/4352, ttl=64 (request ...
66	34.526358045	192.168.0.226	192.168.0.130	ICMP	98	Echo (ping) request id=0x001d, seq=18/4608, ttl=55 (reply in...

Figura 42: Prova de que o host Teresa recebeu os pacotes

Como é possível ver na figura 42, os pacotes são recebidos.

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.

Após analisar melhor a topologia, fomos capazes de detetar que o problema que impedia a existência de conectividade entre o host AfonsoHenriques e o host Teresa estava no router RAGaliza, onde, embora fosse possível enviar pacotes para o host Teresa(request), os pacotes resposta(reply) não chegavam ao host AfonsoHenriques, devido ao facto de faltar a entrada correta na tabela de encaminhamento do router RAGaliza.

```

root@RAGaliza:/tmp/pycore.33491/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
192.168.0.128    0.0.0.0        255.255.255.248 U            0 0      0     eth1
192.168.0.136    172.16.142.1   255.255.255.248 UG           0 0      0     eth0
192.168.0.144    172.16.142.1   255.255.255.248 UG           0 0      0     eth0
192.168.0.152    172.16.142.1   255.255.255.248 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

```

Figura 43: Tabela de encaminhamento do router RAGaliza(Antes)

```

root@RAGaliza:/tmp/pycore.33491/RAGaliza.conf# ip route add 192.168.0.224/29 via 172.16.142.1

```

Figura 44: Adição da rota em falta

```

root@RAGaliza:/tmp/pycore.33491/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
192.168.0.128    0.0.0.0        255.255.255.248 U            0 0      0     eth1
192.168.0.136    172.16.142.1   255.255.255.248 UG           0 0      0     eth0
192.168.0.144    172.16.142.1   255.255.255.248 UG           0 0      0     eth0
192.168.0.152    172.16.142.1   255.255.255.248 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

```

Figura 45: Tabela de encaminhamento do router RAGaliza(Depois)

O problema no router RAGaliza devia-se ao fato de não haver uma rota com a sub-rede do host AfonsoHenriques como destino. Depois de adicionar uma rota com o destino e gateway correspondentes, fomos capazes de obter conectividade entre D.Teresa e D.Afonso Henriques, assim como é mostrado nas figuras 46 e 47 abaixo.

```
root@AfonsoHenriques:/tmp/pycore.33491/AfonsoHenriques.conf# ping 192.168.0.130
PING 192.168.0.130 (192.168.0.130) 56(84) bytes of data.
64 bytes from 192.168.0.130: icmp_seq=1 ttl=55 time=0.505 ms
64 bytes from 192.168.0.130: icmp_seq=2 ttl=55 time=0.218 ms
64 bytes from 192.168.0.130: icmp_seq=3 ttl=55 time=0.234 ms
64 bytes from 192.168.0.130: icmp_seq=4 ttl=55 time=0.257 ms
64 bytes from 192.168.0.130: icmp_seq=5 ttl=55 time=0.254 ms
^C
--- 192.168.0.130 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4097ms
rtt min/avg/max/mdev = 0.218/0.293/0.505/0.106 ms
root@AfonsoHenriques:/tmp/pycore.33491/AfonsoHenriques.conf#
```

Figura 46: Envio de pacotes do host AfonsoHenriques para o host Teresa

```
root@Teresa:/tmp/pycore.33491/Teresa.conf# ping 192.168.0.226
PING 192.168.0.226 (192.168.0.226) 56(84) bytes of data.
64 bytes from 192.168.0.226: icmp_seq=1 ttl=55 time=0.269 ms
64 bytes from 192.168.0.226: icmp_seq=2 ttl=55 time=0.292 ms
64 bytes from 192.168.0.226: icmp_seq=3 ttl=55 time=0.253 ms
64 bytes from 192.168.0.226: icmp_seq=4 ttl=55 time=0.232 ms
64 bytes from 192.168.0.226: icmp_seq=5 ttl=55 time=0.236 ms
^C
--- 192.168.0.226 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4078ms
rtt min/avg/max/mdev = 0.232/0.256/0.292/0.022 ms
root@Teresa:/tmp/pycore.33491/Teresa.conf#
```

Figura 47: Envio de pacotes do host Teresa para o host AfonsoHenriques

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.

```

root@AfonsoHenriques:/tmp/pycore.33491/AfonsoHenriques.conf# traceroute 192.168.0.130
traceroute to 192.168.0.130 (192.168.0.130), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225)  0.079 ms  0.024 ms  0.020 ms
 2 172.16.143.1 (172.16.143.1)  0.053 ms  0.038 ms  0.028 ms
 3 10.0.0.29 (10.0.0.29)  0.062 ms  0.041 ms  0.038 ms
 4 10.0.0.25 (10.0.0.25)  0.071 ms  0.049 ms  0.049 ms
 5 10.0.0.13 (10.0.0.13)  0.080 ms  0.057 ms  0.086 ms
 6 10.0.0.17 (10.0.0.17)  0.101 ms  0.158 ms  0.137 ms
 7 10.0.0.5 (10.0.0.5)  0.129 ms  0.101 ms  0.097 ms
 8 10.0.0.1 (10.0.0.1)  0.148 ms  0.118 ms  0.124 ms
 9 172.16.142.2 (172.16.142.2)  0.155 ms  0.121 ms  0.103 ms
10 192.168.0.130 (192.168.0.130)  0.119 ms  0.147 ms  0.159 ms

```

Figura 48: Output do traceroute do host AfonsoHenriques para o endereço do host Teresa

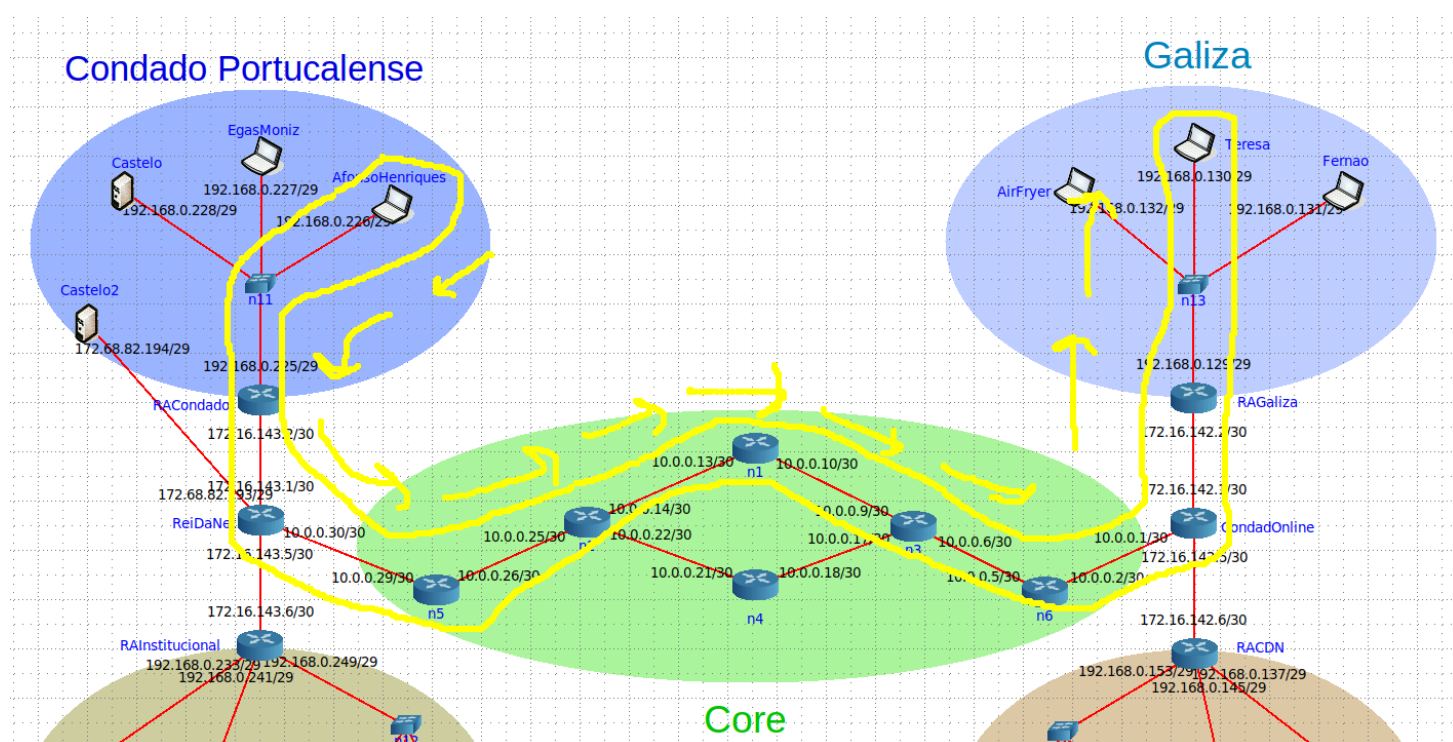


Figura 49: Rota de envio de pacotes do host AfonsoHenriques para o host Teresa

Como podemos ver nas figuras 48 e 49, os pacotes que vão do host AfonsoHenriques para o host Teresa seguem a rota:

- > Host AfonsoHenriques
- > switch n11
- > Router RACondado
- > Router ReiDaNet
- > Router n5
- > Router n2

- > Router n1
- > Router n3
- > Router n6
- > Router CondadoOnline
- > Router RAGaliza
- > switch n13
- > Host Teresa

```

root@Teresa:/tmp/pycore.33491/Teresa.conf# traceroute 192.168.0.226
traceroute to 192.168.0.226 (192.168.0.226), 30 hops max, 60 byte packets
 1 192.168.0.129 (192.168.0.129)  0.136 ms  0.031 ms  0.030 ms
 2 172.16.142.1 (172.16.142.1)  0.045 ms  0.030 ms  0.029 ms
 3 10.0.0.2 (10.0.0.2)  0.077 ms  0.079 ms  0.043 ms
 4 10.0.0.6 (10.0.0.6)  0.105 ms  0.051 ms  0.054 ms
 5 10.0.0.18 (10.0.0.18)  0.139 ms  0.072 ms  0.057 ms
 6 10.0.0.14 (10.0.0.14)  0.147 ms  0.106 ms  0.069 ms
 7 10.0.0.26 (10.0.0.26)  0.102 ms  0.078 ms  0.077 ms
 8 10.0.0.30 (10.0.0.30)  0.098 ms  0.131 ms  0.095 ms
 9 172.16.143.2 (172.16.143.2)  0.106 ms  0.098 ms  0.096 ms
10 192.168.0.226 (192.168.0.226)  0.133 ms  0.139 ms  0.126 ms

```

Figura 50: Output do traceroute do host Teresa para o endereço do host AfonsoHenriques

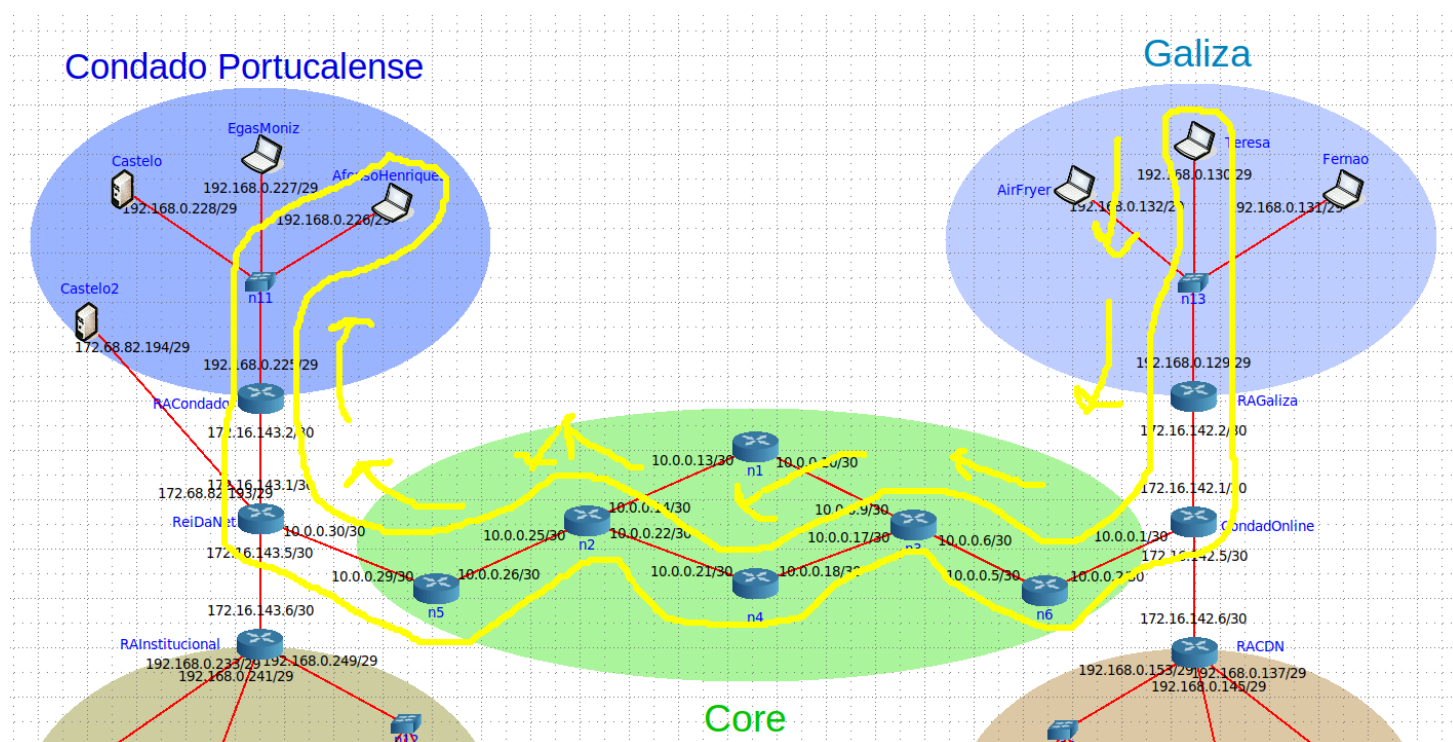


Figura 51: Rota de envio de pacotes do host Teresa para o host AfonsoHenriques

Como é possível observar nas figuras 50 e 51, os pacotes que vão do host Teresa para o host AfonsoHenriques seguem a seguinte rota;

- > Host Teresa
- > switch n13
- > Router RAGaliza
- > Router CondadoOnline
- > Router n6
- > Router n3
- > Router n4
- > Router n2
- > Router n5
- > Router ReiDaNet
- > Router RACondado
- > switch n11
- > Host AfonsoHenriques

Por fim, embora as duas rotas sejam semelhantes elas não são iguais.

e) Estando restabelecida a conectividade entre os dois hosts, obtenha a tabela de encaminhamento de n5 e foque-se na seguinte entrada:

```
ip route 192.168.0.0 255.255.255.0 10.0.0.30
```

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.

```
root@n5:/tmp/pycore.46695/n5.conf# netstat -rn
```

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
192.168.0.0	10.0.0.30	255.255.255.0	UG	0 0		0	eth0
192.168.0.128	10.0.0.25	255.255.255.248	UG	0 0		0	eth1
192.168.0.136	10.0.0.25	255.255.255.248	UG	0 0		0	eth1
192.168.0.144	10.0.0.25	255.255.255.248	UG	0 0		0	eth1
192.168.0.152	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

Figura 53: Tabela de encaminhamento do router n5

Existe uma correspondência para pacotes enviados para os polos Galiza e CDN, visto que a sub-rede 192.168.0.128/29 do polo Galiza e as sub-redes 192.168.0.152/29, 192.168.0.144/29 e 192.168.0.136/29 do CDN fazem parte da sub-rede 192.168.0.0/24.

No entanto, essa entrada não garantiria o funcionamento esperado, uma vez que ela tem como próximo dispositivo o router ReiDaNet que apenas leva ao polo Condado Portucalese e ao Institucional e, além disso, existem rotas mais específicas, como as quatro mencionadas anteriormente, visto que tem máscara /29, que seriam escolhidas ao invés da entrada dada, que tem máscara /24, visto que os routers sempre tem preferência por rotas mais específicas.

Devido aos motivos anteriormente referidos, a entrada não seria utilizada.

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

Os endereços usados nos quatro polos pertencem seguem o padrão 192.168.0.X/29, que pertence à faixa 192.168.0.0/16 que é reservada a redes privadas (RFC 1918).

Já os endereços usados nos ISPs seguem o padrão 172.16.142.X/30 e 172.16.143.X/30, que pertence à faixa 172.16.0.0/12 que também é reservada a redes privadas.

Por último, os endereços usados no core da rede seguem o padrão 10.0.0.X/30, que pertence à faixa 10.0.0.0/8 que é reservada a redes privadas (RFC 1918).

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

Os switches não têm um endereço IP atribuído, visto que são dispositivos de nível/Layer 2(LinK Layer), enquanto que o IP reside no nível 3, logo não há necessidade dos switches terem um endereço IP.

3.3. Exercício 3

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

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.

```
root@n6:/tmp/pycore.46695/n6.conf# netstat -rn
```

Destination	Gateway	Genmask	Flags	MSS Window	irrt	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
192.168.0.128	10.0.0.1	255.255.255.248	UG	0 0	0	eth0
192.168.0.136	10.0.0.1	255.255.255.248	UG	0 0	0	eth0
192.168.0.144	10.0.0.1	255.255.255.248	UG	0 0	0	eth0
192.168.0.152	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

Figura 54: Tabela de encaminhamento do router n6(Antes)

Na figura 54, somos capazes de ver a tabela de encaminhamento do router n6 antes de definirmos um esquema de sumarização de rotas.

```
root@n6:/tmp/pycore.33491/n6.conf# ip route del 192.168.0.128/29
root@n6:/tmp/pycore.33491/n6.conf# ip route del 192.168.0.136/29
root@n6:/tmp/pycore.33491/n6.conf# ip route del 192.168.0.144/29
root@n6:/tmp/pycore.33491/n6.conf# ip route del 192.168.0.152/29
```

Figura 55: Rotas para os polos Galiza e CDN removidas

Já na figura 55, é mostrado o processo de remoção de rotas da tabela de encaminhamento, que tinham como destino sub-redes pertencentes aos polos Galiza e CDN.

```
root@n6:/tmp/pycore.33491/n6.conf# ip route add 192.168.0.128/27 via 10.0.0.1
```

Figura 56: Rota para os polos Galiza e CDN adicionada usando Supernetting

Depois de remover as rotas referentes a Galiza e CDN no dispositivo n6, definimos um esquema de sumarização de rotas(Supernetting) usando uma rota menos específica no lugar das rotas originais mais específicas, assim como é possível ver na figura 56 onde adicionamos a rota com destino para a rede 192.168.0.128/27 que engloba as sub-redes 192.168.0.128/29, 192.168.0.136/29, 192.168.0.144/29 e 192.168.0.152/29, que pertencem aos polos Galiza e CDN, permitindo-nos o uso de apenas uma rota para ambos os polos.

```
root@n6:/tmp/pycore.33491/n6.conf# ip route del 172.16.142.0/30
root@n6:/tmp/pycore.33491/n6.conf# ip route del 172.16.142.4/30
root@n6:/tmp/pycore.33491/n6.conf# ip route add 172.16.142.0/29 via 10.0.0.1
```

Figura 57: Rotas alteradas nos ISPs

Além disso, também definimos um esquema de supernetting para as rotas para os routers RAGaliza e RACDN, eliminando as rotas com destino para as sub-redes 172.16.142.0/30 e 172.16.142.4/30 e adicionando a rota 172.16.142.0/29 que engloba as duas, permitindo-nos ter apenas uma rota para esses routers, como é possível observar na figura 57.

```
root@n6:/tmp/pycore.33491/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        255.255.255.248 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
192.168.0.128     10.0.0.1        255.255.255.224 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
```

Figura 58: Tabela de encaminhamento do router n6(Depois)

Na figura 58, somos capazes de ver a tabela de encaminhamento do dispositivo n6 após as alterações nas suas rotas.

```
root@AfonsoHenriques:/tmp/pycore.33491/AfonsoHenriques.conf# ping 192.168.0.130
PING 192.168.0.130 (192.168.0.130) 56(84) bytes of data.
64 bytes from 192.168.0.130: icmp_seq=1 ttl=55 time=0.283 ms
64 bytes from 192.168.0.130: icmp_seq=2 ttl=55 time=0.446 ms
64 bytes from 192.168.0.130: icmp_seq=3 ttl=55 time=0.319 ms
64 bytes from 192.168.0.130: icmp_seq=4 ttl=55 time=0.219 ms
^C
--- 192.168.0.130 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3056ms
rtt min/avg/max/mdev = 0.219/0.316/0.446/0.082 ms
root@AfonsoHenriques:/tmp/pycore.33491/AfonsoHenriques.conf# ping 192.168.0.154
PING 192.168.0.154 (192.168.0.154) 56(84) bytes of data.
64 bytes from 192.168.0.154: icmp_seq=1 ttl=55 time=0.370 ms
64 bytes from 192.168.0.154: icmp_seq=2 ttl=55 time=0.512 ms
64 bytes from 192.168.0.154: icmp_seq=3 ttl=55 time=0.253 ms
^C
--- 192.168.0.154 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2023ms
rtt min/avg/max/mdev = 0.253/0.378/0.512/0.105 ms
root@AfonsoHenriques:/tmp/pycore.33491/AfonsoHenriques.conf# ping 192.168.0.146
PING 192.168.0.146 (192.168.0.146) 56(84) bytes of data.
64 bytes from 192.168.0.146: icmp_seq=1 ttl=55 time=1.06 ms
64 bytes from 192.168.0.146: icmp_seq=2 ttl=55 time=0.256 ms
64 bytes from 192.168.0.146: icmp_seq=3 ttl=55 time=0.239 ms
^C
--- 192.168.0.146 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.239/0.518/1.061/0.383 ms
root@AfonsoHenriques:/tmp/pycore.33491/AfonsoHenriques.conf# ping 192.168.0.138
PING 192.168.0.138 (192.168.0.138) 56(84) bytes of data.
64 bytes from 192.168.0.138: icmp_seq=1 ttl=55 time=0.291 ms
64 bytes from 192.168.0.138: icmp_seq=2 ttl=55 time=0.225 ms
64 bytes from 192.168.0.138: icmp_seq=3 ttl=55 time=0.257 ms
^C
--- 192.168.0.138 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2010ms
rtt min/avg/max/mdev = 0.225/0.257/0.291/0.026 ms
```

Figura 59: Prova de conectividade com os polos Galiza e CDN

Por fim, na figura 59 podemos ver o output do comando ping quando usado para enviar pacotes do host AfonsoHenriques para vários endereços nos polos Galiza e CDN, como os hosts Teresa(192.168.0.130/29), Panda(192.168.0.154/29), Twitch(192.168.0.146/29) e CondadoGram(192.168.0.138/29), onde é possível observar que os pacotes são recebidos, o que nos permite concluir que a conectividade entre o host AfonsoHenriques e os polos Galiza e CDN é mantida.

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


```
root@n6:/tmp/pycore.33491/n6.conf# ip route del 192.168.0.224/29
root@n6:/tmp/pycore.33491/n6.conf# ip route del 192.168.0.232/29
root@n6:/tmp/pycore.33491/n6.conf# ip route del 192.168.0.240/29
root@n6:/tmp/pycore.33491/n6.conf# ip route del 192.168.0.248/29
```

Figura 60: Rotas para os polos Condado Portucalense e Institucional removidas

Assim como feito na alínea a), na figura 60 é mostrado o processo de remoção de rotas da tabela de encaminhamento, que tinham como destino sub-redes pertencentes aos polos CondadoPortucalense e Institucional.

```
root@n6:/tmp/pycore.33491/n6.conf# ip route add 192.168.0.224/27 via 10.0.0.6
```

Figura 61: Rota para os polos Condado Portucalense e Institucional adicionada usando Supernetting

Apois remover as rotas referentes ao CondadoPortucalense e Institucional no dispositivo n6, definimos um esquema de sumarização de rotas(Supernetting) usando uma rota menos específica no lugar das rotas originais mais específicas, assim como é possível ver na figura 61 onde adicionamos a rota com destino para a rede 192.168.0.224/27 que engloba as sub-redes 192.168.0.224/29, 192.168.0.232/29, 192.168.0.240/29 e 192.168.0.248/29, que pertencem aos polos CondadoPortucalense e Institucional, permitindo-nos o uso de apenas uma rota para ambos os polos.

```
root@n6:/tmp/pycore.33491/n6.conf# ip route del 172.16.143.0/30
root@n6:/tmp/pycore.33491/n6.conf# ip route del 172.16.143.4/30
root@n6:/tmp/pycore.33491/n6.conf# ip route add 172.16.143.0/29 via 10.0.0.6
```

Figura 62: Rotas alteradas nos ISPs

Tal e qual fizemos na alínea a), também definimos um esquema de sumarização de rotas (supernetting) para as rotas para os routers RACondado e RAIstitucional, eliminando as rotas com destino para as sub-redes 172.16.143.0/30 e 172.16.143.4/30 e adicionando a rota 172.16.143.0/29 que engloba as duas, permitindo-nos ter apenas uma rota para esses routers, como é possível observar na figura 62.

```
root@n6:/tmp/pycore.33491/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         255.255.255.248 UG       0 0        0 eth0
172.16.143.0      10.0.0.6         255.255.255.248 UG       0 0        0 eth1
192.168.0.128     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
```

Figura 63: Tabela de encaminhamento do router n6(Fi-
nal)

Na figura 63, somos mais uma vez capazes de ver a tabela de encaminhamento do dispositivo n6 após as alterações nas suas rotas.

```
root@Teresa:/tmp/pycore.33491/Teresa.conf# ping 192.168.0.226
PING 192.168.0.226 (192.168.0.226) 56(84) bytes of data.
64 bytes from 192.168.0.226: icmp_seq=1 ttl=55 time=0.373 ms
64 bytes from 192.168.0.226: icmp_seq=2 ttl=55 time=0.257 ms
64 bytes from 192.168.0.226: icmp_seq=3 ttl=55 time=0.266 ms
64 bytes from 192.168.0.226: icmp_seq=4 ttl=55 time=0.257 ms
^C
--- 192.168.0.226 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.257/0.288/0.373/0.049 ms
root@Teresa:/tmp/pycore.33491/Teresa.conf# ping 192.168.0.234
PING 192.168.0.234 (192.168.0.234) 56(84) bytes of data.
64 bytes from 192.168.0.234: icmp_seq=1 ttl=55 time=0.367 ms
64 bytes from 192.168.0.234: icmp_seq=2 ttl=55 time=0.230 ms
64 bytes from 192.168.0.234: icmp_seq=3 ttl=55 time=0.238 ms
^C
--- 192.168.0.234 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2050ms
rtt min/avg/max/mdev = 0.230/0.278/0.367/0.062 ms
root@Teresa:/tmp/pycore.33491/Teresa.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=55 time=0.301 ms
64 bytes from 192.168.0.242: icmp_seq=2 ttl=55 time=0.209 ms
64 bytes from 192.168.0.242: icmp_seq=3 ttl=55 time=0.211 ms
^C
--- 192.168.0.242 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2011ms
rtt min/avg/max/mdev = 0.209/0.240/0.301/0.042 ms
root@Teresa:/tmp/pycore.33491/Teresa.conf# ping 192.168.0.252
PING 192.168.0.252 (192.168.0.252) 56(84) bytes of data.
64 bytes from 192.168.0.252: icmp_seq=1 ttl=55 time=0.505 ms
64 bytes from 192.168.0.252: icmp_seq=2 ttl=55 time=0.366 ms
64 bytes from 192.168.0.252: icmp_seq=3 ttl=55 time=0.256 ms
^C
--- 192.168.0.252 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2022ms
rtt min/avg/max/mdev = 0.256/0.375/0.505/0.101 ms
```

Figura 64: Prova de conetividade com os polos Condado Portucalense e Institucional

Por fim, na figura 64 podemos ver o output do comando ping quando usado para enviar pacotes do host Teresa para vários endereços nos polos CondadoPortucalense e Institucional, como os hosts AfonsoHneriques(192.168.0.226/29), Uminho(192.168.0.234/29), DI(192.168.0.242/29) e Bombeiros(192.168.0.252/29), onde é possível observar que os pacotes são recebidos, o que nos permite concluir que a conetividade entre o host Teresa e os polos CondadoPortucalense e Institucional é mantida.

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

Supernetting é uma técnica que permite a sumarização de múltiplas redes IP menores, com endereços IP contíguos, em uma rede maior. Ao fazer isso, este método simplifica

o processo de roteamento reduzindo o número de entradas individuais nas tabelas de encaminhamento, o que resulta numa maior eficiência no roteamento. No entanto, além de **aspetos positivos**, Supernetting também possui **aspetos negativos**.

Aspetos Positivos

- **Redução da Tabela de Encaminhamento:** Supernetting reduz significativamente o tamanho da tabela de encaminhamento, já que múltiplos blocos de endereços são representados por uma única entrada na tabela. Isso melhora o desempenho dos routers, reduzindo a sobrecarga de processamento e os recursos de memória necessários nos routers para manter as tabelas de encaminhamento.
- **Uso mais eficiente dos endereços IP:** Ao agregar várias sub-redes em uma única super-rede, evita-se a fragmentação do espaço de endereçamento, otimizando o uso dos endereços disponíveis.
- **Diminuição da quantidade de atualizações de roteamento:** Como menos rotas precisam ser anunciadas entre os routers, há menos atualizações de tabelas de encaminhamento, o que simplifica a sua gestão ao reduzir a complexidade das configurações de roteamento.

Aspetos Negativos

- **Complexidade na Configuração:** Configurar e manter redes supernetting pode ser mais complexo do que o roteamento tradicional, especialmente para redes grandes e complexas. Requer um planejamento cuidadoso para evitar problemas de roteamento e garantir que as rotas sejam agregadas corretamente.
- **Risco de Roteamento Ineficiente:** Se as rotas forem agregadas de forma inadequada, isso pode resultar em roteamento ineficiente ou problemas de conectividade. Por exemplo, se um único bloco de endereços for agregado a um bloco maior, isso pode causar problemas de roteamento para redes menores dentro do bloco original.
- **Dificuldade na Escalabilidade:** Embora o método de supernetting seja eficaz em reduzir a tabela de encaminhamento, pode ser difícil escalar a implementação em redes muito grandes ou em constante mudança. À medida que a rede cresce, torna-se cada vez mais difícil manter a agregação de rotas de forma eficiente e precisa.
- **Pode levar a desperdício de endereços em alguns casos:** Se uma organização agregar muitas redes pequenas em uma única super-rede maior, pode acabar reservando mais endereços do que realmente precisa, resultando em desperdício.

4. Conclusão

Neste trabalho, exploramos e abordamos diversos conceitos essenciais em redes de computadores, tais como o estudo do formato de um pacote ou datagrama IP, a fragmentação de pacotes IP, o endereçamento IP e por último o encaminhamento IP. A execução dos exercícios, no decorrer da realização do projeto, permitiu-nos aprender e compreender o comportamento de pacotes durante o seu envio através da utilização de comandos como o traceroute ou o ping, assim como do software wireshark. Além disso, este projeto também nos deu a conhecer procedimentos a tomar na avaliação do funcionamento de uma rede local e na análise de tabelas de encaminhamento de forma a poder identificar erros e aplicar possíveis correções, assim como a usar o método de sumarização de rotas(Supernetting) e conhecer alguns dos seus aspetos positivos e negativos.

Concluindo, a partir da realização deste trabalho, fomos capazes de aprender mais sobre o IPv4 e as suas principais vertentes, o que nos permitiu ter uma pequena ideia de como funciona esta área na realidade.