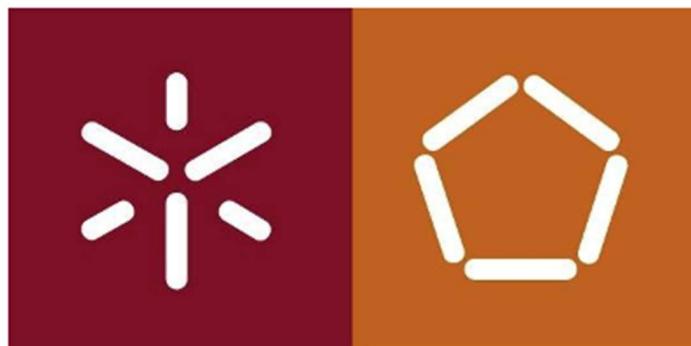
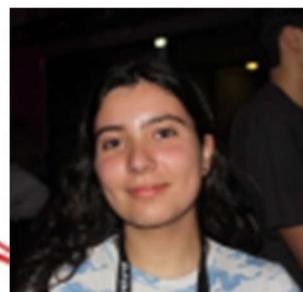


Universidade do Minho

Licenciatura em Engenharia Informática



TP2 - Protocolo IPv4 Datagramas IP e Fragmentação



Trabalho realizado por:
Eduarda Mafalda Martins Vieira, A104098
João Pedro Ferreira e Ferreira, A104539
Maria Leonor Carvalho da Cunha, A103997

Redes de Computadores

PL5 - Grupo 3
março de 2025

Parte 1.....	3
Exercício 1	3
Exercício 2	5
Exercício 3	9
Parte 2.....	15
Exercício 1	15
Exercício 2	19
Exercício 3	32
Conclusão.....	36

Parte 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. Documente e justifique todas as respostas às seguintes alíneas:

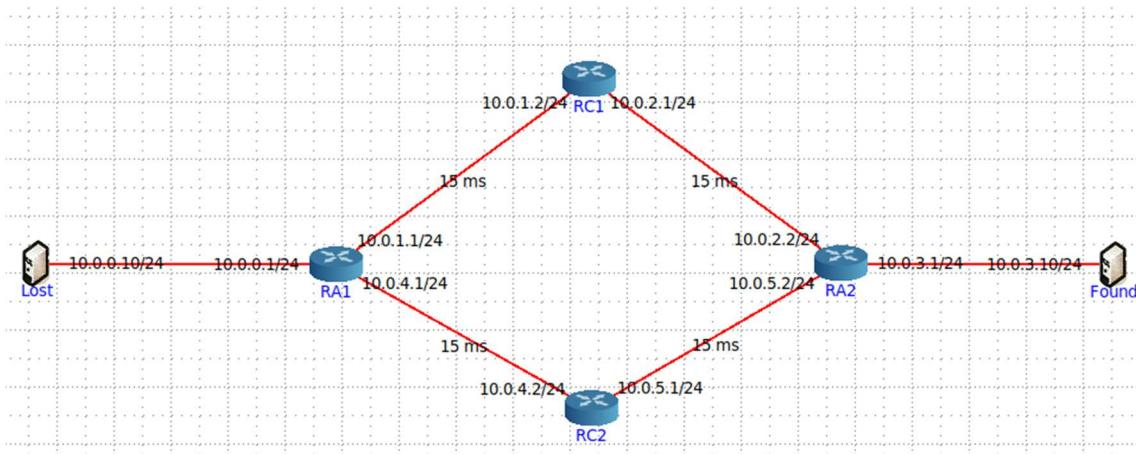


Figura 1- Topologia CORE

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

Resposta:

```
root@Lost:/tmp/pycore.36997/Lost.conf# traceroute -I 10.0.3.10
traceroute to 10.0.3.10 (10.0.3.10), 30 hops max, 60 byte packets
1 10.0.0.1 (10.0.0.1) 0.069 ms 0.011 ms 0.010 ms
2 10.0.1.2 (10.0.1.2) 30.465 ms 30.445 ms 30.438 ms
3 10.0.2.2 (10.0.2.2) 91.748 ms 91.744 ms 91.741 ms
4 10.0.3.10 (10.0.3.10) 91.735 ms 91.731 ms 91.726 ms
root@Lost:/tmp/pycore.36997/Lost.conf#
```

Figura 2 – Output do comando traceroute -I IP Found

No.	Time	Source	Destination	Protocol	Length	Info
26	23.725709390	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=1/256, ttl=1 (no response found!)
27	23.725719557	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
28	23.725728848	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=2/512, ttl=1 (no response found!)
29	23.725735613	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
30	23.725743135	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=3/768, ttl=1 (no response found!)
31	23.725748261	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
32	23.725754815	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=4/1024, ttl=2 (no response found!)
33	23.725777765	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=5/1280, ttl=2 (no response found!)
34	23.725785732	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=6/1536, ttl=2 (no response found!)
35	23.725793461	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=7/1792, ttl=3 (no response found!)
36	23.725799935	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=8/2048, ttl=3 (no response found!)
37	23.725806383	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=9/2304, ttl=4 (no response found!)
38	23.725814588	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=10/2560, ttl=4 (request in 51)
39	23.725821112	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=11/2816, ttl=4 (reply in 58)
40	23.725828907	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=12/3072, ttl=4 (reply in 59)
41	23.725835384	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=13/3328, ttl=5 (reply in 60)
42	23.725842179	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=14/3584, ttl=5 (reply in 61)
43	23.725848924	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=15/3840, ttl=5 (reply in 62)
44	23.725856259	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=16/4096, ttl=6 (reply in 63)
45	23.727062778	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=17/4352, ttl=6 (reply in 64)
46	23.7270915478	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=18/4608, ttl=6 (reply in 65)
47	23.7270923677	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=19/4864, ttl=7 (reply in 66)
48	23.7562194993	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
49	23.7562192123	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
50	23.7562268181	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
51	23.756557333	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=29/5120, ttl=7 (reply in 67)
52	23.756572746	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=21/5376, ttl=7 (reply in 68)
53	23.756583065	10.0.0.10	10.0.3.10	ICMP	74	Echo (ping) request id=0x0021, seq=22/5632, ttl=8 (reply in 69)
54	23.817545838	10.0.2.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
55	23.817546471	10.0.2.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
56	23.817543327	10.0.2.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
57	23.817545833	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=10/2560, ttl=61 (request in 38)
58	23.817548348	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=11/2816, ttl=61 (request in 39)
59	23.817559834	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=12/3072, ttl=61 (request in 40)
60	23.817553398	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=13/3328, ttl=61 (request in 41)
61	23.817555795	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=14/3584, ttl=61 (request in 42)
62	23.817558270	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=15/3840, ttl=61 (request in 43)
63	23.817560858	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=16/4096, ttl=61 (request in 44)
64	23.817563331	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=17/4352, ttl=61 (request in 45)
65	23.817565887	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=18/4608, ttl=61 (request in 46)
66	23.8175658282	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=19/4864, ttl=61 (request in 47)
67	23.817570799	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=20/5120, ttl=61 (request in 51)
68	23.817573283	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=21/5376, ttl=61 (request in 52)
69	23.817575769	10.0.3.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0021, seq=22/5632, ttl=61 (request in 53)

Figura 3 - o tráfego ICMP enviado pelo sistema Lost e o tráfego ICMP recebido como resposta

O computador host Lost envia 3 pacotes através do protocolo ICMP, com um TTL (Time to Leave) crescente, começando em 1 até receber uma resposta do host Found.

- 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.

Resposta: A fim de alcançar o servidor Found, é essencial que o TTL mínimo seja estabelecido em 4. A análise do tráfego ICMP mencionado anteriormente confirma essa informação, uma vez que, ao observar a linha 38, é possível perceber que Lost só recebe uma resposta de Found quando o TTL atinge o valor de 4.

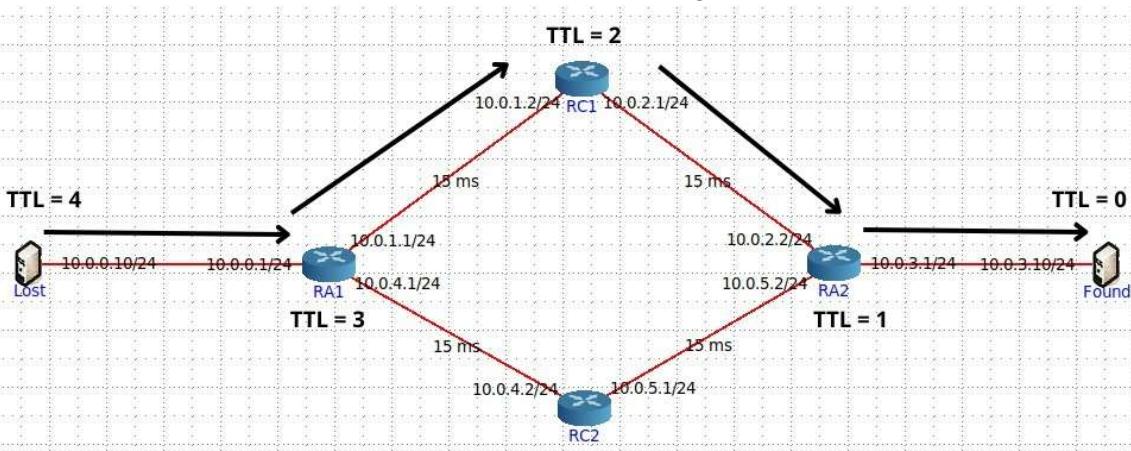


Figura 4 – Esquema com os valores do campo TTL

- 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.

Resposta: Para calcular o RTT fazemos $91.735 + 91.731 + 91.726 = 275,192$. E dividimos 275,192 por 3 (pacotes), resultando em aproximadamente 91,73067 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?

Resposta: Não poderia ser calculado dividindo o RTT por dois, dado que o tempo que demora a ir da origem ao destino, pode não ser (e provavelmente não é) igual ao tempo que demora do destino à origem. O cálculo desta métrica é difícil, porque seria necessário algum recurso contido nos pacotes, de maneira a saber o tempo de chegada a cada router.

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).

Documente e justifique todas as respostas às seguintes alíneas:

```
Tracing route to marco.uminho.pt [193.136.9.240]
over a maximum of 30 hops:

 1      9 ms      1 ms      1 ms  172.26.254.254
 2      5 ms      1 ms      1 ms  172.16.2.1
 3      3 ms      3 ms      1 ms  172.16.115.252
 4     12 ms      2 ms      1 ms  marco.uminho.pt [193.136.9.240]

Trace complete.
```

Figura 5 – Output do comando *traceroute* marco.uminho.pt

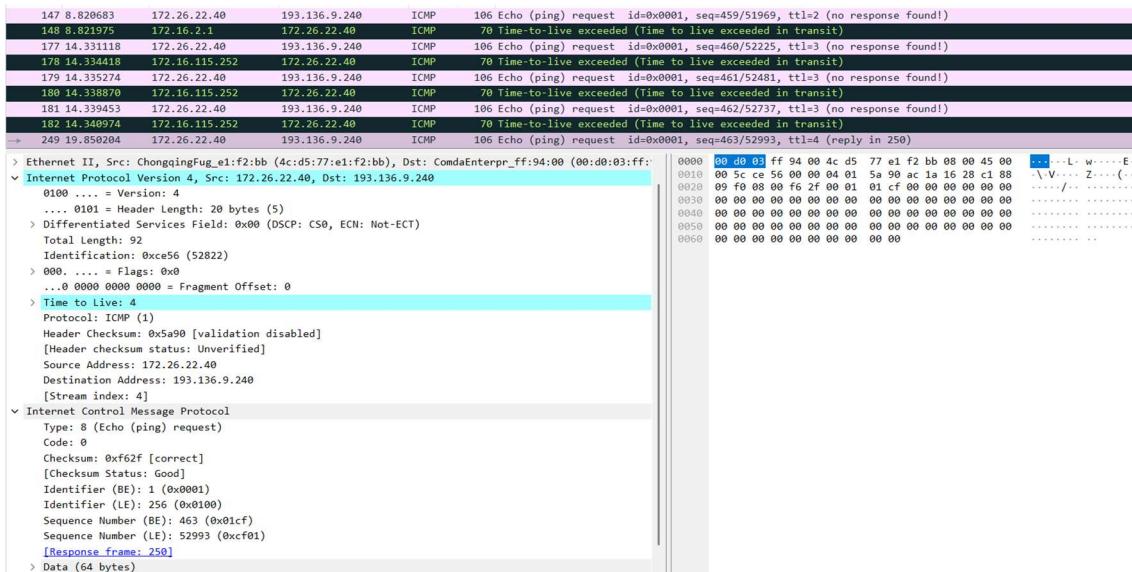


Figura 6- Tab do wireshark correspondente à primeira mensagem capturada

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

Resposta: 172.26.22.40

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

Resposta: O valor do campo protocol, ICMP (Internet Control Message Protocol), pode ser identificado com um valor de campo igual a 1, conforme podemos verificar na figura 6.

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

Resposta: O número de bytes do campo de dados do datagrama é a diferença entre o tamanho total (Total Length) e o tamanho do cabeçalho IPv4 (Header Length). Utilizando os dados que recolhemos, podemos ver na figura 6, o tamanho total corresponde a 92 bytes e o tamanho do cabeçalho a 20 bytes. O campo de dados do datagrama possui, efetuando a diferença, 72 bytes.

d. O datagrama IP foi fragmentado? Justifique.

Resposta: É possível verificar que o IP foi fragmentado observando o valor das Flags e do Fragment Offset. Como ambos os valores se encontram a 0, podemos inferir que o datagrama IP não foi fragmentado. (ver figura 6)

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.

18	2.883609	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=454/50689, ttl=1 (no response found!)
20	2.893333	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=455/50945, ttl=1 (no response found!)
22	2.895135	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=456/51201, ttl=1 (no response found!)
143	8.812433	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=457/51457, ttl=2 (no response found!)
145	8.818937	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=458/51713, ttl=2 (no response found!)
147	8.820683	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=459/51969, ttl=2 (no response found!)
177	14.331118	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=460/52225, ttl=3 (no response found!)
179	14.335274	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=461/52481, ttl=3 (no response found!)
181	14.339453	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=462/52737, ttl=3 (no response found!)
249	19.850204	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=463/52993, ttl=4 (reply in 250)
251	19.863764	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=464/53249, ttl=4 (reply in 252)
253	19.866462	172.26.22.40	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=465/53505, ttl=4 (reply in 254)

Figura 7 – Pacotes gerados a partir do endereço IP atribuído à interface da máquina que utilizamos

```
> Frame 18: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface \Device\NPF_{E284B
> Ethernet II, Src: ChongqingFug_e1:f2:bb (4c:d5:77:e1:f2:bb), Dst: ComdaEnterpr_ff:94:00 (00:d0:03:ff:ff:ff)
└ Internet Protocol Version 4, Src: 172.26.22.40, 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: 92
        Identification: 0xce4d (52813)
    > 000. .... = Flags: 0x0
        ...0 0000 0000 0000 = Fragment Offset: 0
    > Time to Live: 1
        Protocol: ICMP (1)
        Header Checksum: 0x5d99 [validation disabled]
        [Header checksum status: Unverified]
        Source Address: 172.26.22.40
        Destination Address: 193.136.9.240
        [Stream index: 4]
    < Internet Control Message Protocol
```

Figura 8 – Tab do wireshark pacote 1

Resposta: Conforme observado nas figuras 7 e 8, os campos do IPv4 que variam de pacote para pacote são o TTL, a *Identification* e o *Header Checksum*.

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

Resposta: Observando os vários pacotes, podemos concluir que o TTL é incrementado a cada 3 pacotes, a identificação (“Identification”) é incrementada a cada 1 pacote, ao contrário da “Header Checksum” que é decrementada a cada 1 pacote.

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
19	2.892754	172.26.254.254	172.26.22.40	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
21	2.894627	172.26.254.254	172.26.22.40	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
23	2.896543	172.26.254.254	172.26.22.40	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
144	8.818191	172.16.2.1	172.26.22.40	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
146	8.820194	172.16.2.1	172.26.22.40	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
148	8.821975	172.16.2.1	172.26.22.40	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
178	14.334418	172.16.115.252	172.26.22.40	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
180	14.338870	172.16.115.252	172.26.22.40	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
182	14.340974	172.16.115.252	172.26.22.40	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
250	19.862937	193.136.9.240	172.26.22.40	ICMP	106	Echo (ping) reply id=0x0001, seq=463/52993, ttl=61 (request in 249)
252	19.865724	193.136.9.240	172.26.22.40	ICMP	106	Echo (ping) reply id=0x0001, seq=464/53249, ttl=61 (request in 251)
254	19.868123	193.136.9.240	172.26.22.40	ICMP	106	Echo (ping) reply id=0x0001, seq=465/53505, ttl=61 (request in 253)
18	2.883609	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=454/50689, ttl=1 (no response found!)
20	2.893333	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=455/50945, ttl=1 (no response found!)
22	2.895135	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=456/51201, ttl=1 (no response found!)
143	8.812433	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=457/51457, ttl=2 (no response found!)
145	8.818937	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=458/51713, ttl=2 (no response found!)
147	8.820683	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=459/51969, ttl=2 (no response found!)
177	14.331118	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=460/52225, ttl=3 (no response found!)
179	14.335274	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=461/52481, ttl=3 (no response found!)
181	14.339453	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=462/52737, ttl=3 (no response found!)
249	19.850204	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=463/52993, ttl=4 (reply in 250)
251	19.863764	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=464/53249, ttl=4 (reply in 252)
253	19.866462	172.26.22.40	193.136.9.240	ICMP	106	Echo (ping) request id=0x0001, seq=465/53505, ttl=4 (reply in 254)

Figura 9 - Tráfego ICMP ordenado pela coluna do destino

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

Resposta:

Na imagem, observamos várias mensagens ICMP TTL originadas de diferentes roteadores intermediários, e o valor do campo TTL de request recebido é 61. O valor do TTL nesses pacotes reflete o número de saltos percorridos até o momento em que o pacote foi descartado. Esse valor não é constante para todas as respostas ICMP TTL, pois cada uma delas é gerada por um roteador diferente ao longo do caminho até o destino.

Isso ocorre porque, à medida que um pacote trafega pela rede, seu TTL é reduzido a cada salto. Quando esse valor atinge zero, o roteador responsável descarta o pacote e envia uma mensagem de erro ICMP de volta ao remetente. Como cada resposta ICMP TTL corresponde a um roteador distinto, os valores do TTL variam de acordo com a rota e o número de saltos envolvidos no trajeto do pacote.

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

Resposta: As mensagens de respostas ICMP "TTL Exceeded" são originadas com um valor de TTL inicialmente alto na fonte. Esta abordagem pretende garantir que a mensagem alcance o seu destino, independentemente de eventuais alterações na rota de rede desde o envio do pacote original.

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

Resposta: A incorporação das informações do protocolo ICMP no cabeçalho IPv4 traria tanto vantagens quanto desafios que precisam ser considerados. O ICMP é essencial para relatar erros e fornecer informações operacionais sobre a entrega de pacotes, mas sua separação do IPv4 levanta a questão de se a integração poderia melhorar a eficiência da rede ou introduzir complicações adicionais.

Uma das principais vantagens dessa integração seria a simplificação do processamento nos roteadores, eliminando a necessidade de interpretar um cabeçalho ICMP separado. Isso reduziria a carga de trabalho dos dispositivos e tornaria a comunicação de erros mais eficiente. Além disso, ao remover o cabeçalho ICMP, haveria uma redução no tamanho dos pacotes, otimizando a largura de banda. Com as informações do ICMP disponíveis diretamente no IPv4, os roteadores poderiam agir mais rapidamente sobre mensagens de erro, melhorando o desempenho da rede.

No entanto, essa mudança também traria desvantagens significativas. O cabeçalho IPv4, que já é complexo, se tornaria ainda mais extenso e difícil de processar. Além disso, haveria problemas de compatibilidade, pois dispositivos que não suportassem a nova estrutura poderiam ter dificuldades de comunicação, levando a fragmentação da rede. A flexibilidade do ICMP também seria afetada, já que ele é utilizado para

diversos fins além da sinalização de erros, como diagnósticos e notificações de congestionamento. Outra preocupação seria a segurança: o ICMP é frequentemente explorado em ataques de rede, e sua incorporação ao IPv4 poderia dificultar a mitigação dessas ameaças.

Diante desses fatores, embora a integração pudesse trazer ganhos em eficiência e economia de espaço, as desvantagens relacionadas à complexidade, compatibilidade e segurança tornam essa abordagem pouco viável. O modelo atual, que mantém ICMP e IPv4 separados, continua sendo a solução mais equilibrada e funcional para a infraestrutura da Internet.

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, acesse a Edit -> Preferences -> Protocols e em IPv4 desative a opção “Reassemble fragmented IPv4 datagrams”.

Número de bytes do pacote: 3853.

```
PS C:\Users\Leonor Cunha> ping -l 3853 marco.uminho.pt

Pinging marco.uminho.pt [193.136.9.240] with 3853 bytes of data:
Reply from 193.136.9.240: bytes=3853 time=7ms TTL=61
Reply from 193.136.9.240: bytes=3853 time=3ms TTL=61
Reply from 193.136.9.240: bytes=3853 time=3ms TTL=61
Reply from 193.136.9.240: bytes=3853 time=3ms TTL=61

Ping statistics for 193.136.9.240:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 3ms, Maximum = 7ms, Average = 4ms
```

Figura 10 – Output comando ping -l 3853 marco.uminho.pt

21	2.706995	104.18.41.33	172.26.22.40	TCP	66 443 → 51956 [ACK] Seq=1 Ack=2 Win=16 Len=0 SRE=1
22	2.715367	142.250.200.104	172.26.22.40	TCP	66 443 → 51958 [ACK] Seq=1 Ack=2 Win=1045 Len=0 SRE=1
23	2.719306	35.186.224.41	172.26.22.40	TCP	60 443 → 50588 [ACK] Seq=1 Ack=29 Win=1050 Len=0
24	2.735177	35.186.224.41	172.26.22.40	TLSv1.2	78 Application Data
•	25 2.766370	172.26.22.40	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ce90) [Reassembled in #27]
•	26 2.766370	172.26.22.40	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=ce90) [Reassembled in #27]
→	27 2.766370	172.26.22.40	193.136.9.240	ICMP	935 Echo (ping) request id=0x0001, seq=529/4354, ttl=128 (reply in 30)
28	2.773257	193.136.9.240	172.26.22.40	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ee2c) [Reassembled in #30]
29	2.773257	193.136.9.240	172.26.22.40	IPv4	935 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ee2c) [Reassembled in #30]
←	30 2.773257	193.136.9.240	172.26.22.40	ICMP	1514 Echo (ping) reply id=0x0001, seq=529/4354, ttl=61 (request in 27)
31	2.789460	172.26.22.40	35.186.224.41	TCP	54 50588 → 443 [ACK] Seq=29 Ack=25 Win=255 Len=0
32	2.865478	172.26.22.40	13.107.21.239	TCP	55 51960 → 443 [ACK] Seq=3 Ack=1 Win=255 Len=1
33	2.871816	13.107.21.239	172.26.22.40	TCP	66 443 → 51960 [ACK] Seq=1 Ack=2 Win=16384 Len=0 SRE=1

Figura 11 – Tráfego gerado, pacotes ICMP

A nossa primeira mensagem não é ICMP mas é IPv4.

```

> Frame 27: 935 bytes on wire (7480 bits), 935 bytes captured (7480 bits) on interface \Device\NPF_{E284B4
> Ethernet II, Src: ChongqingFug_e1:f2:bb (4c:d5:77:e1:f2:bb), Dst: ComdaEnterpr_ff:94:00 (00:d0:03:ff:94:
< Internet Protocol Version 4, Src: 172.26.22.40, 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: 921
        Identification: 0xce90 (52880)
    > 000. .... = Flags: 0x0
        ...0 0001 0111 0010 = Fragment Offset: 2960
        Time to Live: 128
        Protocol: ICMP (1)
        Header Checksum: 0xd9a6 [validation disabled]
        [Header checksum status: Unverified]
        Source Address: 172.26.22.40
        Destination Address: 193.136.9.240
    > [3 IPv4 Fragments (3861 bytes): #25(1480), #26(1480), #27(901)]
        [Stream index: 7]
< Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xf059 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 529 (0x0211)
    Sequence Number (LE): 4354 (0x1102)
    [Response frame: 30]
    > Data (3853 bytes)

```

Figura 12 – Print da última mensagem (ICMP)

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

Resposta: No nosso caso a primeira mensagem será IPv4 e não ICMP (frame 25) mas houve necessidade de fragmentar o pacote inicial devido ao tamanho do datagrama exceder o limite máximos de bytes permitidos para envio em uma única transmissão. O MTU que estamos a usar só é capaz de enviar pacotes com tamanho de 1500 bytes e o nosso tem 3853 bytes, surgindo a necessidade de fragmentar.

- 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?

```

> Frame 25: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{E284B4
> Ethernet II, Src: ChongqingFug_e1:f2:bb (4c:d5:77:e1:f2:bb), Dst: ComdaEnterpr_ff:94:00 (00:d0:03:ff:94:00)
< Internet Protocol Version 4, Src: 172.26.22.40, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        Total Length: 1500
        Identification: 0xce90 (52880)
    < 001. .... = Flags: 0x1, More fragments
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..1. .... = More fragments: Set
        ...0 0000 0000 0000 = Fragment Offset: 0
        Time to Live: 128
        Protocol: ICMP (1)
        Header Checksum: 0xb8d5 [validation disabled]
        [Header checksum status: Unverified]
        Source Address: 172.26.22.40
        Destination Address: 193.136.9.240
        [Reassembled IPv4 in frame: 27]
        [Stream index: 7]
    < Data (1480 bytes)
        Data [...]: 0800f059000102116162636465666768696a6b6c6d6e6f70717273747576776162636465666768696a6b6c6d6e6f707
        [Length: 1480]

```

Figura 13 – Primeiro fragmento do datagrama IP

Resposta: A informação contida no campo *Flags* permite-nos chegar a conclusão de que o pacote foi fragmentado, o *Fragment Offset* está a 0 e *More Fragments* está a 1, também indicando que se trata do primeiro fragmento. O tamanho do datagrama é 1500 bytes.

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

```

> Frame 26: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{E284B4
> Ethernet II, Src: ChongqingFug_e1:f2:bb (4c:d5:77:e1:f2:bb), Dst: ComdaEnterpr_ff:94:00 (00:d0:03:ff:94:00)
< Internet Protocol Version 4, Src: 172.26.22.40, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        Total Length: 1500
        Identification: 0xce90 (52880)
    < 001. .... = Flags: 0x1, More fragments
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..1. .... = More fragments: Set
        ...0 0000 1011 1001 = Fragment Offset: 1480
        Time to Live: 128
        Protocol: ICMP (1)
        Header Checksum: 0xb81c [validation disabled]
        [Header checksum status: Unverified]
        Source Address: 172.26.22.40
        Destination Address: 193.136.9.240
        [Reassembled IPv4 in frame: 27]
        [Stream index: 7]
    < Data (1480 bytes)
        Data [...]: 6162636465666768696a6b6c6d6e6f70717273747576776162636465666768696a6b6c6d6e6f7071727374757677616
        [Length: 1480]

```

Figura 14 – Segundo fragmento do datagrama IP segmentado

Resposta: À semelhança da questão anterior, observando o campo *Flags* podemos responder a estas perguntas. Neste caso o valor de *More Fragments* é 1, o que indica que há mais fragmentos ainda, no entanto, o valor de *Fragments Offset* é 1480, ou seja, não é o primeiro fragmento.

- 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.

Resposta: Foram criados 3 fragmentos a partir do datagrama original (sabendo que o que identificamos agora é o último e contamos anteriormente mais dois).

De forma a identificar o último fragmento de um datagrama IP, utilizamos o campo *More fragments*. Quando esse campo é igual a 0 (`ip.flags.mf == 0`), significa que o fragmento é o último ou único. A fim de filtrar apenas o último fragmento do primeiro datagrama segmentado, é necessário adicionar uma condição ao filtro anterior. Além de `ip.flags.mf == 0`, também devemos incluir `ip.id == 0xce90`, onde 0xce90 é o identificador do primeiro datagrama IP segmentado.

ip.flags.mf == 0 && ip.id == 0xce90						
No.	Time	Source	Destination	Protocol	Length	Info
→ 27	2.766370	172.26.22.40	193.136.9.240	ICMP	935	Echo (ping) request id=0x0001, seq=529/4354, ttl=128 (reply in 30)

Figura 15 – Último fragmento do primeiro datagrama IP segmentado

- 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.

Resposta: Os campos que mudam no cabeçalho IP entre os diferentes fragmentos são *Fragment Offset* e o *More Fragments*. A flag *Fragment Offset* informa a ordem em que os fragmentos devem ser organizados, uma vez que aparecem em ordem crescente, enquanto a flag *More Fragments* permite verificar se o datagrama original possui mais fragmentos. Ao combinar essas duas flags, é possível reconstruir o datagrama original.

- 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.

Resposta: O MTU que estamos a usar só permite enviar pacotes com o tamanho máximo de 1500 bytes, para além disso 20 bytes fazem parte do cabeçalho IP (*Header Length*), então, só serão enviados 1480 bytes de dados por fragmento ($1500 - 20 = 1480$). Neste caso em específico precisamos de enviar 3853 bytes, sendo necessário fragmentar a nossa mensagem em 3 (uma vez que $3853 / 1480 = 2.60$, arredondamos para 3 pois o número de fragmentos deve ser um número inteiro). Para determinar o número de bytes no último fragmento subtraímos a 3853 duas vezes 1480, o que resulta em 893 bytes.

Concluindo, em teoria, o datagrama é fragmentado em 3 fragmentos sendo transportados 1480 bytes no primeiro e segundo fragmento e 893 bytes no terceiro e último fragmento.

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.

Resposta: O motivo pelo qual somente o primeiro fragmento de cada pacote é reconhecido como um pacote ICMP é porque ele é o único que contém o cabeçalho do protocolo da camada superior, enquanto os outros fragmentos contêm apenas dados. Neste caso, como podemos verificar nas figuras 11 e 12, aconteceu-nos o contrário e apenas o último fragmento é reconhecido como ICMP, pelo que o cabeçalho do protocolo da camada superior foi enviado no último fragmento.

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?

Resposta: O tamanho do datagrama está diretamente ligado à MTU da rede. Quando o datagrama excede o limite de tamanho, é preciso dividi-lo em partes menores para que possa ser transmitido. Alterar o valor máximo da MTU pode ter diferentes impactos na rede.

Aumentar a MTU resulta em uma rede mais eficiente, pois permite transportar mais dados em cada pacote, reduzindo a necessidade de fragmentação para transmitir uma determinada quantidade de dados. No entanto, esse aumento pode causar problemas de desempenho e confiabilidade, devido à possível sobrecarga nos routers ou à necessidade de retransmitir fragmentos perdidos.

Por outro lado, reduzir o tamanho da MTU causaria uma diminuição na eficiência, já que seria necessário fazer mais fragmentações para transmitir um mesmo datagrama. No entanto, essa redução poderia melhorar a confiabilidade, pois haveria menos risco de perda de dados e menor possibilidade de congestionamento devido a transferências grandes.

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.

Resposta: O tamanho máximo para evitar a fragmentação do pacote é de 1472 bytes. Esse valor resulta da capacidade máxima do MTU, que é de 1500 bytes, com 20 bytes destinados ao cabeçalho e 8 bytes para a identificação do tipo de serviço ($1500 - 20 - 8 = 1472$), deixando assim 1472 bytes para o conteúdo do pacote.

```
ping -f -l 1473 marco.uminho.pt

Pinging marco.uminho.pt [193.136.9.240] with 1473 bytes of data:
Packet needs to be fragmented but DF set.

Ping statistics for 193.136.9.240:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PS C:\Users\Leonor Cunha> ping -f -l 1472 marco.uminho.pt

Pinging marco.uminho.pt [193.136.9.240] with 1472 bytes of data:
Reply from 193.136.9.240: bytes=1472 time=17ms TTL=61
Reply from 193.136.9.240: bytes=1472 time=6ms TTL=61
Reply from 193.136.9.240: bytes=1472 time=5ms TTL=61
Reply from 193.136.9.240: bytes=1472 time=2ms TTL=61

Ping statistics for 193.136.9.240:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 17ms, Average = 7ms
PS C:\Users\Leonor Cunha> |
```

Figura 16 – Comando ping com size 1473 e 1472

Parte 2

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.

Resposta: Para atender ao requisito de 5 hosts por sub-rede, precisamos de sub-redes com pelo menos 8 endereços ($2^3 = 8$), ou seja, uma máscara de /29.

Sub-rede 1: 172.68.53.192/29

Intervalo de endereços de host: 172.68.53.193 - 172.68.53.198

Endereço de Broadcast: 172.68.53.199

Sub-rede 2: 172.68.53.200/29

Intervalo de endereços de host: 172.68.53.201 - 172.68.53.206

Endereço de Broadcast: 172.68.53.207

Sub-rede 3: 172.68.53.208/29

Intervalo de endereços de host: 172.68.53.209 - 172.68.53.214

Endereço de Broadcast: 172.68.53.215

Sub-rede 4: 172.68.53.216/29

Intervalo de endereços de host: 172.68.53.217 - 172.68.53.222

Endereço de Broadcast: 172.68.53.223

Sub-rede 5: 172.68.53.224/29

Intervalo de endereços de host: 172.68.53.225 - 172.68.53.230

Endereço de Broadcast: 172.68.53.231

Sub-rede 6: 172.68.53.232/29

Intervalo de endereços de host: 172.68.53.233 - 172.68.53.238

Endereço de Broadcast: 172.68.53.239

Sub-rede 7: 172.68.53.240/29

Intervalo de endereços de host: 172.68.53.241 - 172.68.53.246

Endereço de Broadcast: 172.68.53.247

Sub-rede 8: 172.68.53.248/29

Intervalo de endereços de host: 172.68.53.249 - 172.68.53.254

Endereço de Broadcast: 172.68.53.255

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.

Resposta:

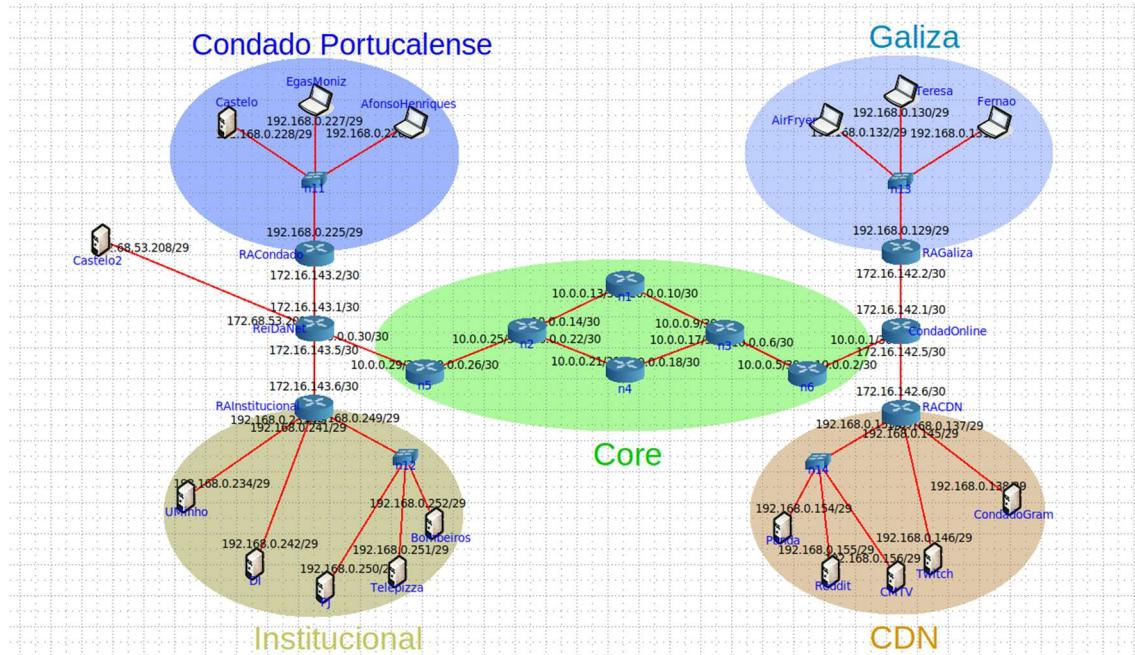


Figura 17 – Topologia da Rede Utilizada

```

root@Castelo2:/tmp/pycore.39069/Castelo2.conf# ping 172.68.53.209
PING 172.68.53.209 (172.68.53.209) 56(84) bytes of data.
64 bytes from 172.68.53.209: icmp_seq=1 ttl=64 time=0.061 ms
64 bytes from 172.68.53.209: icmp_seq=2 ttl=64 time=0.068 ms
64 bytes from 172.68.53.209: icmp_seq=3 ttl=64 time=0.063 ms
64 bytes from 172.68.53.209: icmp_seq=4 ttl=64 time=0.046 ms
64 bytes from 172.68.53.209: icmp_seq=5 ttl=64 time=0.052 ms
64 bytes from 172.68.53.209: icmp_seq=6 ttl=64 time=0.057 ms
64 bytes from 172.68.53.209: icmp_seq=7 ttl=64 time=0.056 ms
^C
--- 172.68.53.209 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6124ms
rtt min/avg/max/mdev = 0.046/0.057/0.068/0.006 ms
root@Castelo2:/tmp/pycore.39069/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=0.109 ms
64 bytes from 192.168.0.228: icmp_seq=2 ttl=62 time=0.093 ms
64 bytes from 192.168.0.228: icmp_seq=3 ttl=62 time=0.066 ms
64 bytes from 192.168.0.228: icmp_seq=4 ttl=62 time=0.113 ms
64 bytes from 192.168.0.228: icmp_seq=5 ttl=62 time=0.131 ms
64 bytes from 192.168.0.228: icmp_seq=6 ttl=62 time=0.110 ms
64 bytes from 192.168.0.228: icmp_seq=7 ttl=62 time=0.109 ms
64 bytes from 192.168.0.228: icmp_seq=8 ttl=62 time=0.108 ms
64 bytes from 192.168.0.228: icmp_seq=9 ttl=62 time=0.076 ms
64 bytes from 192.168.0.228: icmp_seq=10 ttl=62 time=0.107 ms
^C
--- 192.168.0.228 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9215ms
rtt min/avg/max/mdev = 0.066/0.102/0.131/0.018 ms
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# ping 192.168.0.227
PING 192.168.0.227 (192.168.0.227) 56(84) bytes of data.
64 bytes from 192.168.0.227: icmp_seq=1 ttl=62 time=0.113 ms
64 bytes from 192.168.0.227: icmp_seq=2 ttl=62 time=0.115 ms
64 bytes from 192.168.0.227: icmp_seq=3 ttl=62 time=0.095 ms
64 bytes from 192.168.0.227: icmp_seq=4 ttl=62 time=0.086 ms
64 bytes from 192.168.0.227: icmp_seq=5 ttl=62 time=0.080 ms
64 bytes from 192.168.0.227: icmp_seq=6 ttl=62 time=0.074 ms
64 bytes from 192.168.0.227: icmp_seq=7 ttl=62 time=0.108 ms
^C
--- 192.168.0.227 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6133ms
rtt min/avg/max/mdev = 0.074/0.095/0.115/0.015 ms
root@Castelo2:/tmp/pycore.39069/Castelo2.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=62 time=0.098 ms
64 bytes from 192.168.0.226: icmp_seq=2 ttl=62 time=0.097 ms
64 bytes from 192.168.0.226: icmp_seq=3 ttl=62 time=0.098 ms
64 bytes from 192.168.0.226: icmp_seq=4 ttl=62 time=0.079 ms
64 bytes from 192.168.0.226: icmp_seq=5 ttl=62 time=0.109 ms
64 bytes from 192.168.0.226: icmp_seq=6 ttl=62 time=0.115 ms
64 bytes from 192.168.0.226: icmp_seq=7 ttl=62 time=0.069 ms
64 bytes from 192.168.0.226: icmp_seq=8 ttl=62 time=0.074 ms
^C
--- 192.168.0.226 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7163ms
rtt min/avg/max/mdev = 0.069/0.092/0.115/0.015 ms
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# █

```

Figura 18 – Conectividade do Castelo2 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. Explicite ainda a utilidade de uma rota default.

Resposta:

```
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
0.0.0.0         172.68.53.209  0.0.0.0        UG      0 0          0 eth0
172.68.53.208  0.0.0.0        255.255.255.248 U        0 0          0 eth0
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# route delete default
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
172.68.53.208  0.0.0.0        255.255.255.248 U        0 0          0 eth0
```

Figura 19 - Apagar rota default

```
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# route add -net 192.168.0.224 netmask 255.255.255.248 gw 172.68.53.209 dev eth0
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags  Type       State      I-Node Path
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
172.68.53.208  0.0.0.0        255.255.255.248 U        0 0          0 eth0
192.168.0.224  172.68.53.209  255.255.255.248 UG      0 0          0 eth0
```

Figura 20 - Configurar rota Condado Portucalense

```
root@Castelo2:/tmp/pycore.39069/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=0.088 ms
64 bytes from 192.168.0.228: icmp_seq=2 ttl=62 time=0.076 ms
64 bytes from 192.168.0.228: icmp_seq=3 ttl=62 time=0.112 ms
64 bytes from 192.168.0.228: icmp_seq=4 ttl=62 time=0.122 ms
64 bytes from 192.168.0.228: icmp_seq=5 ttl=62 time=0.102 ms
^C
--- 192.168.0.228 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4100ms
rtt min/avg/max/mdev = 0.076/0.100/0.122/0.016 ms
```

Figura 21 - Conetividade com o Castelo do Condado Portucalense

```
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# route add -net 192.168.0.248 netmask 255.255.255.248 gw 172.68.53.209 dev eth0
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
172.68.53.208  0.0.0.0        255.255.255.248 U        0 0          0 eth0
192.168.0.224  172.68.53.209  255.255.255.248 UG      0 0          0 eth0
192.168.0.248  172.68.53.209  255.255.255.248 UG      0 0          0 eth0
root@Castelo2:/tmp/pycore.39069/Castelo2.conf#
```

Figura 22 - Configurar rota institucional

```
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# route add -net 192.168.0.232 netmask 255.255.255.248 gw 172.68.53.209 dev eth0
root@Castelo2:/tmp/pycore.39069/Castelo2.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
172.68.53.208  0.0.0.0        255.255.255.248 U        0 0          0 eth0
192.168.0.224  172.68.53.209  255.255.255.248 UG      0 0          0 eth0
192.168.0.232  172.68.53.209  255.255.255.248 UG      0 0          0 eth0
root@Castelo2:/tmp/pycore.39069/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=0.174 ms
64 bytes from 192.168.0.234: icmp_seq=2 ttl=62 time=0.075 ms
64 bytes from 192.168.0.234: icmp_seq=3 ttl=62 time=0.124 ms
64 bytes from 192.168.0.234: icmp_seq=4 ttl=62 time=0.075 ms
64 bytes from 192.168.0.234: icmp_seq=5 ttl=62 time=0.129 ms
64 bytes from 192.168.0.234: icmp_seq=6 ttl=62 time=0.203 ms
64 bytes from 192.168.0.234: icmp_seq=7 ttl=62 time=0.102 ms
64 bytes from 192.168.0.234: icmp_seq=8 ttl=62 time=0.098 ms
64 bytes from 192.168.0.234: icmp_seq=9 ttl=62 time=0.081 ms
64 bytes from 192.168.0.234: icmp_seq=10 ttl=62 time=0.094 ms
64 bytes from 192.168.0.234: icmp_seq=11 ttl=62 time=0.163 ms
64 bytes from 192.168.0.234: icmp_seq=12 ttl=62 time=0.108 ms
64 bytes from 192.168.0.234: icmp_seq=13 ttl=62 time=0.081 ms
64 bytes from 192.168.0.234: icmp_seq=14 ttl=62 time=0.075 ms
64 bytes from 192.168.0.234: icmp_seq=15 ttl=62 time=0.087 ms
64 bytes from 192.168.0.234: icmp_seq=16 ttl=62 time=0.094 ms
64 bytes from 192.168.0.234: icmp_seq=17 ttl=62 time=0.101 ms
^C
--- 192.168.0.234 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16364ms
rtt min/avg/max/mdev = 0.075/0.109/0.203/0.036 ms
```

Figura 23 - Configurar rota institucional e teste de conetividade com Uminho

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
172.68.53.208	0.0.0.0	255.255.255.248	U	0	0	0	eth0
192.168.0.224	172.68.53.209	255.255.255.248	UG	0	0	0	eth0
192.168.0.232	172.68.53.209	255.255.255.248	UG	0	0	0	eth0

Figura 24 - Tabela de encaminhamento resultante

A rota default é uma configuração essencial em redes, pois define um caminho para pacotes cujo destino não está explicitamente especificado na tabela de roteamento. Quando um sistema recebe um pacote para um endereço desconhecido, ele utiliza a rota default para encaminhá-lo ao gateway apropriado, garantindo conectividade com redes externas.

Sem essa configuração, o sistema não sabe como encaminhar pacotes para destinos não especificados manualmente, resultando na perda de conectividade com redes fora das rotas definidas.

No caso do Castelo2, ao eliminar a rota default, foi necessário configurar rotas específicas para manter a comunicação com as redes essenciais, como o Condado Portucalense e a rede Institucional. Se o objetivo fosse garantir acesso global à internet ou a outras redes não especificadas na configuração manual, a solução mais eficiente seria manter uma rota default apontando para um gateway adequado, evitando a necessidade de adicionar múltiplas rotas individuais.

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.

Resposta:

```
<pycore.39069/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=0.271 ms
64 bytes from 192.168.0.155: icmp_seq=2 ttl=55 time=0.154 ms
64 bytes from 192.168.0.155: icmp_seq=3 ttl=55 time=0.154 ms
64 bytes from 192.168.0.155: icmp_seq=4 ttl=55 time=0.135 ms
64 bytes from 192.168.0.155: icmp_seq=5 ttl=55 time=0.160 ms
64 bytes from 192.168.0.155: icmp_seq=6 ttl=55 time=0.215 ms
^C
--- 192.168.0.155 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5104ms
rtt min/avg/max/mdev = 0.135/0.181/0.271/0.047 ms
root@AfonsoHenriques:/tmp/pycore.39069/AfonsoHenriques.conf#
```

Figura 25 – Comando Ping de AfonsoHenriques para Reddit

```

<core.39069/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=0.155 ms
64 bytes from 192.168.0.146: icmp_seq=2 ttl=55 time=0.134 ms
64 bytes from 192.168.0.146: icmp_seq=3 ttl=55 time=0.130 ms
64 bytes from 192.168.0.146: icmp_seq=4 ttl=55 time=0.134 ms
64 bytes from 192.168.0.146: icmp_seq=5 ttl=55 time=0.133 ms
^C
--- 192.168.0.146 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4084ms
rtt min/avg/max/mdev = 0.130/0.137/0.155/0.009 ms
root@AfonsoHenriques:/tmp/pycore.39069/AfonsoHenriques.conf# 

```

Figura 26 – Comando Ping de AfonsoHenriques para Twitch

Como podemos observar nas imagens 25 e 26, é possível confirmar que AfonsoHenriques tem conectividade com os servidores Reddit e Twitch.

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.

Resposta:

```

root@AfonsoHenriques:/tmp/pycore.39069/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.39069/AfonsoHenriques.conf# 

```

Figura 27 – Comando netstat -rn do dispositivo AfonsoHenriques

```

root@Teresa:/tmp/pycore.39069/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.39069/Teresa.conf# 

```

Figura 28 – Comando netstat -rn do dispositivo Teresa

Na Figura 27, é notável que as configurações de roteamento para o dispositivo AfonsoHenriques estão sem problemas aparentes. A primeira linha detalha que o endereço de destino padrão (0.0.0.0) deve ser direcionado para o gateway 192.168.0.225 via a interface de rede eth0. Esta rota é identificada com a flag UG, sinalizando ser a rota padrão, e que o gateway deve ser usado para encaminhar todos os pacotes de rede que não têm correspondência com outras rotas na tabela. A segunda linha especifica que o endereço de destino 192.168.0.224 deve ser acedido diretamente pela interface de rede eth0, sem necessidade de passar por um gateway. A máscara de sub-rede 255.255.255.248 indica que essa rota é para uma rede local com 8 endereços IP disponíveis (192.168.0.224 a 192.168.0.231).

Na Figura 27, é possível observar as configurações de roteamento para o dispositivo Teresa e não aparentemente não mostram problemas. A primeira linha mostra que o endereço de destino padrão (0.0.0.0) deve ser direcionado para o gateway 192.168.0.129 via a interface de rede eth0. Esta rota possui a flag UG, sinalizando ser a rota padrão, e que o gateway deve ser usado para encaminhar todos os pacotes de

rede que não têm correspondência com outras rotas na tabela.. A segunda linha indica que o endereço de destino 192.168.0.128 deve ser acedido diretamente pela interface eth0, sem necessidade de gateway. A máscara de sub-rede 255.255.255.248 indica que essa rota é para uma rede local com 8 endereços IP disponíveis (192.168.0.128 a 192.168.0.135).

Em ambos os casos, essas configurações de roteamento são essenciais para garantir o correto encaminhamento dos pacotes de rede no sistema e para que cheguem aos seus destinos adequados. A utilidade de cada entrada varia conforme o contexto e a configuração específica do sistema, sendo frequentemente utilizadas para garantir uma conectividade de rede confiável e eficiente.

c. Analise o comportamento dos routers do core da rede (n1 a n6) quando tenta estabelecer comunicação entre os hosts AfonsoHenriques e Teresa. Indique que dispositivo(s) não permite(m) o encaminhamento correto dos pacotes. Seguidamente, avalie e explique a(s) causa(s) do funcionamento incorreto do dispositivo. Utilize o comando ip route add/del para adicionar as rotas necessárias ou remover rotas incorretas. Verifique a sintaxe completa do comando a usar com man ip-route ou man route. Poderá também utilizar o comando traceroute para se certificar do caminho nó a nó. Considere a alínea resolvida assim que houver tráfego a chegar ao ISP CondadOnline.

Resposta:

Quando tentamos fazer o traceroute entre AfonsoHenriques e Teresa percebemos que há uma falha na ligação.

```
<ycore.39069/AfonsoHenriques.conf# traceroute -I 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)  0.041 ms  0.010 ms  0.009 ms
 2  172.16.143.1 (172.16.143.1)  0.025 ms  0.015 ms  0.012 ms
 3  10.0.0.29 (10.0.0.29)  0.028 ms  0.015 ms  0.015 ms
 4  10.0.0.25 (10.0.0.25)  0.046 ms  0.035 ms  0.019 ms
 5  10.0.0.25 (10.0.0.25)  3064.956 ms !H  3064.934 ms !H  3064.925 ms !H
```

Figura 29 - Falha na ligação entre AfonsoHenriques e Teresa

Os dispositivos que não permitem o encaminhamento correto dos pacotes são os routers: n1, n2 e n3. A partir das seguintes figuras conseguimos perceber e identificar o funcionamento incorreto de cada um destes dispositivos.

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 30 – Tabela de encaminhamento do router n1

Como podemos ver na figura 30, a tabela de encaminhamento do router n1, a rota 192.168.0.128 Teresa é enviada por eth1 e deveria estar a ser enviada por eth0.

root@n1:/tmp/pycore.39069/n1.conf# ip route add 192.168.0.128/29 via 10.0.0.9							
root@n1:/tmp/pycore.39069/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

Figura 31 – comando -add e tabela de encaminhamento

Com a execução dos comandos –add, n1 fica com a sua rota corrigida, como podemos ver na figura acima.

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

No router n2, como podemos ver na figura 32, a rota 192.168.0.130 não deveria existir pois é uma sub-rede de uma sub-rede.

root@n2:/tmp/pycore.39069/n2.conf# ip route delete 192.168.0.130/31							
root@n2:/tmp/pycore.39069/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

Figura 33 - Comando –del e tabela de encaminhamento

Efetuamos o comando delete da rota de ip indesejada corrigindo assim o problema em n2, como podemos verificar na figura 33.

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 34 - Tabela de encaminhamento router n3

No router n3 não há ligação para 192.168.0.128 tornando impossível receber ou enviar pacotes assim, como observamos acima (figura 34).

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

Figura 35 – Comando ip route que adiciona 192.168.0.128

Como podemos ver na figura e pelo comando “ip route add”, adiciona-se a rota que faltava ao router n3.

Após resolvemos os problemas encontrados nos routers n1, n2 e n3, verificarmos todos os outros routers no caso de haver mais problemas. Como tal não se verificou, confirmamos o recebimento de tráfego em CondadOnline.

```
<pycore.39069/AfonsoHenriques.conf# ping 172.16.142.1
PING 172.16.142.1 (172.16.142.1) 56(84) bytes of data.
64 bytes from 172.16.142.1: icmp_seq=1 ttl=57 time=0.131 ms
64 bytes from 172.16.142.1: icmp_seq=2 ttl=57 time=0.280 ms
64 bytes from 172.16.142.1: icmp_seq=3 ttl=57 time=0.186 ms
64 bytes from 172.16.142.1: icmp_seq=4 ttl=57 time=0.225 ms
64 bytes from 172.16.142.1: icmp_seq=5 ttl=57 time=0.259 ms
^C
--- 172.16.142.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4072ms
rtt min/avg/max/mdev = 0.131/0.216/0.280/0.053 ms
root@AfonsoHenriques:/tmp/pycore.39069/AfonsoHenriques.conf# 
```

Figura 36 – Ping do AfonsoHenriques para CondadOnline

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	16.0.0.1	224.0.0.5	OSPF	78	Hello Packet
2	2.000191892	16.0.0.1	224.0.0.5	OSPF	78	Hello Packet
3	3.693888254	192.168.0.226	172.16.142.1	ICMP	98	Echo (ping) request id=0x0038, seq=1/256, ttl=57 (reply in 4)
4	3.693901337	172.16.142.1	192.168.0.226	ICMP	98	Echo (ping) reply id=0x0038, seq=1/256, ttl=64 (request in 3)
5	4.000399884	16.0.0.1	224.0.0.5	OSPF	78	Hello Packet
6	4.6866666667	16.0.0.1	172.16.142.1	ICMP	98	Echo (ping) request id=0x0038, seq=2/512, ttl=57 (reply in 7)
7	4.994638119	172.16.142.1	192.168.0.226	ICMP	98	Echo (ping) reply id=0x0038, seq=2/512, ttl=64 (request in 6)
8	5.718328236	192.168.0.226	172.16.142.1	ICMP	98	Echo (ping) request id=0x0038, seq=3/768, ttl=57 (reply in 9)
9	5.7183477529	172.16.142.1	192.168.0.226	ICMP	98	Echo (ping) reply id=0x0038, seq=3/768, ttl=64 (request in 8)
10	5.855882432	fe80::200:ff:fea:21	f092::5	OSPF	98	Hello Packet
11	6.000928353	16.0.0.1	224.0.0.5	OSPF	78	Hello Packet
12	6.742692799	192.168.0.226	172.16.142.1	ICMP	98	Echo (ping) request id=0x0038, seq=4/1024, ttl=57 (reply in 13)
13	6.742714857	172.16.142.1	192.168.0.226	ICMP	98	Echo (ping) reply id=0x0038, seq=4/1024, ttl=64 (request in 12)
14	7.766328137	192.168.0.226	172.16.142.1	ICMP	98	Echo (ping) request id=0x0038, seq=5/1280, ttl=57 (reply in 15)
15	7.766328136	172.16.142.1	192.168.0.226	ICMP	98	Echo (ping) reply id=0x0038, seq=5/1280, ttl=64 (request in 14)
16	8.918174954	00:00:00:aa:00:21	224.0.0.5	OSPF	78	Hello Packet
17	8.918174954	00:00:00:aa:00:21	00:00:00:aa:00:29	ARP	42	Who has 10.0.0.1? Tell 10.0.0.1
18	8.918367342	00:00:00:aa:00:20	00:00:00:aa:00:21	ARP	42	Who has 10.0.0.17? Tell 10.0.0.2
19	8.918534487	00:00:00:aa:00:20	00:00:00:aa:00:21	ARP	42	10.0.0.2 is at 00:00:00:aa:00:20
20	8.918506108	00:00:00:aa:00:21	00:00:00:aa:00:20	ARP	42	10.0.0.1 is at 00:00:00:aa:00:21
21	10.002356183	16.0.0.1	224.0.0.5	OSPF	78	Hello Packet

Figura 37 – Tráfego wireshark recebido no CondadOnline

Como pode ser observado nas figuras 36 e 37, existe tráfego a chegar ao ISP CondadOnline, pelo que damos a alínea como concluída.

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.

```
<core.39069/AfonsoHenriques.conf# ping 192.168.0.130
PING 192.168.0.130 (192.168.0.130) 56(84) bytes of data.
From 10.0.0.17 icmp_seq=1 Destination Net Unreachable
From 10.0.0.17 icmp_seq=9 Destination Net Unreachable
^C
--- 192.168.0.130 ping statistics ---
259 packets transmitted, 0 received, +2 errors, 100% packet loss, time 264199ms
root@AfonsoHenriques:/tmp/pycore.39069/AfonsoHenriques.conf# 
```

Figura 38 - Ping de AfonsoHenriques para Teresa, os pacotes são enviados, mas não recebe resposta

108 1/0.2422181410 00:00:00:aa:00:10	broadcast	ARP	42 Who has 192.168.0.130? Tell 192.168.0.129
109 176.242205151 00:00:00:aa:00:13	00:00:00_aa:00:10	ARP	42 192.168.0.130 is at 00:00:00:aa:00:13
110 176.242233178 192.168.0.226	192.168.0.130	ICMP	98 Echo (ping) request id=0x002f, seq=51/13056, ttl=64 (request in 110)
111 176.242251179 192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x002f, seq=51/13056, ttl=64 (request in 111)
112 176.242272255 192.168.0.129	192.168.0.130	ICMP	126 Destination unreachable (Network unreachable)
113 177.266605569 192.168.0.226	192.168.0.130	ICMP	98 Echo (ping) request id=0x002f, seq=52/13312, ttl=64 (request in 114)
114 177.266608795 192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x002f, seq=52/13312, ttl=64 (request in 113)
115 177.266109595 192.168.0.129	192.168.0.130	ICMP	126 Destination unreachable (Network unreachable)
116 178.065557541 192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
117 178.298338626 192.168.0.226	192.168.0.130	ICMP	98 Echo (ping) request id=0x002f, seq=53/13568, ttl=64 (request in 117)
118 178.290348116 192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x002f, seq=53/13568, ttl=64 (request in 118)
119 178.314570848 192.168.0.129	192.168.0.130	ICMP	126 Destination unreachable (Network unreachable)
120 179.31457346 192.168.0.226	192.168.0.130	ICMP	98 Echo (ping) request id=0x002f, seq=54/13824, ttl=64 (request in 121)
121 179.31457346 192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x002f, seq=54/13824, ttl=64 (request in 120)
122 179.314570848 192.168.0.129	192.168.0.130	ICMP	126 Destination unreachable (Network unreachable)
123 189.066064852 192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
124 189.338908482 192.168.0.226	192.168.0.130	ICMP	98 Echo (ping) request id=0x002f, seq=55/14080, ttl=64 (request in 124)
125 189.338929227 192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x002f, seq=55/14080, ttl=64 (request in 125)
126 189.362198879 fe80::200:ff:fea:10 ff02::5	OSPF	78 Hello Packet	
127 189.266619878 00:00:00_aa:00:13	00:00:00_aa:00:10	ARP	42 Who has 192.168.0.129? Tell 192.168.0.130
128 189.266725448 00:00:00_aa:00:13	00:00:00_aa:00:10	ARP	42 192.168.0.129 is at 00:00:00_aa:00:10
129 189.362012623 192.168.0.226	192.168.0.130	ICMP	98 Echo (ping) request id=0x002f, seq=56/14336, ttl=64 (request in 129)
130 189.362035682 192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x002f, seq=56/14336, ttl=64 (request in 129)
131 189.067653695 192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
132 189.386207481 192.168.0.226	192.168.0.130	ICMP	98 Echo (ping) request id=0x002f, seq=57/14592, ttl=64 (request in 133)
133 189.386234717 192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x002f, seq=57/14592, ttl=64 (request in 132)
134 189.410692433 192.168.0.226	192.168.0.130	ICMP	98 Echo (ping) request id=0x002f, seq=58/14848, ttl=64 (request in 135)
135 189.410716023 192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x002f, seq=58/14848, ttl=64 (request in 134)
136 189.410716023 192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
137 189.434119415 192.168.0.226	192.168.0.130	ICMP	98 Echo (ping) request id=0x002f, seq=59/15104, ttl=64 (request in 138)
138 189.434148664 192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x002f, seq=59/15104, ttl=64 (request in 137)
139 189.458143957 192.168.0.226	192.168.0.130	ICMP	98 Echo (ping) request id=0x002f, seq=60/15360, ttl=64 (request in 140)
140 189.458160746 192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x002f, seq=60/15360, ttl=64 (request in 139)
141 189.069859772 192.168.0.193	224.0.0.5	OSPF	78 Hello Packet

Figura 39 – Tráfego wireshark da Teresa que recebe e envia mas as respostas não chegam a AfonsoHenriques

- 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.

Resposta:

Como podemos ver nas figuras acima, 38 e 39, embora a Teresa receba os pacotes de AfonsoHenriques, ela não consegue rotear de volta as respostas para ele.

Analisamos o router RAGaliza, na figura abaixo, e observamos uma falha no IP 192.168.0.224, indicando necessidade de adicionar a rota do RAGaliza para o Condado Portucalense.

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		eth0
10.0.0.4	172.16.142.1	255.255.255.252	UG	0	0		eth0
10.0.0.8	172.16.142.1	255.255.255.252	UG	0	0		eth0
10.0.0.12	172.16.142.1	255.255.255.252	UG	0	0		eth0
10.0.0.16	172.16.142.1	255.255.255.252	UG	0	0		eth0
10.0.0.20	172.16.142.1	255.255.255.252	UG	0	0		eth0
10.0.0.24	172.16.142.1	255.255.255.252	UG	0	0		eth0
10.0.0.28	172.16.142.1	255.255.255.252	UG	0	0		eth0
172.0.0.0	172.16.142.1	255.0.0.0	UG	0	0		eth0
172.16.142.0	0.0.0.0	255.255.255.252	U	0	0		eth0
172.16.142.4	172.16.142.1	255.255.255.252	UG	0	0		eth0
172.16.143.0	172.16.142.1	255.255.255.252	UG	0	0		eth0
172.16.143.4	172.16.142.1	255.255.255.252	UG	0	0		eth0
192.168.0.128	0.0.0.0	255.255.255.248	U	0	0		eth1
192.168.0.136	172.16.142.1	255.255.255.248	UG	0	0		eth0
192.168.0.144	172.16.142.1	255.255.255.248	UG	0	0		eth0
192.168.0.152	172.16.142.1	255.255.255.248	UG	0	0		eth0
192.168.0.192	0.0.0.0	255.255.255.248	U	0	0		eth1
192.168.0.200	172.16.142.1	255.255.255.248	UG	0	0		eth0
192.168.0.208	172.16.142.1	255.255.255.248	UG	0	0		eth0
192.168.0.216	172.16.142.1	255.255.255.248	UG	0	0		eth0
192.168.0.232	172.16.142.1	255.255.255.248	UG	0	0		eth0
192.168.0.240	172.16.142.1	255.255.255.248	UG	0	0		eth0
192.168.0.248	172.16.142.1	255.255.255.248	UG	0	0		eth0

Figura 40 – Tabela de encaminhamento do router RAGaliza não tem caminho para o Condado Portucalense

```

root@RAGaliza:/tmp/pycore.39069/RAGaliza.conf# ip route add 192.168.0.224/29 via 192.168.0.130
root@RAGaliza:/tmp/pycore.39069/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 eth1
192.168.0.192    0.0.0.0       255.255.255.248 U          0 0          0 eth1
192.168.0.200    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.208    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.216    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.224    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.232    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.240    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.248    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
root@RAGaliza:/tmp/pycore.39069/RAGaliza.conf#

```

Figura 41 – Tabela de encaminhamento do RAGaliza depois de adicionamos rota para o Condado Portucalense

Após adicionarmos a rota verificamos se a Teresa já conseguiria enviar a resposta.

```

<core.39069/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=0.214 ms
64 bytes from 192.168.0.130: icmp_seq=2 ttl=55 time=0.198 ms
64 bytes from 192.168.0.130: icmp_seq=3 ttl=55 time=0.231 ms
64 bytes from 192.168.0.130: icmp_seq=4 ttl=55 time=0.212 ms
64 bytes from 192.168.0.130: icmp_seq=5 ttl=55 time=0.266 ms
64 bytes from 192.168.0.130: icmp_seq=6 ttl=55 time=0.151 ms
64 bytes from 192.168.0.130: icmp_seq=7 ttl=55 time=0.162 ms
64 bytes from 192.168.0.130: icmp_seq=8 ttl=55 time=0.171 ms
64 bytes from 192.168.0.130: icmp_seq=9 ttl=55 time=0.230 ms
64 bytes from 192.168.0.130: icmp_seq=10 ttl=55 time=0.177 ms
64 bytes from 192.168.0.130: icmp_seq=11 ttl=55 time=0.296 ms
64 bytes from 192.168.0.130: icmp_seq=12 ttl=55 time=0.178 ms
64 bytes from 192.168.0.130: icmp_seq=13 ttl=55 time=0.179 ms
64 bytes from 192.168.0.130: icmp_seq=14 ttl=55 time=0.145 ms
64 bytes from 192.168.0.130: icmp_seq=15 ttl=55 time=0.206 ms
64 bytes from 192.168.0.130: icmp_seq=16 ttl=55 time=0.167 ms
64 bytes from 192.168.0.130: icmp_seq=17 ttl=55 time=0.182 ms
64 bytes from 192.168.0.130: icmp_seq=18 ttl=55 time=0.285 ms
64 bytes from 192.168.0.130: icmp_seq=19 ttl=55 time=0.233 ms
64 bytes from 192.168.0.130: icmp_seq=20 ttl=55 time=0.382 ms
64 bytes from 192.168.0.130: icmp_seq=21 ttl=55 time=0.250 ms
64 bytes from 192.168.0.130: icmp_seq=22 ttl=55 time=0.270 ms
64 bytes from 192.168.0.130: icmp_seq=23 ttl=55 time=0.152 ms
64 bytes from 192.168.0.130: icmp_seq=24 ttl=55 time=0.266 ms
64 bytes from 192.168.0.130: icmp_seq=25 ttl=55 time=0.184 ms
64 bytes from 192.168.0.130: icmp_seq=26 ttl=55 time=0.230 ms
64 bytes from 192.168.0.130: icmp_seq=27 ttl=55 time=0.195 ms
64 bytes from 192.168.0.130: icmp_seq=28 ttl=55 time=0.230 ms
64 bytes from 192.168.0.130: icmp_seq=29 ttl=55 time=0.248 ms
64 bytes from 192.168.0.130: icmp_seq=30 ttl=55 time=0.235 ms
64 bytes from 192.168.0.130: icmp_seq=31 ttl=55 time=0.232 ms
64 bytes from 192.168.0.130: icmp_seq=32 ttl=55 time=0.137 ms
64 bytes from 192.168.0.130: icmp_seq=33 ttl=55 time=0.176 ms
64 bytes from 192.168.0.130: icmp_seq=34 ttl=55 time=0.167 ms
64 bytes from 192.168.0.130: icmp_seq=35 ttl=55 time=0.275 ms
^C
--- 192.168.0.130 ping statistics ---
35 packets transmitted, 35 received, 0% packet loss, time 34797ms
rtt min/avg/max/mdev = 0.137/0.214/0.382/0.051 ms

```

Figura 42 – Ping de AfonsoHenriques para Teresa

No.	Time	Source	Destination	Protocol	Length Info
1	0.0000000000	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
2	1.9910560606	fe80::200:ff:fea:10	ff02::5	OSPF	98 Hello Packet
3	2.0006865305	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
4	4.0814320322	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
5	6.0006865305	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
6	8.0839806309	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
7	8.3994828342	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=1/256, ttl=64 (reply in 8)
8	8.3994280111	192.168.0.130	192.168.0.126	ICMP	98 Echo (ping) reply id=0x0037, seq=1/256, ttl=64 (request in 7)
9	9.404815539	192.168.0.126	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=2/512, ttl=64 (reply in 10)
10	9.404831276	192.168.0.126	192.168.0.126	ICMP	98 Echo (ping) reply id=0x0037, seq=2/512, ttl=64 (request in 9)
11	10.005122988	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
12	10.428828392	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=3/768, ttl=64 (reply in 13)
13	10.428873463	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=3/768, ttl=64 (request in 12)
14	11.452843633	192.168.0.130	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=4/1024, ttl=64 (reply in 15)
15	11.452863363	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=4/1024, ttl=64 (request in 14)
16	12.005829781	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
17	12.0072208397	fe80::200:ff:fea:10	ff02::5	OSPF	98 Hello Packet
18	12.476876681	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=5/1280, ttl=64 (reply in 19)
19	12.476876681	192.168.0.130	192.168.0.126	ICMP	98 Echo (ping) reply id=0x0037, seq=5/1280, ttl=64 (request in 18)
20	13.4567151509	00:00:00:00:00:13	00:00:00:00:aa:00:10	ARP	42 Who has 192.168.0.129? Tell 192.168.0.130
21	13.4569008016	00:00:00:00:00:10	00:00:00:00:aa:00:13	ARP	42 Who has 192.168.0.130? Tell 192.168.0.129
22	13.4378622932	00:00:00:00:00:10	00:00:00:00:aa:00:19	ARP	42 192.168.0.129 is at 00:00:00:aa:00:19
23	13.436973728	00:00:00:00:00:10	00:00:00:00:aa:00:13	ARP	42 192.168.0.130 is at 00:00:00:aa:00:13
24	13.5098847154	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=6/1536, ttl=64 (reply in 25)
25	13.509862799	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=6/1536, ttl=64 (request in 24)
26	14.0065369192	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
27	14.525347872	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=7/1792, ttl=64 (reply in 28)
28	14.525363944	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=7/1792, ttl=64 (request in 27)
29	15.548999525	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=8/2048, ttl=64 (reply in 30)
30	15.548999525	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=8/2048, ttl=64 (request in 29)
31	16.0076859557	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
32	16.572875739	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=9/2304, ttl=64 (reply in 33)
33	16.5728998861	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=9/2304, ttl=64 (request in 32)
34	17.0076859557	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=10/2560, ttl=64 (reply in 35)
35	17.5096590515	192.168.0.193	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=10/2560, ttl=64 (request in 34)
36	18.007404425	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
37	18.620993179	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=11/2816, ttl=64 (reply in 38)
38	18.621817567	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=11/2816, ttl=64 (request in 37)
39	19.645928997	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=12/3072, ttl=64 (reply in 40)
40	19.645944404	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=12/3072, ttl=64 (request in 39)
41	20.008272663	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
42	20.668939259	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=13/3328, ttl=64 (reply in 43)
43	20.66895674	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=13/3328, ttl=64 (request in 42)
44	21.69350072	192.168.0.130	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=14/3584, ttl=64 (reply in 45)
45	21.69351553	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=14/3584, ttl=64 (request in 44)
46	22.009457424	192.168.0.193	224.0.0.5	OSPF	78 Hello Packet
47	22.011405447	fe80::200:ff:fea:10	ff02::5	OSPF	98 Hello Packet
48	22.7100000001	192.168.0.193	192.168.0.130	ICMP	98 Echo (ping) request id=0x0037, seq=15/3840, ttl=64 (reply in 49)
49	22.7160711667	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) reply id=0x0037, seq=15/3840, ttl=64 (request in 48)
50	23.741573624	192.168.0.130	192.168.0.226	ICMP	98 Echo (ping) request id=0x0037, seq=16/4096, ttl=64 (reply in 51)
51	23.741589110	192.168.0.130	192.168.0.226	ICMP	98 Echo (final) reply id=0x0037, seq=16/4096, ttl=64 (request in 50)

Figura 43 – Tráfego do Wireshark, Teresa a receber e enviar pacotes

Como verificamos, já é possível enviar e receber o tráfego entre AfonsoHenriques e Teresa.

- 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.

Resposta:

```
<pycore.39069/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)  0.050 ms  0.009 ms  0.008 ms
 2  172.16.143.1 (172.16.143.1)  0.041 ms  0.014 ms  0.011 ms
 3  10.0.0.29 (10.0.0.29)  0.024 ms  0.014 ms  0.013 ms
 4  10.0.0.25 (10.0.0.25)  0.033 ms  0.016 ms  0.018 ms
 5  10.0.0.13 (10.0.0.13)  0.037 ms  0.020 ms  0.019 ms
 6  10.0.0.17 (10.0.0.17)  0.049 ms  0.054 ms  0.025 ms
 7  10.0.0.5 (10.0.0.5)  0.047 ms  0.027 ms  0.025 ms
 8  10.0.0.1 (10.0.0.1)  0.045 ms  0.030 ms  0.028 ms
 9  172.16.142.2 (172.16.142.2)  0.066 ms  0.034 ms  0.033 ms
10  192.168.0.130 (192.168.0.130)  0.057 ms  0.038 ms  0.035 ms
root@AfonsoHenriques:/tmp/pycore.39069/AfonsoHenriques.conf#
```

Figura 44 – Comando -traceroute do AfonsoHenriques para Teresa

```

root@Teresa:/tmp/pycore.39069/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)  0.036 ms  0.009 ms  0.007 ms
 2  172.16.142.1 (172.16.142.1)  0.023 ms  0.011 ms  0.014 ms
 3  10.0.0.2 (10.0.0.2)  0.025 ms  0.014 ms  0.013 ms
 4  10.0.0.6 (10.0.0.6)  0.027 ms  0.017 ms  0.017 ms
 5  10.0.0.18 (10.0.0.18)  0.028 ms  0.019 ms  0.019 ms
 6  10.0.0.14 (10.0.0.14)  0.038 ms  0.049 ms  0.024 ms
 7  10.0.0.26 (10.0.0.26)  0.035 ms  0.025 ms  0.025 ms
 8  10.0.0.30 (10.0.0.30)  0.036 ms  0.031 ms  0.029 ms
 9  172.16.143.2 (172.16.143.2)  0.041 ms  0.032 ms  0.031 ms
10  192.168.0.226 (192.168.0.226)  0.043 ms  0.034 ms  0.034 ms
root@Teresa:/tmp/pycore.39069/Teresa.conf# 

```

Figura 45 – Comando -traceroute da Teresa para AfonsoHenriques

As rotas dos pacotes ICMP echo reply não são as mesmas (em ordem inversa), do que as rotas dos pacotes ICMP echo request enviadas entre AfonsoHenriques e Teresa. A rede está a usar roteamento assimétrico onde os pacotes usam diferentes caminhos quando viajam de AfonsoHenriques para Teresa,e diferentes caminhos quando retornam.

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.

Resposta:

```

root@n5:/tmp/pycore.39069/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
root@n5:/tmp/pycore.39069/n5.conf# 

```

Figura 46 – Tabela de encaminhamento de n5

Existe correspondência nesta entrada para pacotes enviados para o polo Galiza. O polo Galiza tem o endereço IP 192.168.0.128, que está dentro da rede 192.168.0.0/24. Portanto, os pacotes destinados a Galiza corresponderão a esta entrada.

Para CDN não existe correspondência. O endereço IP do CDN é 172.16.142.5, que não está na rede 192.168.0.0/24. Esta entrada não será usada para pacotes destinados ao CDN.

Para o polo Galiza, caso seja essa a entrada utilizada para o encaminhamento, permitirá o funcionamento esperado do dispositivo. Esta entrada encaminhará pacotes para o gateway 10.0.0.30 através da interface eth0. No entanto, conforme podemos ver na topologia, o n5 está conectado ao n6 através da interface eth0, e o n6 está no caminho para Galiza através do gateway 10.0.0.2, não 10.0.0.30. Isso sugere que esta entrada pode não ser a mais eficiente ou correta para alcançar o polo Galiza.

Para o CDN, esta entrada não é relevante pois o CDN tem um endereço IP (172.16.142.5) que não corresponde a esta entrada. Na tabela de rotas, existe uma entrada 172.16.142.0 com gateway 10.0.0.25, que seria a entrada usada para encaminhar pacotes para o CDN.

A entrada 192.168.0.0/24 será utilizada para pacotes destinados ao polo Galiza (192.168.0.128), a menos que exista uma entrada mais específica. Analisando a tabela de rotas completa, há uma entrada específica para 192.168.0.128 com máscara 255.255.255.248. Esta entrada mais específica (com prefixo mais longo) terá precedência sobre a entrada 192.168.0.0/24 para pacotes destinados a 192.168.0.128. Para o CDN (172.16.142.5), a entrada 192.168.0.0/24 não é utilizada. Em vez disso, será utilizada a entrada para a rede 172.16.142.0.

Concluindo, a entrada mencionada não é, provavelmente, a que seria utilizada para nenhum dos dois destinos na prática: para o polo Galiza seria usada a entrada mais específica para 192.168.0.128, e para o CDN seria usada a entrada para 172.16.142.0.

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

Resposta:

Endereços dos quatro polos,

Polo Condado Portucalense:

Dispositivos: Castelo (192.168.0.228/29), EgasMoniz (192.168.0.227/29), AfonsoHenriques (192.168.0.226/29)

Roteador RACondado: 192.168.0.225/29

Polo Galiza:

Dispositivos: Teresa (192.168.0.130/29), AirFry (192.168.0.132/29), Fernão (192.168.0.131/29)

Roteador RAGaliza: 192.168.0.129/29

Polo Institucional:

Dispositivos: UMinho (192.168.0.234/29), Tabacaria (192.168.0.250/29), Barceiros (192.168.0.251/29)

Roteador RAIstitucional: 192.168.0.241/29

Polo CDN:

Dispositivos: Reddit (192.168.0.155/29), CCTV (192.168.0.156/29), Twitch (192.168.0.146/29), CondadoGram (192.168.0.139/29)

Roteador RACDN: 192.168.0.145/29

Análise dos endereços dos polos,

Todos os endereços dos quatro polos começam com 192.168.x.x, que pertencem à faixa 192.168.0.0 - 192.168.255.255. Esta faixa está definida na RFC 1918 como endereços IP privados reservados para redes locais privadas. Portanto, os endereços utilizados pelos quatro polos são endereços privados.

Endereços do core da rede/ISPs,

No core da rede (área verde), observamos endereços como:

10.0.0.x/30 (interfaces entre roteadores n1, n2, n3, n4, n5, n6)

172.16.142.x/30, 172.16.143.x/30, etc. (conexões entre roteadores e os polos)

Análise dos endereços do core,

Endereços começando com 10.x.x.x pertencem ao intervalo 10.0.0.0 - 10.255.255.255

Endereços começando com 172.16.x.x pertencem ao intervalo 172.16.0.0 -

172.31.255.255

Ambos estes intervalos também estão definidos na RFC 1918 como endereços IP privados. Portanto, os endereços utilizados no core da rede também são endereços privados.

Justificativa:

A topologia de rede apresentada utiliza exclusivamente endereços IP privados, uma escolha arquitetônica que oferece diversas vantagens para a organização da infraestrutura. Esta abordagem proporciona múltiplos benefícios que se complementam para criar uma rede eficiente e segura.

A economia de endereços IP públicos é um fator crucial nesta implementação. Ao utilizar endereços privados em toda a infraestrutura, a rede preserva os valiosos endereços IP públicos, que são um recurso extremamente limitado no protocolo IPv4. Esta prática é especialmente importante considerando a escassez global de endereços IPv4.

Do ponto de vista de segurança, a utilização de endereços privados representa uma vantagem significativa. Estes endereços não são roteáveis diretamente pela Internet, o que cria uma barreira natural contra tentativas de acesso externo não autorizado. Esta característica funciona como uma camada adicional de proteção para os dispositivos da rede. A estrutura hierárquica implementada demonstra um planejamento cuidadoso, com a segregação da rede em diferentes segmentos utilizando faixas de endereços distintas. Os dispositivos finais nos polos são configurados com endereços 192.168.0.x/29, enquanto as interconexões no core utilizam 10.0.0.x/30, e as conexões entre o core e os roteadores de acesso empregam 172.16.x.x/30. Esta compartmentalização facilita a administração e o diagnóstico de problemas.

A conectividade entre os diversos segmentos da rede é garantida através do core central. Esta arquitetura permite que os quatro polos (Condado Portucalense, Galiza,

Institucional e CDN) se comuniquem eficientemente, mantendo ao mesmo tempo uma clara segmentação lógica. Esta organização facilita a aplicação de políticas de controle de acesso e melhora a performance geral da rede.

O subnetting implementado demonstra eficiência no uso dos endereços disponíveis. As máscaras /29 utilizadas nos polos possibilitam a alocação de até 6 hosts por subrede, uma capacidade adequada para os pequenos grupos de dispositivos presentes em cada polo. Por outro lado, as máscaras /30 empregadas no core permitem apenas 2 hosts por subrede, uma configuração ideal para os links ponto a ponto entre os roteadores, evitando o desperdício de endereços.

Concluindo, toda a infraestrutura de rede mostrada na topologia está utilizando exclusivamente endereços IP privados, tanto nos polos quanto no core da rede, seguindo práticas comuns de design de redes corporativas e de campus.

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

Resposta:

Os switches, por atuarem na camada 2, não possuem um endereço IP atribuído, pois sua função é encaminhar pacotes com base nos endereços MAC dos dispositivos conectados a eles. Essa característica diferencia-os dos roteadores, que operam na camada 3 e utilizam endereços IP para o encaminhamento de pacotes entre redes.

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 planear um novo ataque, mas constata que o seu exército não só perde bastante tempo a decidir que direção tomar a cada salto como, por vezes, inclusivamente se perde.

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.

Resposta:

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 47 – Comando Netsat

Usa-se o comando Netstat no router n6 para verificar as rotas que temos.

```
root@n6:/tmp/pycore.39069/n6.conf# ip route del 192.168.0.128/29
root@n6:/tmp/pycore.39069/n6.conf# ip route del 192.168.0.136/29
```

Figura 48 – Comando -del

Utilizamos este comando para apagar as rotas referentes a galiza e CDN.

```
root@n6:/tmp/pycore.39069/n6.conf# ip route del 192.168.0.144/29
root@n6:/tmp/pycore.39069/n6.conf#
```

Figura 49 – Comando -del

Apagamos também as que podia englobar a nova rota.

Porque escolher a nova rota?

Para criar uma única rota que englobe ambos os polos, precisamos encontrar o prefixo comum que cubra desde 192.168.0.128 até aproximadamente 192.168.0.147.

Convertendo para binário:

- 192.168.0.128 = 11000000.10101000.00000000.10000000
- 192.168.0.147 = 11000000.10101000.00000000.10010011

O prefixo comum mais longo entre esses endereços é:

11000000.10101000.00000000.1000xxxx = 192.168.0.128/28

Esta máscara /28 (255.255.255.240) cobre os endereços de 192.168.0.128 até 192.168.0.143.

No entanto, como os endereços do CDN vão até 192.168.0.146, precisamos de uma máscara maior: 192.168.0.128/27 (255.255.255.224) que cobre 192.168.0.128 até 192.168.0.159.

Esta rota única irá direcionar o tráfego destinado tanto para Galiza (192.168.0.128/29) quanto para CDN (192.168.0.136/29 e afins) através do mesmo caminho, simplificando

a tabela de roteamento e a tomada de decisão.

```
root@n6:/tmp/pycore.39069/n6.conf# ip route add 192.168.0.128/27 via 10.0.0.1
root@n6:/tmp/pycore.39069/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      255.255.255.224 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
root@n6:/tmp/pycore.39069/n6.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=62 time=0.085 ms
64 bytes from 192.168.0.130: icmp_seq=2 ttl=62 time=0.099 ms
64 bytes from 192.168.0.130: icmp_seq=3 ttl=62 time=0.081 ms
64 bytes from 192.168.0.130: icmp_seq=4 ttl=62 time=0.080 ms
64 bytes from 192.168.0.130: icmp_seq=5 ttl=62 time=0.070 ms
64 bytes from 192.168.0.130: icmp_seq=6 ttl=62 time=0.068 ms
^C
--- 192.168.0.130 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5104ms
rtt min/avg/max/mdev = 0.068/0.080/0.099/0.010 ms
root@n6:/tmp/pycore.39069/n6.conf#
```

Figura 50 – Comando Netsat e ping

Adicionamos a nova rota à tabela de encaminhamento e utilizamos o comando Netstat para confirmar a conectividade. De forma a confirmar se a conectividade é mantida efetivamente fizemos -ping para a Teresa na Galiza e obtivemos o resultado acima.

```
root@n6:/tmp/pycore.39069/n6.conf# ping 192.168.0.137
PING 192.168.0.137 (192.168.0.137) 56(84) bytes of data.
64 bytes from 192.168.0.137: icmp_seq=1 ttl=63 time=0.050 ms
64 bytes from 192.168.0.137: icmp_seq=2 ttl=63 time=0.067 ms
64 bytes from 192.168.0.137: icmp_seq=3 ttl=63 time=0.062 ms
64 bytes from 192.168.0.137: icmp_seq=4 ttl=63 time=0.070 ms
64 bytes from 192.168.0.137: icmp_seq=5 ttl=63 time=0.126 ms
^C
--- 192.168.0.137 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4080ms
rtt min/avg/max/mdev = 0.050/0.075/0.126/0.026 ms
root@n6:/tmp/pycore.39069/n6.conf#
```

Figura 51 – Ping para RACDN

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

Resposta:

```
root@n6:/tmp/pycore.39069/n6.conf# ip route del 192.168.0.224/29
root@n6:/tmp/pycore.39069/n6.conf# ip route del 192.168.0.232/29
root@n6:/tmp/pycore.39069/n6.conf# ip route del 192.168.0.240/29
```

Figura 52 – Comando -del

```
root@n6:/tmp/pycore.39069/n6.conf# ip route del 192.168.0.248/29
```

Figura 53 – Comando -del

Usamos este comando para a eliminação das rotas

```
root@n6:/tmp/pycore.39069/n6.conf# ip route add 192.168.0.224/27 via 10.0.0.6
root@n6:/tmp/pycore.39069/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      255.255.255.224 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      255.255.255.224 UG       0 0          0 eth1
root@n6:/tmp/pycore.39069/n6.conf#
```

Figura 54 – Tabela de encaminhamento final mais o comando que adiciona a nova rota

```
root@n6:/tmp/pycore.39069/n6.conf# ping 192.168.0.226
PING 192.168.0.226 (192.168.0.226) 56(84) bytes of data.
64 bytes from 192.168.0.226: icmp_seq=1 ttl=58 time=0.153 ms
64 bytes from 192.168.0.226: icmp_seq=2 ttl=58 time=0.130 ms
64 bytes from 192.168.0.226: icmp_seq=3 ttl=58 time=0.113 ms
64 bytes from 192.168.0.226: icmp_seq=4 ttl=58 time=0.136 ms
64 bytes from 192.168.0.226: icmp_seq=5 ttl=58 time=0.129 ms
^C
--- 192.168.0.226 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4085ms
rtt min/avg/max/mdev = 0.113/0.132/0.153/0.012 ms
root@n6:/tmp/pycore.39069/n6.conf# ping 192.168.0.234
PING 192.168.0.234 (192.168.0.234) 56(84) bytes of data.
64 bytes from 192.168.0.234: icmp_seq=1 ttl=58 time=0.165 ms
64 bytes from 192.168.0.234: icmp_seq=2 ttl=58 time=0.116 ms
64 bytes from 192.168.0.234: icmp_seq=3 ttl=58 time=0.128 ms
64 bytes from 192.168.0.234: icmp_seq=4 ttl=58 time=0.130 ms
^C
--- 192.168.0.234 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3047ms
rtt min/avg/max/mdev = 0.116/0.134/0.165/0.018 ms
root@n6:/tmp/pycore.39069/n6.conf#
```

Figura 55 – Ping com AfonsoHneriques e Uminho

Usamos este ping para testar a conectividade.

c. Comente os aspectos positivos e negativos do uso do Supernetting.

Resposta:

O Supernetting é uma técnica de design de rede que consiste em agrupar várias redes IP menores em uma única rede maior, com o objetivo de simplificar e reduzir o tamanho das tabelas de roteamento. Essa abordagem permite diminuir o número de

entradas na tabela de roteamento, melhorando o desempenho da rede e reduzindo a carga de processamento nos routers.

Uma das principais vantagens do Supernetting é a otimização da tabela de roteamento. Ao consolidar várias sub-redes em uma única rede maior, há uma redução significativa na quantidade de informações que os routers precisam processar, o que torna o roteamento mais eficiente. Além disso, essa técnica melhora o aproveitamento do espaço de endereçamento IP, pois minimiza o desperdício de endereços ao permitir que menos IP's sejam necessários para representar a rede agregada.

Por outro lado, o Supernetting também apresenta algumas desvantagens. Uma delas é a perda de flexibilidade, uma vez que a agregação de sub-redes pode dificultar futuras alterações. Se for necessário modificar ou remover uma sub-rede específica, será preciso reconfigurar toda a rede agregada, tornando a administração da rede mais complexa. Além disso, essa técnica pode aumentar o risco de falha, pois uma interrupção na rede agregada pode impactar todas as sub-redes associadas, dificultando a identificação e resolução de problemas.

Em resumo, o Supernetting é uma solução eficiente para reduzir o tamanho das tabelas de roteamento e otimizar o uso de endereços IP, contribuindo para um melhor desempenho da rede. No entanto, ao implementá-lo, é fundamental considerar suas limitações, como a menor flexibilidade e o risco de falhas que podem comprometer a estabilidade da rede.

Conclusão

Este projeto foi dividido em duas partes principais, cada uma apresentando um nível de dificuldade e complexidade crescente. Na primeira parte, realizamos uma análise do tráfego de rede por meio de diferentes comandos, como o traceroute, e exploramos conceitos fundamentais como TTL, RTT e os protocolos ICMP e IP. O uso do Wireshark foi essencial para estabelecer uma relação entre a topologia da rede, os comandos executados no terminal e o tráfego capturado, permitindo uma compreensão mais prática desses conceitos.

Na segunda parte do projeto, aprofundamos os nossos estudos em máscara de rede e sub-redes no protocolo IPv4, além de abordarmos o conceito de Sub-endereçamento IP. Essa etapa se revelou mais desafiadora e exigente, o que nos levou a buscar auxílio do professor para esclarecer dúvidas e superar obstáculos. No entanto, conseguimos ultrapassar estas dificuldades e prosseguir com o nosso trabalho.

Em conclusão, este permitiu-nos aplicar e fortalecer conceitos fundamentais de Redes de Computadores. Através da prática, conseguimos não apenas responder satisfatoriamente às questões propostas, mas também compreender melhor a utilidade de ferramentas como o Core e o Wireshark, que se mostraram indispensáveis para a análise e configuração de redes.