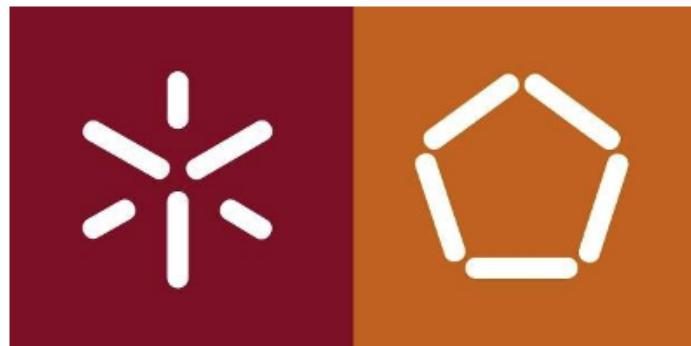
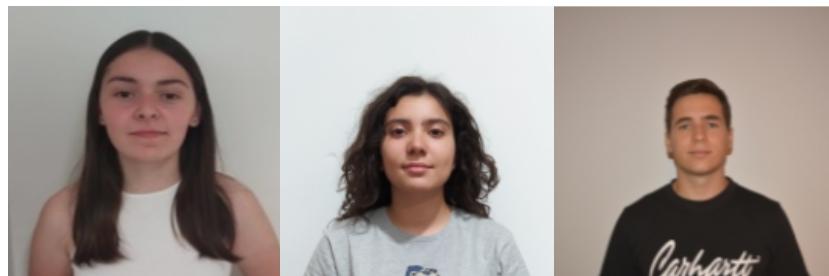


Universidade do Minho

Licenciatura em Engenharia Informática



TP2 - Protocolo IPv4 Datagramas IP e Fragmentação



Trabalho realizado por:
Margarida Cunha da Silva, A104357
Maria Leonor Carvalho da Cunha, A103997
Tiago Rodrigues Barros, A104530.

Redes de Computadores

PL9 - Grupo 91
March 4, 2024

Índice

TP2 - Protocolo IPv4 Datagramas IP e Fragmentação.....	1
1.1. Exercício 1.....	3
1.2. Exercício 2.....	5
1.3. Exercício 3.....	9
2.1. Exercício 1.....	15
2.2. Exercício 2.....	16
2.3. Exercício 3.....	17

1. Parte 1

1.1. Exercício 1

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

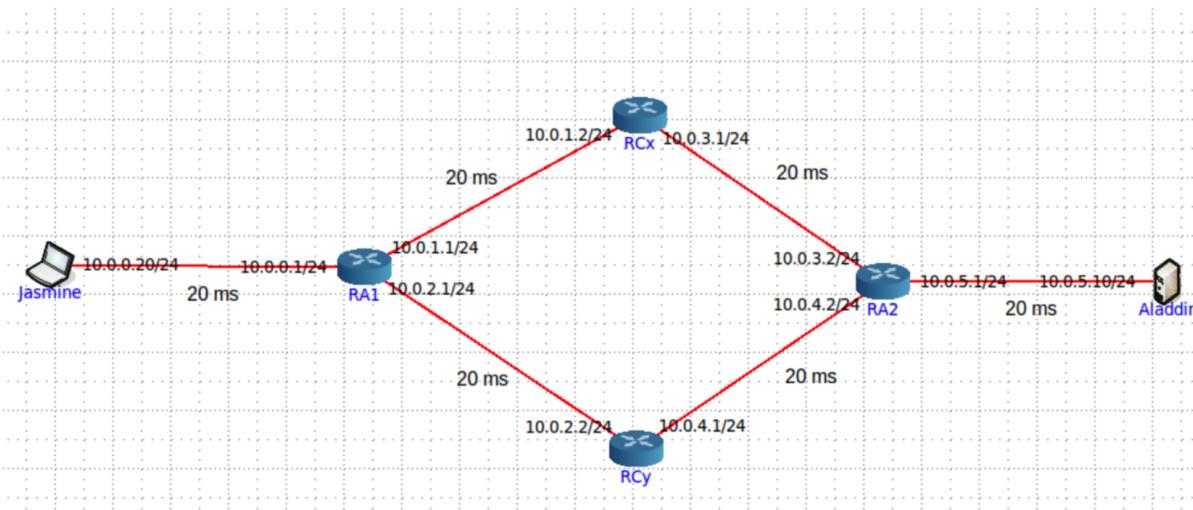


Figura 1: Topologia CORE

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

Resposta:

```
root@Jasmine:/tmp/pycore.38695/Jasmine.conf# traceroute -I 10.0.5.10
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.036 ms  0.010 ms  0.007 ms
 2  10.0.1.2 (10.0.1.2)  0.043 ms  0.009 ms  0.008 ms
 3  10.0.3.2 (10.0.3.2)  0.033 ms  0.011 ms  0.011 ms
 4  10.0.5.10 (10.0.5.10)  0.032 ms  0.014 ms  0.013 ms
root@Jasmine:/tmp/pycore.38695/Jasmine.conf#
```

Figura 2: Output do comando traceroute -I IP Aladdin

icmp						
No.	Time	Source	Destination	Protocol	Length	Info
38	13.585778414	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=1/256, ttl=1 (no response..)
39	13.585803691	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
40	13.585814041	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=2/512, ttl=1 (no response..)
41	13.585828313	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
42	13.585825122	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=3/768, ttl=1 (no response..)
43	13.585829400	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
44	13.585833618	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=4/1024, ttl=2 (no respons..)
45	13.585874164	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
46	13.585878863	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=5/1280, ttl=2 (no respons..)
47	13.585885806	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
48	13.585889583	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=6/1536, ttl=2 (no respons..)
49	13.585895634	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
50	13.585899752	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=7/1792, ttl=3 (no respons..)
51	13.585936751	10.0.3.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
52	13.585934808	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=8/2048, ttl=3 (no respons..)
53	13.585943635	10.0.3.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
54	13.585947402	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=9/2304, ttl=3 (no respons..)
55	13.585956680	10.0.3.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
56	13.585961168	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=10/2560, ttl=4 (reply in ..)
57	13.585991595	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0019, seq=10/2560, ttl=61 (request ..)
58	13.585996955	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=11/2816, ttl=4 (reply in ..)
59	13.586008437	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0019, seq=11/2816, ttl=61 (request ..)
60	13.586012294	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=12/3072, ttl=4 (reply in ..)
61	13.586023375	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0019, seq=12/3072, ttl=61 (request ..)
62	13.586027443	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=13/3328, ttl=5 (reply in ..)
63	13.586038123	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0019, seq=13/3328, ttl=61 (request ..)
64	13.586042610	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=14/3584, ttl=5 (reply in ..)
65	13.586052560	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0019, seq=14/3584, ttl=61 (request ..)
66	13.586056498	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=15/3840, ttl=5 (reply in ..)
67	13.586066907	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0019, seq=15/3840, ttl=61 (request ..)
68	13.586070935	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0019, seq=16/4096, ttl=6 (reply in ..)
69	13.586081244	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0019, seq=16/4096, ttl=61 (request ..)

Figura 3: o tráfego ICMP enviado pelo sistema Jasmine e o tráfego ICMP recebido como resposta

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

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

Resposta: A fim de alcançar o Servidor1, 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 56, é possível perceber que Jasmine só recebe uma resposta de Aladdin quando o TTL atinge o valor de 4.

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

Resposta: Para calcular o RTT fazemos $0.032 + 0.014 + 0.013 = 0.059$. E dividimos 0.059 por 3, resultando em aproximadamente 0.0197 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, 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.

1.2. Exercício 2

```
traceroute to marco.uminho.pt (193.136.9.240), 64 hops max, 72 byte packets
 1  172.26.254.254 (172.26.254.254)  4.327 ms  3.728 ms  3.361 ms
 2  172.16.2.1 (172.16.2.1)  3.636 ms  2.988 ms  3.009 ms
 3  172.16.115.252 (172.16.115.252)  3.529 ms  7.039 ms  4.467 ms
 4  marco.uminho.pt (193.136.9.240)  3.816 ms  3.669 ms  3.377 ms
```

Figura 4: Output do comando: traceroute -l marco.uminho.pt

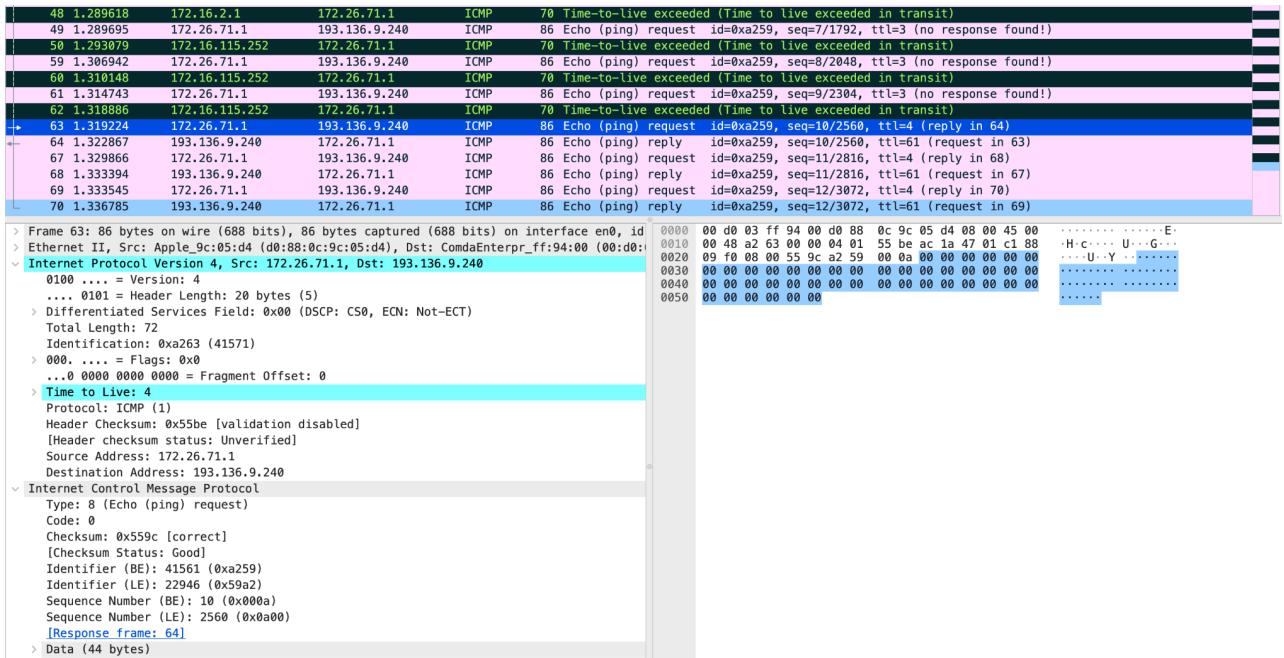


Figura 5: Tab do wireshark correspondente à primeira mensagem capturada

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

Resposta: 172.26.71.1

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

Resposta: O protocolo ICMP (Protocolo de Mensagem de Controle da Internet) é identificado com um valor de campo igual a 4, conforme indicado na figura 5 apresentada acima.

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: Considerando que o número de bytes do campo de dados do datagrama é calculado como a diferença entre o tamanho total (Total Length) e o tamanho do cabeçalho IPv4 (Header Length), que são 72 e 20 bytes, respectivamente, concluímos que o campo de dados do datagrama possui 52 bytes. Os valores mencionados podem ser verificados na Figura 5.

d) O datagrama IP foi fragmentado? Justifique.

Resposta: Considerando que o valor da Flag e do Fragment Offset está definido como 0, conforme evidenciado na Figura 5, podemos inferir que o datagrama IP não foi fragmentado.

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

29	1.247745	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=1/256, ttl=1 (no response found!)
33	1.257326	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=2/512, ttl=1 (no response found!)
35	1.261058	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=3/768, ttl=1 (no response found!)
37	1.264422	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=4/1024, ttl=2 (no response found!)
45	1.283681	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=5/1280, ttl=2 (no response found!)
47	1.286673	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=6/1536, ttl=2 (no response found!)
49	1.289695	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=7/1792, ttl=2 (no response found!)
59	1.306942	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=8/2048, ttl=2 (no response found!)
61	1.314743	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=9/2304, ttl=3 (no response found!)
63	1.319224	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=10/2560, ttl=4 (reply in 64)
67	1.329866	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=11/2816, ttl=4 (reply in 68)
69	1.333545	172.26.71.1	193.136.9.240	ICMP	86 Echo (ping) request id=0xa259, seq=12/3072, ttl=4 (reply in 70)

Figura 6: Pacotes ordenados

```
> Frame 29: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface en0, id
> Ethernet II, Src: Apple_9c:05:d4 (d0:88:0c:9c:05:d4), Dst: ComdaEnterpr_ff:94:00 (00:d0:
< Internet Protocol Version 4, Src: 172.26.71.1, 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: 72
        Identification: 0xa25a (41562)
    > 000. .... = Flags: 0x0
        ...0 0000 0000 0000 = Fragment Offset: 0
    > Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x58c7 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.71.1
    Destination Address: 193.136.9.240
```

Figura 7: Pacote 1

Resposta: Conforme observado nas figuras 6 e 7, 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 do TTL?

Resposta: Após uma análise mais detalhada, concluímos que o TTL é incrementado a cada 3 pacotes, a "Identification" é incrementada a cada 1 pacote, e a "Header Checksum" 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.

30	1.251607	172.26.254.254	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
34	1.260949	172.26.254.254	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
36	1.264323	172.26.254.254	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
38	1.267981	172.16.2.1	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
46	1.286543	172.16.2.1	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
48	1.289618	172.16.2.1	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
50	1.293079	172.16.115.252	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
60	1.310148	172.16.115.252	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
62	1.318886	172.16.115.252	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
64	1.322867	193.136.9.240	172.26.71.1	ICMP	86	Echo (ping) reply id=0xa259, seq=10/2560, ttl=61 (request in 63)
68	1.333394	193.136.9.240	172.26.71.1	ICMP	86	Echo (ping) reply id=0xa259, seq=11/2816, ttl=61 (request in 67)
70	1.336785	193.136.9.240	172.26.71.1	ICMP	86	Echo (ping) reply id=0xa259, seq=12/3072, ttl=61 (request in 69)
29	1.247745	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=1/256, ttl=1 (no response found!)
33	1.257326	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=2/512, ttl=1 (no response found!)
35	1.261050	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=3/768, ttl=1 (no response found!)
37	1.264422	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=4/1024, ttl=2 (no response found!)
45	1.283681	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=5/1280, ttl=2 (no response found!)
47	1.286673	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=6/1536, ttl=2 (no response found!)
49	1.289695	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=7/1792, ttl=3 (no response found!)
59	1.306942	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=8/2048, ttl=3 (no response found!)
61	1.314743	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=9/2304, ttl=3 (no response found!)
63	1.319224	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=10/2560, ttl=4 (reply in 64)
67	1.329866	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=11/2816, ttl=4 (reply in 68)
69	1.333545	172.26.71.1	193.136.9.240	ICMP	86	Echo (ping) request id=0xa259, seq=12/3072, ttl=4 (reply in 70)

Figura 8: Tráfego ICMP Ordenado pela Coluna Destination

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: O valor do TTL não é constante em todas as mensagens de resposta, pois varia conforme o número de routers pelo qual o pacote passa ao longo do caminho. Num determinado momento, o TTL pode ser 4, mas pode ser diferente noutras mensagens de resposta devido às variações na rota percorrida.

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

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

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

Resposta: Ao considerarmos a possibilidade de incorporar as informações do protocolo ICMP no cabeçalho IPv4, surgem vantagens e desvantagens a serem ponderadas.

Vantagens:

Simplificação do processamento: A eliminação do cabeçalho ICMP reduziria a carga de processamento nos routers, pois não seria necessário analisar um cabeçalho adicional para obter detalhes sobre o tipo de mensagem e o código de erro.

Redução do tamanho do pacote: A ausência do cabeçalho ICMP resultaria em uma economia de espaço em cada pacote, o que é particularmente valioso em redes com largura de banda limitada, contribuindo para uma utilização mais eficiente dos recursos.

Melhoria na eficiência: Com as informações do ICMP integradas diretamente ao cabeçalho IPv4, os routers teriam acesso imediato a esses dados, o que poderia otimizar o roteamento e a entrega de pacotes, resultando num desempenho geral aprimorado da rede.

Desvantagens:

Aumento da complexidade do cabeçalho IPv4: O cabeçalho IPv4 já é intrinsecamente complexo, e adicionar informações do ICMP poderia complicá-lo ainda mais, tornando-o mais difícil de entender e implementar.

Incompatibilidade com versões anteriores: Dispositivos que não suportam a nova estrutura do cabeçalho IPv4 seriam incapazes de comunicar-se com dispositivos que a utilizam, o que poderia levar a problemas de interoperabilidade e fragmentação na rede.

Perda de flexibilidade: O ICMP é um protocolo altamente flexível que pode ser utilizado para uma variedade de propósitos. Incorporá-lo diretamente ao cabeçalho IPv4 poderia limitar essa flexibilidade, restringindo as possibilidades de adaptação a diferentes cenários e requisitos de comunicação.

Portanto, enquanto a inclusão das informações do ICMP no cabeçalho IPv4 poderia trazer benefícios em termos de simplificação, economia de espaço e eficiência, ela também apresentaria desafios significativos em termos de complexidade, compatibilidade e flexibilidade, exigindo uma cuidadosa avaliação de seus prós e contras antes de ser implementada.

1.3. Exercício 3

Pretende-se agora analisar a fragmentação de pacotes IP. Usando o wireshark, capture e observe o tráfego gerado depois do tamanho de pacote ter sido definido para (3xy0) bytes, em que xy é o número do seu grupo de trabalho (e.g., o grupo PL19 deve usar um tamanho de pacote de 3190 bytes). De modo a poder visualizar os fragmentos, aceda a Edit -> Preferences -> Protocols e em IPv4 desative a opção “Reassemble fragmented IPv4 datagrams”. Documente e justifique todas as respostas às seguintes alíneas:

Número de bytes do pacote: 3910.

25	1.566029	172.26.71.1	35.186.224.39	TCP	66	64658 → 443 [ACK] Seq=29 Ack=25 Win=2047 Len=0 TSval=2453174835 TSecr=831303286
26	1.595928	172.26.71.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0xa899, seq=1/256, ttl=1 (no response found!)
27	1.595953	172.26.71.1	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=a89a)
28	1.595964	172.26.71.1	193.136.9.240	IPv4	964	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=a89a)
29	1.599923	172.26.254.254	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
30	1.606028	172.26.71.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0xa899, seq=2/512, ttl=1 (no response found!)
31	1.606046	172.26.71.1	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=a89b)
32	1.606059	172.26.71.1	193.136.9.240	IPv4	964	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=a89b)
33	1.604418	172.26.254.254	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
34	1.604577	172.26.71.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0xa899, seq=3/768, ttl=1 (no response found!)
35	1.604592	172.26.71.1	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=a89c)
36	1.604603	172.26.71.1	193.136.9.240	IPv4	964	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=a89c)
37	1.607862	172.26.254.254	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
38	1.608058	172.26.71.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0xa899, seq=4/1024, ttl=2 (no response found!)
39	1.608059	172.26.71.1	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=a89d)
40	1.608061	172.26.71.1	193.136.9.240	IPv4	964	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=a89d)
41	1.611952	172.16.2.1	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
42	1.612989	172.26.71.1	193.137.16.65	DNS	83	Standard query 0xaf4c PTR 1.2.16.172.in-addr.arpa
43	1.616743	193.137.16.65	172.26.71.1	DNS	83	Standard query response 0xaf4c Refused PTR 1.2.16.172.in-addr.arpa
44	1.617082	172.26.71.1	193.137.16.145	DNS	83	Standard query 0xaf4c PTR 1.2.16.172.in-addr.arpa
45	1.621279	193.137.16.145	172.26.71.1	DNS	83	Standard query response 0xaf4c Refused PTR 1.2.16.172.in-addr.arpa
46	1.621567	172.26.71.1	193.137.16.75	DNS	83	Standard query 0xaf4c PTR 1.2.16.172.in-addr.arpa
47	1.626003	193.137.16.75	172.26.71.1	DNS	83	Standard query response 0xaf4c Refused PTR 1.2.16.172.in-addr.arpa
48	1.626476	172.26.71.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0xa899, seq=5/1280, ttl=2 (no response found!)
49	1.626493	172.26.71.1	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=a89e)
50	1.626503	172.26.71.1	193.136.9.240	IPv4	964	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=a89e)
51	1.630832	172.16.2.1	172.26.71.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
52	1.630848	172.26.71.1	193.136.9.240	ICMP	1514	Echo (ping) request id=0xa899, seq=6/1536, ttl=2 (no response found!)

```
> Frame 85: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface en0, id 0
> Ethernet II, Src: Apple_9c:05:d4 (00:88:0c:9c:05:d4), Dst: ComdaEnterpr_fff:94:00 (00:00:00:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.71.1, 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: 0xa8a3 (43171)
  > 001. .... = Flags: 0x1, More fragments
  ...0 0000 0000 0000 = Fragment Offset: 0
> Time To Live: 4
  Protocol: ICMP (1)
  Header Checksum: 0x29ea [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.71.1
```

Figura 9: Tráfego ICMP

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

Resposta: Devido ao tamanho do datagrama exceder o limite máximo de bytes permitidos

para envio em uma única transmissão, foi essencial realizar a fragmentação da primeira mensagem ICMP.

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

```
Frame 26: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface en0, id 0
Ethernet II, Src: Apple_9c:05:d4 (d0:88:0c:9c:05:d4), Dst: ComdaEnterpr_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.71.1, 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: 0xa89a (43162)
> 001. .... = Flags: 0x1, More fragments
  ...0 0000 0000 0000 = Fragment Offset: 0
> Time to Live: 1
  Protocol: ICMP (1)
  Header Checksum: 0x2cf3 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.71.1
  Destination Address: 193.136.9.240
Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x4f65 [unverified] [fragmented datagram]
  [Checksum Status: Unverified]
  Identifier (BE): 43161 (0xa899)
  Identifier (LE): 39336 (0x99a8)
  Sequence Number (BE): 1 (0x0001)
  Sequence Number (LE): 256 (0x0100)
> [No response seen]
> Data (1472 bytes)
```

Figura 10: Primeiro fragmento do datagrama IP original

Resposta: Através do campo Flags, é possível obter a indicação de que o pacote foi fragmentado. Ao verificar que a flag "Mais fragmentos" possui o valor 1, podemos concluir que existem mais fragmentos associados a este datagrama. No campo "Deslocamento de fragmento", observamos que o valor é 0, o que nos leva a concluir que se trata do primeiro fragmento. O tamanho do datagrama IP é de 1500 (Indicado no Total Length).

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 27: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface en0, id 0
> Ethernet II, Src: Apple_9c:05:d4 (d0:88:0c:9c:05:d4), Dst: ComdaEnterpr_ff:94:00 (00:d0:03:ff:94:00)
< Internet Protocol Version 4, Src: 172.26.71.1, 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: 0xa89a (43162)
> 001. .... = Flags: 0x1, More fragments
  ...0 0000 1011 1001 = Fragment Offset: 1480
> Time to Live: 1
  Protocol: ICMP (1)
  Header Checksum: 0x2c3a [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.71.1
  Destination Address: 193.136.9.240
> Data (1480 bytes)

```

Figura 11: Segundo fragmento do datagrama IP original

Resposta: Como foi dito na questão anterior, sabemos que se trata do primeiro fragmento quando o valor da flag 'Fragment Offset' é 0 e o valor de 'More fragments' é 1. Neste caso, o valor da flag 'Fragment Offset' é diferente de zero, o que nos leva a concluir que não é o primeiro fragmento. Uma vez que o valor da flag 'More fragments' continua a ter o valor 1 podemos concluir que ainda existem mais fragmentos.

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

Resposta: Para determinar o número de fragmentos gerados, é necessário primeiramente obter o "Maximum Transmission Unit" (MTU) de 1500 bytes e o tamanho do cabeçalho IP de 20 bytes (Header Length). Subtraindo o tamanho do cabeçalho IP do valor do MTU ($1500 - 20 = 1480$ bytes), obtemos o resultado. Para descobrir o número de fragmentos, dividimos 3910 por 1480, resultando em aproximadamente 2,64. Como o número de fragmentos deve ser um número inteiro, arredondamos para cima, obtendo 3 fragmentos. Para determinar o número de bytes no último fragmento, subtraímos 3910 por 2 vezes 1480, resultando em 950 bytes. Em teoria, o datagrama é fragmentado em 3 fragmentos, sendo transportados 950 bytes no último fragmento.

```

> Frame 28: 964 bytes on wire (7712 bits), 964 bytes captured (7712 bits) on interface en0, id 0
> Ethernet II, Src: Apple_9c:05:d4 (d0:88:0c:9c:05:d4), Dst: ComdaEnterpr_ff:94:00 (00:d0:03:ff:94:00)
  Internet Protocol Version 4, Src: 172.26.71.1, 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: 950
      Identification: 0xa89a (43162)
    < 000. .... = Flags: 0x0
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
      ...0 0001 0111 0010 = Fragment Offset: 2960
    > Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x4da7 [validation disabled]
      [Header checksum status: Unverified]
    Source Address: 172.26.71.1
    Destination Address: 193.136.9.240
  > Data (930 bytes)

```

Figura 12: Terceiro Fragmento do Datagrama IP Original

Ao analisarmos o Wireshark é possível confirmar que o datagrama foi fragmentado em 3 fragmentos uma vez que no terceiro fragmento a flag “More fragments” tem valor 0. Podemos ainda verificar que o número de bytes transportados no último datagrama é igual ao valor calculado anteriormente (Total Length: 950).

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

Resposta: Para identificar o último fragmento de um datagrama IP, podemos utilizar 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 == 0xa89a, onde 0xa89a é o identificador do primeiro datagrama IP segmentado.

ip.flags.mf == 0 && ip.id == 0xa89a					
No.	Time	Source	Destination	Protocol	Length/Info
28	1.595964	172.26.71.1	193.136.9.240	IPv4	964 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=a89a)
29	1.599923	172.26.254.254	172.26.71.1	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

Figura 13: Resultados Depois da Aplicação do Filtro

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

Resposta: A reconstrução é efetuada no destino final, que é o dispositivo que recebe os pacotes. Isto acontece devido ao facto de que cada fragmento contém informações sobre sua posição no datagrama original, permitindo que o dispositivo de destino reconstrua o datagrama por completo. A reconstrução poderia ser realizada se um router ao longo do caminho estivesse configurado para tal, entretanto, nenhum router está configurado para desempenhar essa função neste caso.

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

Resposta:

No cabeçalho IP, os campos que sofrem alterações são o "Fragment Offset" e o "More fragments". A flag "fragment offset" informa a ordem em que os fragmentos devem ser organizados, visto que aparecem em ordem crescente, enquanto a flag "more fragments" possibilita verificar se o datagrama original possui mais fragmentos. Ao combinar essas duas flags, é factível reconstruir o datagrama original.

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

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.

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

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.

j) Sabendo que no comando ping a opção -f (Windows), -M do (Linux) ou –D (Mac) ativa a flag “Don’t Fragment” (DF) no cabeçalho do IPv4, usando ping 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:

```
[margarida@MacBook-Pro-de-Margarida ~ % ping -D -s 1473 marco.uminho.pt ]
PING marco.uminho.pt (193.136.9.240): 1473 data bytes
ping: sendto: Message too long
ping: sendto: Message too long
Request timeout for icmp_seq 0
ping: sendto: Message too long
Request timeout for icmp_seq 1
ping: sendto: Message too long
Request timeout for icmp_seq 2
ping: sendto: Message too long
Request timeout for icmp_seq 3
ping: sendto: Message too long
Request timeout for icmp_seq 4
ping: sendto: Message too long
Request timeout for icmp_seq 5
ping: sendto: Message too long
Request timeout for icmp_seq 6
^C
--- marco.uminho.pt ping statistics ---
8 packets transmitted, 0 packets received, 100.0% packet loss
[margarida@MacBook-Pro-de-Margarida ~ % ping -D -s 1472 marco.uminho.pt ]
PING marco.uminho.pt (193.136.9.240): 1472 data bytes
1480 bytes from 193.136.9.240: icmp_seq=0 ttl=61 time=3.448 ms
1480 bytes from 193.136.9.240: icmp_seq=1 ttl=61 time=2.752 ms
1480 bytes from 193.136.9.240: icmp_seq=2 ttl=61 time=5.084 ms
1480 bytes from 193.136.9.240: icmp_seq=3 ttl=61 time=3.282 ms
1480 bytes from 193.136.9.240: icmp_seq=4 ttl=61 time=3.898 ms
1480 bytes from 193.136.9.240: icmp_seq=5 ttl=61 time=3.299 ms
1480 bytes from 193.136.9.240: icmp_seq=6 ttl=61 time=3.795 ms
1480 bytes from 193.136.9.240: icmp_seq=7 ttl=61 time=3.303 ms
1480 bytes from 193.136.9.240: icmp_seq=8 ttl=61 time=3.403 ms
1480 bytes from 193.136.9.240: icmp_seq=9 ttl=61 time=3.428 ms
1480 bytes from 193.136.9.240: icmp_seq=10 ttl=61 time=4.236 ms
1480 bytes from 193.136.9.240: icmp_seq=11 ttl=61 time=3.714 ms
1480 bytes from 193.136.9.240: icmp_seq=12 ttl=61 time=3.193 ms
1480 bytes from 193.136.9.240: icmp_seq=13 ttl=61 time=2.778 ms
1480 bytes from 193.136.9.240: icmp_seq=14 ttl=61 time=2.615 ms
1480 bytes from 193.136.9.240: icmp_seq=15 ttl=61 time=3.436 ms
1480 bytes from 193.136.9.240: icmp_seq=16 ttl=61 time=4.224 ms
1480 bytes from 193.136.9.240: icmp_seq=17 ttl=61 time=2.718 ms
^C
--- marco.uminho.pt ping statistics ---
18 packets transmitted, 18 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 2.615/3.478/5.084/0.604 ms
```

Figura 14: Comando ping com Size = 1472 e Size = 1473

Ao utilizar o comando ping com as opções corretas, é possível determinar que o tamanho máximo para evitar a fragmentação do pacote é de 1472 bytes. Esse valor é resultado 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, deixando assim 1472 bytes para o conteúdo do pacote.

2. Parte 2

2.1. Exercício 1

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

a) De modo a garantir uma posição estratégicamente mais vantajosa e ter casa de férias para relaxar entre batalhas, ordena a construção de um segundo Castelo, em Braga. Não tendo qualquer queixa do serviço prestado, recorre aos serviços do ISP ReiDaNet, que já utiliza no condado, para ter acesso à rede no segundo Castelo. O ISP atribuiu-lhe o endereço de rede IP 172.XX.33.128/26 em que XX corresponde ao seu número de grupo (PLXX). Defina um esquema de endereçamento que permita o estabelecimento de pelo menos 3 redes e que garanta que cada uma destas possa ter 12 ou mais hosts. Assuma que todos os endereços de sub-redes são utilizáveis.

Resposta:

Sub-rede 1 : 172.91.33.128/26

Intervalo de endereços de host: 172.12.33.129 – 172.12.33.142

Endereço de Broadcast: 172.12.33.143

Sub-rede 2 : 172.91.33.144/28

Intervalo de endereços de host: 172.12.33.145 – 172.12.33.158

Endereço de Broadcast: 172.12.33.159

Sub-rede 3 : 172.91.33.160/28

Intervalo de endereços de host: 172.12.33.161 – 172.12.33.174

Endereço de Broadcast: 172.12.33.175

b) Ligue um novo host Castelo2 diretamente ao router ReiDaNet. Associe-lhe um endereço, à sua escolha, pertencente a uma sub-rede disponível das criadas na alínea anterior (garanta que a interface do router ReiDaNet utiliza o primeiro endereço da sub-rede escolhida). Verifique que tem conectividade com os dispositivos do Condado Portucalense.

Resposta:

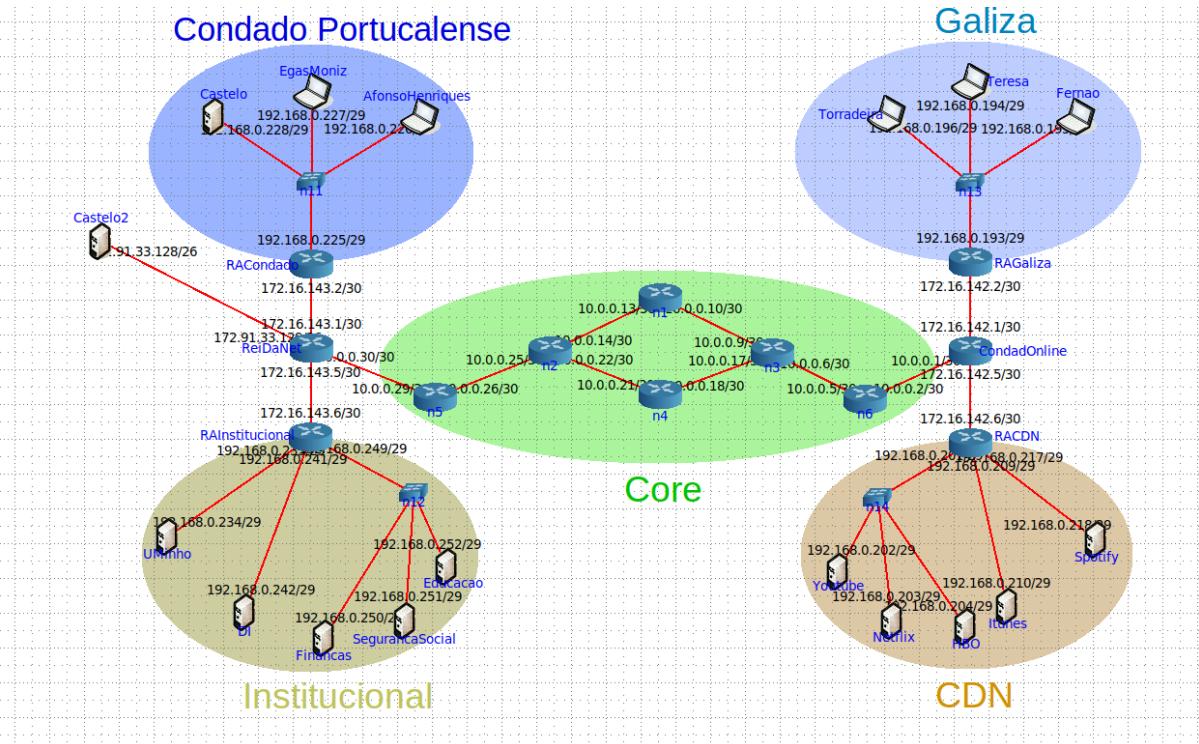


Figura 15: Topologia da Rede Utilizada

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.36427/Castelo2.conf# route add -net 192.168.0.228 ne>
root@Castelo2:/tmp/pycore.36427/Castelo2.conf# route add -net 192.168.0.227 ne>
root@Castelo2:/tmp/pycore.36427/Castelo2.conf# route add -net 192.168.0.226 ne>
root@Castelo2:/tmp/pycore.36427/Castelo2.conf# route add -net 192.168.0.234 ne>
root@Castelo2:/tmp/pycore.36427/Castelo2.conf# route add -net 192.168.0.242 ne>
root@Castelo2:/tmp/pycore.36427/Castelo2.conf# route add -net 192.168.0.250 ne>
root@Castelo2:/tmp/pycore.36427/Castelo2.conf# route add -net 192.168.0.251 ne>
root@Castelo2:/tmp/pycore.36427/Castelo2.conf# route add -net 192.168.0.252 ne>
```

Figura 16: Comandos para adicionar as Rotas necessárias ao Castelo 2.

```
root@Castelo2:/tmp/pycore.36427/Castelo2.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
172.91.33.128  0.0.0.0        255.255.255.192 U        0 0          0 eth0
192.168.0.226  172.91.33.128  255.255.255.255 UGH      0 0          0 eth0
192.168.0.227  172.91.33.128  255.255.255.255 UGH      0 0          0 eth0
192.168.0.228  172.91.33.128  255.255.255.255 UGH      0 0          0 eth0
192.168.0.234  172.91.33.128  255.255.255.255 UGH      0 0          0 eth0
192.168.0.242  172.91.33.128  255.255.255.255 UGH      0 0          0 eth0
192.168.0.250  172.91.33.128  255.255.255.255 UGH      0 0          0 eth0
192.168.0.251  172.91.33.128  255.255.255.255 UGH      0 0          0 eth0
192.168.0.252  172.91.33.128  255.255.255.255 UGH      0 0          0 eth0
```

Figura 17: Comando netstat -rn

2.2. Exercício 2

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

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

```
<pycore.36427/AfonsoHenriques.conf# ping 192.168.0.250
PING 192.168.0.250 (192.168.0.250) 56(84) bytes of data.
64 bytes from 192.168.0.250: icmp_seq=1 ttl=61 time=0.113 ms
64 bytes from 192.168.0.250: icmp_seq=2 ttl=61 time=0.066 ms
64 bytes from 192.168.0.250: icmp_seq=3 ttl=61 time=0.084 ms
64 bytes from 192.168.0.250: icmp_seq=4 ttl=61 time=0.065 ms
64 bytes from 192.168.0.250: icmp_seq=5 ttl=61 time=0.090 ms
64 bytes from 192.168.0.250: icmp_seq=6 ttl=61 time=0.038 ms
^C
--- 192.168.0.250 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5123ms
rtt min/avg/max/mdev = 0.038/0.076/0.113/0.023 ms
root@AfonsoHenriques:/tmp/pycore.36427/AfonsoHenriques.conf# █
```

Figura 18: Ping de AfonsoHenriques para o servidor Financas

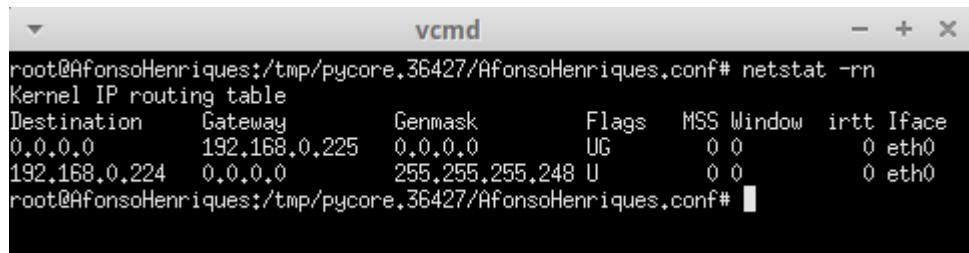
```
<pycore.36427/AfonsoHenriques.conf# ping 172.16.142.6
PING 172.16.142.6 (172.16.142.6) 56(84) bytes of data.
From 172.16.143.1 icmp_seq=1 Destination Net Unreachable
From 172.16.143.1 icmp_seq=2 Destination Net Unreachable
From 172.16.143.1 icmp_seq=3 Destination Net Unreachable
From 172.16.143.1 icmp_seq=4 Destination Net Unreachable
^C
--- 172.16.142.6 ping statistics ---
14 packets transmitted, 0 received, +4 errors, 100% packet loss, time 13296ms
root@AfonsoHenriques:/tmp/pycore.36427/AfonsoHenriques.conf# █
```

Figura 19: Ping de AfonsoHenriques para os servidores do CDN

Resposta: Por meio da execução do comando ping, é evidente que AfonsoHenriques estabelece com sucesso conexão tanto com o servidor Finanças quanto com os servidores da CDN.

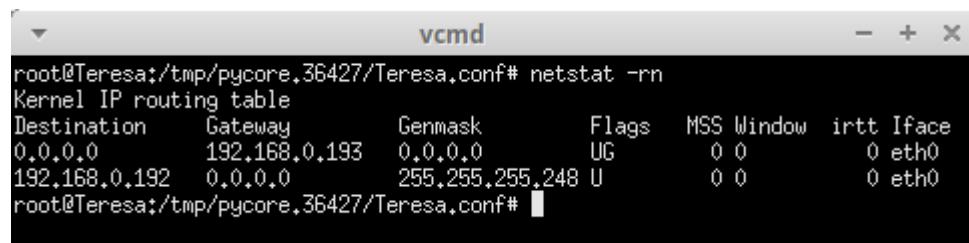
- b) Recorrendo ao comando netstat -rn, analise as tabelas de encaminhamento dos dispositivos AfonsoHenriques e Teresa. Existe algum problema com as suas entradas? Identifique e descreva a utilidade de cada uma das entradas destes dois hosts.

Resposta:



```
vcmd
root@AfonsoHenriques:/tmp/pycore.36427/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.36427/AfonsoHenriques.conf#
```

Figura 20: Comando -netstat -rn do dispositivo AfonsoHenriques



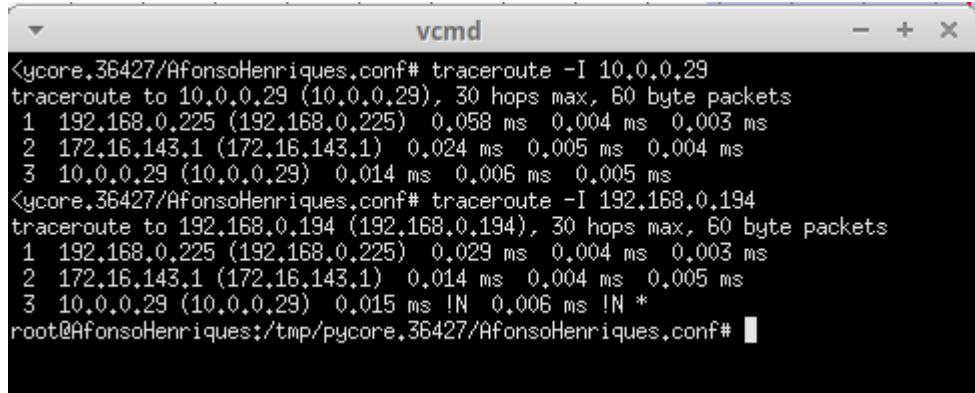
```
vcmd
root@Teresa:/tmp/pycore.36427/Teresa.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
0.0.0.0         192.168.0.193  0.0.0.0        UG        0 0          0 eth0
192.168.0.192  0.0.0.0        255.255.255.248 U          0 0          0 eth0
root@Teresa:/tmp/pycore.36427/Teresa.conf#
```

Figura 21: Comando -netstat -rn do dispositivo Teresa

Na Figura 20, 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 acessado diretamente pela interface de rede eth0, sem a 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). Similarmente, embora com valores diferentes, não há problemas visíveis nas configurações para o dispositivo Teresa, com uma explicação análoga. Em ambos os casos, essas rotas são essenciais para garantir o correto roteamento dos pacotes de rede no sistema e para que cheguem aos seus destinos corretos. A utilidade de cada entrada varia conforme o contexto e a configuração específica do sistema, sendo comumente empregadas 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:



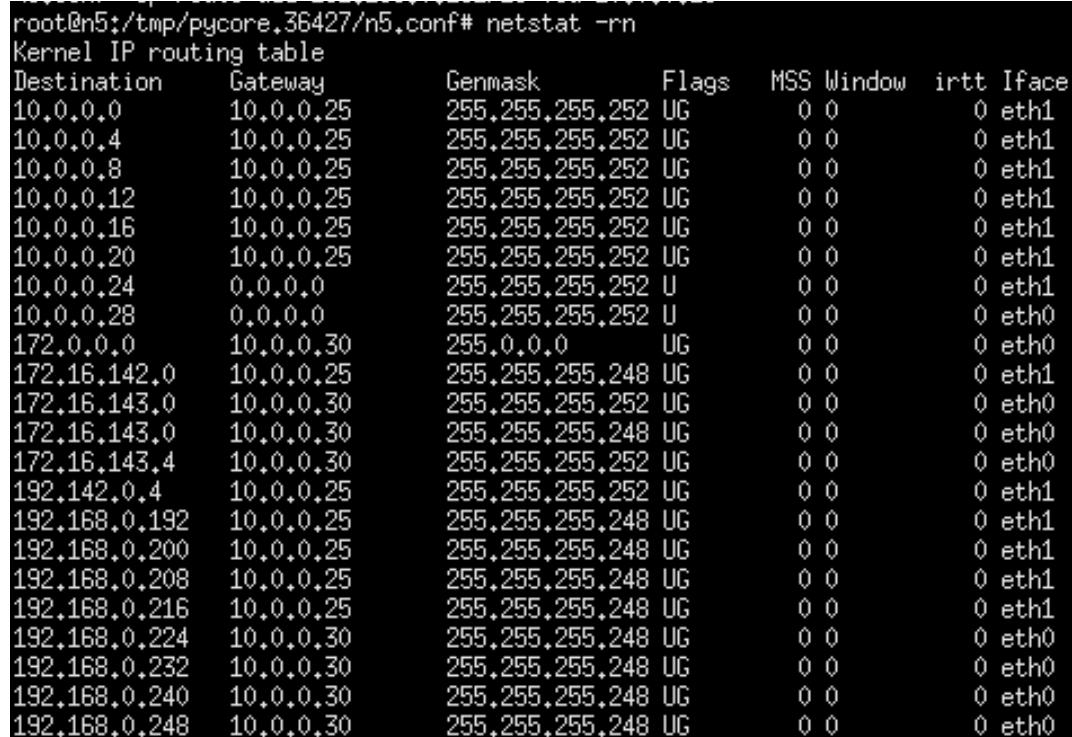
```
<pycore.36427/AfonsoHenriques.conf# traceroute -I 10.0.0.29
traceroute to 10.0.0.29 (10.0.0.29), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.058 ms  0.004 ms  0.003 ms
 2  172.16.143.1 (172.16.143.1)  0.024 ms  0.005 ms  0.004 ms
 3  10.0.0.29 (10.0.0.29)  0.014 ms  0.006 ms  0.005 ms
<pycore.36427/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.029 ms  0.004 ms  0.003 ms
 2  172.16.143.1 (172.16.143.1)  0.014 ms  0.004 ms  0.005 ms
 3  10.0.0.29 (10.0.0.29)  0.015 ms !N  0.006 ms !N *
root@AfonsoHenriques:/tmp/pycore.36427/AfonsoHenriques.conf#
```

Figura 22: Comando Traceroute

Analizando a imagem, nota-se uma discrepância no endereço IP 10.0.0.29, indicando que o dispositivo n5 não está realizando o encaminhamento adequado dos pacotes. Diante disso, é necessário incluir uma rota para resolver essa falha.

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

Figura 23: Comando -add



```
root@n5:/tmp/pycore.36427/n5.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
10.0.0.0        10.0.0.25      255.255.255.252 UG        0 0          0 eth1
10.0.0.4        10.0.0.25      255.255.255.252 UG        0 0          0 eth1
10.0.0.8        10.0.0.25      255.255.255.252 UG        0 0          0 eth1
10.0.0.12       10.0.0.25      255.255.255.252 UG        0 0          0 eth1
10.0.0.16       10.0.0.25      255.255.255.252 UG        0 0          0 eth1
10.0.0.20       10.0.0.25      255.255.255.252 UG        0 0          0 eth1
10.0.0.24       0.0.0.0        255.255.255.252 U        0 0          0 eth1
10.0.0.28       0.0.0.0        255.255.255.252 U        0 0          0 eth0
172.0.0.0        10.0.0.30      255.0.0.0        UG       0 0          0 eth0
172.16.142.0    10.0.0.25      255.255.255.248 UG       0 0          0 eth1
172.16.143.0    10.0.0.30      255.255.255.252 UG       0 0          0 eth0
172.16.143.0    10.0.0.30      255.255.255.248 UG       0 0          0 eth0
172.16.143.4    10.0.0.30      255.255.255.252 UG       0 0          0 eth0
192.142.0.4     10.0.0.25      255.255.255.252 UG       0 0          0 eth1
192.168.0.192   10.0.0.25      255.255.255.248 UG       0 0          0 eth1
192.168.0.200   10.0.0.25      255.255.255.248 UG       0 0          0 eth1
192.168.0.208   10.0.0.25      255.255.255.248 UG       0 0          0 eth1
192.168.0.216   10.0.0.25      255.255.255.248 UG       0 0          0 eth1
192.168.0.224   10.0.0.30      255.255.255.248 UG       0 0          0 eth0
192.168.0.232   10.0.0.30      255.255.255.248 UG       0 0          0 eth0
192.168.0.240   10.0.0.30      255.255.255.248 UG       0 0          0 eth0
192.168.0.248   10.0.0.30      255.255.255.248 UG       0 0          0 eth0
```

Figura 24: Comando -netstat -rn

```
<ycore.36427/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.028 ms  0.005 ms  0.003 ms
 2  172.16.143.1 (172.16.143.1)  0.014 ms  0.004 ms  0.004 ms
 3  10.0.0.29 (10.0.0.29)  0.015 ms  0.006 ms  0.005 ms
 4  10.0.0.25 (10.0.0.25)  0.035 ms  0.007 ms  0.006 ms
 5  10.0.0.25 (10.0.0.25)  3075.858 ms !H  3075.851 ms !H  3075.848 ms !H
```

Figura 25: Comando -traceroute

Após a adição da nova rota conforme as imagens fornecidas, ainda persiste uma falha no IP 10.0.0.25, com o dispositivo n2 sendo responsável pelo encaminhamento inadequado. Considerando que as rotas de n2 para n1 e de n2 para n4 já estão definidas, é crucial identificar e corrigir a rota incorreta entre esses dois dispositivos.

```
root@n2:/tmp/pycore.36427/n2.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
10.0.0.0         10.0.0.13      255.255.255.252 UG        0 0          0 eth1
10.0.0.4         10.0.0.21      255.255.255.252 UG        0 0          0 eth0
10.0.0.8         10.0.0.13      255.255.255.252 UG        0 0          0 eth1
10.0.0.12        0.0.0.0        255.255.255.252 U          0 0          0 eth1
10.0.0.16        10.0.0.13      255.255.255.252 UG        0 0          0 eth1
10.0.0.20        0.0.0.0        255.255.255.252 U          0 0          0 eth0
10.0.0.24        0.0.0.0        255.255.255.252 U          0 0          0 eth2
10.0.0.28        10.0.0.26      255.255.255.252 UG        0 0          0 eth2
172.0.0.0         10.0.0.26      255.0.0.0        UG        0 0          0 eth2
172.16.142.0     10.0.0.13      255.255.255.252 UG        0 0          0 eth1
172.16.142.4     10.0.0.21      255.255.255.252 UG        0 0          0 eth0
172.16.143.0     10.0.0.26      255.255.255.252 UG        0 0          0 eth2
172.16.143.4     10.0.0.26      255.255.255.252 UG        0 0          0 eth2
192.168.0.192    10.0.0.13      255.255.255.248 UG        0 0          0 eth1
192.168.0.194    10.0.0.25      255.255.255.254 UG        0 0          0 eth2
192.168.0.200    10.0.0.21      255.255.255.248 UG        0 0          0 eth0
192.168.0.208    10.0.0.21      255.255.255.248 UG        0 0          0 eth0
192.168.0.216    10.0.0.21      255.255.255.248 UG        0 0          0 eth0
192.168.0.224    10.0.0.26      255.255.255.248 UG        0 0          0 eth2
192.168.0.232    10.0.0.26      255.255.255.248 UG        0 0          0 eth2
192.168.0.240    10.0.0.26      255.255.255.248 UG        0 0          0 eth2
192.168.0.248    10.0.0.26      255.255.255.248 UG        0 0          0 eth2
<2.conf# route del -net 192.168.0.194 netmask 255.255.255.254
root@n2:/tmp/pycore.36427/n2.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
10.0.0.0         10.0.0.13      255.255.255.252 UG        0 0          0 eth1
10.0.0.4         10.0.0.21      255.255.255.252 UG        0 0          0 eth0
10.0.0.8         10.0.0.13      255.255.255.252 UG        0 0          0 eth1
10.0.0.12        0.0.0.0        255.255.255.252 U          0 0          0 eth1
10.0.0.16        10.0.0.13      255.255.255.252 UG        0 0          0 eth1
10.0.0.20        0.0.0.0        255.255.255.252 U          0 0          0 eth0
10.0.0.24        0.0.0.0        255.255.255.252 U          0 0          0 eth2
10.0.0.28        10.0.0.26      255.255.255.252 UG        0 0          0 eth2
172.0.0.0         10.0.0.26      255.0.0.0        UG        0 0          0 eth2
172.16.142.0     10.0.0.13      255.255.255.252 UG        0 0          0 eth1
172.16.142.4     10.0.0.21      255.255.255.252 UG        0 0          0 eth0
172.16.143.0     10.0.0.26      255.255.255.252 UG        0 0          0 eth2
172.16.143.4     10.0.0.26      255.255.255.252 UG        0 0          0 eth2
192.168.0.192    10.0.0.13      255.255.255.248 UG        0 0          0 eth1
192.168.0.200    10.0.0.21      255.255.255.248 UG        0 0          0 eth0
192.168.0.208    10.0.0.21      255.255.255.248 UG        0 0          0 eth0
192.168.0.216    10.0.0.21      255.255.255.248 UG        0 0          0 eth0
192.168.0.224    10.0.0.26      255.255.255.248 UG        0 0          0 eth2
192.168.0.232    10.0.0.26      255.255.255.248 UG        0 0          0 eth2
192.168.0.240    10.0.0.26      255.255.255.248 UG        0 0          0 eth2
192.168.0.248    10.0.0.26      255.255.255.248 UG        0 0          0 eth2
```

Figura 26: Comando -netstat e Comando -del

Analisando a figura fornecida, fica claro que a rota que desejamos eliminar é aquela que direciona o tráfego de 192.168.0.194 para 10.0.0.25, pois não há intenção de rotear do n1 para o n2.

```
<1.conf# route del -net 192.168.0.192 netmask 255.255.255.248
root@n1:/tmp/pycore.36427/n1.conf# ip route add 192.168.0.192/29 via 10.0.0.9
root@n1:/tmp/pycore.36427/n1.conf#
```

Figura 27: Comando -del e -add

Ao utilizar o comando "-add" conforme ilustrado na Figura 27, inserimos uma nova rota, estabelecendo a conexão do n1 para o n3, ou seja, de 10.0.0.13 para 10.0.0.9.

```
<ycore.36427/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.030 ms  0.004 ms  0.004 ms
 2  172.16.143.1 (172.16.143.1)  0.015 ms  0.006 ms  0.005 ms
 3  10.0.0.29 (10.0.0.29)  0.016 ms  0.007 ms  0.006 ms
 4  10.0.0.25 (10.0.0.25)  0.018 ms  0.008 ms  0.008 ms
 5  10.0.0.13 (10.0.0.13)  0.046 ms  0.010 ms  0.009 ms
 6  10.0.0.17 (10.0.0.17)  0.073 ms  0.018 ms  0.010 ms
 7  10.0.0.5 (10.0.0.5)  0.032 ms  0.011 ms  0.010 ms
 8  10.0.0.1 (10.0.0.1)  0.035 ms  0.013 ms  0.014 ms
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
```

Figura 28: Comando -traceroute

Graças às adaptações feitas, conseguimos direcionar o fluxo de dados até o provedor de serviços de Internet CondadOnline. Isso é evidenciado pela chegada ao endereço IP 10.0.0.1, conforme ilustrado pelo comando -traceroute na figura mencionada.

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

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

Resposta: Os asteriscos na figura 29 indicam que o programa não está recebendo respostas do router.

```
root@Teresa:/tmp/pycore.36427/Teresa.conf# traceroute -I 192.168.0.226
traceroute to 192.168.0.226 (192.168.0.226), 30 hops max, 60 byte packets
 1  192.168.0.193 (192.168.0.193)  0.052 ms !N  0.004 ms !N *
root@Teresa:/tmp/pycore.36427/Teresa.conf#
```

Figura 29: Comando -traceroute

Assim sendo, embora Teresa receba os pacotes de AfonsoHenriques, ela não consegue rotear de volta as respostas para AfonsoHenriques. Observamos, na Figura 30, uma falha no IP 192.168.0.193, indicando a necessidade de adicionar a rota do RAGaliza para o CondadOnline, conforme demonstrado na figura abaixo.

```
root@RAGaliza:/tmp/pycore.36427/RAGaliza.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
10.0.0.0         172.16.142.1   255.255.255.252 UG      0 0          0 eth0
10.0.0.4         172.16.142.1   255.255.255.252 UG      0 0          0 eth0
10.0.0.8         172.16.142.1   255.255.255.252 UG      0 0          0 eth0
10.0.0.12        172.16.142.1   255.255.255.252 UG      0 0          0 eth0
10.0.0.16        172.16.142.1   255.255.255.252 UG      0 0          0 eth0
10.0.0.20        172.16.142.1   255.255.255.252 UG      0 0          0 eth0
10.0.0.24        172.16.142.1   255.255.255.252 UG      0 0          0 eth0
10.0.0.28        172.16.142.1   255.255.255.252 UG      0 0          0 eth0
172.0.0.0         172.16.142.1   255.0.0.0       UG      0 0          0 eth0
172.16.142.0     0.0.0.0       255.255.255.252 U      0 0          0 eth0
172.16.142.4     172.16.142.1   255.255.255.252 UG      0 0          0 eth0
172.16.143.0     172.16.142.1   255.255.255.252 UG      0 0          0 eth0
172.16.143.4     172.16.142.1   255.255.255.252 UG      0 0          0 eth0
192.168.0.192    0.0.0.0       255.255.255.248 U      0 0          0 eth1
192.168.0.200    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.208    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.216    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.232    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.240    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
192.168.0.248    172.16.142.1   255.255.255.248 UG      0 0          0 eth0
<6427/RAGaliza.conf# ip route add 192.168.0.224/29 via 172.16.142.1
```

Figura 30: Comando -netstat -rn e Comando -add

Após adicionar a rota, confirmamos por meio do comando -traceroute que agora conseguimos encaminhar a mensagem até AfonsoHenriques.

```
root@Teresa:/tmp/pycore.36427/Teresa.conf# traceroute -I 192.168.0.226
traceroute to 192.168.0.226 (192.168.0.226), 30 hops max, 60 byte packets
 1  192.168.0.193 (192.168.0.193)  0.026 ms  0.005 ms  0.004 ms
 2  172.16.142.1 (172.16.142.1)  0.013 ms  0.005 ms  0.004 ms
 3  10.0.0.2 (10.0.0.2)  0.015 ms  0.006 ms  0.005 ms
 4  10.0.0.6 (10.0.0.6)  0.028 ms  0.007 ms  0.007 ms
 5  10.0.0.18 (10.0.0.18)  0.020 ms  0.009 ms  0.007 ms
 6  10.0.0.14 (10.0.0.14)  0.024 ms  0.030 ms  0.010 ms
 7  10.0.0.26 (10.0.0.26)  0.022 ms  0.010 ms  0.011 ms
 8  10.0.0.30 (10.0.0.30)  0.019 ms  0.011 ms  0.017 ms
 9  172.16.143.2 (172.16.143.2)  0.022 ms  0.013 ms  0.013 ms
10  192.168.0.226 (192.168.0.226)  0.022 ms  0.015 ms  0.014 ms
root@Teresa:/tmp/pycore.36427/Teresa.conf#
```

Figura 31: Comando -traceroute

- 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.36427/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.029 ms  0.004 ms  0.003 ms
 2  172.16.143.1 (172.16.143.1)  0.014 ms  0.005 ms  0.004 ms
 3  10.0.0.29 (10.0.0.29)  0.016 ms  0.006 ms  0.005 ms
 4  10.0.0.25 (10.0.0.25)  0.015 ms  0.007 ms  0.007 ms
 5  10.0.0.13 (10.0.0.13)  0.016 ms  0.008 ms  0.007 ms
 6  10.0.0.17 (10.0.0.17)  0.022 ms  0.017 ms  0.009 ms
 7  10.0.0.5 (10.0.0.5)  0.018 ms  0.010 ms  0.010 ms
 8  10.0.0.1 (10.0.0.1)  0.020 ms  0.012 ms  0.012 ms
 9  172.16.142.2 (172.16.142.2)  0.021 ms  0.013 ms  0.013 ms
10  192.168.0.194 (192.168.0.194)  0.022 ms  0.014 ms  0.014 ms
root@AfonsoHenriques:/tmp/pycore.36427/AfonsoHenriques.conf# █
```

Figura 32: Comando -traceroute do AfonsoHenriques.

```
root@Teresa:/tmp/pycore.36427/Teresa.conf# traceroute -I 192.168.0.226
traceroute to 192.168.0.226 (192.168.0.226), 30 hops max, 60 byte packets
 1  192.168.0.193 (192.168.0.193)  0.028 ms  0.005 ms  0.003 ms
 2  172.16.142.1 (172.16.142.1)  0.014 ms  0.005 ms  0.004 ms
 3  10.0.0.2 (10.0.0.2)  0.015 ms  0.005 ms  0.006 ms
 4  10.0.0.6 (10.0.0.6)  0.016 ms  0.007 ms  0.007 ms
 5  10.0.0.18 (10.0.0.18)  0.016 ms  0.008 ms  0.009 ms
 6  10.0.0.14 (10.0.0.14)  0.022 ms  0.033 ms  0.011 ms
 7  10.0.0.26 (10.0.0.26)  0.018 ms  0.011 ms  0.010 ms
 8  10.0.0.30 (10.0.0.30)  0.020 ms  0.011 ms  0.012 ms
 9  172.16.143.2 (172.16.143.2)  0.022 ms  0.013 ms  0.013 ms
10  192.168.0.226 (192.168.0.226)  0.027 ms  0.014 ms  0.013 ms
root@Teresa:/tmp/pycore.36427/Teresa.conf# █
```

Figura 33: Comando -traceroute da Teresa.

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

192.168.0.192	192.168.0.194	255.255.255.255/32	v v	v Conn
192.168.0.192	10.0.0.18	255.255.255.240 UG	0 0	0 eth1
192.168.0.192	10.0.0.15	255.255.255.240 UG	1 1	1 eth0

Figura 34: Entrada sugerida pelo enunciado

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: Quando os pacotes são enviados para o polo Galiza e o polo CDN, não há

correspondência na entrada selecionada. Na configuração atual, quando o destino é a Galiza, o Gateway sugerido é 10.0.0.18, o que é problemático, pois seria um caminho de retorno. Isso resultaria em um loop entre o n3 e o n4, conforme mostrado na Figura 36. Portanto, o Gateway adequado a ser usado é 10.0.0.5, garantindo assim o funcionamento esperado do dispositivo.

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

Figura 35: Tabela de encaminhamento de n3.

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.0.0	10.0.0.17	255.255.255.252	UG	0	0	0	eth0
10.0.0.4	10.0.0.17	255.255.255.252	UG	0	0	0	eth0
10.0.0.8	10.0.0.17	255.255.255.252	UG	0	0	0	eth0
10.0.0.12	10.0.0.22	255.255.255.252	UG	0	0	0	eth1
10.0.0.16	0.0.0.0	255.255.255.252	U	0	0	0	eth0
10.0.0.20	0.0.0.0	255.255.255.252	U	0	0	0	eth1
10.0.0.24	10.0.0.22	255.255.255.252	UG	0	0	0	eth1
10.0.0.28	10.0.0.22	255.255.255.252	UG	0	0	0	eth1
172.0.0.0	10.0.0.22	255.0.0.0	UG	0	0	0	eth1
172.16.142.0	10.0.0.17	255.255.255.252	UG	0	0	0	eth0
172.16.142.4	10.0.0.17	255.255.255.252	UG	0	0	0	eth0
172.16.143.0	10.0.0.22	255.255.255.252	UG	0	0	0	eth1
172.16.143.4	10.0.0.22	255.255.255.252	UG	0	0	0	eth1
192.168.0.192	10.0.0.17	255.255.255.248	UG	0	0	0	eth0
192.168.0.200	10.0.0.17	255.255.255.248	UG	0	0	0	eth0
192.168.0.208	10.0.0.17	255.255.255.248	UG	0	0	0	eth0
192.168.0.216	10.0.0.17	255.255.255.248	UG	0	0	0	eth0
192.168.0.224	10.0.0.22	255.255.255.248	UG	0	0	0	eth1
192.168.0.232	10.0.0.22	255.255.255.248	UG	0	0	0	eth1
192.168.0.240	10.0.0.22	255.255.255.248	UG	0	0	0	eth1
192.168.0.248	10.0.0.22	255.255.255.248	UG	0	0	0	eth1

Figura 36: Tabela de encaminhamento de n4.

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

Resposta: Para determinar se os endereços são privados, é crucial verificar se eles se encontram dentro das faixas designadas para endereços IP privados. Estas incluem o intervalo de endereços de Classe A privada, que vai de 0.0.0.0 a 10.255.255.255, o intervalo de endereços de Classe B privada, que vai de 172.16.0.0 a 172.31.255.255, e o intervalo de endereços de Classe C privada, que vai de 192.168.0.0 a 192.168.255.255. Portanto, os endereços utilizados pelos quatro polos são considerados privados, pois todos eles estão dentro do intervalo de Classe C privada: 192.168.0.0 a 192.168.255.255. Além disso, os endereços utilizados no core da rede também são privados, pois todos eles estão dentro do intervalo de Classe A privada: 0.0.0.0 a 10.255.255.255. Esses endereços IP privados são reservados para a comunicação interna entre dispositivos em redes locais.

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

Resposta: Os switches não possuem um endereço IP designado devido à sua natureza como dispositivos de camada 2, o que os limita a encaminhar pacotes com base nos endereços MAC - Media Access Control dos dispositivos conectados a eles.

2.3. Exercício 3

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

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

Resposta: Primeiramente, de acordo com a figura abaixo, foi executado o comando "netstat -rn" no n6 para visualizar os diversos caminhos disponíveis.

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Ifac
10.0.0.0	0.0.0.0	255.255.255.252	U	0	0	0	eth0
10.0.0.4	0.0.0.0	255.255.255.252	U	0	0	0	eth1
10.0.0.8	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
10.0.0.12	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
10.0.0.16	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
10.0.0.20	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
10.0.0.24	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
10.0.0.28	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
172.0.0.0	10.0.0.6	255.0.0.0	UG	0	0	0	eth1
172.16.142.0	10.0.0.1	255.255.255.252	UG	0	0	0	eth0
172.16.142.4	10.0.0.1	255.255.255.252	UG	0	0	0	eth0
172.16.143.0	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
172.16.143.4	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
192.168.0.192	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.200	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.208	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.216	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.224	10.0.0.6	255.255.255.248	UG	0	0	0	eth1
192.168.0.232	10.0.0.6	255.255.255.248	UG	0	0	0	eth1
192.168.0.240	10.0.0.6	255.255.255.248	UG	0	0	0	eth1
192.168.0.248	10.0.0.6	255.255.255.248	UG	0	0	0	eth1

Figura 37: Comando -netstat -rn.

Posteriormente, procedeu-se com a remoção das rotas relacionadas à Galiza e ao CDN, conforme ilustrado na figura 37. Adicionalmente, foi adicionada a rota de rede 192.168.0.192/27, uma vez que a máscara de sub-rede /27 é capaz de abranger todas as redes pertinentes.

```
root@n6:/tmp/pycore.36427/n6.conf# ip route del 192.168.0.192 via 10.0.0.1
RTNETLINK answers: No such process
root@n6:/tmp/pycore.36427/n6.conf# ip route del 192.168.0.192/29 via 10.0.0.1
root@n6:/tmp/pycore.36427/n6.conf# ip route del 192.168.0.200/29 via 10.0.0.1
root@n6:/tmp/pycore.36427/n6.conf# ip route del 192.168.0.208/29 via 10.0.0.1
root@n6:/tmp/pycore.36427/n6.conf# ip route del 192.168.0.216/29 via 10.0.0.1
root@n6:/tmp/pycore.36427/n6.conf# ip route del 192.168.0.192/27 via 10.0.0.1
RTNETLINK answers: No such process
root@n6:/tmp/pycore.36427/n6.conf# ■
```

Figura 38: Comandos -del e -add.

Por fim, foi executado novamente o comando "netstat -rn" e o "traceroute" para verificar a conectividade após as alterações realizadas.

root@n6:/tmp/pycore.36427/n6.conf# netstat -rn							
Kernel IP routing table							
Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Ifac
10.0.0.0	0.0.0.0	255.255.255.252	U	0	0	0	eth0
10.0.0.4	0.0.0.0	255.255.255.252	U	0	0	0	eth1
10.0.0.8	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
10.0.0.12	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
10.0.0.16	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
10.0.0.20	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
10.0.0.24	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
10.0.0.28	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
172.0.0.0	10.0.0.6	255.0.0.0	UG	0	0	0	eth1
172.16.142.0	10.0.0.1	255.255.255.252	UG	0	0	0	eth0
172.16.142.4	10.0.0.1	255.255.255.252	UG	0	0	0	eth0
172.16.143.0	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
172.16.143.4	10.0.0.6	255.255.255.252	UG	0	0	0	eth1
192.168.0.192	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.200	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.208	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.216	10.0.0.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.224	10.0.0.6	255.255.255.248	UG	0	0	0	eth1
192.168.0.232	10.0.0.6	255.255.255.248	UG	0	0	0	eth1
192.168.0.240	10.0.0.6	255.255.255.248	UG	0	0	0	eth1
192.168.0.248	10.0.0.6	255.255.255.248	UG	0	0	0	eth1

Figura 39: Comando -netstat -rn.

```
</7/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.034 ms  0.006 ms  0.006 ms
 2  172.16.143.1 (172.16.143.1)  0.014 ms  0.005 ms  0.004 ms
 3  10.0.0.29 (10.0.0.29)  0.015 ms  0.006 ms  0.006 ms
 4  10.0.0.25 (10.0.0.25)  0.017 ms  0.008 ms  0.007 ms
 5  10.0.0.13 (10.0.0.13)  0.020 ms  0.008 ms  0.008 ms
 6  10.0.0.17 (10.0.0.17)  0.028 ms  0.018 ms  0.010 ms
 7  10.0.0.5 (10.0.0.5)  0.023 ms !N  0.011 ms !N *
<7/AfonsoHenriques.conf# traceroute -I 192.168.0.218
traceroute to 192.168.0.218 (192.168.0.218), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.029 ms  0.005 ms  0.003 ms
 2  172.16.143.1 (172.16.143.1)  0.015 ms  0.004 ms  0.005 ms
 3  10.0.0.29 (10.0.0.29)  0.015 ms  0.006 ms  0.006 ms
 4  10.0.0.25 (10.0.0.25)  0.015 ms  0.006 ms  0.007 ms
 5  10.0.0.21 (10.0.0.21)  0.018 ms  0.011 ms  0.008 ms
 6  10.0.0.17 (10.0.0.17)  0.018 ms  0.017 ms  0.010 ms
 7  10.0.0.5 (10.0.0.5)  0.018 ms !N  0.011 ms !N *
<7/AfonsoHenriques.conf# traceroute -I 192.168.0.204
traceroute to 192.168.0.204 (192.168.0.204), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.024 ms  0.004 ms  0.003 ms
 2  172.16.143.1 (172.16.143.1)  0.013 ms  0.004 ms  0.004 ms
 3  10.0.0.29 (10.0.0.29)  0.015 ms  0.007 ms  0.005 ms
 4  10.0.0.25 (10.0.0.25)  0.015 ms  0.006 ms  0.006 ms
 5  10.0.0.21 (10.0.0.21)  0.015 ms  0.007 ms  0.007 ms
 6  10.0.0.17 (10.0.0.17)  0.016 ms  0.016 ms  0.009 ms
 7  10.0.0.5 (10.0.0.5)  0.017 ms !N  0.010 ms !N *
<7/AfonsoHenriques.conf# traceroute -I 192.168.0.210
traceroute to 192.168.0.210 (192.168.0.210), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.027 ms  0.004 ms  0.003 ms
 2  172.16.143.1 (172.16.143.1)  0.015 ms  0.004 ms  0.004 ms
 3  10.0.0.29 (10.0.0.29)  0.015 ms  0.006 ms  0.005 ms
 4  10.0.0.25 (10.0.0.25)  0.014 ms  0.007 ms  0.006 ms
 5  10.0.0.21 (10.0.0.21)  0.014 ms  0.011 ms  0.010 ms
 6  10.0.0.17 (10.0.0.17)  0.016 ms  0.017 ms  0.010 ms
 7  10.0.0.5 (10.0.0.5)  0.018 ms !N  0.010 ms !N *
```

Figura 40: Comando -traceroute para Teresa, Spotify, HBO e Itunes.

```

root@n6:/tmp/pycore.36427/n6.conf# ping 192.168.0.204
PING 192.168.0.204 (192.168.0.204) 56(84) bytes of data.
64 bytes from 192.168.0.204: icmp_seq=1 ttl=62 time=0.058 ms
64 bytes from 192.168.0.204: icmp_seq=2 ttl=62 time=0.120 ms
64 bytes from 192.168.0.204: icmp_seq=3 ttl=62 time=0.131 ms
64 bytes from 192.168.0.204: icmp_seq=4 ttl=62 time=0.105 ms
64 bytes from 192.168.0.204: icmp_seq=5 ttl=62 time=0.124 ms
^C
--- 192.168.0.204 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4083ms
rtt min/avg/max/mdev = 0.058/0.107/0.131/0.026 ms

```

Figura 41: Comando ping para HBO.

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

```

root@n6:/tmp/pycore.36427/n6.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
10.0.0.0         0.0.0.0       255.255.255.252 U        0 0          0 eth0
10.0.0.4         0.0.0.0       255.255.255.252 U        0 0          0 eth1
10.0.0.8         10.0.0.6      255.255.255.252 UG       0 0          0 eth1
10.0.0.12        10.0.0.6      255.255.255.252 UG       0 0          0 eth1
10.0.0.16        10.0.0.6      255.255.255.252 UG       0 0          0 eth1
10.0.0.20        10.0.0.6      255.255.255.252 UG       0 0          0 eth1
10.0.0.24        10.0.0.6      255.255.255.252 UG       0 0          0 eth1
10.0.0.28        10.0.0.6      255.255.255.252 UG       0 0          0 eth1
172.0.0.0         10.0.0.6      255.0.0.0        UG       0 0          0 eth1
172.16.142.0     10.0.0.1      255.255.255.252 UG       0 0          0 eth0
172.16.142.4     10.0.0.1      255.255.255.252 UG       0 0          0 eth0
172.16.143.0     10.0.0.6      255.255.255.252 UG       0 0          0 eth1
172.16.143.4     10.0.0.6      255.255.255.252 UG       0 0          0 eth1
192.168.0.192    10.0.0.1      255.255.255.248 UG       0 0          0 eth0
192.168.0.224    10.0.0.6      255.255.255.224 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.36427/n6.conf# █

```

Figura 42: Comando -netstat -rn.

```

root@Teresa:/tmp/pycore.36427/Teresa.conf# traceroute -I 192.168.0.226
traceroute to 192.168.0.226 (192.168.0.226), 30 hops max, 60 byte packets
 1  192.168.0.193 (192.168.0.193)  0.050 ms  0.005 ms  0.005 ms
 2  172.16.142.1 (172.16.142.1)  0.018 ms  0.008 ms  0.008 ms
 3  10.0.0.2 (10.0.0.2)  0.038 ms  0.012 ms  0.011 ms
 4  10.0.0.6 (10.0.0.6)  0.028 ms  0.015 ms  0.013 ms
 5  10.0.0.18 (10.0.0.18)  0.030 ms  0.037 ms  0.020 ms
 6  10.0.0.14 (10.0.0.14)  0.045 ms  0.046 ms  0.019 ms
 7  10.0.0.26 (10.0.0.26)  0.038 ms  0.020 ms  0.020 ms
 8  10.0.0.30 (10.0.0.30)  0.031 ms  0.050 ms  0.026 ms
 9  172.16.143.2 (172.16.143.2)  0.035 ms  0.027 ms  0.025 ms
10  192.168.0.226 (192.168.0.226)  0.042 ms  0.028 ms  0.028 ms
root@Teresa:/tmp/pycore.36427/Teresa.conf# traceroute -I 192.168.0.234
traceroute to 192.168.0.234 (192.168.0.234), 30 hops max, 60 byte packets
 1  192.168.0.193 (192.168.0.193)  0.035 ms  0.035 ms  0.008 ms
 2  172.16.142.1 (172.16.142.1)  0.017 ms  0.007 ms  0.007 ms
 3  10.0.0.2 (10.0.0.2)  0.021 ms  0.010 ms  0.010 ms
 4  10.0.0.6 (10.0.0.6)  0.041 ms  0.012 ms  0.012 ms
 5  10.0.0.10 (10.0.0.10)  0.040 ms  0.015 ms  0.016 ms
 6  10.0.0.14 (10.0.0.14)  0.028 ms  0.030 ms  0.017 ms
 7  10.0.0.26 (10.0.0.26)  0.027 ms  0.058 ms  0.030 ms
 8  10.0.0.30 (10.0.0.30)  0.031 ms  0.023 ms  0.026 ms
 9  172.16.143.6 (172.16.143.6)  0.063 ms  0.026 ms  0.025 ms
10  192.168.0.234 (192.168.0.234)  0.059 ms  0.030 ms  0.029 ms
root@Teresa:/tmp/pycore.36427/Teresa.conf# traceroute -I 192.168.0.242
traceroute to 192.168.0.242 (192.168.0.242), 30 hops max, 60 byte packets
 1  192.168.0.193 (192.168.0.193)  0.034 ms  0.005 ms  0.005 ms
 2  172.16.142.1 (172.16.142.1)  0.017 ms  0.008 ms  0.008 ms
 3  10.0.0.2 (10.0.0.2)  0.019 ms  0.009 ms  0.010 ms
 4  10.0.0.6 (10.0.0.6)  0.020 ms  0.013 ms  0.011 ms
 5  10.0.0.10 (10.0.0.10)  0.022 ms  0.014 ms  0.014 ms
 6  10.0.0.14 (10.0.0.14)  0.024 ms  0.031 ms  0.017 ms
 7  10.0.0.26 (10.0.0.26)  0.026 ms  0.018 ms  0.018 ms
 8  10.0.0.30 (10.0.0.30)  0.030 ms  0.022 ms  0.021 ms
 9  172.16.143.6 (172.16.143.6)  0.032 ms  0.024 ms  0.024 ms
10  192.168.0.242 (192.168.0.242)  0.067 ms  0.025 ms  0.025 ms
root@Teresa:/tmp/pycore.36427/Teresa.conf# traceroute -I 192.168.0.252
traceroute to 192.168.0.252 (192.168.0.252), 30 hops max, 60 byte packets
 1  192.168.0.193 (192.168.0.193)  0.033 ms  0.005 ms  0.005 ms
 2  172.16.142.1 (172.16.142.1)  0.017 ms  0.007 ms  0.007 ms
 3  10.0.0.2 (10.0.0.2)  0.019 ms  0.010 ms  0.009 ms
 4  10.0.0.6 (10.0.0.6)  0.020 ms  0.015 ms  0.012 ms
 5  10.0.0.10 (10.0.0.10)  0.023 ms  0.015 ms  0.016 ms
 6  10.0.0.14 (10.0.0.14)  0.026 ms  0.028 ms  0.016 ms
 7  10.0.0.26 (10.0.0.26)  0.026 ms  0.019 ms  0.020 ms
 8  10.0.0.30 (10.0.0.30)  0.028 ms  0.022 ms  0.021 ms
 9  172.16.143.6 (172.16.143.6)  0.031 ms  0.023 ms  0.024 ms
10  192.168.0.252 (192.168.0.252)  0.072 ms  0.026 ms  0.026 ms

```

Figura 43: Comando -traceroute para Teresa, Spotify, HBO e Itunes.

```
root@m6:/tmp/pycore.36427/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=0.131 ms
64 bytes from 192.168.0.226: icmp_seq=2 ttl=58 time=0.280 ms
64 bytes from 192.168.0.226: icmp_seq=3 ttl=58 time=0.197 ms
64 bytes from 192.168.0.226: icmp_seq=4 ttl=58 time=0.526 ms
64 bytes from 192.168.0.226: icmp_seq=5 ttl=58 time=0.210 ms
64 bytes from 192.168.0.226: icmp_seq=6 ttl=58 time=0.368 ms
64 bytes from 192.168.0.226: icmp_seq=7 ttl=58 time=0.575 ms
64 bytes from 192.168.0.226: icmp_seq=8 ttl=58 time=0.164 ms
64 bytes from 192.168.0.226: icmp_seq=9 ttl=58 time=0.244 ms
^C
--- 192.168.0.226 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8170ms
rtt min/avg/max/mdev = 0.131/0.299/0.575/0.149 ms
root@m6:/tmp/pycore.36427/n6.conf# █
```

Figura 44: Comando Ping AfonsoHenriques

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

O Supernetting é uma técnica de design de rede que consiste em agrupar várias redes IP menores numa única rede maior, com o objetivo de simplificar e reduzir o tamanho das tabelas de roteamento. Ao agrupar as sub-redes numa única rede, é possível reduzir o número de entradas na tabela de roteamento, o que melhora o desempenho da rede e diminui a carga de processamento nos routers.

Uma das vantagens do Supernetting é a redução do tamanho da tabela de roteamento. Ao agrupar várias sub-redes numa única rede maior, é possível diminuir o tamanho da tabela de roteamento, o que contribui para melhorar o desempenho da rede. Além disso, essa técnica também otimiza o uso de endereços IP, uma vez que menos endereços são necessários para representar a rede agregada.

No entanto, o Supernetting também apresenta algumas desvantagens. Uma delas é a perda de flexibilidade. Quando várias sub-redes são agrupadas numa única rede maior, a flexibilidade da rede pode ser comprometida. Por exemplo, se uma sub-rede precisar de ser alterada ou removida, será necessário reconfigurar toda a rede agregada.

Outra desvantagem é o risco de falha de rede. Se a rede agregada for interrompida, todas as sub-redes contidas nessa rede também serão afetadas. Isso pode aumentar o risco de falhas de rede e dificultar a resolução de problemas.

Em resumo, o Supernetting é uma técnica que traz benefícios como a redução do tamanho da tabela de roteamento e a otimização do uso de endereços IP. No entanto, é importante considerar as desvantagens, como a perda de flexibilidade e o risco de falha de rede, ao implementar essa técnica.

Conclusão

Este projeto foi dividido em duas partes principais, cada uma com um nível de dificuldade e complexidade crescente. Na primeira parte, realizamos uma análise do tráfego utilizando diferentes comandos, como o traceroute, e exploramos conceitos como TTL, RTT e os protocolos ICMP e IP. Além disso, o uso do Wireshark permitiu-nos estabelecer uma relação entre a topologia, o comando executado no terminal e o tráfego capturado.

Na segunda parte do projeto prático, exploramos conceitos de máscara de rede e sub-redes no protocolo IPv4. Além disso, também abordamos conceitos como o Sub-endereçamento IP. Essa segunda parte do projeto revelou-se mais desafiadora e exigente, o que nos levou a buscar ajuda da professora. No entanto, conseguimos superar essas dificuldades e dar continuidade ao trabalho. Com a realização desse projeto prático, conseguimos aprimorar nossos conhecimentos teóricos e compreender melhor a utilidade de ferramentas como o Core e o Wireshark.

Em conclusão, esse projeto foi bem-sucedido, permitindo-nos aprofundar e consolidar diversos conceitos importantes de Redes de Computadores. Conseguimos responder a todas as perguntas de forma satisfatória.