Redes - TP2

Trabalho realizado por:

Carlos Miguel Luzia de Carvalho

Ruben César Ferreira Lucas

Paulo Silva Sousa

A89605
Carlos Carvalho

A89487
Ruben Lucas

A89465
Paulo Sousa

Grupo 47
2020/2021
Universidade do Minho

Índice

1 Parte 1 **1**
1.1 Pergunta 1 1
1.2 Pergunta 2 3
1.3 Pergunta 3 8

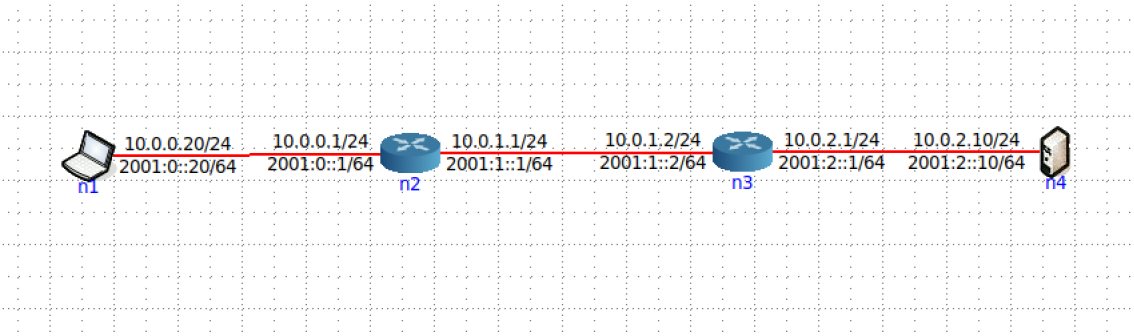
2 Parte 2 **10**
2.1 Pergunta 1 10
2.2 Pergunta 2 13
2.3 Pergunta 3 16

3 Conclusão **18**

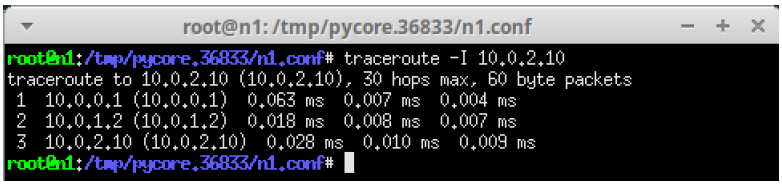
1 Parte 1

1.1 Pergunta 1

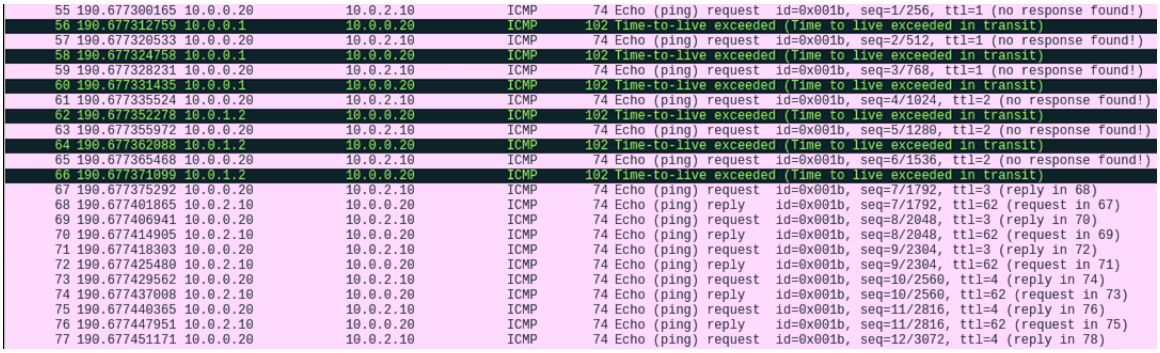
Prepare uma topologia CORE para verificar o comportamento dotraceroute. Ligue um host (pc) Cliente1 a um router R2; o routerR2 a um router R3, que por sua vez, se liga a um host (servidor) Servidor1. (Note que pode não existir conectividade IP imediataentre o Cliente1 e o Servidor1 até que o anúncio de rotasestabilize). Ajuste o nome dos equipamentos atribuídos por defeitopara a topologia do enunciado.



A Active o wireshark ou o tcpdump no Cliente1. Numa shell doCliente1, execute o comando traceroute -I para o endereçoIP do Servidor1.



B Registe e analise o tráfego ICMP enviado pelo Cliente1 e otráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.



55	190.677300165	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=1/256, ttl=1 (no response found!)
56	190.677312759	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
57	190.677320533	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=2/512, ttl=1 (no response found!)
58	190.677324758	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
59	190.677328231	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=3/768, ttl=1 (no response found!)
60	190.677331435	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
61	190.677335524	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=4/1024, ttl=2 (no response found!)
62	190.677352278	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
63	190.677355972	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=5/1280, ttl=2 (no response found!)
64	190.677362088	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
65	190.677365468	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=6/1536, ttl=2 (no response found!)
66	190.677371099	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
67	190.677375292	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=7/1792, ttl=3 (reply in 68)
68	190.677401865	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=7/1792, ttl=62 (request in 67)
69	190.677406941	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=8/2048, ttl=3 (reply in 70)
70	190.677414905	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=8/2048, ttl=62 (request in 69)
71	190.677418303	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=9/2304, ttl=3 (reply in 72)
72	190.677425480	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=9/2304, ttl=62 (request in 71)
73	190.677429562	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=10/2560, ttl=4 (reply in 74)
74	190.677437008	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=10/2560, ttl=62 (request in 73)
75	190.677440365	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=11/2816, ttl=4 (reply in 76)
76	190.677447951	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=11/2816, ttl=62 (request in 75)
77	190.677451171	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=12/3072, ttl=4 (reply in 78)

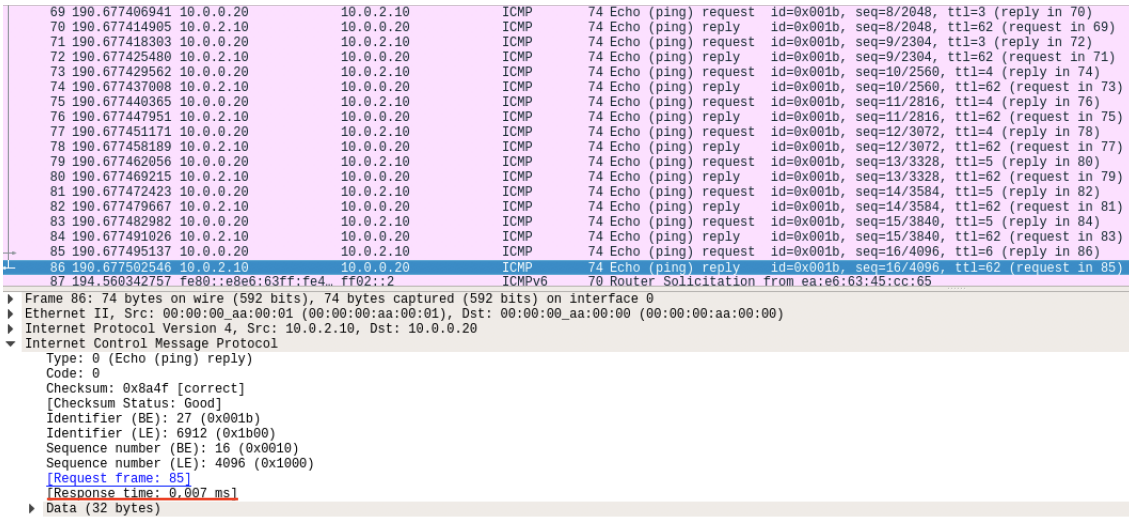
Neste caso em específico, os packets são enviados em conjuntos de 3, os primeiros 2 conjuntos, como têm um TTL inferior a 3 (o número necessário de saltos para ir da source ao destino), não chegam ao destino e, por isso, recebemos as mensagens de erro ICMP.

C Qual deve ser o valor inicial mínimo do campo TTL para alcançar o Servidor 1? Verifique na prática que a sua resposta está correta.

67	190.677375292	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=7/1792, ttl=3 (reply in 68)
68	190.677401865	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=7/1792, ttl=62 (request in 67)

O mínimo TTL para alcançar o servidor 1 deverá ser 3 pelo motivo referenciado na alínea anterior.

D Calcule o valor médio do tempo de ida-e-volta (Round-Trip Time) obtido?



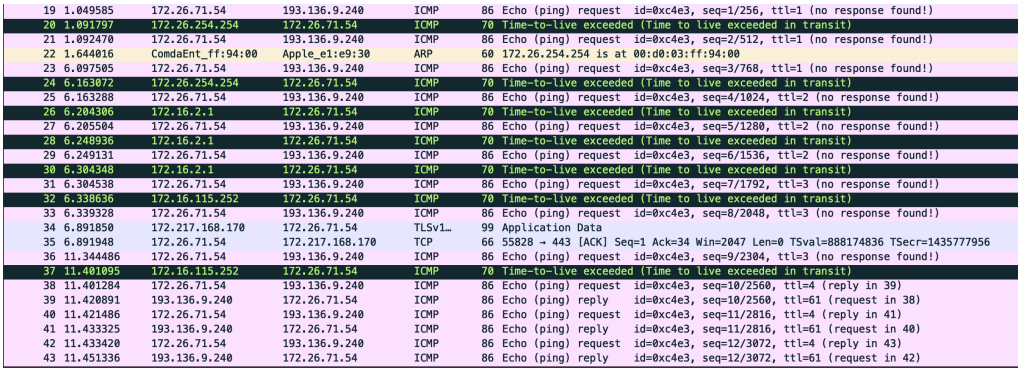
69	190.677406941	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=8/2048, ttl=3 (reply in 70)
70	190.677414905	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=8/2048, ttl=62 (request in 69)
71	190.677418303	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=9/2304, ttl=3 (reply in 72)
72	190.677425480	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=9/2304, ttl=62 (request in 71)
73	190.677429562	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=10/2560, ttl=4 (reply in 74)
74	190.677437008	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=10/2560, ttl=62 (request in 73)
75	190.677440365	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=11/2816, ttl=4 (reply in 76)
76	190.677447951	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=11/2816, ttl=62 (request in 75)
77	190.677451171	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=12/3072, ttl=4 (reply in 78)
78	190.677458189	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=12/3072, ttl=62 (request in 77)
79	190.677462056	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=13/3328, ttl=5 (reply in 80)
80	190.677469215	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=13/3328, ttl=62 (request in 79)
81	190.677472423	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=14/3584, ttl=5 (reply in 82)
82	190.677479667	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=14/3584, ttl=62 (request in 81)
83	190.677482982	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=15/3840, ttl=5 (reply in 84)
84	190.677491026	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=15/3840, ttl=62 (request in 83)
85	190.677495137	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x001b, seq=16/4096, ttl=6 (reply in 86)
86	190.677502546	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=16/4096, ttl=62 (request in 85)
87	194.560342757	fe80::e8e6:63ff:fe4…	ff02::2	ICMPv6	70 Router Solicitation from ea:e6:63:45:cc:65

```
▸ Frame 86: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_aa:00:01 (00:00:00:aa:00:01), Dst: 00:00:00_aa:00:00 (00:00:00:aa:00:00)
▸ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.0.20
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x8a4f [correct]
    [Checksum Status: Good]
    Identifier (BE): 27 (0x001b)
    Identifier (LE): 6912 (0x1b00)
    Sequence number (BE): 16 (0x0010)
    Sequence number (LE): 4096 (0x1000)
    [Request frame: 85]
    [Response time: 0.007 ms]
  ▸ Data (32 bytes)
```

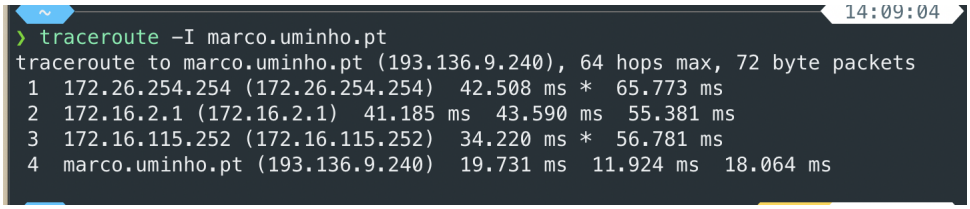
O valor deverá ser 0.007 ms.

1.2 Pergunta 2

Pretende-se agora usar o traceroute na sua máquina nativa, e gerar de datagramas IP de diferentes tamanhos. Usando o wireshark capture o tráfego gerado pelo traceroute para os seguintes tamanhos de pacote: (i) usando o tamanho por defeito. Selecione a primeira mensagem ICMP capturada e centre a análise no nível protocolar IP.

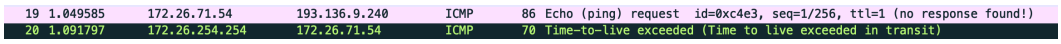


19	1.049585	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=1/256, ttl=1 (no response found!)
20	1.091797	172.26.254.254	172.26.71.54	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
21	1.092470	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=2/512, ttl=1 (no response found!)
22	1.644016	ComdaEnt_ff:94:00	Apple_e1:e9:30	ARP	60 172.26.254.254 is at 00:d0:03:ff:94:00
23	6.097505	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=3/768, ttl=1 (no response found!)
24	6.163072	172.26.254.254	172.26.71.54	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
25	6.163288	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=4/1024, ttl=2 (no response found!)
26	6.204306	172.16.2.1	172.26.71.54	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
27	6.205504	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=5/1280, ttl=2 (no response found!)
28	6.248936	172.16.2.1	172.26.71.54	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
29	6.249131	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=6/1536, ttl=2 (no response found!)
30	6.304348	172.16.2.1	172.26.71.54	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
31	6.304538	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=7/1792, ttl=3 (no response found!)
32	6.338636	172.16.115.252	172.26.71.54	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
33	6.339328	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=8/2048, ttl=3 (no response found!)
34	6.891850	172.217.168.170	172.26.71.54	TLSv1…	99 Application Data
35	6.891948	172.26.71.54	172.217.168.170	TCP	66 55828 → 443 [ACK] Seq=1 Ack=34 Win=2047 Len=0 TSval=888174836 TSecr=1435777956
36	11.344486	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=9/2304, ttl=3 (no response found!)
37	11.401095	172.16.115.252	172.26.71.54	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
38	11.401284	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=10/2560, ttl=4 (reply in 39)
39	11.420891	193.136.9.240	172.26.71.54	ICMP	86 Echo (ping) reply id=0xc4e3, seq=10/2560, ttl=61 (request in 38)
40	11.421486	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=11/2816, ttl=4 (reply in 41)
41	11.433325	193.136.9.240	172.26.71.54	ICMP	86 Echo (ping) reply id=0xc4e3, seq=11/2816, ttl=61 (request in 40)
42	11.433420	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=12/3072, ttl=4 (reply in 43)
43	11.451336	193.136.9.240	172.26.71.54	ICMP	86 Echo (ping) reply id=0xc4e3, seq=12/3072, ttl=61 (request in 42)



```
~                                                                    14:09:04
❯ traceroute -I marco.uminho.pt
traceroute to marco.uminho.pt (193.136.9.240), 64 hops max, 72 byte packets
 1  172.26.254.254 (172.26.254.254)  42.508 ms *  65.773 ms
 2  172.16.2.1 (172.16.2.1)  41.185 ms  43.590 ms  55.381 ms
 3  172.16.115.252 (172.16.115.252)  34.220 ms *  56.781 ms
 4  marco.uminho.pt (193.136.9.240)  19.731 ms  11.924 ms  18.064 ms
```

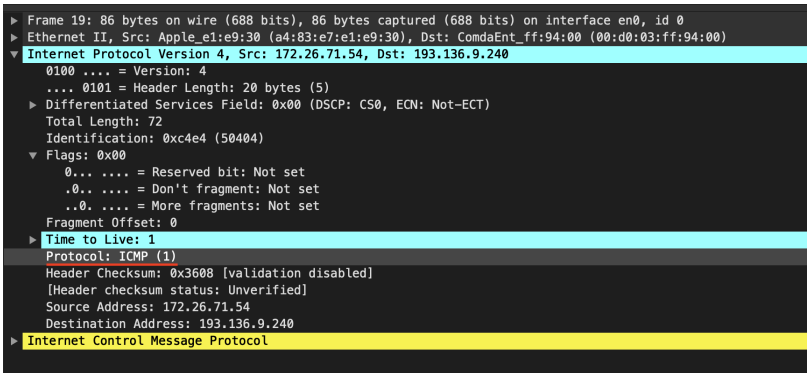
A Qual é o endereço IP da interface ativa do seu computador?



19	1.049585	172.26.71.54	193.136.9.240	ICMP	86 Echo (ping) request id=0xc4e3, seq=1/256, ttl=1 (no response found!)
20	1.091797	172.26.254.254	172.26.71.54	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

O endereço de IP, como podemos ver pela figura, é 172.26.71.54.

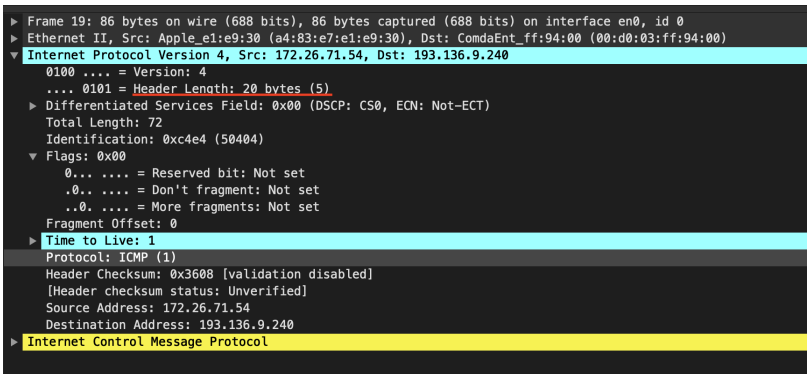
B Qual é o valor do campo protocolo? O que identifica?



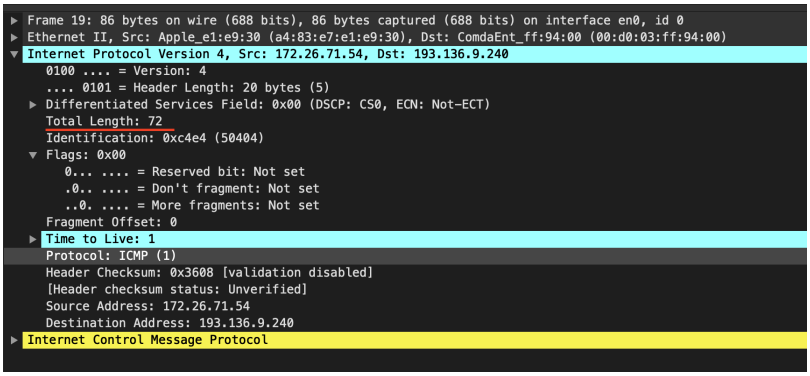
```
▶ Frame 19: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface en0, id 0
▶ Ethernet II, Src: Apple_e1:e9:30 (a4:83:e7:e1:e9:30), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.71.54, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 72
    Identification: 0xc4e4 (50404)
  ▼ Flags: 0x00
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    Fragment Offset: 0
  ▶ Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x3608 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.71.54
    Destination Address: 193.136.9.240
▶ Internet Control Message Protocol
```

Como podemos ver pela figura, o valor do campo do protocolo é 1, que representa o protocolo ICMP.

C Quantos bytes tem o cabeçalho IP(v4)? Quantos bytes tem o campo de dados (payload) do datagrama? Como se calcula o tamanho do payload?



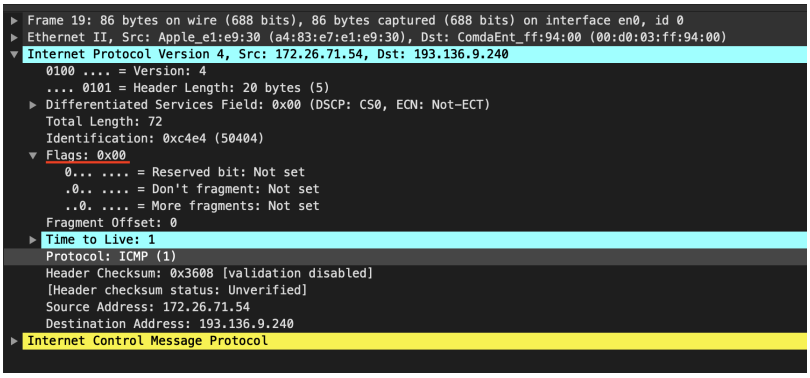
O número de bytes do cabeçalho IP(v4) é 20, como podemos ver na figura.



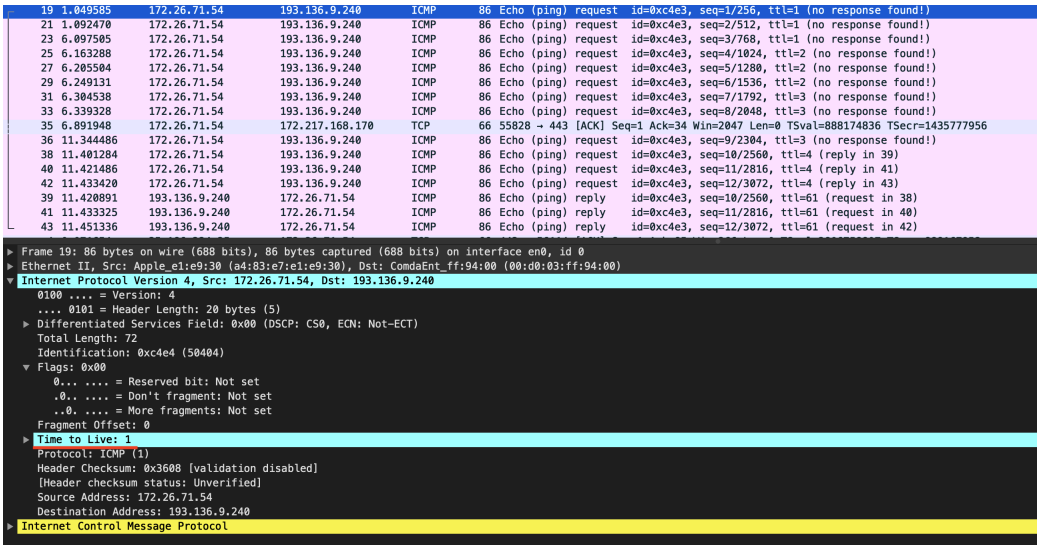
O campo de dados do datagrama tem 52 bytes. Este valor é obtido subtraindo o número de bytes do cabeçalho ao número total de bytes (72, tal como podemos observar na figura).

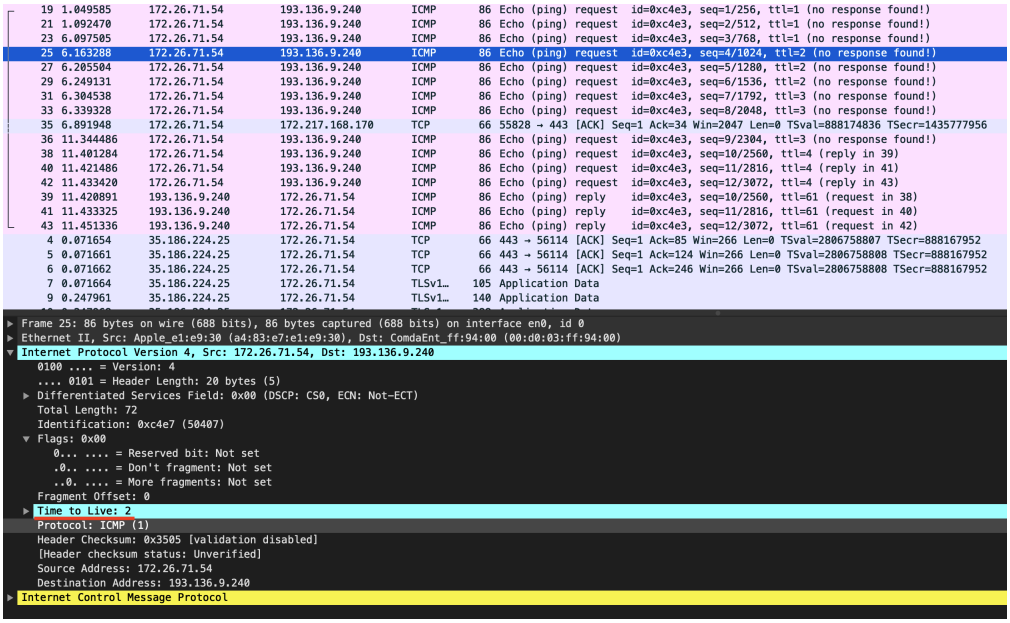
D O datagrama IP foi fragmentado? Justifique.

O datagrama do IP não foi fragmentado porque, tal como podemos ver na imagem, as flags e o offset têm ambos valor 0.



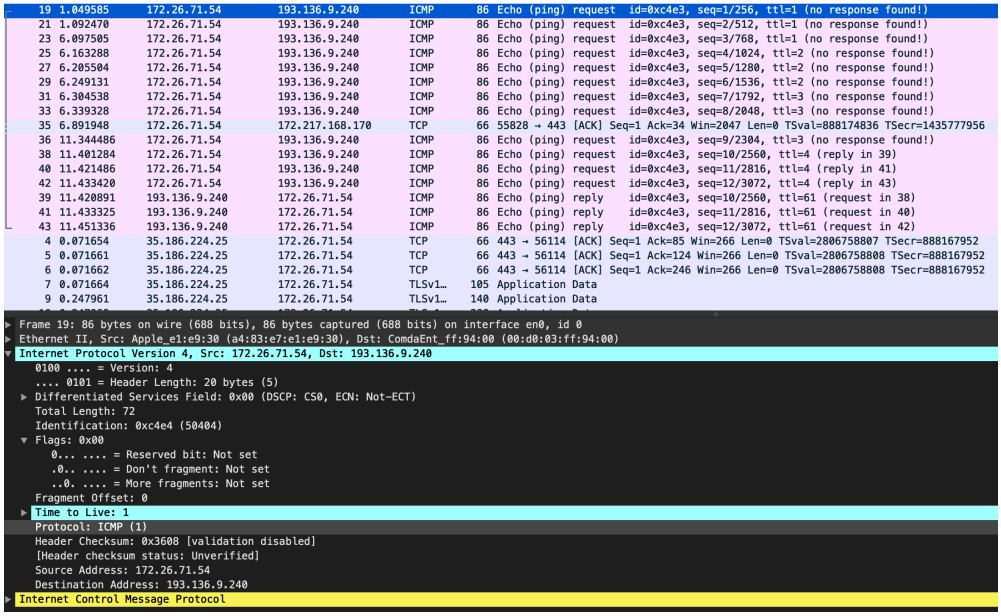
E Ordene os pacotes capturados de acordo com o endereço IP fonte, e analise a sequência de tráfego ICMP gerado a partir do endereço IP atribuído à interface da sua máquina. Para a sequência de mensagens ICMP enviadas pelo seu computador, indique que campos do cabeçalho IP variam de pacote para pacote.

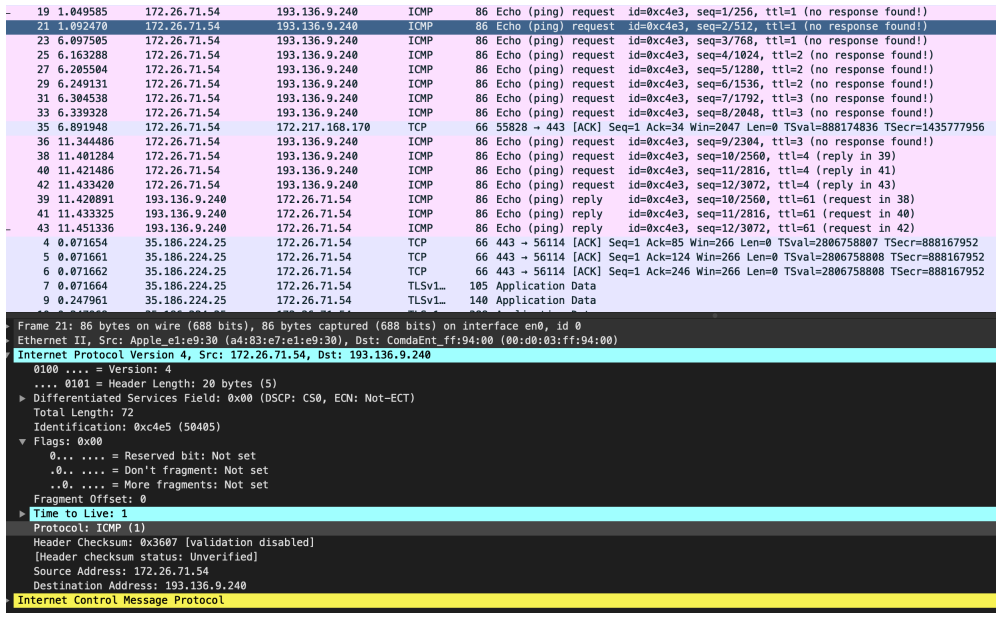


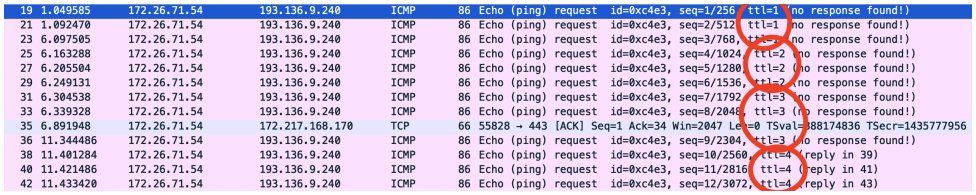


Como podemos ver nas figuras, os campos do cabeçalho IP que variam com o tempo são o Time to Live e o Identification.

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

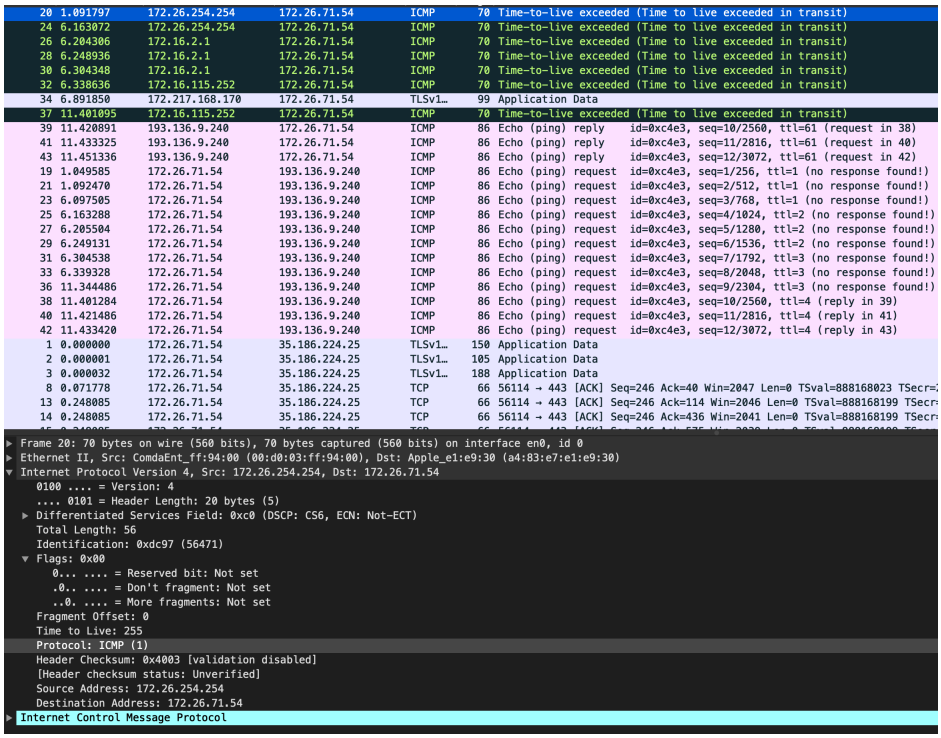


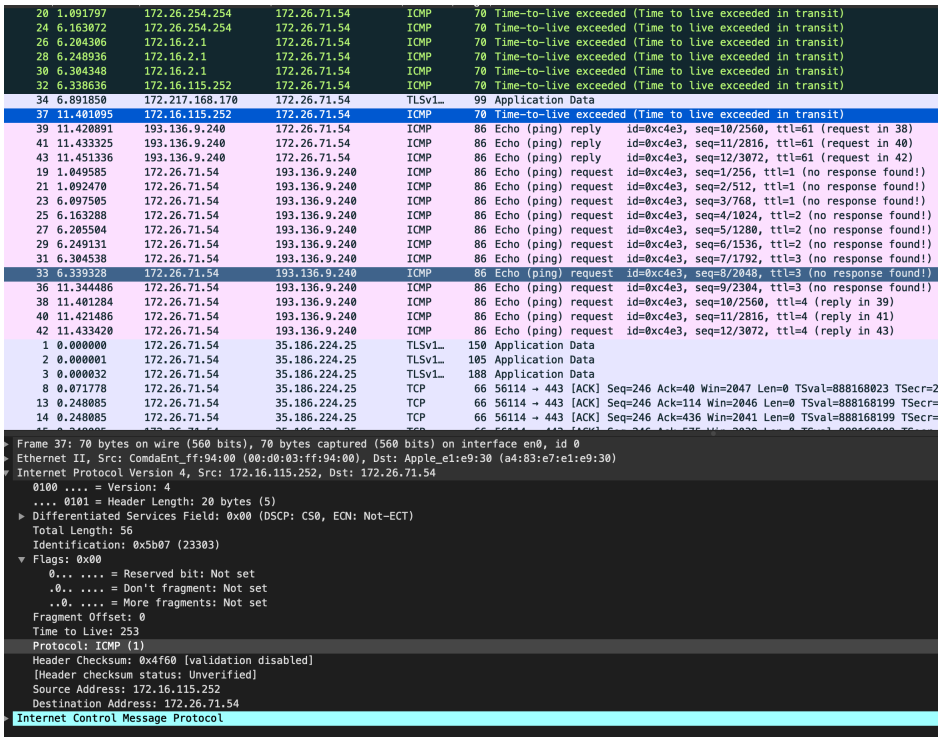




Através da análise das três figuras, concluímos que o valor do campo Time to Live é incrementado a cada 3 pacotes.

G Ordene o tráfego capturado por endereço destino e encontre a série de respostas ICMP TTL exceeded enviadas ao seu computador. Qual é o valor do campo TTL? Esse valor permanece constante para todas as mensagens de resposta ICMP TTL exceeded enviados ao seu host? Porquê?

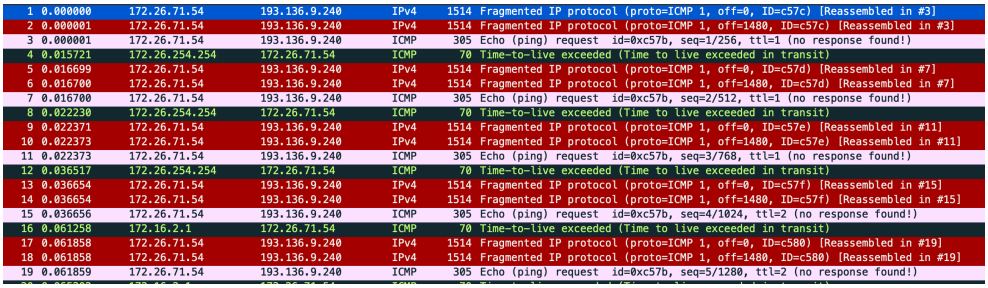




As mensagem de erro são enviadas, por predefinição, com ttl 256. Ao ser enviada pelo n4 chega ao n3 com 255 ttl, ao n2 com 254 ttl e, por último, a n1 com 253 ttl.

1.3 Pergunta 3

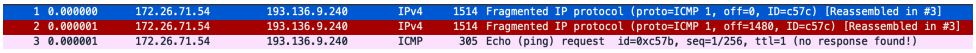
Pretende-se agora analisar a fragmentação de pacotes IP. Reponha a ordem do tráfego capturado usando a coluna do tempo de captura. Observe o tráfego depois do tamanho de pacote ter sido definido para 3251 bytes.



1	0.000000	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=c57c) [Reassembled in #3]
2	0.000001	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=c57c) [Reassembled in #3]
3	0.000001	172.26.71.54	193.136.9.240	ICMP	305	Echo (ping) request id=0xc57b, seq=1/256, ttl=1 (no response found!)
4	0.015721	172.26.254.254	172.26.71.54	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
5	0.016699	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=c57d) [Reassembled in #7]
6	0.016700	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=c57d) [Reassembled in #7]
7	0.016700	172.26.71.54	193.136.9.240	ICMP	305	Echo (ping) request id=0xc57b, seq=2/512, ttl=1 (no response found!)
8	0.022230	172.26.254.254	172.26.71.54	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
9	0.022371	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=c57e) [Reassembled in #11]
10	0.022373	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=c57e) [Reassembled in #11]
11	0.022373	172.26.71.54	193.136.9.240	ICMP	305	Echo (ping) request id=0xc57b, seq=3/768, ttl=1 (no response found!)
12	0.036517	172.26.254.254	172.26.71.54	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
13	0.036654	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=c57f) [Reassembled in #15]
14	0.036654	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=c57f) [Reassembled in #15]
15	0.036656	172.26.71.54	193.136.9.240	ICMP	305	Echo (ping) request id=0xc57b, seq=4/1024, ttl=2 (no response found!)
16	0.061258	172.16.2.1	172.26.71.54	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
17	0.061858	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=c580) [Reassembled in #19]
18	0.061858	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=c580) [Reassembled in #19]
19	0.061859	172.26.71.54	193.136.9.240	ICMP	305	Echo (ping) request id=0xc57b, seq=5/1280, ttl=2 (no response found!)

```
❯ traceroute -I marco.uminho.pt 3251
traceroute to marco.uminho.pt (193.136.9.240), 64 hops max, 3251 byte packets
 1  172.26.254.254 (172.26.254.254)  16.081 ms  5.668 ms  14.272 ms
 2  172.16.2.1 (172.16.2.1)  24.696 ms  3.425 ms  8.193 ms
 3  172.16.115.252 (172.16.115.252)  50.999 ms  5.827 ms  9.106 ms
 4  marco.uminho.pt (193.136.9.240)  76.213 ms  11.569 ms  4.618 ms
```

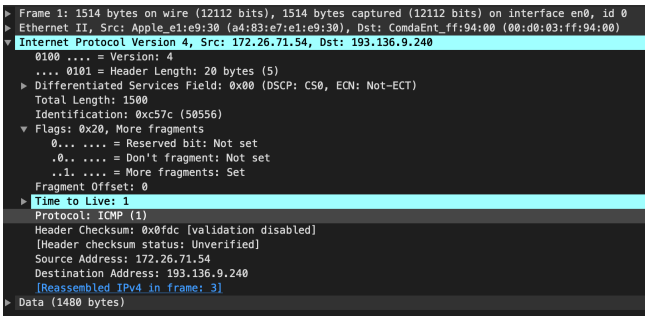
A Localize a primeira mensagem ICMP. Porque é que houve necessidade de fragmentar o pacote inicial?



1	0.000000	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=c57c) [Reassembled in #3]
2	0.000001	172.26.71.54	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=c57c) [Reassembled in #3]
3	0.000001	172.26.71.54	193.136.9.240	ICMP	305	Echo (ping) request id=0xc57b, seq=1/256, ttl=1 (no response found!)

Uma vez que a MTU que estamos a usar só consegue enviar pacotes com tamanho 1500 e o pacote que estamos a tentar enviar tem 3251, este tem de ser fragmentado para poder ser enviado.

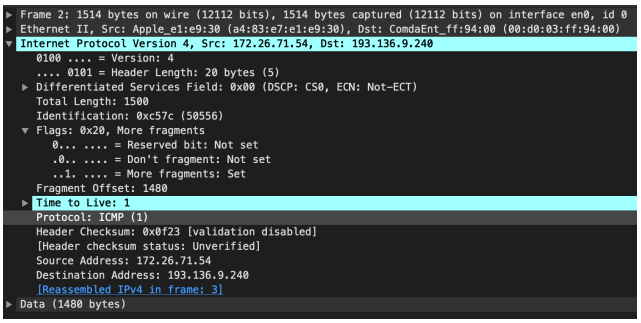
B Imprima o primeiro fragmento do datagrama IP segmentado. Que informação no cabeçalho indica que o datagrama foi fragmentado? Que informação no cabeçalho IP indica que se trata do primeiro fragmento? Qual é o tamanho deste datagrama IP?



```
▶ Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface en0, id 0
▶ Ethernet II, Src: Apple_e1:e9:30 (a4:83:e7:e1:e9:30), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.71.54, 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: 0xc57c (50556)
  ▼ Flags: 0x20, More fragments
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..1. .... = More fragments: Set
    Fragment Offset: 0
  ▶ Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x0fdc [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.71.54
    Destination Address: 193.136.9.240
    [Reassembled IPv4 in frame: 3]
▶ Data (1480 bytes)
```

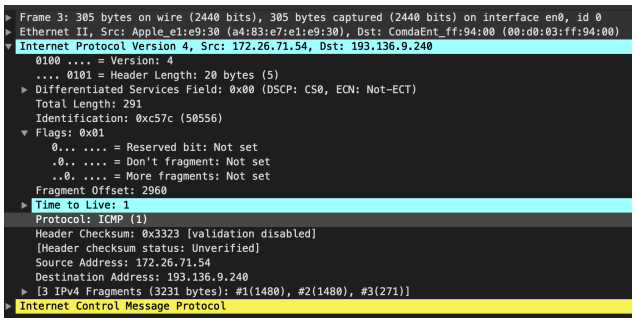
A informação contida em Flags permite-nos saber que o datagrama foi fragmentado uma vez que o offset se encontra com valor zero e o More fragments com valor 1, estes valores indicam-nos também que se trata do primeiro fragmento. Quanto ao tamanho do datagrama sabemos que é 1500.

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



Como vimos na alínea acima, trata-se do 1º fragmento quando o valor do offset se encontra a zero, o que não se verifica como podemos ver na imagem. Logo este fragmento não é o 1º fragmento. Relativamente à existência de mais fragmentos, uma vez que o parâmetro More Fragments se encontra a 1, podemos concluir que existem ainda mais fragmentos.

D Quantos fragmentos foram criados a partir do datagrama original? Como se detecta o último fragmento correspondente ao datagrama original?



Sabemos que o último fragmento do datagrama tem valor 0 para a flag more fragments, como podemos ver na figura. Assim, como sabemos que este fragmento é o último, podemos simplesmente contar o número de fragmentos, ou seja, 3.

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.

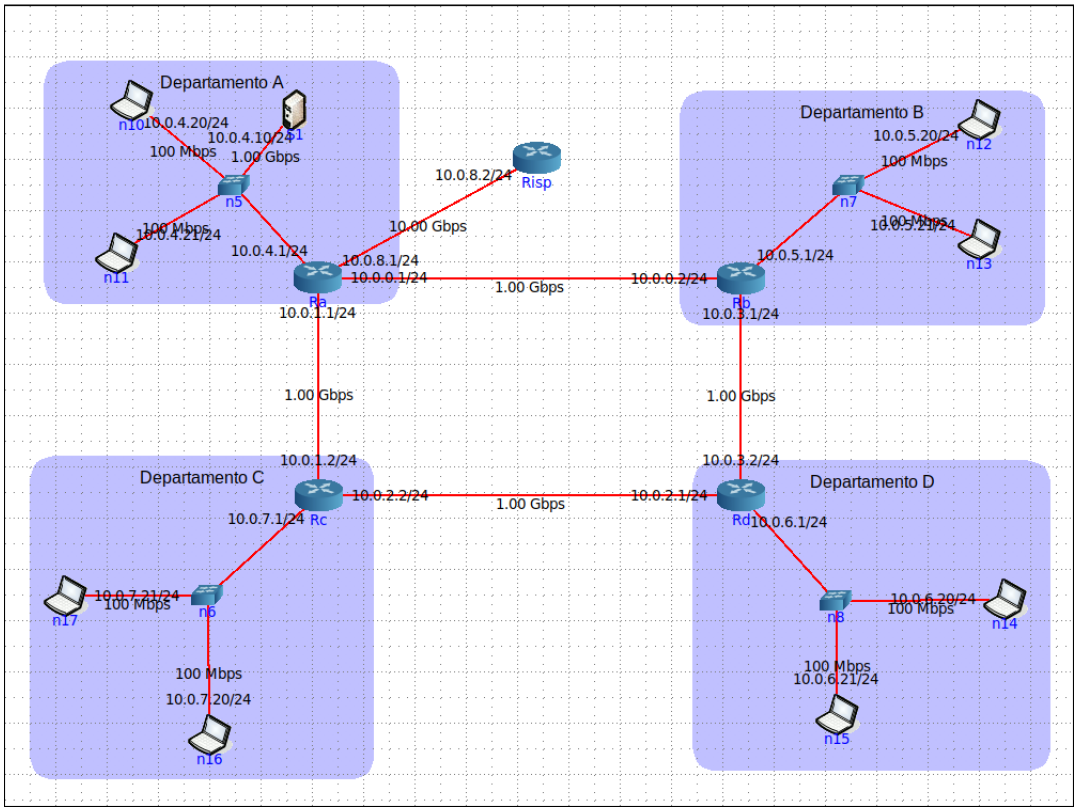
Entre os diferentes fragmentos variam os valores da flag offset e da flag more fragments. A flag more segments permite-nos saber se o packet em questão é o último ou não. A flag offset permite-nos saber a posição inicial do packet original a partir da qual obtemos o packet fragmentado.

2 Parte 2

Considere que a organização MIEI-RC é constituída por quatro departamentos (A, B, C e D) e cada departamento possui um router de acesso à sua rede local. Estes routers de acesso (RA, RB, RC e RD) estão interligados entre si por ligações Ethernet a 1Gbps, formando um anel. Por sua vez, existe um servidor (S1) na rede do departamento A e, pelo menos, dois laptops por departamento, interligados ao router respetivo através de um comutador (switch). S1 tem uma ligação a 1Gbps e os laptops ligações a 100Mbps. Considere apenas a existência de um comutador por departamento.

2.1 Pergunta 1

Atenda aos endereços IP atribuídos automaticamente pelo CORE aos diversos equipamentos da topologia.



A Indique que endereços IP e máscaras de rede foram atribuídos pelo CORE a cada equipamento. Para simplificar, pode incluir uma imagem que ilustre de forma clara a topologia definida e o endereçamento usado.

Os equipamentos têm a máscara /24, que corresponde a 255.255.255.0. Os IPs de cada equipamento estão disponíveis na figura a cima.

B Tratam-se de endereços públicos ou privados? Porquê?

Como todos os endereços entre 10.0.0.0 e 10.255.255.255 são endereços privados, e todos os endereços IP atribuidos pertencem a esta gama, podemos concluir que os endereços são privados.

C Porque razão não é atribuído um endereço IP aos switches?

Devido à principal funcionalidade de um equipamento switch ser reencaminhamento de informação, apenas registando os endereços MAC dos dispositivos ligados a cada porta, não lhe é atribuido um endereço IP pelo facto de não ser necessário, sendo que estes só reencaminham os pacotes para o destino.

D Usando o comando ping certifique-se que existe conectividade IP entre os laptops dos vários departamentose o servidor do departamento A (basta certificar-se da conectividade de um laptop por departamento).

```
64 bytes from 10.0.4.10: icmp_seq=30 ttl=64 time=0.039 ms
64 bytes from 10.0.4.10: icmp_seq=31 ttl=64 time=0.042 ms
64 bytes from 10.0.4.10: icmp_seq=32 ttl=64 time=0.070 ms
64 bytes from 10.0.4.10: icmp_seq=33 ttl=64 time=0.038 ms
64 bytes from 10.0.4.10: icmp_seq=34 ttl=64 time=0.039 ms
64 bytes from 10.0.4.10: icmp_seq=35 ttl=64 time=0.064 ms
64 bytes from 10.0.4.10: icmp_seq=36 ttl=64 time=0.069 ms
64 bytes from 10.0.4.10: icmp_seq=37 ttl=64 time=0.038 ms
64 bytes from 10.0.4.10: icmp_seq=38 ttl=64 time=0.039 ms
64 bytes from 10.0.4.10: icmp_seq=39 ttl=64 time=0.093 ms
^C
--- 10.0.4.10 ping statistics ---
39 packets transmitted, 39 received, 0% packet loss, time 38928ms
rtt min/avg/max/mdev = 0.032/0.049/0.093/0.016 ms
root@n10:/tmp/pycore.46839/n10.conf#
```

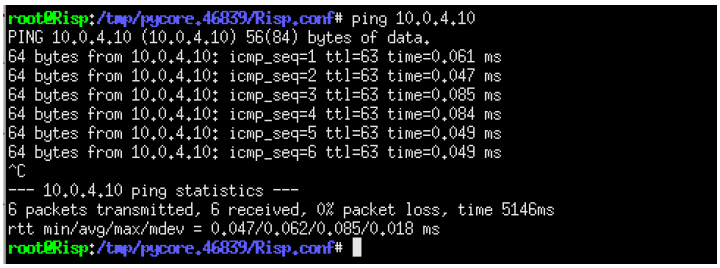
```
root@n13:/tmp/pycore.46839/n13.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=0.056 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=0.047 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=0.060 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=62 time=0.056 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=62 time=0.083 ms
64 bytes from 10.0.4.10: icmp_seq=6 ttl=62 time=0.354 ms
64 bytes from 10.0.4.10: icmp_seq=7 ttl=62 time=0.061 ms
^C
--- 10.0.4.10 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6187ms
rtt min/avg/max/mdev = 0.047/0.102/0.354/0.103 ms
root@n13:/tmp/pycore.46839/n13.conf#
```

```
root@n17:/tmp/pycore.46839/n17.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=0.073 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=0.063 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=0.050 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=62 time=0.101 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=62 time=0.529 ms
64 bytes from 10.0.4.10: icmp_seq=6 ttl=62 time=0.096 ms
64 bytes from 10.0.4.10: icmp_seq=7 ttl=62 time=0.054 ms
^C
--- 10.0.4.10 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6377ms
rtt min/avg/max/mdev = 0.050/0.138/0.529/0.160 ms
root@n17:/tmp/pycore.46839/n17.conf#
```

```
root@n15:/tmp/pycore.46839/n15.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=61 time=0.114 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=61 time=0.113 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=61 time=0.069 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=61 time=0.118 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=61 time=0.115 ms
64 bytes from 10.0.4.10: icmp_seq=6 ttl=61 time=0.050 ms
64 bytes from 10.0.4.10: icmp_seq=7 ttl=61 time=0.057 ms
64 bytes from 10.0.4.10: icmp_seq=8 ttl=61 time=0.073 ms
^C
--- 10.0.4.10 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7206ms
rtt min/avg/max/mdev = 0.050/0.088/0.118/0.029 ms
root@n15:/tmp/pycore.46839/n15.conf#
```

Ao usar o comando ping a partir dos laptops de cada departamento, pode-se comprovar que a packet loss é de 0%, logo existe conectividade IP entre os laptops de cada departamento.

E Verifique se existe conectividade IP do router de acesso RISP para o servidor S1.



Ao usar o comando ping a partir de Ra, como não existe packet loss, comprovamos que existe conectividade IP do router Risp e o Servidor 1.

2.2 Pergunta 2

Para o router e um laptop do departamento C:

A Execute o comando netstat –rn por forma a poder consultara tabela de encaminhamento unicast (IPv4). Inclua no seu relatório as tabelas de encaminhamento obtidas; interprete as várias entradas de cada tabela. Se necessário, consulte o manual respetivo (man netstat).

```
root@n16:/tmp/pycore.46839/n16.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.7.1        0.0.0.0         UG        0 0          0 eth0
10.0.7.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
root@n16:/tmp/pycore.46839/n16.conf#
```

```
root@Rc:/tmp/pycore.46839/Rc.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
10.0.3.0        10.0.2.1        255.255.255.0   UG        0 0          0 eth1
10.0.4.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.5.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.6.0        10.0.2.1        255.255.255.0   UG        0 0          0 eth1
10.0.7.0        0.0.0.0         255.255.255.0   U         0 0          0 eth2
10.0.8.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
root@Rc:/tmp/pycore.46839/Rc.conf#
```

A primeira figura faz referência ao laptop e nela podemos identificar o destino default para o envio de pacotes, sendo este 0.0.0.0. Já a segunda linha em que tem como destino 10.0.7.0 é referente a quando o laptop quer enviar os pacotes para a própria rede. A segunda figura com as redes e respetivos gateway é referente ao router e quando o destino do pacote é uma das redes presetes na coluna destino este é enviado pelo Gateway respetivo.

B Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico (sugestão: analise que processos estão a correr em cada sistema, por exemplo, ps -ax).

```
root@n16:/tmp/pycore.46839/n16.conf# ps -ax
  PID TTY      STAT   TIME COMMAND
    1 ?        S      0:00 /usr/local/bin/vnoded -v -c /tmp/pycore.46839/n16 -l
   52 pts/4    Ss     0:00 /bin/bash
   60 pts/4    R+     0:00 ps -ax
root@n16:/tmp/pycore.46839/n16.conf#
```

```
root@Rc:/tmp/pycore.46839/Rc.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
10.0.3.0        10.0.2.1        255.255.255.0   UG        0 0          0 eth1
10.0.4.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.5.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.6.0        10.0.2.1        255.255.255.0   UG        0 0          0 eth1
10.0.7.0        0.0.0.0         255.255.255.0   U         0 0          0 eth2
10.0.8.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
root@Rc:/tmp/pycore.46839/Rc.conf# ps -ax
  PID TTY      STAT   TIME COMMAND
    1 ?        S      0:00 /usr/local/bin/vnoded -v -c /tmp/pycore.46839/Rc -l /
   56 ?        Ss     0:00 /usr/sbin/zebra -d
   62 ?        Ss     0:00 /usr/sbin/ospf6d -d
   66 ?        Ss     0:00 /usr/sbin/ospfd -d
  108 pts/2    Ss     0:00 /bin/bash
  117 pts/2    R+     0:00 ps -ax
root@Rc:/tmp/pycore.46839/Rc.conf#
```

Um encaminhamento estático é configurado quando uma tabela de roteamento estático é construída manualmente pelo administrador do sistema. Tabelas estáticas não se ajustam automaticamente às alterações na rede, portanto devem ser utilizadas somente onde as rotas não sofrerem alterações, ou seja, as rotas permanecem fixas e são baseadas nas rotas pré-definidas. Por isso, não existe nenhum processo a correr além dos da própria máquina.

Por outro lado, no encaminhamento dinâmico, as rotas são calculadas dinamicamente recorrendo a protocolos de encaminhamento dinâmico que "respondem" e se adaptam a possíveis alterações da rede, ou seja, os routers trocam informação de routing entre si, sendo as rotas atualizadas ao longo do tempo.

Posto isto, podemos deduzir que, na imagem referente a um laptop, se está a utilizar encaminhamento estático, sendo que não existem processos a correr além do da própria maquina. Já na segunda, que se trata de um router, está-se a utilizar encaminhamento dinâmico, uma vez que nesta se podem observar que existem processos a correr em OSPF e ZEBRA.

C Admita que, por questões administrativas, a rota por defeito (0.0.0.0 ou default) deve ser retirada definitivamente da tabela de encaminhamento do servidor S1 localizado no departamento A. Use o comando route delete para o efeito. Que implicações tem esta medida para os utilizadores da organização MIEI-RC que acedem ao servidor.

```
root@S1:/tmp/pycore.46839/S1.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.4.1        0.0.0.0         UG        0 0          0 eth0
10.0.4.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
root@S1:/tmp/pycore.46839/S1.conf# route delete default
root@S1:/tmp/pycore.46839/S1.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.4.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
root@S1:/tmp/pycore.46839/S1.conf#
```

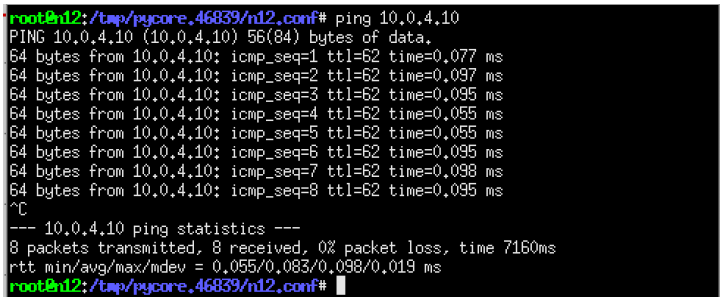
Com o comando *route delete default* a rota entre o Servidor 1 e o router A deixa de existir. Com isto, os outros departamentos deixam de ter acesso ao Servidor 1, uma vez que o seu ponto de acesso seria através do router A (Ra). Porém, os laptops do Departamento A continuam a ter acesso a este, uma vez que não dependem do router, mas sim do Switch n5 para se conectarem ao servidor.

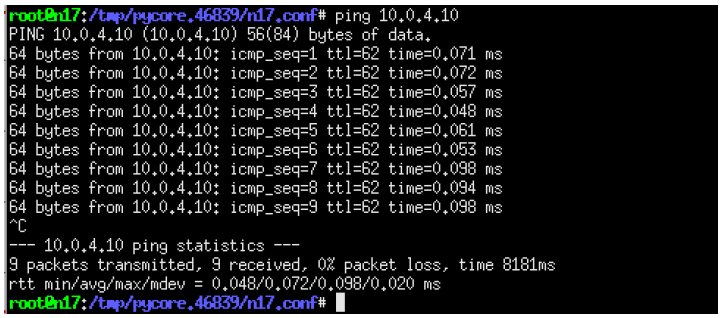
D Adicione as rotas estáticas necessárias para restaurar aconectividade para o servidor S1, por forma a contornar a restrição imposta na alínea c). Utilize para o efeito o comando route add e registe os comandos que usou.

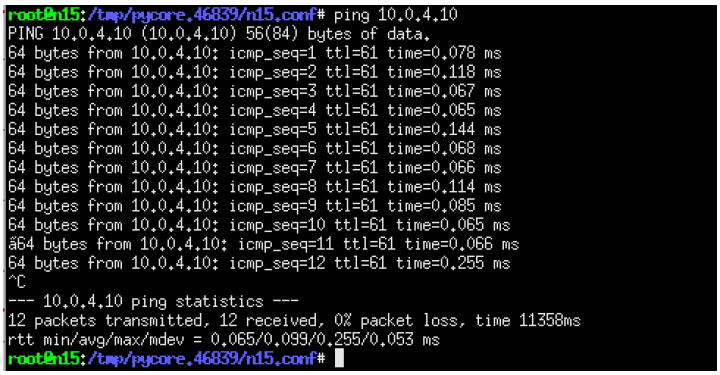
```
root@S1:/tmp/pycore.46839/S1.conf# route add -net 10.0.5.0 netmask 255.255.255.0
 gw 10.0.4.1
root@S1:/tmp/pycore.46839/S1.conf# route add -net 10.0.6.0 netmask 255.255.255.0
 gw 10.0.4.1
root@S1:/tmp/pycore.46839/S1.conf# route add -net 10.0.7.0 netmask 255.255.255.0
 gw 10.0.4.1
root@S1:/tmp/pycore.46839/S1.conf# route add -net 10.0.8.0 netmask 255.255.255.0
 gw 10.0.4.1
root@S1:/tmp/pycore.46839/S1.conf#
```

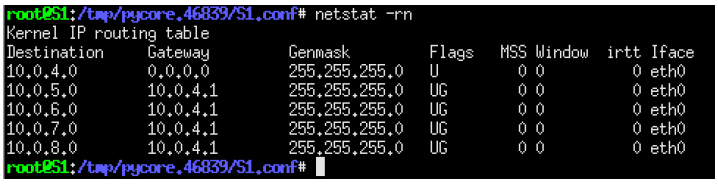
Para criar rotas entre o servidor e os vários departamentos, utilizamos o comando *route add -net [IP] netmask 255.255.255.0 gw 10.0.4.1.*

E Teste a nova política de encaminhamento garantindo que o servidor está novamente acessível, utilizando para o efeito o comando ping. Registe a nova tabela de encaminhamento do servidor.









Como é observável pela última imagem, a conectividade foi restabelecida, visto que o laptop no departamento A consegue transmitir pacotes entre ele e o Servidor 1. Isto é também verificável para os restantes Departamentos e para o Servidor 1.

2.3 Pergunta 3

Considere a topologia definida anteriormente. Assuma que o endereçamento entre os routers se mantém inalterado, contudo, o endereçamento em cada departamento deve ser redefinido.

1 Considere que dispõe apenas do endereço de rede IP 130.XX.96.0/19, em que XX é o decimal correspondendo ao seu número de grupo (PLXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo a rede de acesso e core inalteradas) e atribua endereços às interfaces dos vários sistemas envolvidos. Assuma que todos os endereços de sub-redes são usáveis.

O nosso endereço de rede IP é 130.51.96.0/19. Como temos uma máscara de 19, podemos usar todos os endereços entre 130.51.96.1 e 130.51.127.254.

Sendo que, por convenção, o primeiro e o último endereços para sub-redes são reservados e, sendo que necessitamos de 4 sub-redes para os departamentos, iremos precisar de 3 bits para representar as sub-redes.

136.51.11 |XXX| 00.0

000	Reservado	
001	Livre	Departamento A
010	Livre	Departamento B
011	Livre	
100	Livre	Departamento C
101	Livre	
110	Livre	Departamento D
111	Reservado	

A máscara tem de aumentar 3 bits, (de /19 para /22), devido a serem necessários 3 bits para satisfazer as condições necessárias para criar 4 subredes.

Com esta ditribuição conseguimos no futuro fazer uma super-rede para o departamento B e C. Sendo assim ficamos com:

Departamento A	130.51.100.0/22	130.51.100.1 a 130.51.101.254
Departamento B	130.51.104.0/22	130.51.104.1 a 130.51.107.254
Departamento C	130.51.112.0/22	130.51.112.1 a 130.51.115.254
Departamento D	130.51.120.0/22	130.51.120.1 a 130.51.123.254

Uma configuração possível para os departamentos será:

Departamento A	IP	Departamento B	IP
Ra	130.51.100.1/22	Rb	130.51.104.1/22
S1	130.51.100.2/22	n3	130.51.104.2/22
n1	130.51.100.3/22	n4	130.51.104.3/22
n2	130.51.100.4/22		
Departamento C	IP	Departamento D	IP
Rc	130.51.112.1/22	Rd	130.51.120.1/22
n5	130.51.112.2/22	n7	130.51.120.2/22
n6	130.51.112.3/22	n8	130.51.120.3/22

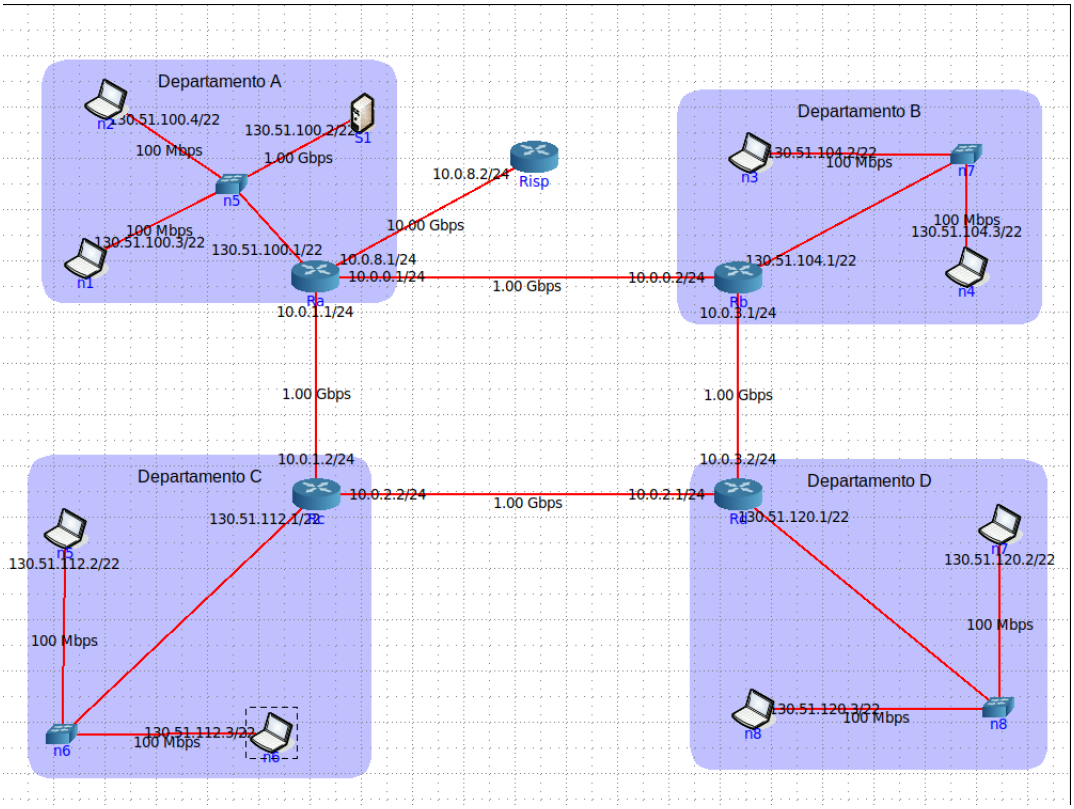
2 Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP pode interligar em cada departamento?

Como foi referido na alínea anterior, são necessários 3 bits para poder criar 4 subredes, isso implica que a máscara em formato decimal seja 255.255.252.0.

Não se pode mexer nos primeiros 22 bits, logo podem-se alterar 10 bits. Sendo assim, podemos interligar $2^{10} - 2 = 1022$ hosts, sendo que o primeiro é reservado para o IP e o último para o broadcasting.

3 Garanta e verifique que conectividade IP entre as várias redes locais da organização MIEI-RC é mantida. Explique como procedeu.

Inicialmente, trocamos os IPs dos computadores, servidor e routers dos departamentos, para os endereços de subneting.



Seguidamente, para comprovar que o subneting funcionou, realizamos o comando *ping [IP]* a partir de um computador no departamento C, para um computador de cada outro departamento.

```
root@n6:/tmp/pycore.33643/n6.conf# ping 130.51.100.4 -c 5
PING 130.51.100.4 (130.51.100.4) 56(84) bytes of data.
64 bytes from 130.51.100.4: icmp_seq=1 ttl=62 time=0.046 ms
64 bytes from 130.51.100.4: icmp_seq=2 ttl=62 time=0.054 ms
64 bytes from 130.51.100.4: icmp_seq=3 ttl=62 time=0.097 ms
64 bytes from 130.51.100.4: icmp_seq=4 ttl=62 time=0.090 ms
64 bytes from 130.51.100.4: icmp_seq=5 ttl=62 time=0.055 ms

--- 130.51.100.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4098ms
rtt min/avg/max/mdev = 0.046/0.068/0.097/0.022 ms
root@n6:/tmp/pycore.33643/n6.conf#
```

```
root@n6:/tmp/pycore.33643/n6.conf# ping 130.51.104.2 -c 5
PING 130.51.104.2 (130.51.104.2) 56(84) bytes of data.
64 bytes from 130.51.104.2: icmp_seq=1 ttl=61 time=0.059 ms
64 bytes from 130.51.104.2: icmp_seq=2 ttl=61 time=0.063 ms
64 bytes from 130.51.104.2: icmp_seq=3 ttl=61 time=0.064 ms
64 bytes from 130.51.104.2: icmp_seq=4 ttl=61 time=0.065 ms
64 bytes from 130.51.104.2: icmp_seq=5 ttl=61 time=0.092 ms

--- 130.51.104.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4200ms
rtt min/avg/max/mdev = 0.059/0.068/0.092/0.014 ms
root@n6:/tmp/pycore.33643/n6.conf#
```

```
root@n6:/tmp/pycore.33643/n6.conf# ping 130.51.120.2 -c 5
PING 130.51.120.2 (130.51.120.2) 56(84) bytes of data.
64 bytes from 130.51.120.2: icmp_seq=1 ttl=62 time=0.106 ms
64 bytes from 130.51.120.2: icmp_seq=2 ttl=62 time=0.067 ms
64 bytes from 130.51.120.2: icmp_seq=3 ttl=62 time=0.052 ms
64 bytes from 130.51.120.2: icmp_seq=4 ttl=62 time=0.052 ms
64 bytes from 130.51.120.2: icmp_seq=5 ttl=62 time=0.050 ms

--- 130.51.120.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4083ms
rtt min/avg/max/mdev = 0.050/0.065/0.106/0.022 ms
root@n6:/tmp/pycore.33643/n6.conf#
```

3 Conclusão

Este trabalho foi uma grande ajuda, para sedimentar os conhecimentos da disciplina.

Na primeira fase deste, abordamos a transmissão de dados referente a máquinas ligadas à mesma rede e a necessidade, ou não, de fragmentação no envio de pacotes de dados de modo a possibilitar a mesma. Em suma, trabalhamos sobre o protocolo IPV4.

Na segunda parte, abordamos o funcionamento do encaminhamento e endereçamento entre vários departamentos distintos, cada um com a sua sub-rede, e a diferença entre os dois tipos de encaminhamento e formas de conectar/desconectar equipamentos.