Trabalho Prático Nº2 – Protocolo IPv4 :: Datagramas IP e Fragmentação

Grupo 17 - Ana Rita Poças (a97284) , Bernard Georges (a96326) e João Pedro Braga (a97368)

Parte I

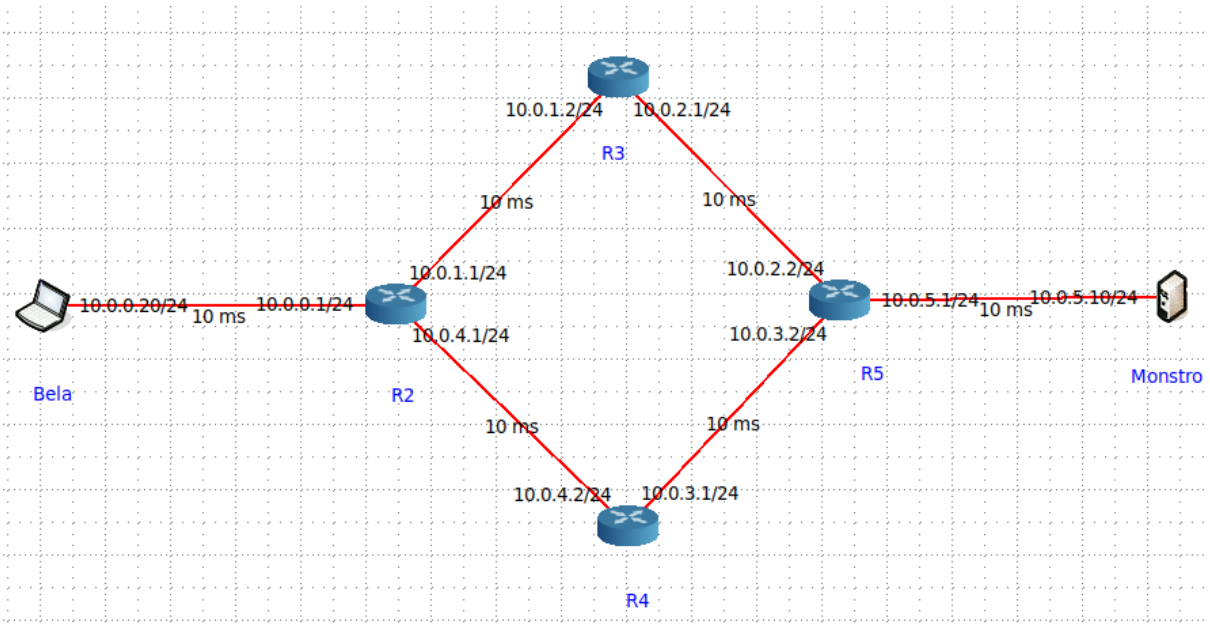
Questões

1. Prepare uma topologia CORE para verificar o comportamento do traceroute. Na topologia deve existir: um host (pc) cliente designado Bela cujo router de acesso é R2; o router R2 está simultaneamente ligado a dois routers R3 e R4; estes estão conectados a um router R5, que por sua vez, se liga a um host (servidor) designado Monstro. Ajuste o nome dos equipamentos atribuídos por defeito para o enunciado. Nas ligações (links) da rede de core estabeleça um tempo de propagação de 10ms. Após ativar a topologia, note que pode não existir conectividade IP imediata entre a Bela e o Monstro até que o anúncio de rotas entre routers estabilize.

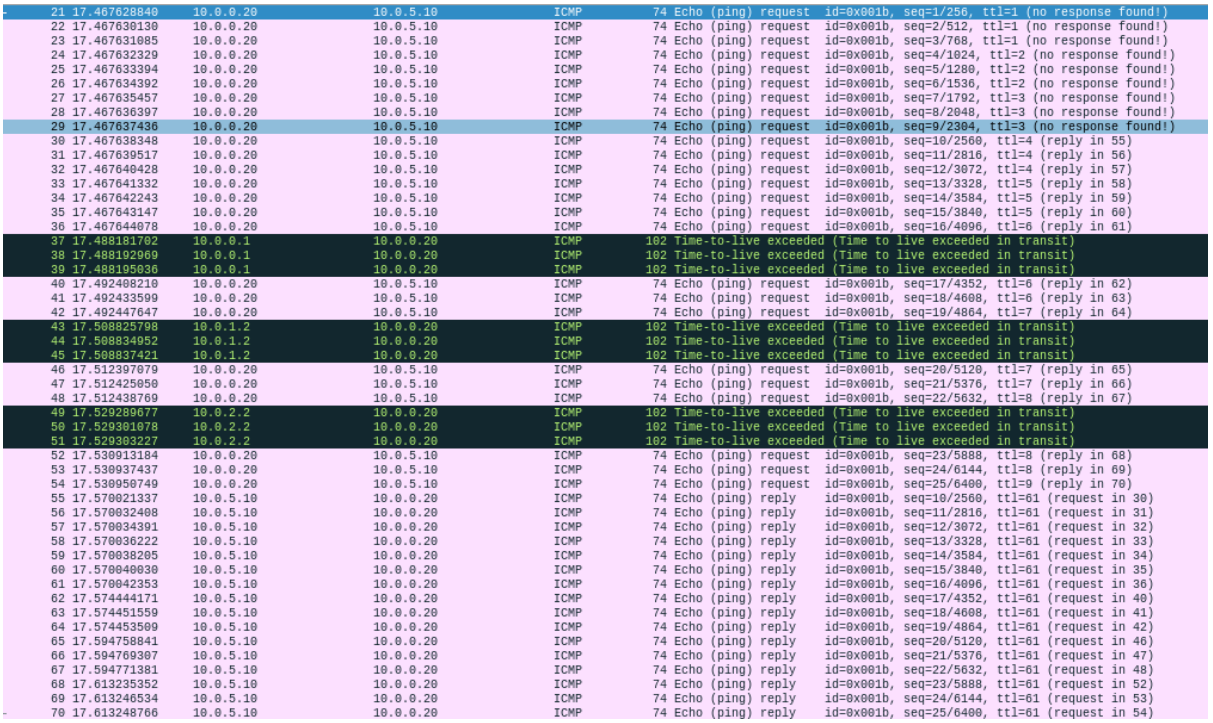
a. Active o wireshark ou o tcpdump no host Bela. Numa shell de Bela execute o comando traceroute -I para o endereço IP do Monstro.

R:

```
root@Bela:/tmp/pycore.40059/Bela.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)  41.145 ms  41.081 ms  41.075 ms
 2  10.0.1.2 (10.0.1.2)  61.706 ms  61.705 ms  61.701 ms
 3  10.0.2.2 (10.0.2.2)  82.155 ms  82.153 ms  82.151 ms
 4  10.0.5.10 (10.0.5.10)  122.870 ms  122.869 ms  122.866 ms
```



b. Registe e analise o tráfego ICMP enviado pelo sistema Bela e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.
R:

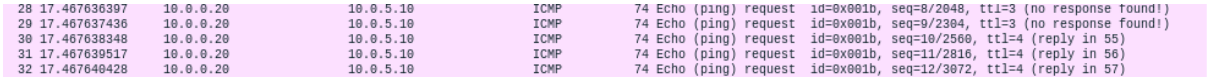


21	17.467628840	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=1/256, ttl=1 (no response found!)
22	17.467630130	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=2/512, ttl=1 (no response found!)
23	17.467631085	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=3/768, ttl=1 (no response found!)
24	17.467632329	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=4/1024, ttl=2 (no response found!)
25	17.467633394	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=5/1280, ttl=2 (no response found!)
26	17.467634392	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=6/1536, ttl=2 (no response found!)
27	17.467635457	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=7/1792, ttl=3 (no response found!)
28	17.467636397	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=8/2048, ttl=3 (no response found!)
29	17.467637436	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=9/2304, ttl=3 (no response found!)
30	17.467638348	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=10/2560, ttl=4 (reply in 55)
31	17.467639517	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=11/2816, ttl=4 (reply in 56)
32	17.467640428	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=12/3072, ttl=4 (reply in 57)
33	17.467641332	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=13/3328, ttl=5 (reply in 58)
34	17.467642243	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=14/3584, ttl=5 (reply in 59)
35	17.467643147	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=15/3840, ttl=5 (reply in 60)
36	17.467644078	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=16/4096, ttl=6 (reply in 61)
37	17.488181702	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
38	17.488192969	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
39	17.488195036	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
40	17.492408210	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=17/4352, ttl=6 (reply in 62)
41	17.492433599	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=18/4608, ttl=6 (reply in 63)
42	17.492447647	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=19/4864, ttl=7 (reply in 64)
43	17.508825798	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
44	17.508834952	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
45	17.508837421	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
46	17.512397079	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=20/5120, ttl=7 (reply in 65)
47	17.512425050	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=21/5376, ttl=7 (reply in 66)
48	17.512438769	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=22/5632, ttl=8 (reply in 67)
49	17.529289677	10.0.2.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
50	17.529301078	10.0.2.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
51	17.529303227	10.0.2.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
52	17.530913184	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=23/5888, ttl=8 (reply in 68)
53	17.530937437	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=24/6144, ttl=8 (reply in 69)
54	17.530950749	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=25/6400, ttl=9 (reply in 70)
55	17.570021337	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=10/2560, ttl=61 (request in 30)
56	17.570032408	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=11/2816, ttl=61 (request in 31)
57	17.570034391	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=12/3072, ttl=61 (request in 32)
58	17.570036222	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=13/3328, ttl=61 (request in 33)
59	17.570038205	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=14/3584, ttl=61 (request in 34)
60	17.570040030	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=15/3840, ttl=61 (request in 35)
61	17.570042353	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=16/4096, ttl=61 (request in 36)
62	17.574444171	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=17/4352, ttl=61 (request in 40)
63	17.574451559	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=18/4608, ttl=61 (request in 41)
64	17.574453509	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=19/4864, ttl=61 (request in 42)
65	17.594758841	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=20/5120, ttl=61 (request in 46)
66	17.594769307	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=21/5376, ttl=61 (request in 47)
67	17.594771381	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=22/5632, ttl=61 (request in 48)
68	17.613235352	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=23/5888, ttl=61 (request in 52)
69	17.613246534	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=24/6144, ttl=61 (request in 53)
70	17.613248766	10.0.5.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x001b, seq=25/6400, ttl=61 (request in 54)

O sistema Bela envia, através do protocolo ICMP para o sistema Monstro, pacotes de TTL incrementado, iniciando em 1, que só são respondidos pelo destino quando o seu TTL é igual ou superior a 4. Os resultados obtidos são coerentes, visto que quando o TTL não permite ao pacote chegar ao destino (TTL < 4), o router que recebeu esse pacote com o TTL no momento a 0 envia uma mensagem (pacote) de volta à fonte a informar que o pacote inicial não chegou devidamente ao destino.

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

R: O valor mínimo para o campo TTL é 4, que é provado pela existência de uma resposta apenas quando um pacote ICMP é enviado com TTL igual ou superior a 4.



28	17.467636397	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=8/2048, ttl=3 (no response found!)
29	17.467637436	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=9/2304, ttl=3 (no response found!)
30	17.467638348	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=10/2560, ttl=4 (reply in 55)
31	17.467639517	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=11/2816, ttl=4 (reply in 56)
32	17.467640428	10.0.0.20	10.0.5.10	ICMP	74 Echo (ping) request id=0x001b, seq=12/3072, ttl=4 (reply in 57)

d. Calcule o valor médio do tempo de ida-e-volta (RTT - Round-Trip Time) obtido no acesso ao servidor. Para melhorar a média, poderá alterar o número pacotes de prova com a opção `-q`.

R:

Sem a opção -q : 122.868 ms. Depois da opção -q: 84.873 ms

```
root@Bela:/tmp/pycore.40059/Bela.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)  41.145 ms  41.081 ms  41.075 ms
 2  10.0.1.2 (10.0.1.2)  61.706 ms  61.705 ms  61.701 ms
 3  10.0.2.2 (10.0.2.2)  82.155 ms  82.153 ms  82.151 ms
 4  10.0.5.10 (10.0.5.10)  122.870 ms  122.869 ms  122.866 ms
```

```
root@Bela:/tmp/pycore.40059/Bela.conf# traceroute -q 10 -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)  21.098 ms  21.039 ms  21.027 ms  21.019 ms  21.011 ms
21.002 ms * * * *
 2  10.0.1.2 (10.0.1.2)  41.212 ms  41.204 ms  41.196 ms  41.189 ms  41.180 ms
41.173 ms * * * *
 3  10.0.2.2 (10.0.2.2)  61.593 ms  61.594 ms  62.608 ms  62.591 ms  62.590 ms
62.588 ms * * * *
 4  10.0.5.10 (10.0.5.10)  88.061 ms  88.055 ms  84.642 ms  84.609 ms  84.935 ms
  84.904 ms  84.896 ms  84.889 ms  81.887 ms  81.854 ms
```

e. 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?

R: Não, uma vez que existem outros fatores a influenciar, visto que:
1) sendo o encaminhamento feito ponto a ponto, não temos como garantir que o caminho de ida seja o mesmo do caminho de volta;
2) o tempo que o router demora a processar cada um dos pacotes que recebe a cada instante;
3) não temos como garantir que o protocolo num ponto já tenha sido executado;

2. Pretende-se agora usar o traceroute na sua máquina nativa, e gerar datagramas IP de diferentes tamanhos.

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

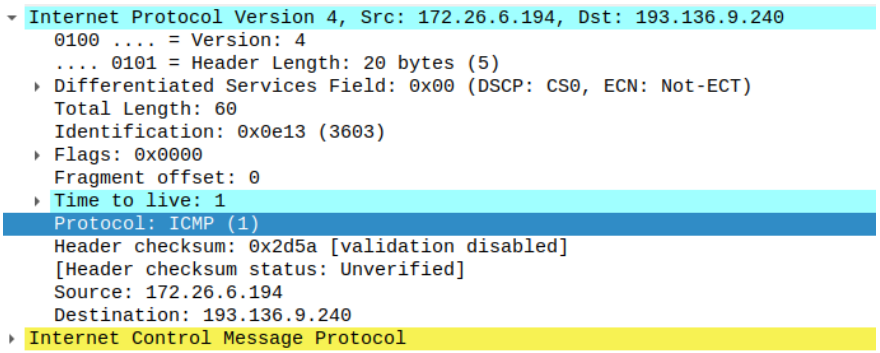
R: Source: 172.26.6.194

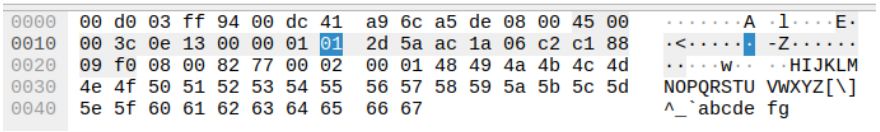
```
▾ Internet Protocol Version 4, Src: 172.26.6.194, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x0e13 (3603)
  ▸ Flags: 0x0000
    Fragment offset: 0
  ▸ Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x2d5a [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.6.194
    Destination: 193.136.9.240
▸ Internet Control Message Protocol
```

```
0000  00 d0 03 ff 94 00 dc 41  a9 6c a5 de 08 00 45 00   ······A ·l····E·
0010  00 3c 0e 13 00 00 01 01  2d 5a ac 1a 06 c2 c1 88   ·<····· -Z······
0020  09 f0 08 00 82 77 00 02  00 01 48 49 4a 4b 4c 4d   ·····w·· ··HIJKLM
0030  4e 4f 50 51 52 53 54 55  56 57 58 59 5a 5b 5c 5d   NOPQRSTU VWXYZ[\]
0040  5e 5f 60 61 62 63 64 65  66 67                     ^_`abcde fg
```

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

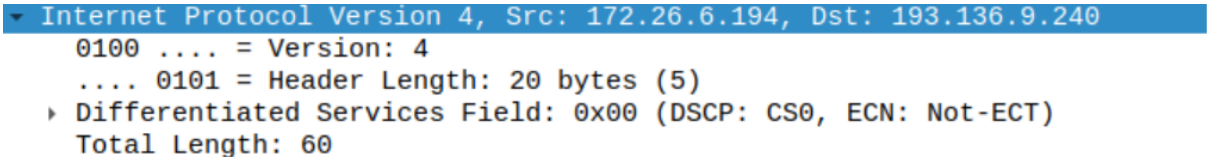
R: O valor do campo protocolo é ICMP (1), que permite identificar o ICMP como o protocolo usado.





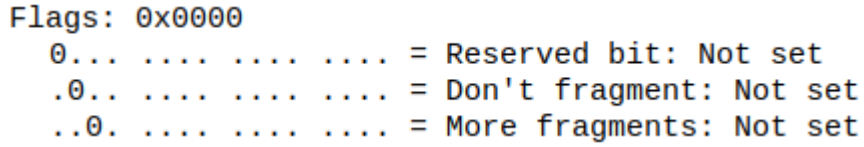
c. Quantos bytes tem o cabeçalho IPv4? Quantos bytes tem o campo de dados (payload) do datagrama? Como se calcula o tamanho do payload?

R: O cabeçalho contém 20 bytes e como o campo total contém 60 bytes, deduz-se então que os dados efetivos enviados são 40 bytes (60-20=40).



d. O datagrama IP foi fragmentado? Justifique.

R: O datagrama não foi fragmentado, isso é verificado através da análise da flag de fragmentação abaixo apresentada:



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.

R:Ordenação dos pacotes capturados:

7	1.851047057	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=1/256, ttl=1 (no response…
8	1.851055636	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=2/512, ttl=1 (no response…
9	1.851057409	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=3/768, ttl=1 (no response…
10	1.851059092	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=4/1024, ttl=2 (no respons…
11	1.851060634	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=5/1280, ttl=2 (no respons…
12	1.851062079	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=6/1536, ttl=2 (no respons…
13	1.851063677	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=7/1792, ttl=3 (no respons…
14	1.851065072	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=8/2048, ttl=3 (no respons…
15	1.851066714	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=9/2304, ttl=3 (no respons…
16	1.851068355	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=10/2560, ttl=4 (reply in …
17	1.851069823	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=11/2816, ttl=4 (reply in …
18	1.851071306	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=12/3072, ttl=4 (reply in …
19	1.851072985	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=13/3328, ttl=5 (reply in …
20	1.851074436	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=14/3584, ttl=5 (reply in …
21	1.851075919	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=15/3840, ttl=5 (reply in …
22	1.851077569	172.26.6.194	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=16/4096, ttl=6 (reply in …

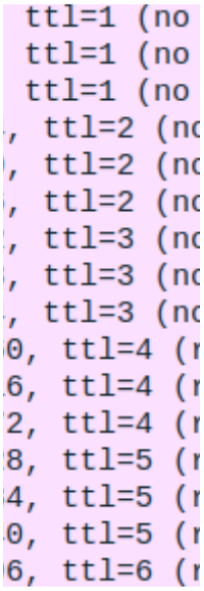
Os cabeçalhos IP variaram na sua identificação (*Identification*), no *Header checksum* e no TTL. Os primeiros dois alteram-se sempre, enquanto que o TTL nem sempre é alterado.

```
▸ Frame 7: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp11s0, id 0
▸ Ethernet II, Src: dc:41:a9:6c:a5:de (dc:41:a9:6c:a5:de), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▾ Internet Protocol Version 4, Src: 172.26.6.194, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x0e13 (3603)
  ▸ Flags: 0x0000
    Fragment offset: 0
  ▸ Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x2d5a [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.6.194
    Destination: 193.136.9.240
```

```
▸ Frame 16: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp11s0, id 0
▸ Ethernet II, Src: dc:41:a9:6c:a5:de (dc:41:a9:6c:a5:de), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▾ Internet Protocol Version 4, Src: 172.26.6.194, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x0e1c (3612)
  ▸ Flags: 0x0000
    Fragment offset: 0
  ▸ Time to live: 4
    Protocol: ICMP (1)
    Header checksum: 0x2a51 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.6.194
    Destination: 193.136.9.240
```

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

R: A imagem abaixo demonstra o facto de o TTL incrementar por 1 unidade a cada 3 pacotes:



De outro modo a identificação é sempre alterada de pacote em pacote, aumentando por uma unidade. As imagens abaixo representam dois pacotes enviados de seguida:

```
Frame 7: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp11s0, id 0
Ethernet II, Src: dc:41:a9:6c:a5:de (dc:41:a9:6c:a5:de), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.6.194, Dst: 193.136.9.240
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 60
  Identification: 0x0e13 (3603)
```

```
Frame 8: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp11s0, id 0
Ethernet II, Src: dc:41:a9:6c:a5:de (dc:41:a9:6c:a5:de), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.6.194, Dst: 193.136.9.240
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 60
  Identification: 0x0e14 (3604)
```

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?

R:

Ordenação do tráfego:

23	1.853704496	172.16.115.252	172.26.6.194	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
24	1.853705218	172.26.254.254	172.26.6.194	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
25	1.853705553	172.16.2.1	172.26.6.194	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
26	1.853705268	172.26.254.254	172.26.6.194	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
27	1.853705601	172.16.2.1	172.26.6.194	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
28	1.853707258	193.136.9.240	172.26.6.194	ICMP	74 Echo (ping) reply id=0x0002, seq=10/2560, ttl=61 (request in 16)
29	1.853705288	172.26.254.254	172.26.6.194	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
30	1.853705623	172.16.2.1	172.26.6.194	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
31	1.853707307	193.136.9.240	172.26.6.194	ICMP	74 Echo (ping) reply id=0x0002, seq=11/2816, ttl=61 (request in 17)
32	1.853707328	193.136.9.240	172.26.6.194	ICMP	74 Echo (ping) reply id=0x0002, seq=12/3072, ttl=61 (request in 18)
33	1.853707342	193.136.9.240	172.26.6.194	ICMP	74 Echo (ping) reply id=0x0002, seq=13/3328, ttl=61 (request in 19)
34	1.853707356	193.136.9.240	172.26.6.194	ICMP	74 Echo (ping) reply id=0x0002, seq=14/3584, ttl=61 (request in 20)
35	1.853707370	193.136.9.240	172.26.6.194	ICMP	74 Echo (ping) reply id=0x0002, seq=15/3840, ttl=61 (request in 21)
36	1.853707383	193.136.9.240	172.26.6.194	ICMP	74 Echo (ping) reply id=0x0002, seq=16/4096, ttl=61 (request in 22)
38	1.854090962	172.16.115.252	172.26.6.194	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
39	1.854090996	172.16.115.252	172.26.6.194	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

Alguns valores TTL destes pacotes *TTL exceeded*:

```
Time to live: 253
```

```
Time to live: 255
```

```
Time to live: 254
```

Os valores de TTL mudam de forma desordenada entre os pacotes que apresentam *time exceeded* contendo sempre um valor de um entre 253 - 255. Sendo o valor inicial 255, o decremento que ocorre é representativo do número de saltos que são efetuados até chegar ao nosso dispositivo.

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 (4000 + X) bytes.

No caso do nosso grupo, o tamanho do pacote será definido por 4000 + 17 = 4017 bytes.

```
lykifyar ~ traceroute -I marco.uminho.pt 4017
traceroute to marco.uminho.pt (193.136.9.240), 30 hops max, 4017 byte packets
 1  * * *
 2  172.16.2.1 (172.16.2.1)  8.297 ms  8.272 ms  8.246 ms
 3  172.16.115.252 (172.16.115.252)  8.217 ms  8.193 ms  8.169 ms
 4  marco.uminho.pt (193.136.9.240)  8.172 ms  8.148 ms  8.213 ms
```

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

R: A primeira mensagem ICMP é a seguinte:

```
▸ Frame 101: 1071 bytes on wire (8568 bits), 1071 bytes captured (8568 bits) on interface wlp11s0, id 0
▾ Ethernet II, Src: dc:41:a9:6c:a5:de (dc:41:a9:6c:a5:de), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
    ▸ Destination: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
    ▸ Source: dc:41:a9:6c:a5:de (dc:41:a9:6c:a5:de)
      Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 172.26.6.194, 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: 1057
      Identification: 0x22b7 (8887)
    ▸ Flags: 0x0172
      Fragment offset: 2960
    ▸ Time to live: 1
      Protocol: ICMP (1)
      Header checksum: 0x135f [validation disabled]
      [Header checksum status: Unverified]
      Source: 172.26.6.194
      Destination: 193.136.9.240
    ▸ [3 IPv4 Fragments (3997 bytes): #99(1480), #100(1480), #101(1037)]
▸ Internet Control Message Protocol
```

O objetivo desta fragmentação é dividir um pacote grande em pequenos blocos de dados de forma a mandar sem haver congestionamento de dados. Como este pacote inicial é maior que 1500 bytes é necessário dividi-lo de forma a ser possível o seu envio.

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?

R: Um pacote fragmentado contém a sua flag a 1, o que demonstra que há continuação da sua informação, já o primeiro contém o seu offset a 0.

```
Flags: 0x2000, More fragments
  0... .... .... .... = Reserved bit: Not set
  .0.. .... .... .... = Don't fragment: Not set
  ..1. .... .... .... = More fragments: Set
```

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?

R: O facto de o campo *Fragment offset* ser diferente de 0, indica que não se trata do primeiro fragmentos. Podemos afirmar também que há mais fragmentos uma vez que está indicado no cabeçalho *More Fragments*.

```
▾ Flags: 0x20b9, More fragments
    0... .... .... .... = Reserved bit: Not set
    .0.. .... .... .... = Don't fragment: Not set
    ..1. .... .... .... = More fragments: Set
  Fragment offset: 1480
```

d. Quantos fragmentos foram criados a partir do datagrama original?

R: Foram criados 3 fragmentos a partir do original, uma vez que o *More Fragments* está a 0 e o *offset* é o dobro de 1480.

```
 99 4.227804096   172.26.6.194    193.136.9.240   IPv4    1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=22b7) [Reassembled in #101]
100 4.227830285   172.26.6.194    193.136.9.240   IPv4    1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=22b7) [Reassembled in #101]
101 4.227835405   172.26.6.194    193.136.9.240   ICMP    1071 Echo (ping) request  id=0x0003, seq=1/256, ttl=1 (no response found!)
```

```
▾ Flags: 0x0172
    0... .... .... .... = Reserved bit: Not set
    .0.. .... .... .... = Don't fragment: Not set
    ..0. .... .... .... = More fragments: Not set
  Fragment offset: 2960
```

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.

R: Os campos alterados são a *flag*, mais particularmente o More Fragments, o *fragment offset* e o *header checksum*. Para reconstruir o datagrama original, é utilizado o Fragment offset para indicar a posição dos dados do atual fragmento no datagrama original e o More Fragments para indicar a existência de mais fragmentos além do atual.

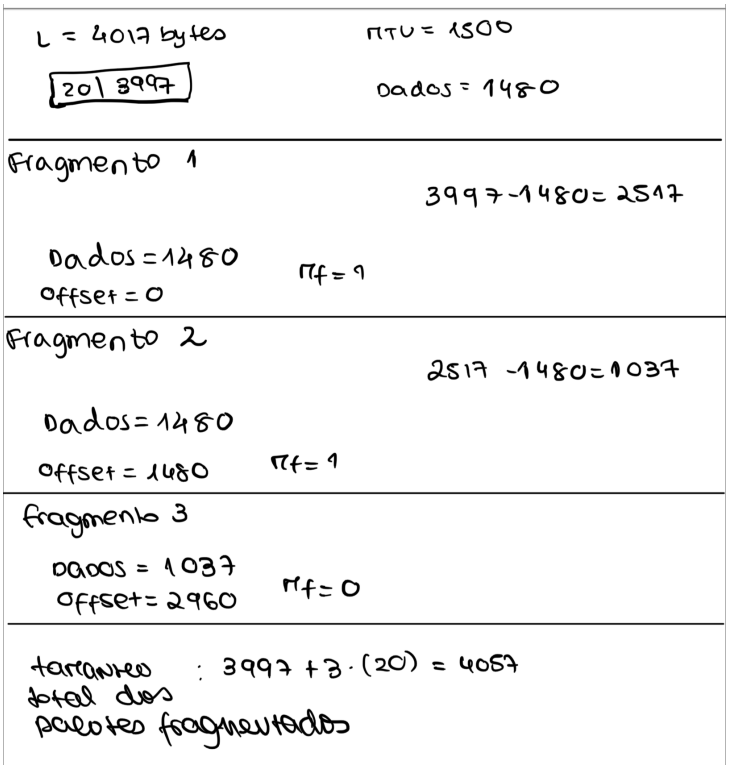
```
Frame 99: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface wlp11s0,
Ethernet II, Src: dc:41:a9:6c:a5:de (dc:41:a9:6c:a5:de), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:0
Internet Protocol Version 4, Src: 172.26.6.194, Dst: 193.136.9.240
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 1500
  Identification: 0x22b7 (8887)
▾ Flags: 0x2000, 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: 0xf315 [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.6.194
  Destination: 193.136.9.240
  Reassembled IPv4 in frame: 101
```

```
Frame 100: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface wlp11s0, id 0
Ethernet II, Src: dc:41:a9:6c:a5:de (dc:41:a9:6c:a5:de), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.6.194, 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: 0x22b7 (8887)
▾ Flags: 0x20b9, More fragments
    0... .... .... .... = Reserved bit: Not set
    .0.. .... .... .... = Don't fragment: Not set
    ..1. .... .... .... = More fragments: Set
  Fragment offset: 1480
▸ Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0xf25c [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.6.194
  Destination: 193.136.9.240
  Reassembled IPv4 in frame: 101
```

```
Frame 101: 1071 bytes on wire (8568 bits), 1071 bytes captured (8568 bits) on interface wlp11s0, id 0
Ethernet II, Src: dc:41:a9:6c:a5:de (dc:41:a9:6c:a5:de), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.6.194, 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: 1057
  Identification: 0x22b7 (8887)
▾ Flags: 0x0172
    0... .... .... .... = Reserved bit: Not set
    .0.. .... .... .... = Don't fragment: Not set
    ..0. .... .... .... = More fragments: Not set
  Fragment offset: 2960
▸ Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x135f [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.6.194
  Destination: 193.136.9.240
▸ [3 IPv4 Fragments (3997 bytes): #99(1480), #100(1480), #101(1037)]
```

f. Verifique o processo de fragmentação através de um processo de cálculo.

R: Sabendo que o comprimento total do pacote é de 1500 bytes e a mensagem total tem 4017 bytes e 20 bytes são de header, conclui-se que não é possível enviar os nossos pacotes, uma vez que o tamanho do pacote original é muito maior do que o que somos capazes de enviar.



g. Escreva uma expressão lógica que permita detetar o último fragmento correspondente ao datagrama original.

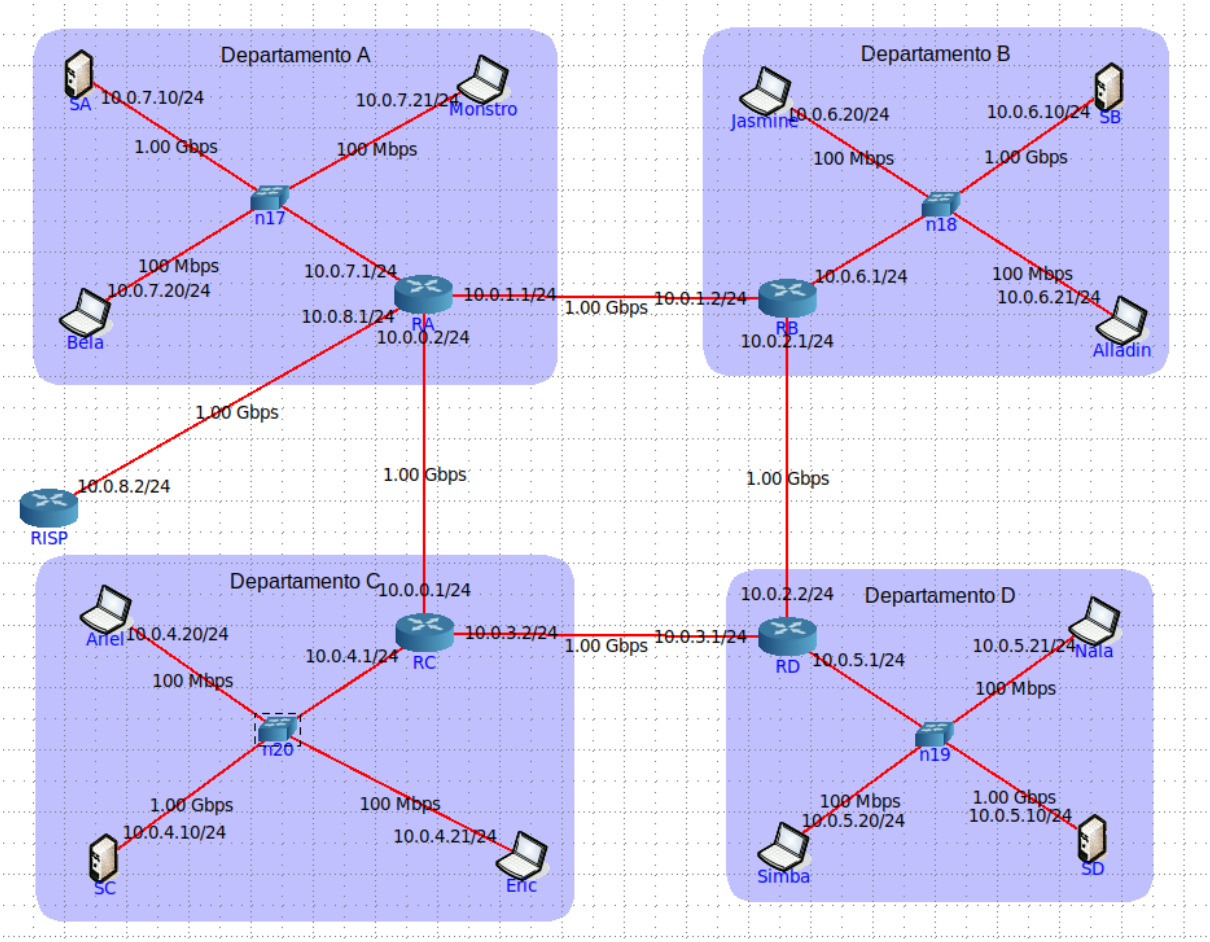
R: More Fragments == 0 && Fragment offset != 0 && $\forall\{1<=n_i<=n\}$ $IDn_i = IDn_{i+1}$

Parte II

Questões

Caso de estudo: Considere que a topologia de rede LEI-RC é distribuída 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 por departamento (SA, SB, SC, SD) e dois portáteis (pc) por departamento (A - Bela, Monstro; B - Jasmine, Alladin; C - Ariel, Eric; D - Simba e Nala), todos interligados ao router respetivo através de um comutador (switch). Cada servidor S tem uma ligação a 1Gbps e os laptops ligações a 100Mbps. Considere apenas a existência de um comutador por departamento.

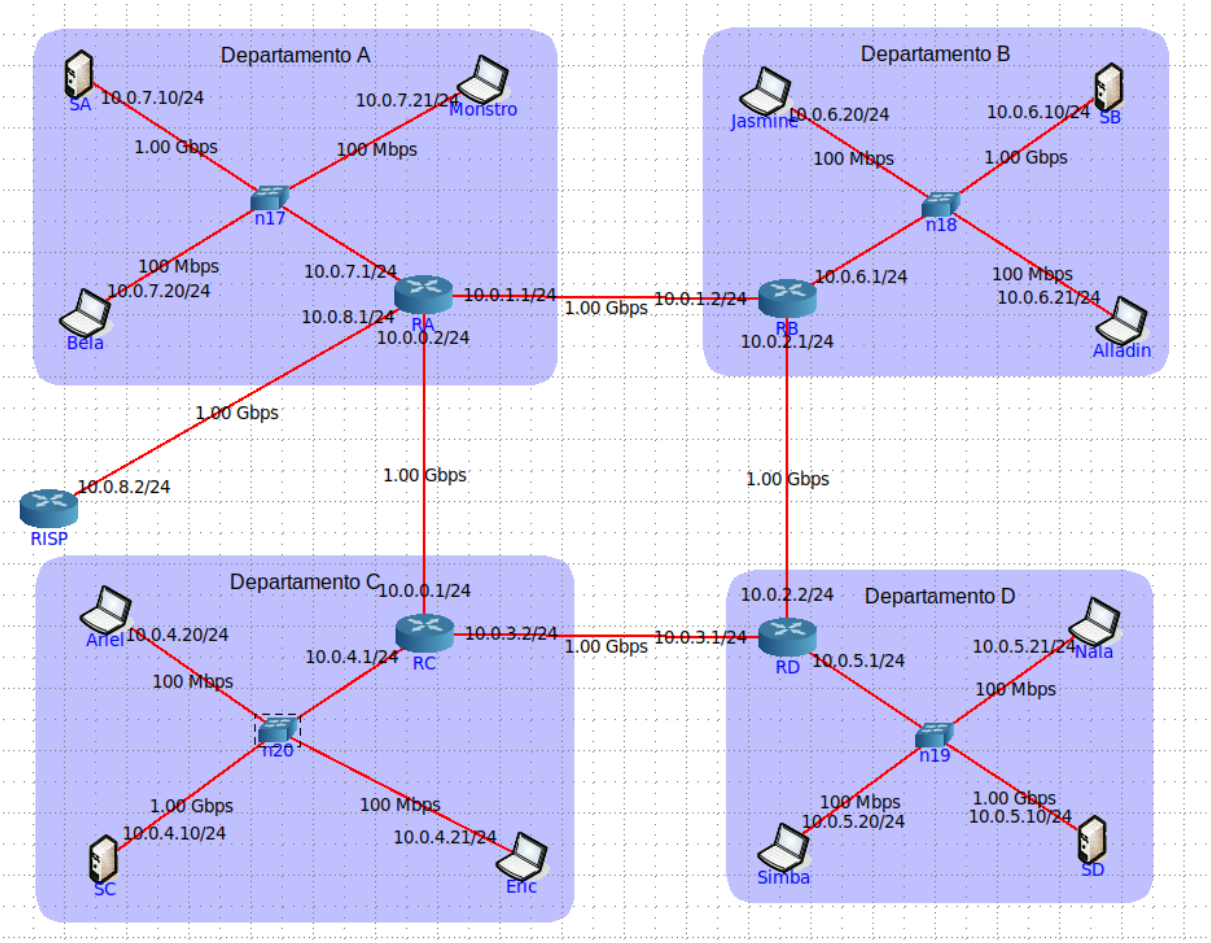
A conectividade IP externa da organização é assegurada através de um router de acesso RISP conectado a RA por uma ligação ponto-a-ponto a 1 Gbps. Construa uma topologia CORE que reflita a rede local da organização. Atribua as designações corretas aos equipamentos. Para facilitar a visualização pode ocultar o endereçamento IPv6. Grave a topologia para eventual reposição futura.



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.

R:



Na imagem acima é possível visualizar toda a topologia e endereçamento presente. Podemos observar que todos os dispositivos da rede possuem a mesma máscara de rede: 255.255.255.0

b. Tratam-se de endereços públicos ou privados? Porquê?

R: Tratam-se de endereços privados, pois pertencem a uma gama de valores IP reservados a redes privadas (neste caso a gama 10.0.0.0 a 10.255.255.255).

c. Por que razão não é atribuído um endereço IP aos switches?

R: O switch não tem um endereço IP atribuído, devido ao facto de ser apenas uma camada intermediária entre os diferentes dispositivos de rede.

d. Usando o comando ping certifique-se que existe conectividade IP interna a cada departamento (e.g. entre um laptop e o servidor respectivo).

R:

Bela

```
root@Bela:/tmp/pycore.34339/Bela.conf# ping 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data.
64 bytes from 10.0.7.10: icmp_seq=1 ttl=64 time=0.974 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=64 time=0.067 ms
64 bytes from 10.0.7.10: icmp_seq=3 ttl=64 time=0.269 ms
64 bytes from 10.0.7.10: icmp_seq=4 ttl=64 time=0.213 ms
```

Monstro

```
root@Bela:/tmp/pycore.34339/Bela.conf# ping 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data.
64 bytes from 10.0.7.10: icmp_seq=1 ttl=64 time=0.974 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=64 time=0.067 ms
64 bytes from 10.0.7.10: icmp_seq=3 ttl=64 time=0.269 ms
64 bytes from 10.0.7.10: icmp_seq=4 ttl=64 time=0.213 ms
```

Jasmine

```
root@Jasmine:/tmp/pycore.34339/Jasmine.conf# ping 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data.
64 bytes from 10.0.6.10: icmp_seq=1 ttl=64 time=1.26 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=64 time=0.223 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=64 time=0.208 ms
```

Alladin

```
root@Alladin:/tmp/pycore.34339/Alladin.conf# ping 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data.
64 bytes from 10.0.6.10: icmp_seq=1 ttl=64 time=0.533 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=64 time=0.221 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=64 time=0.191 ms
```

Ariel

```
root@Ariel:/tmp/pycore.34339/Ariel.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=64 time=1.31 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=64 time=0.215 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=64 time=0.214 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=64 time=0.189 ms
```

Eric

```
root@Eric:/tmp/pycore.34339/Eric.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=64 time=1.31 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=64 time=0.235 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=64 time=0.207 ms
```

Simba

```
root@Simba:/tmp/pycore.34339/Simba.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
64 bytes from 10.0.5.10: icmp_seq=1 ttl=64 time=1.11 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=64 time=0.210 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=64 time=0.202 ms
```

Nala

```
root@Nala:/tmp/pycore.34339/Nala.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
64 bytes from 10.0.5.10: icmp_seq=1 ttl=64 time=1.65 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=64 time=0.231 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=64 time=0.215 ms
```

e. Execute o número mínimo de comandos ping que lhe permite verificar a existência de conectividade IP entre departamentos.

R:

A para B

```
root@RA:/tmp/pycore.34339/RA.conf# ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=0.410 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=0.100 ms
```

A para C

```
root@RA:/tmp/pycore.34339/RA.conf# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=1.38 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.240 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.230 ms
```

A para D

```
root@RA:/tmp/pycore.34339/RA.conf# ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=63 time=0.497 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=63 time=0.374 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=63 time=0.376 ms
```

B para C

```
root@RB:/tmp/pycore.34339/RB.conf# ping 10.0.3.2
PING 10.0.3.2 (10.0.3.2) 56(84) bytes of data.
64 bytes from 10.0.3.2: icmp_seq=1 ttl=63 time=0.925 ms
64 bytes from 10.0.3.2: icmp_seq=2 ttl=63 time=0.381 ms
64 bytes from 10.0.3.2: icmp_seq=3 ttl=63 time=0.399 ms
```

B para D

```
root@RB:/tmp/pycore.34339/RB.conf# ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=0.681 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=64 time=0.105 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=64 time=0.219 ms
```

C para D

```
root@RC:/tmp/pycore.34339/RC.conf# ping 10.0.3.1
PING 10.0.3.1 (10.0.3.1) 56(84) bytes of data.
64 bytes from 10.0.3.1: icmp_seq=1 ttl=64 time=1.01 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=64 time=0.275 ms
64 bytes from 10.0.3.1: icmp_seq=3 ttl=64 time=0.178 ms
```

f. Verifique se existe conectividade IP do portátil Bela para o router de acesso RISP.

R:

```
root@Bela:/tmp/pycore.34339/Bela.conf# ping 10.0.8.2
PING 10.0.8.2 (10.0.8.2) 56(84) bytes of data.
64 bytes from 10.0.8.2: icmp_seq=1 ttl=63 time=0.398 ms
64 bytes from 10.0.8.2: icmp_seq=2 ttl=63 time=0.309 ms
64 bytes from 10.0.8.2: icmp_seq=3 ttl=63 time=0.309 ms
```

2) Para o router RA e o portátil Bela:

a. Execute o comando `netstat -rn` por forma a poder consultar a 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`).

R:

Tabela de endereçamento da Bela:

```
root@Bela:/tmp/pycore.34339/Bela.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
```

Router RA:

```
root@RA:/tmp/pycore.34339/RA.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
10.0.2.0        10.0.1.2        255.255.255.0   UG        0 0          0 eth1
10.0.3.0        10.0.0.1        255.255.255.0   UG        0 0          0 eth0
10.0.4.0        10.0.0.1        255.255.255.0   UG        0 0          0 eth0
10.0.5.0        10.0.0.1        255.255.255.0   UG        0 0          0 eth0
10.0.6.0        10.0.1.2        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        0.0.0.0         255.255.255.0   U         0 0          0 eth3
```

O portátil Bela apresenta uma entrada para os dispositivos da rede a que pertence (10.0.7.0) enquanto que a entrada default (0.0.0.0) define que qualquer pacote com outro destino fora da rede deve seguir para a gateway 10.0.7.1, o router A.
Já o router A possui entradas para as várias sub redes a que se consegue comunicar diretamente (sem gateway), como é o caso da sua própria rede (10.0.7.0), as ligações com

as redes adjacentes (10.0.0.0, 10.0.1.0) e o ISP (10.0.8.0), como também possui entradas para comunicação com redes não ligadas diretamente, como é o caso dos outros departamentos B, C e D (10.0.6.0 , 10.0.4.0, 10.0.5.0) e das subredes formadas entre os routers de B e D (10.0.2.0) e C e D (10.0.3.0).

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` ou equivalente).

R:

É um encaminhamento dinâmico, visto que estamos a usar o OSPFD. Este facto também pode ser observado pelo facto de sermos capazes de, ao correrrmos a topologia no core, todos os sistemas estarem ligados, sem ser necessária configuração prévia.

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 SA. Use o comando `route delete` para o efeito. Que implicações tem esta medida para os utilizadores da LEI-RC que acedem ao servidor. Justifique.

R:

Como é possível observar, ao fazermos delete da rota default, verificamos que a conectividade com as sub redes dos outros departamentos foi perdida, bem como o resto das redes acessíveis ao utilizador de LEI-RC. A única ligação que se mantém é a ligação com os dispositivos da própria rede.

d. Não volte a repor a rota por defeito. Adicione todas as rotas estáticas necessárias para restaurar a conectividade para o servidor SA, por forma a contornar a restrição imposta na alínea c). Utilize para o efeito o comando route add e registe os comandos que usou.

R: Comandos usados:

```
route add -net 10.0.4.0 netmask 255.255.255.0 gw 10.0.7.1
route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.7.1
route add -net 10.0.6.0 netmask 255.255.255.0 gw 10.0.7.1
route add -net 10.0.8.0 netmask 255.255.255.0 gw 10.0.7.1
```

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.

R:

Departamento C está novamente acessível a partir do A:

```
root@SA:/tmp/pycore.34339/SA.conf# ping 10.0.4.20
PING 10.0.4.20 (10.0.4.20) 56(84) bytes of data.
64 bytes from 10.0.4.20: icmp_seq=1 ttl=62 time=1.51 ms
64 bytes from 10.0.4.20: icmp_seq=2 ttl=62 time=0.127 ms
64 bytes from 10.0.4.20: icmp_seq=3 ttl=62 time=0.291 ms
```

Departamento B está novamente acessível a partir do A:

```
root@SA:/tmp/pycore.34339/SA.conf# ping 10.0.6.21
PING 10.0.6.21 (10.0.6.21) 56(84) bytes of data.
64 bytes from 10.0.6.21: icmp_seq=1 ttl=62 time=1.61 ms
64 bytes from 10.0.6.21: icmp_seq=2 ttl=62 time=0.442 ms
64 bytes from 10.0.6.21: icmp_seq=3 ttl=62 time=0.227 ms
```

Departamento D está novamente acessível a partir do A:

```
root@SA:/tmp/pycore.34339/SA.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
64 bytes from 10.0.5.10: icmp_seq=1 ttl=61 time=3.44 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=61 time=0.566 ms
```

RISC está novamente acessível a partir do A:

```
root@SA:/tmp/pycore.34339/SA.conf# ping 10.0.8.2
PING 10.0.8.2 (10.0.8.2) 56(84) bytes of data.
64 bytes from 10.0.8.2: icmp_seq=1 ttl=63 time=1.58 ms
64 bytes from 10.0.8.2: icmp_seq=2 ttl=63 time=0.300 ms
```

```
root@SA:/tmp/pycore.40309/SA.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.4.0        10.0.7.1        255.255.255.0   UG        0 0          0 eth0
10.0.5.0        10.0.7.1        255.255.255.0   UG        0 0          0 eth0
10.0.6.0        10.0.7.1        255.255.255.0   UG        0 0          0 eth0
10.0.7.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.8.0        10.0.7.1        255.255.255.0   UG        0 0          0 eth0
```

3. Definição de Sub-redes

Por forma a minimizar a falta de endereços IPv4 é comum a utilização de sub-redes. Além disso, a definição de sub-redes permite uma melhor organização do espaço de endereçamento das redes em questão. Para definir endereços de sub-rede é necessário usar a parte prevista para endereçamento de host, não sendo possível alterar o endereço de rede original. Recorda-se que o subnetting, ao recorrer ao espaço de endereçamento para host, implica que possam ser endereçados menos hosts. Considere a topologia definida anteriormente. Assuma que o endereçamento entre os routers (rede de backbone) se mantém inalterado, contudo, o endereçamento em cada departamento deve ser redefinido.

1) Considere que dispõe apenas do endereço de rede IP 192.168.XXX.128/25, em que XXX é o decimal correspondente ao seu número de grupo (PLXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo as redes de acesso externo e backbone inalteradas), sabendo que o número de departamentos pode vir a aumentar no curto prazo. Atribua endereços às interfaces dos vários sistemas envolvidos. Assuma que todos os endereços de subredes são usáveis. Justifique as opções tomadas no planeamento.

R:

Subnetting

PL17 →

192.168.17.128/25

Hosts = 4

Maximizar # subredes

$2^m - 2 \geq 4 \Leftrightarrow m \geq \log_2(6) \Leftrightarrow m \geq 2,\ldots$

↓

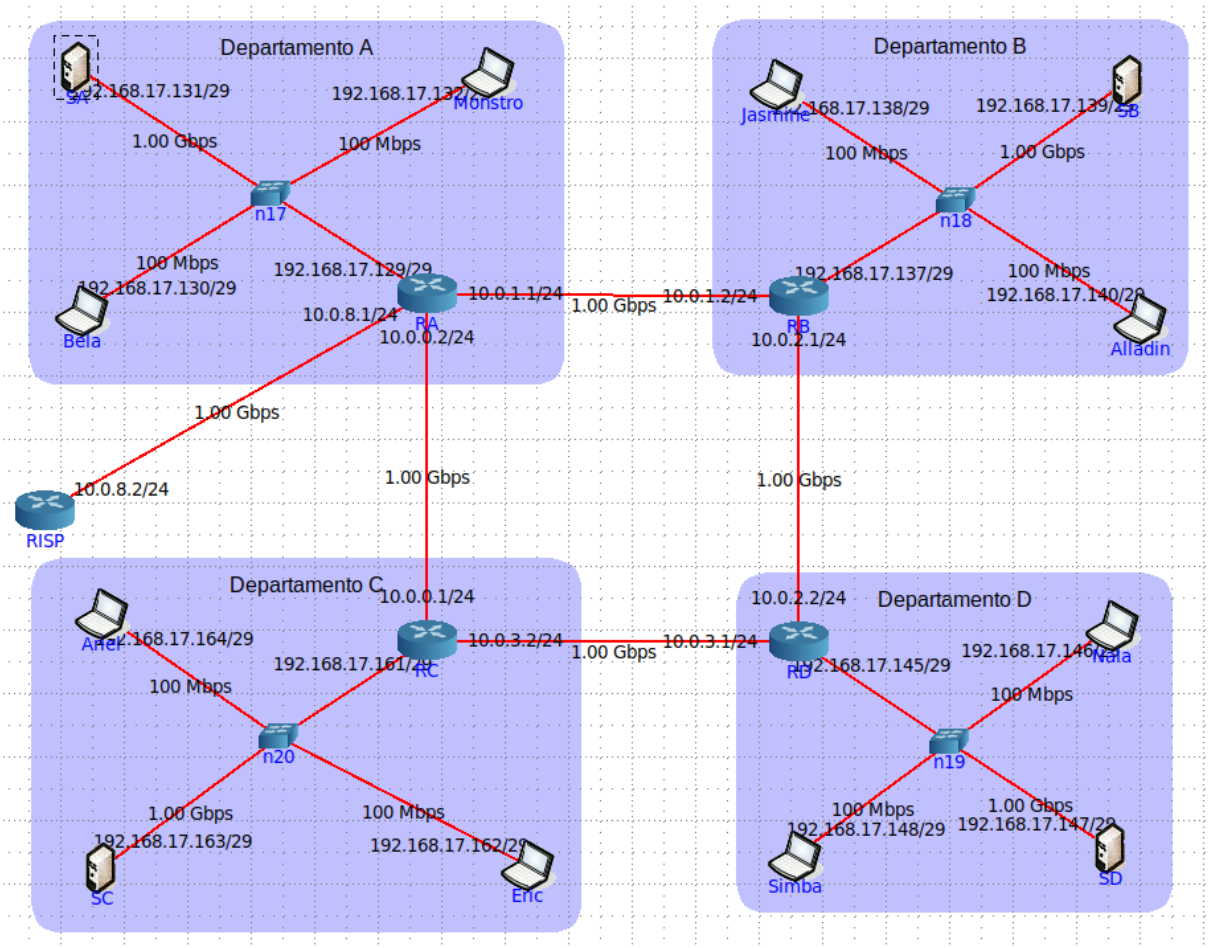
endereços que não podemos usar:
- → o da própria subrede
- → o de broadcast

3 bits para hosts ↓

32 - 3 - 25 = 4 bits ↓ para subredes

Subredes disponíveis

0	0000	4	0100	8	1000	12	1100
1	0001	5	0101	9	1001	13	1101
2	0010	6	0110	10	1010	14	1110
3	0011	7	0111	11	1011	15	1111



2) Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP pode interligar em cada departamento? Quantos prefixos de sub-rede ficam disponíveis para uso futuro? Justifique.

R: Usamos a máscara de rede 255.255.255.248;
Podemos interligar 6 endereços hosts;
Para uso futuro, ficam disponíveis 11 sub redes.

3) Verifique e garanta que a conectividade IP interna na rede local LEI-RC é mantida. No caso de não existência de conetividade, reveja a atribuição de endereços efetuada e eventuais erros de encaminhamento por forma a realizar as correções necessárias. Explique como procedeu.

R:

```
root@SA:/tmp/pycore.34339/SA.conf# ping 192.168.17.147
PING 192.168.17.147 (192.168.17.147) 56(84) bytes of data.
64 bytes from 192.168.17.147: icmp_seq=1 ttl=61 time=1.35 ms
64 bytes from 192.168.17.147: icmp_seq=2 ttl=61 time=0.833 ms
64 bytes from 192.168.17.147: icmp_seq=3 ttl=61 time=0.153 ms
```

```
root@SA:/tmp/pycore.34339/SA.conf# ping 192.168.17.163
PING 192.168.17.163 (192.168.17.163) 56(84) bytes of data.
64 bytes from 192.168.17.163: icmp_seq=1 ttl=62 time=0.923 ms
64 bytes from 192.168.17.163: icmp_seq=2 ttl=62 time=0.550 ms
64 bytes from 192.168.17.163: icmp_seq=3 ttl=62 time=0.407 ms
```

```
root@SA:/tmp/pycore.34339/SA.conf# ping 192.168.17.139
PING 192.168.17.139 (192.168.17.139) 56(84) bytes of data.
64 bytes from 192.168.17.139: icmp_seq=1 ttl=62 time=1.69 ms
64 bytes from 192.168.17.139: icmp_seq=2 ttl=62 time=0.380 ms
64 bytes from 192.168.17.139: icmp_seq=3 ttl=62 time=0.423 ms
```

Utilizamos o comando ping para verificar a conectividade entre dispositivos dos vários departamentos, com o exemplo acima do dispositivo Bela e a conexão entre dispositivos dos outros departamentos.

Conclusão

A realização deste trabalho prático, possibilitou o nosso primeiro contacto com os softwares CORE e WIRESHARK, assim como a consolidação dos conceitos que tínhamos adquirido, sobretudo os conceitos de Datagrama IP, Fragmentação, Endereçamento e Encaminhamento, que serão necessários para o bom aproveitamento à Unidade Curricular e necessários no decorrer das nossas aprendizagens como futuros Engenheiros Informáticos.