

Universidade do Minho  
Redes de Computadores  
Trabalho Prático II

Ano letivo 2021/2022

a97363, Gabriel Alexandre Monteiro da Silva

a96106, Miguel Silva Pinto

a97613, Pedro Miguel Castilho Martins

## Parte - 1

### 1. Exercício 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.

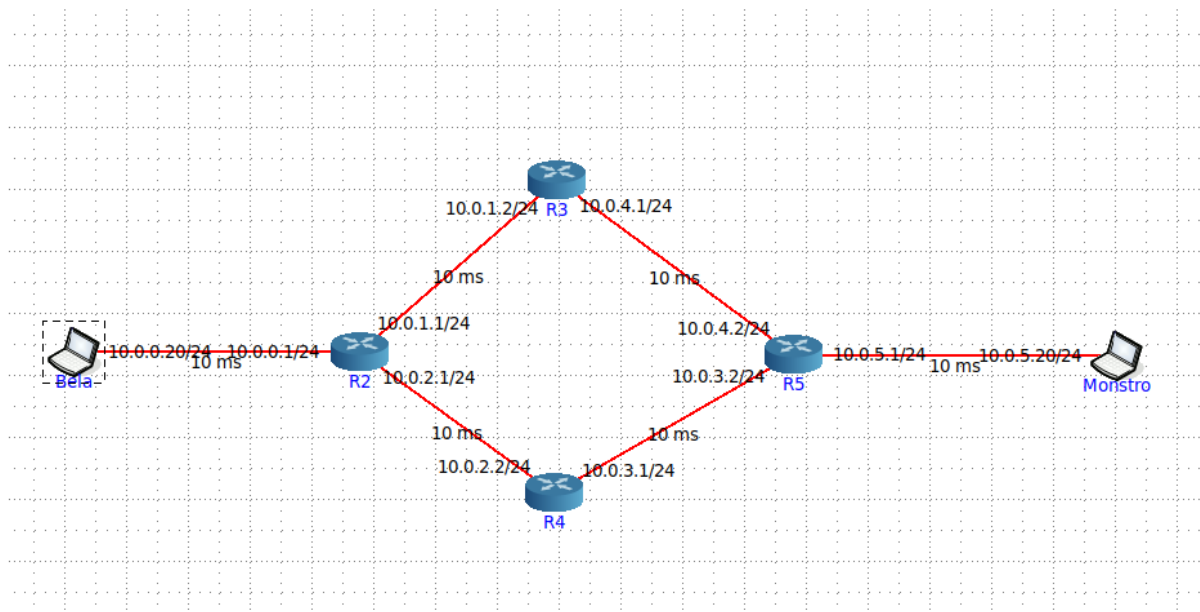


Fig. 1 - Topologia CORE.

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

```
vcmd
root@Bela:/tmp/pycore.37587/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) 42.930 ms 42.743 ms 42.737 ms
 2 10.0.1.2 (10.0.1.2) 63.012 ms 63.011 ms 63.009 ms
 3 10.0.2.2 (10.0.2.2) 83.677 ms 83.676 ms 83.676 ms
 4 10.0.5.10 (10.0.5.10) 125.950 ms 125.951 ms 125.949 ms
root@Bela:/tmp/pycore.37587/Bela.conf#
```

Fig. 2 - Comando traceroute para o endereço IP Monstro

**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:** O host “Bela” envia pings com diferentes valores de TTL com o intuito de descobrir a quantidade de entidades pelas quais necessita de passar de modo a atingir o host “Monstro”. O esperado era que fossem primeiramente enviados pings com valor 1 de TTL e o host “Bela” aguardasse resposta, seja o reply do host “Monstro” ou resposta dada pelo router onde o ping parou (linhas pretas) devido ao valor do TTL ser inferior ao necessário.

No.	Time	Source	Destination	Protocol	Length	Info
42	39.738939148	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=1/256, ttl=1 (no response found!)
43	39.738939465	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=2/512, ttl=1 (no response found!)
44	39.738939801	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=3/768, ttl=1 (no response found!)
45	39.738940094	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=4/1024, ttl=2 (no response found!)
46	39.738940349	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=5/1280, ttl=2 (no response found!)
47	39.738940683	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=6/1536, ttl=2 (no response found!)
48	39.738941013	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=7/1792, ttl=3 (no response found!)
49	39.738941337	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=8/2048, ttl=3 (no response found!)
50	39.738941592	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=9/2304, ttl=3 (no response found!)
51	39.738942155	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=10/2560, ttl=4 (reply in 76)
52	39.738942155	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=11/2816, ttl=4 (reply in 77)
53	39.738942425	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=12/3072, ttl=4 (reply in 78)
54	39.738942690	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=13/3328, ttl=5 (reply in 79)
55	39.738942968	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=14/3584, ttl=5 (reply in 80)
56	39.738943367	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=15/3840, ttl=5 (reply in 81)
57	39.738943655	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=16/4096, ttl=6 (reply in 82)
58	39.759874445	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
59	39.759873275	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
60	39.760972940	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
61	39.760563105	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=17/4352, ttl=6 (reply in 83)
62	39.760569760	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=18/4608, ttl=6 (reply in 84)
63	39.760573741	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=19/4864, ttl=7 (reply in 85)
64	39.760569356	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
65	39.780952815	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
66	39.780953791	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
67	39.780261840	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=20/5120, ttl=7 (reply in 86)
68	39.780270090	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=21/5376, ttl=7 (reply in 87)
69	39.780274051	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=22/5632, ttl=8 (reply in 88)
70	39.800967591	10.0.3.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
71	39.800968255	10.0.3.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
72	39.800968044	10.0.3.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
73	39.800991479	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=23/5888, ttl=8 (reply in 89)
74	39.800997544	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=24/6144, ttl=8 (reply in 90)
75	39.801001292	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=25/6400, ttl=9 (reply in 91)
76	39.841940892	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=10/2560, ttl=4 (request in 51)
77	39.842424821	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=11/2816, ttl=4 (request in 52)
78	39.842426432	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=12/3072, ttl=4 (request in 53)
79	39.842426999	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=13/3328, ttl=5 (request in 54)
80	39.842427533	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=14/3584, ttl=5 (request in 55)
81	39.842428113	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=15/3840, ttl=5 (request in 56)
82	39.842428642	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=16/4096, ttl=6 (request in 57)
83	39.842429160	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=17/4352, ttl=6 (request in 58)
84	39.842429793	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=18/4608, ttl=6 (request in 59)
85	39.842430466	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=19/4864, ttl=7 (request in 60)
86	39.862728586	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=20/5120, ttl=7 (request in 61)
87	39.862732638	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=21/5376, ttl=7 (request in 62)
88	39.862733428	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=22/5632, ttl=8 (request in 63)
89	39.882809714	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=23/5888, ttl=8 (request in 64)
90	39.882875920	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=24/6144, ttl=8 (request in 65)
91	39.882879657	10.0.5.20	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0018, seq=25/6400, ttl=9 (request in 66)

Fig. 3 - Tráfego ICMP.

**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 TTL mínimo para alcançar o host “Monstro” é 4.

42	39.738939148	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=1/256, ttl=1 (no response found!)
43	39.738939465	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=2/512, ttl=1 (no response found!)
44	39.738939801	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=3/768, ttl=1 (no response found!)
45	39.738940094	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=4/1024, ttl=2 (no response found!)
46	39.738940349	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=5/1280, ttl=2 (no response found!)
47	39.738940683	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=6/1536, ttl=2 (no response found!)
48	39.738941013	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=7/1792, ttl=3 (no response found!)
49	39.738941337	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=8/2048, ttl=3 (no response found!)
50	39.738941592	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=9/2304, ttl=3 (no response found!)
51	39.738942155	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=10/2560, ttl=4 (reply in 76)
52	39.738942155	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=11/2816, ttl=4 (reply in 77)
53	39.738942425	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=12/3072, ttl=4 (reply in 78)
54	39.738942690	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=13/3328, ttl=5 (reply in 79)
55	39.738942968	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=14/3584, ttl=5 (reply in 80)
56	39.738943367	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=15/3840, ttl=5 (reply in 81)
57	39.738943655	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x0018, seq=16/4096, ttl=6 (reply in 82)

Fig. 4 - “Protocolos ICMP, Wireshark”

Como podemos ver na figura acima, os pings enviados pelo host “Bela” apenas obtém resposta a partir da linha 51 onde o TTL utilizado tem valor 4.

**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:** O tempo médio de ida-e-volta (RTT - Round-Trip Time) é de aproximadamente 85.7366 ms.

```
27 ms 105,420 ms 105,419 ms 105,409 ms 81,519 ms 81,468 ms
root@Bela:/tmp/pycore.37587/Bela.conf# traceroute -I -q 10 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) 20,999 ms 20,978 ms 20,974 ms 20,971 ms 20,968 ms * * * * *
 2 10.0.1.2 (10.0.1.2) 41,274 ms 41,274 ms 41,271 ms 41,269 ms 41,266 ms * * * * *
 3 10.0.2.2 (10.0.2.2) 62,010 ms 63,227 ms 63,180 ms 63,171 ms 63,165 ms * * * * *
 4 10.0.5.10 (10.0.5.10) 85,457 ms 84,838 ms 84,093 ms 84,057 ms 84,049 ms 84,043 ms 89,135 ms 88,962 ms 88,936 ms 88,930 ms
root@Bela:/tmp/pycore.37587/Bela.conf#
```

Fig. 5 - “Comando (traceroute -I -q 10 10.0.5.10)”

**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:** O valor médio do atraso num sentido (One-Way Delay) não pode ser calculado com precisão dividindo o RTT por 2 porque o tempo de ida e de volta podem não ser o mesmo. O cálculo desta métrica é difícil porque pode não existir sincronização entre host.

## 2. Exercício 2

Usando o wireshark capture o tráfego gerado pelo traceroute para os seguintes tamanhos de pacote: situação (i) sem especificar, i.e., usando o tamanho do pacote de prova por defeito; e situação (ii) (4000 + X) bytes, em que X é o número do grupo de trabalho. Utilize como máquina destino o host marco.uminho.pt. Pare a captura. Com base no tráfego capturado, identifique os pedidos ICMP Echo Request e o conjunto de mensagens devolvidas como resposta.

Selecione a primeira mensagem ICMP capturada (referente à situação (i) tamanho por defeito) e centre a análise no nível protocolar IP (expandir o tab correspondente na janela de detalhe do wireshark). Através da análise do cabeçalho IP diga:

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

```
▶ Frame 11: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
▶ Linux cooked capture
▼ Internet Protocol Version 4, Src: 172.26.36.203, 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: 0xfbe1 (64481)
    ▶ Flags: 0x0000
      Fragment offset: 0
    ▶ Time to live: 1
      Protocol: ICMP (1)
      Header checksum: 0x2182 [validation disabled]
```

**Fig. 6** - Indicação do endereço fonte do pacote.

**R:** Src: 172.26.36.203.

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

```
▶ Frame 11: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
▶ Linux cooked capture
▼ Internet Protocol Version 4, Src: 172.26.36.203, 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: 0xfbe1 (64481)
    ▶ Flags: 0x0000
      Fragment offset: 0
    ▶ Time to live: 1
      Protocol: ICMP (1)
      Header checksum: 0x2182 [validation disabled]
```

**Fig. 7** - Indicação do protocolo.

**R:** O valor do campo protocolo é ICMP (1), permitindo identificar o protocolo usado para enviar a mensagem.

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

```
▶ Frame 11: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
▶ Linux cooked capture
▼ Internet Protocol Version 4, Src: 172.26.36.203, 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: 0xfbe1 (64481)
    ▶ Flags: 0x0000
      Fragment offset: 0
    ▶ Time to live: 1
      Protocol: ICMP (1)
      Header checksum: 0x2182 [validation disabled]
```

**Fig. 8** - Indicação do tamanho das componentes do pacote.

**R:** O cabeçalho tem 20 bytes. O payload do datagrama tem 40 bytes. O payload é calculado a partir da expressão: 60(Total length) - 20(Header length) = 40.

d) O datagrama IP foi fragmentado? Justifique.

```

Internet Protocol Version 4, Src: 172.26.36.203, 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: 0x96ff (38655)
  Flags: 0x0000
    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)

```

Fig. 9 - Verificação da fragmentação do pacote.

**R:** O datagrama não foi fragmentado, pois o Fragment offset apresenta valor 0 e a flag “More Fragments” também tem valor 0.

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.

No.	Time	Source	Destination	Protocol	Length	Info
258	20.146754737	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=1/256, ttl=1 (no response...
259	20.146790317	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=2/512, ttl=1 (no response...
260	20.146799758	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=3/768, ttl=1 (no response...
261	20.146808223	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=4/1024, ttl=2 (no response...
262	20.146815777	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=5/1280, ttl=2 (no response...
263	20.146823206	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=6/1536, ttl=2 (no response...
264	20.146830678	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=7/1792, ttl=3 (no response...
265	20.146837495	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=8/2048, ttl=3 (no response...
266	20.146844704	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=9/2304, ttl=3 (no response...
267	20.146853505	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=10/2560, ttl=4 (reply in ...
268	20.146861121	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=11/2816, ttl=4 (reply in ...
269	20.146869290	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=12/3072, ttl=4 (reply in ...
270	20.146879355	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=13/3328, ttl=5 (reply in ...
271	20.146887097	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=14/3584, ttl=5 (reply in ...
272	20.146894947	172.26.36.203	193.136.9.240	ICMP	76	Echo (ping) request id=0x000b, seq=15/3840, ttl=5 (reply in ...

Fig. 10 - Datagramas ordenados de acordo com o endereço IP..

**R:** Os campos alterados de pacote para pacote são, o TTL,o Header Checksum e a Identification.

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

**R:** A identificação do datagrama IP incrementa 1 a cada pacote enviado. No caso do TTL, o valor standard de pacotes enviados com o mesmo valor TTL é 3, pelo que, o valor de TTL incrementa a cada 3 pacotes enviados.

g) Ordene o tráfego capturado por endereço destino e encontre a série de respostas ICMP TTL exceeded enviadas ao seu computador. 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: O valor do TTL não permanece constante variando entre 253 e 255, uma vez que passa por 3 routers intermédios.

### 3. Exercício 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 4044 bytes.

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

No.	Time	Source	Destination	Protocol	Length	Info
26	2.753960955	127.0.0.53	127.0.0.1	DNS	88	Standard query response 0x362a AAAA marco.uminho.pt OPT
27	2.754031649	127.0.0.53	127.0.0.1	DNS	104	Standard query response 0x72d9 A marco.uminho.pt A 193.136.9.240 OPT
28	2.754124811	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol (proto=ICMP 1, off=0, ID=8e2f) [Reassembled in #30]
29	2.754136171	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=8e2f) [Reassembled in #30]
30	2.754139029	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id=0x000c, seq=1/256, ttl=1 (no response found!)
31	2.754147315	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol (proto=ICMP 1, off=0, ID=8e30) [Reassembled in #33]
32	2.754149825	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=8e30) [Reassembled in #33]
33	2.754152082	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id=0x000c, seq=2/512, ttl=1 (no response found!)

Frame 28: 1516 bytes on wire (12128 bits), 1516 bytes captured (12128 bits) on interface any, id 0  
 Linux cooked capture  
 Internet Protocol Version 4, Src: 172.26.36.203, 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: 0x8e2f (36399)  
 Flags: 0x2000, More fragments  
 Fragment offset: 0  
 Time to live: 1

Fig. 11 - Fragmentação da primeira mensagem.

R: O limite máximo de bytes enviados é 1500, no entanto, os pacotes enviados possuem 4044 bytes de tamanho sendo necessária a fragmentação do pacote.

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?



No.	Time	Source	Destination	Protocol	Length	Info
109	29.303401356	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
110	29.303432163	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
111	29.303438046	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
112	29.303456761	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
113	29.303462387	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
114	29.303466235	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
115	29.303480903	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
116	29.303486874	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
117	29.303491340	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
118	29.303506990	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
119	29.303511869	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
120	29.303515837	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
121	29.303530922	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol

▶ Frame 109: 1516 bytes on wire (12128 bits), 1516 bytes captured (12128 bits) on interface any, id 0  
 ▶ Linux cooked capture  
 ▶ Internet Protocol Version 4, Src: 172.26.36.203, 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: 0x7b62 (31586)  
     ▶ 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

Fig. 12 - Fragmentação da primeira mensagem.

**R:** No cabeçalho conseguimos saber se o datagrama foi fragmentado a partir das flags. A flag “More fragments” está a 1 logo sabemos que o datagrama foi fragmentado. Se o campo “Fragment offset” estiver a 0 significa que este é o primeiro fragmento do datagrama. O tamanho do datagrama IP é 1500 dado pelo campo “Total Length”.

**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?**

No.	Time	Source	Destination	Protocol	Length	Info
109	29.303401356	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
110	29.303432163	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
111	29.303438046	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
112	29.303456761	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
113	29.303462387	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
114	29.303466235	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
115	29.303480903	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
116	29.303486874	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
117	29.303491340	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
118	29.303506990	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
119	29.303511869	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
120	29.303515837	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
121	29.303530922	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol

▶ Frame 110: 1516 bytes on wire (12128 bits), 1516 bytes captured (12128 bits) on interface any, id 0  
 ▶ Linux cooked capture  
 ▶ Internet Protocol Version 4, Src: 172.26.36.203, 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: 0x7b62 (31586)  
     ▶ 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

Fig. 13 - 2º fragmento do datagrama.

**R:** O campo “Fragment offset” não está a 0 logo não se trata do primeiro fragmento. Existem mais fragmentos porque a flag “More fragments” está a 1.



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

No.	Time	Source	Destination	Protocol	Length	Info
109	29.303401356	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
110	29.303432163	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
111	29.303438046	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
112	29.303456761	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
113	29.303462387	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
114	29.303466235	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
115	29.303480903	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
116	29.303486874	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
117	29.303491340	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
118	29.303506990	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
119	29.303511869	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
120	29.303515837	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
121	29.303530922	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
Frame 111: 1100 bytes on wire (8800 bits), 1100 bytes captured (8800 bits) on interface any, id 0						
Linux cooked capture						
Internet Protocol Version 4, Src: 172.26.36.203, 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: 1084						
Identification: 0x7b62 (31586)						
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						

Fig. 14 - 3º fragmento do datagrama.

R: Foram criados 3 fragmentos pois existem 3 datagramas com a mesma “Identification” e o último datagrama possui a flag “More fragments” a 0.

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.

No.	Time	Source	Destination	Protocol	Length	Info
108	29.303137268	127.0.0.53	127.0.0.1	DNS	88	Standard query response
109	29.303401356	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
110	29.303432163	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
111	29.303438046	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
112	29.303456761	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
113	29.303462387	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
114	29.303466235	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
115	29.303480903	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
116	29.303486874	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
117	29.303491340	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
118	29.303506990	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
119	29.303511869	172.26.36.203	193.136.9.240	IPv4	1516	Fragmented IP protocol
120	29.303515837	172.26.36.203	193.136.9.240	ICMP	1100	Echo (ping) request id
Frame 109: 1516 bytes on wire (12128 bits), 1516 bytes captured (12128 bits) on interface any, id 0						
Linux cooked capture						
Internet Protocol Version 4, Src: 172.26.36.203, 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: 0xb62 (31586)						
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: 0x7c61 [validation disabled]						
[Header checksum status: Unverified]						
Source: 172.26.36.203						
Destination: 193.136.9.240						

Fig. 15 - Indicação dos campos que mudam.

**R:** Os campos que mudam são o “Total length”, “Fragment offset” e o “Header checksum”. Estes campos permitem reconstruir o datagrama original porque sabemos onde cada fragmento começa a partir do “Fragment offset” e o tamanho de cada um a partir do “Total length”, e como todos têm a mesma “Identification” sabemos que pertencem ao mesmo datagrama.

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

**R:** O tamanho é de 4044 bytes logo, com o limite de 1500 bytes e um header de tamanho 20, cada datagrama só pode levar 1480 bytes de cada vez, logo os fragmentos ficam  $20 + 1480 + 1480 + 1064$ .

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

**R:** (“MoreFragments” == 0 && “FragmentOffset” != 0 && “Identification” == (id do datagrama original)).

## Parte - 2

- 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.

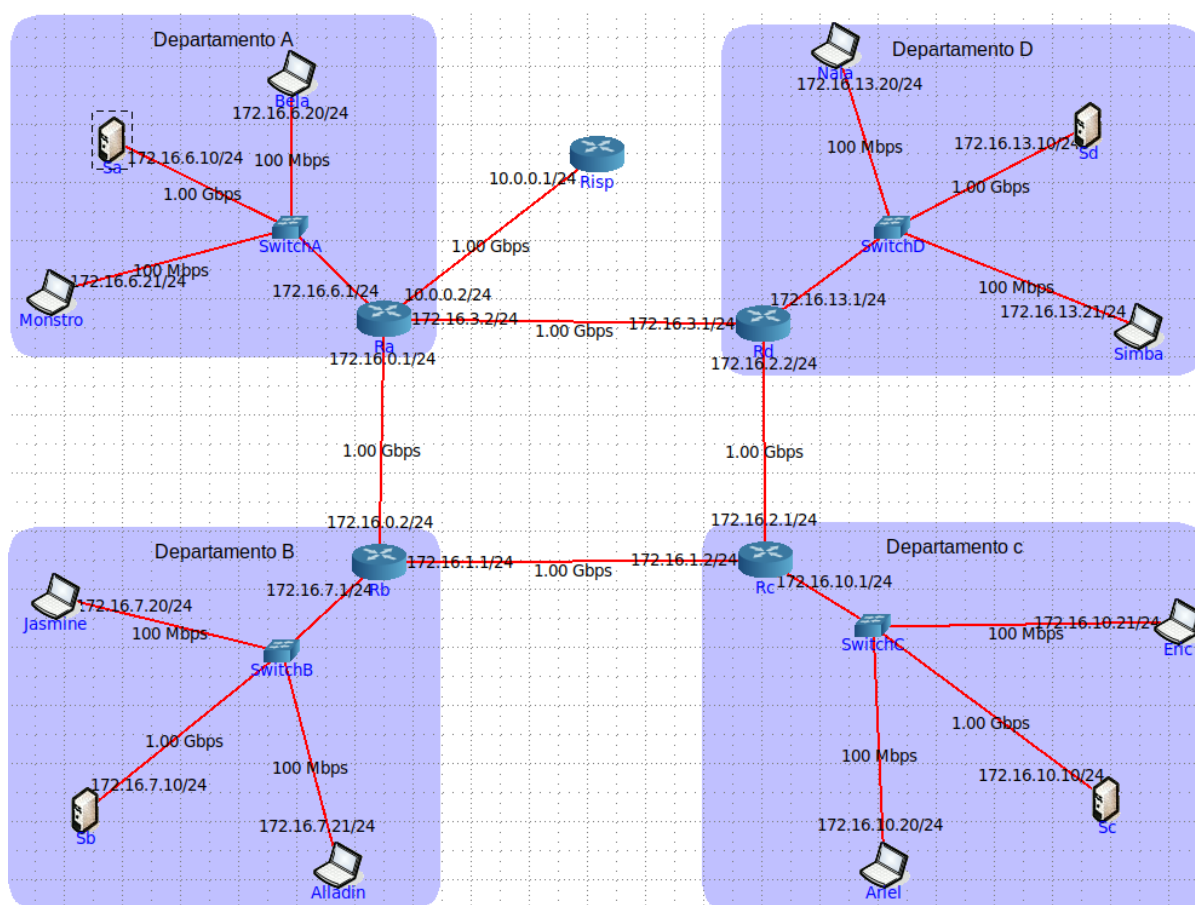


Fig. 16 - Topologia Core.

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

10.0.0.0	-	10.255.255.255 (10/8 prefix)
172.16.0.0	-	172.31.255.255 (172.16/12 prefix)
192.168.0.0	-	192.168.255.255 (192.168/16 prefix)

**R:** Tratam-se de endereços privados, pois estão no espaço reservado a endereços privados.

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

**R:** Os switches não têm um endereço de IP, pois um switch não funciona ao nível da rede, mas sim ao nível de ligação de dados.

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

```
vcmd
root@Sa:/tmp/pycore.37477/Sa.conf# ping 172.16.6.20
PING 172.16.6.20 (172.16.6.20) 56(84) bytes of data.
64 bytes from 172.16.6.20: icmp_seq=1 ttl=64 time=1.03 ms
64 bytes from 172.16.6.20: icmp_seq=2 ttl=64 time=0.393 ms
64 bytes from 172.16.6.20: icmp_seq=3 ttl=64 time=1.17 ms
64 bytes from 172.16.6.20: icmp_seq=4 ttl=64 time=0.365 ms
64 bytes from 172.16.6.20: icmp_seq=5 ttl=64 time=0.208 ms
^C
--- 172.16.6.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4045ms
rtt min/avg/max/mdev = 0.208/0.632/1.170/0.388 ms
root@Sa:/tmp/pycore.37477/Sa.conf# ping 172.16.6.21
PING 172.16.6.21 (172.16.6.21) 56(84) bytes of data.
64 bytes from 172.16.6.21: icmp_seq=1 ttl=64 time=0.706 ms
64 bytes from 172.16.6.21: icmp_seq=2 ttl=64 time=0.400 ms
64 bytes from 172.16.6.21: icmp_seq=3 ttl=64 time=0.264 ms
64 bytes from 172.16.6.21: icmp_seq=4 ttl=64 time=0.428 ms
64 bytes from 172.16.6.21: icmp_seq=5 ttl=64 time=0.401 ms
^C
--- 172.16.6.21 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4078ms
rtt min/avg/max/mdev = 0.264/0.439/0.706/0.144 ms
root@Sa:/tmp/pycore.37477/Sa.conf# S
```

Fig. 17 - Certificação da conectividade

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

**R:** Para testar a conectividade entre os departamentos seriam precisos 6 pings, A->(b,c,d); B->(c,d); C->(d).

```
root@Bela:/tmp/pycore.35889/Bela.conf# ping 172.16.7.20
PING 172.16.7.20 (172.16.7.20) 56(84) bytes of data.
64 bytes from 172.16.7.20: icmp_seq=1 ttl=62 time=0.676 ms
64 bytes from 172.16.7.20: icmp_seq=2 ttl=62 time=0.370 ms
64 bytes from 172.16.7.20: icmp_seq=3 ttl=62 time=0.365 ms
64 bytes from 172.16.7.20: icmp_seq=4 ttl=62 time=0.418 ms
^C
--- 172.16.7.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3070ms
rtt min/avg/max/mdev = 0.365/0.457/0.676/0.127 ms
root@Bela:/tmp/pycore.35889/Bela.conf# ping 172.16.10.20
PING 172.16.10.20 (172.16.10.20) 56(84) bytes of data.
64 bytes from 172.16.10.20: icmp_seq=1 ttl=61 time=1.10 ms
64 bytes from 172.16.10.20: icmp_seq=2 ttl=61 time=0.679 ms
64 bytes from 172.16.10.20: icmp_seq=3 ttl=61 time=0.445 ms
64 bytes from 172.16.10.20: icmp_seq=4 ttl=61 time=0.494 ms
^C
--- 172.16.10.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3053ms
rtt min/avg/max/mdev = 0.445/0.680/1.104/0.259 ms
root@Bela:/tmp/pycore.35889/Bela.conf# ping 172.16.13.20
PING 172.16.13.20 (172.16.13.20) 56(84) bytes of data.
64 bytes from 172.16.13.20: icmp_seq=1 ttl=62 time=0.770 ms
64 bytes from 172.16.13.20: icmp_seq=2 ttl=62 time=0.829 ms
64 bytes from 172.16.13.20: icmp_seq=3 ttl=62 time=0.480 ms
64 bytes from 172.16.13.20: icmp_seq=4 ttl=62 time=0.575 ms
^C
--- 172.16.13.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3056ms
rtt min/avg/max/mdev = 0.480/0.663/0.829/0.141 ms
root@Bela:/tmp/pycore.35889/Bela.conf#
```

**A->(b,c,d)**

```

root@Jasmine:/tmp/pycore.35889/Jasmine.conf# ping 172.16.10.20
PING 172.16.10.20 (172.16.10.20) 56(84) bytes of data.
64 bytes from 172.16.10.20: icmp_seq=1 ttl=62 time=0.718 ms
64 bytes from 172.16.10.20: icmp_seq=2 ttl=62 time=0.512 ms
64 bytes from 172.16.10.20: icmp_seq=3 ttl=62 time=0.612 ms
64 bytes from 172.16.10.20: icmp_seq=4 ttl=62 time=0.480 ms
^C
--- 172.16.10.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3070ms
rtt min/avg/max/mdev = 0.480/0.580/0.718/0.093 ms
root@Jasmine:/tmp/pycore.35889/Jasmine.conf# ping 172.16.13.20
PING 172.16.13.20 (172.16.13.20) 56(84) bytes of data.
64 bytes from 172.16.13.20: icmp_seq=1 ttl=61 time=1.05 ms
64 bytes from 172.16.13.20: icmp_seq=2 ttl=61 time=0.771 ms
64 bytes from 172.16.13.20: icmp_seq=3 ttl=61 time=0.647 ms
64 bytes from 172.16.13.20: icmp_seq=4 ttl=61 time=0.847 ms
^C
--- 172.16.13.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3045ms
rtt min/avg/max/mdev = 0.647/0.829/1.051/0.146 ms
root@Jasmine:/tmp/pycore.35889/Jasmine.conf# █

```

B->(c,d)

```

root@Ariel:/tmp/pycore.35889/Ariel.conf# ping 172.16.13.20
PING 172.16.13.20 (172.16.13.20) 56(84) bytes of data.
64 bytes from 172.16.13.20: icmp_seq=1 ttl=62 time=0.596 ms
64 bytes from 172.16.13.20: icmp_seq=2 ttl=62 time=2.13 ms
64 bytes from 172.16.13.20: icmp_seq=3 ttl=62 time=0.498 ms
64 bytes from 172.16.13.20: icmp_seq=4 ttl=62 time=0.795 ms
^C
--- 172.16.13.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3031ms
rtt min/avg/max/mdev = 0.498/1.005/2.132/0.659 ms
root@Ariel:/tmp/pycore.35889/Ariel.conf# █

```

C->(d)

Figs. 18 - Certificação de conexão entre departamentos.

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

```

vcmd
root@Bela:/tmp/pycore.37477/Bela.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=63 time=0.788 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=63 time=0.507 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=63 time=0.571 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=63 time=0.265 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=63 time=0.110 ms
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4069ms
rtt min/avg/max/mdev = 0.110/0.448/0.788/0.237 ms
root@Bela:/tmp/pycore.37477/Bela.conf# █

```

Fig. 19 - Certificação da conectividade entre Risp e Bela

- 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`).

```
root@Bela:/tmp/pycore.44465/Bela.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 172.16.6.1 0.0.0.0 UG 0 0 0 eth0
172.16.6.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@Bela:/tmp/pycore.44465/Bela.conf#
```

```
root@Ra:/tmp/pycore.44465/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 eth3
172.16.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
172.16.1.0 172.16.0.2 255.255.255.0 UG 0 0 0 eth0
172.16.2.0 172.16.3.1 255.255.255.0 UG 0 0 0 eth1
172.16.3.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
172.16.6.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
172.16.7.0 172.16.0.2 255.255.255.0 UG 0 0 0 eth0
172.16.10.0 172.16.0.2 255.255.255.0 UG 0 0 0 eth0
172.16.13.0 172.16.3.1 255.255.255.0 UG 0 0 0 eth1
root@Ra:/tmp/pycore.44465/Ra.conf#
```

Fig. 20 - Tabelas de encaminhamento.

R: A coluna “destination” refere-se aos endereços IP destino, havendo uma porta de saída (“gateway”) associada a cada endereço de IP da tabela. A coluna “Genmask” refere-se à máscara que está a ser utilizada, neste caso, utilizando 24 bits como máscara de rede. As “Flags” apresentam informação adicional, apresentado ‘U’ quando o Gateway não está definido e ‘UG’ quando o Gateway está definido. A coluna “Iface” refere-se à interface de saída da máquina local.

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).

```
root@Ra:/tmp/pycore.44465/Ra.conf# ps -ax
PID TTY STAT TIME COMMAND
1 ? S 0:00 vncnode -v -c /tmp/pycore.44465/Ra -l /tmp/pycore.44465/Ra.log -p /tmp/
69 ? Ss 0:00 /usr/local/sbin/zebra -d
75 ? Ss 0:00 /usr/local/sbin/ospf6d -d
79 ? Ss 0:00 /usr/local/sbin/ospf6d -d
87 pts/2 Ss 0:00 /bin/bash
95 pts/2 R+ 0:00 ps -ax
root@Ra:/tmp/pycore.44465/Ra.conf#
```

Fig. 21 - Processos em execução no router A.

**R:** Está a ser usado encaminhamento dinâmico. Como podemos ver na lista de processos, existe um protocolo OSPF6 a ser executado, tratando-se de um protocolo de encaminhamento dinâmico.

**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.

```
root@Sa:/tmp/pycore.44465/Sa.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0          172.16.6.1     0.0.0.0         UG        0 0        0 eth2
172.16.6.0       0.0.0.0        255.255.255.0   U        0 0        0 eth2
root@Sa:/tmp/pycore.44465/Sa.conf# route delete default
root@Sa:/tmp/pycore.44465/Sa.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt Iface
172.16.6.0       0.0.0.0        255.255.255.0   U        0 0        0 eth2
root@Sa:/tmp/pycore.44465/Sa.conf# S
```

Fig. 22 - Alterações verificadas após remoção da rota default.

```
root@Sa:/tmp/pycore.44465/Sa.conf# ping 172.16.7.20
ping: connect: Network is unreachable
root@Sa:/tmp/pycore.44465/Sa.conf#
```

Fig. 23 - Teste de ping.

**R:** Ao retirar a rota por defeito da tabela de encaminhamento do SA, o servidor apenas conseguirá saber encaminhar pacotes cujo endereço destino é 172.16.6.0, ou seja, para a sua rede local. Com esta alteração o servidor não será capaz de comunicar com outros departamentos, como exemplificamos ao tentar estabelecer uma comunicação com um pc (Jasmine) do departamento B, na figura 23.

**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.

```
root@Sa:/tmp/pycore.41251/Sa.conf# route add 10.0.0.1 gw 172.16.6.1
root@Sa:/tmp/pycore.41251/Sa.conf# route add -net 172.16.7.0 netmask 255.255.255.0 gw 172.16.6.1
root@Sa:/tmp/pycore.41251/Sa.conf# route add -net 172.16.13.0 netmask 255.255.255.0 gw 172.16.6.1
root@Sa:/tmp/pycore.41251/Sa.conf# route add -net 172.16.10.0 netmask 255.255.255.0 gw 172.16.6.1
root@Sa:/tmp/pycore.41251/Sa.conf#
```

Fig. 24 - Comandos usados.

**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.



```

root@Sa:/tmp/pycore.41251/Sa.conf# route add 10.0.0.1 gw 172.16.6.1
root@Sa:/tmp/pycore.41251/Sa.conf# route add -net 172.16.7.0 netmask 255.255.255.0 gw 172.16.6.1
root@Sa:/tmp/pycore.41251/Sa.conf# route add -net 172.16.13.0 netmask 255.255.255.0 gw 172.16.6.1
root@Sa:/tmp/pycore.41251/Sa.conf# route add -net 172.16.10.0 netmask 255.255.255.0 gw 172.16.6.1
root@Sa:/tmp/pycore.41251/Sa.conf# ping 172.16.7.20
PING 172.16.7.20 (172.16.7.20) 56(84) bytes of data.
64 bytes from 172.16.7.20: icmp_seq=1 ttl=62 time=1.06 ms
64 bytes from 172.16.7.20: icmp_seq=2 ttl=62 time=0.520 ms
64 bytes from 172.16.7.20: icmp_seq=3 ttl=62 time=0.512 ms
64 bytes from 172.16.7.20: icmp_seq=4 ttl=62 time=0.625 ms
64 bytes from 172.16.7.20: icmp_seq=5 ttl=62 time=0.230 ms
^C
--- 172.16.7.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4066ms
rtt min/avg/max/mdev = 0.230/0.589/1.059/0.268 ms
root@Sa:/tmp/pycore.41251/Sa.conf# ping 172.16.13.20
PING 172.16.13.20 (172.16.13.20) 56(84) bytes of data.
64 bytes from 172.16.13.20: icmp_seq=1 ttl=62 time=0.890 ms
64 bytes from 172.16.13.20: icmp_seq=2 ttl=62 time=0.514 ms
64 bytes from 172.16.13.20: icmp_seq=3 ttl=62 time=0.175 ms
64 bytes from 172.16.13.20: icmp_seq=4 ttl=62 time=0.509 ms
64 bytes from 172.16.13.20: icmp_seq=5 ttl=62 time=0.700 ms
^C
--- 172.16.13.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4096ms
rtt min/avg/max/mdev = 0.175/0.557/0.890/0.237 ms
root@Sa:/tmp/pycore.41251/Sa.conf# ping 172.16.10.20
PING 172.16.10.20 (172.16.10.20) 56(84) bytes of data.
64 bytes from 172.16.10.20: icmp_seq=1 ttl=61 time=1.04 ms
64 bytes from 172.16.10.20: icmp_seq=2 ttl=61 time=0.703 ms
64 bytes from 172.16.10.20: icmp_seq=3 ttl=61 time=0.673 ms
64 bytes from 172.16.10.20: icmp_seq=4 ttl=61 time=0.681 ms
64 bytes from 172.16.10.20: icmp_seq=5 ttl=61 time=0.687 ms
^C
--- 172.16.10.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4051ms
rtt min/avg/max/mdev = 0.673/0.757/1.043/0.143 ms
root@Sa:/tmp/pycore.41251/Sa.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=63 time=0.220 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=63 time=0.498 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=63 time=0.560 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=63 time=0.549 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=63 time=0.514 ms
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4087ms
rtt min/avg/max/mdev = 0.220/0.468/0.560/0.126 ms
root@Sa:/tmp/pycore.41251/Sa.conf# █

```

Fig. 25 - Testes com o comando "ping".

```

root@Sa:/tmp/pycore.41251/Sa.conf# netstat -rn
Kernel IP routing table

```

Destination	Gateway	Genmask	Flags	MSS	Window	irrt	Iface
10.0.0.1	172.16.6.1	255.255.255.255	UGH	0	0	0	eth2
172.16.6.0	0.0.0.0	255.255.255.0	U	0	0	0	eth2
172.16.7.0	172.16.6.1	255.255.255.0	UG	0	0	0	eth2
172.16.10.0	172.16.6.1	255.255.255.0	UG	0	0	0	eth2
172.16.13.0	172.16.6.1	255.255.255.0	UG	0	0	0	eth2

```

root@Sa:/tmp/pycore.41251/Sa.conf# █

```

Fig. 26 - Nova tabela de encaminhamento.

### - 3) Definição de Sub-redes

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 192.168.44.128/25, em que 44 é o decimal correspondendo ao seu número de grupo (PL44). 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.

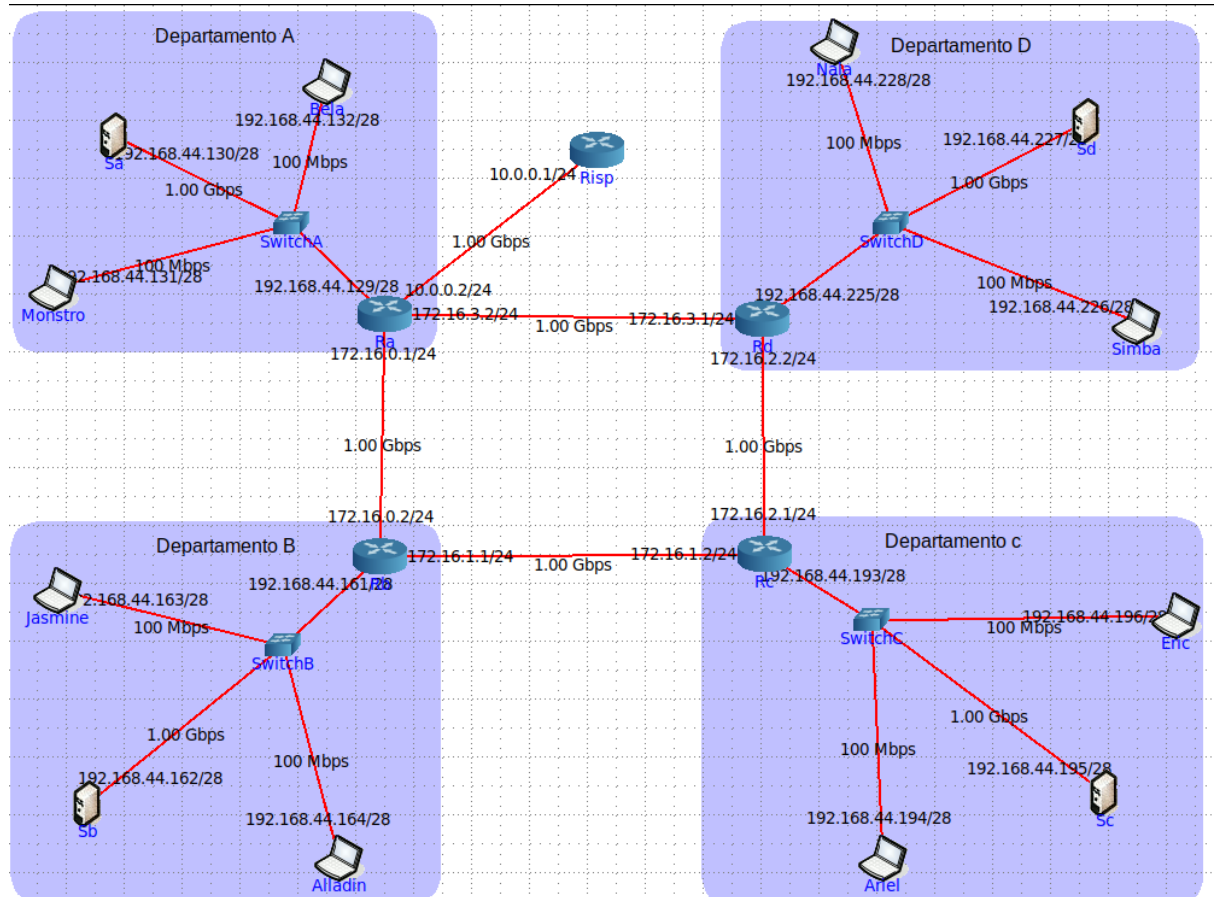


Fig. 27 - Novo esquema de endereçamento.

**R:** Existem 4 subredes, logo precisaríamos no mínimo de 2 bits para as identificar, uma vez que todos os endereços de subrede são usáveis. Mas como o número de departamentos virá a aumentar, decidimos atribuir mais um bit para a distinção de subredes, permitindo a futura adição de mais 4 departamentos (subredes). Assim, ficamos com 3 bits para a identificação de subredes (permitindo 8 subredes diferentes), restando 4 bits para a identificação de hosts em cada subrede (permitindo  $2^4 - 2 = 14$  hosts diferentes em cada subrede).

**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:** Foi usada a máscara /28 = 255.255.255.240. Pode interligar (2 elevado ao número de bits que restam para identificar os hosts, menos 2)  $2^4 - 2 = 14$  hosts IP em cada departamento. Visto que são usados 3 bits para definir uma subrede, há espaço para 8 diferentes prefixos de subrede, uma vez que não existem endereços de subrede reservados e como estão a ser utilizados 4 endereços de subrede, restam 4 disponíveis para uso futuro.

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

```
root@Sa:/tmp/pycore.35889/Sa.conf# ping 192.168.44.163
PING 192.168.44.163 (192.168.44.163) 56(84) bytes of data.
64 bytes from 192.168.44.163: icmp_seq=1 ttl=62 time=0.637 ms
64 bytes from 192.168.44.163: icmp_seq=2 ttl=62 time=0.478 ms
64 bytes from 192.168.44.163: icmp_seq=3 ttl=62 time=0.746 ms
64 bytes from 192.168.44.163: icmp_seq=4 ttl=62 time=0.558 ms
^C
--- 192.168.44.163 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.478/0.604/0.746/0.099 ms
root@Sa:/tmp/pycore.35889/Sa.conf# ping 192.168.44.194
PING 192.168.44.194 (192.168.44.194) 56(84) bytes of data.
64 bytes from 192.168.44.194: icmp_seq=1 ttl=61 time=0.942 ms
64 bytes from 192.168.44.194: icmp_seq=2 ttl=61 time=0.364 ms
64 bytes from 192.168.44.194: icmp_seq=3 ttl=61 time=0.679 ms
64 bytes from 192.168.44.194: icmp_seq=4 ttl=61 time=0.825 ms
^C
--- 192.168.44.194 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3037ms
rtt min/avg/max/mdev = 0.364/0.702/0.942/0.216 ms
root@Sa:/tmp/pycore.35889/Sa.conf# ping 192.168.44.228
PING 192.168.44.228 (192.168.44.228) 56(84) bytes of data.
64 bytes from 192.168.44.228: icmp_seq=1 ttl=62 time=0.896 ms
64 bytes from 192.168.44.228: icmp_seq=2 ttl=62 time=0.679 ms
64 bytes from 192.168.44.228: icmp_seq=3 ttl=62 time=0.648 ms
64 bytes from 192.168.44.228: icmp_seq=4 ttl=62 time=0.725 ms
^C
--- 192.168.44.228 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3073ms
rtt min/avg/max/mdev = 0.648/0.737/0.896/0.095 ms
root@Sa:/tmp/pycore.35889/Sa.conf#
```

```

root@Jasmine:/tmp/pycore.35889/Jasmine.conf# ping 192.168.44.194
PING 192.168.44.194 (192.168.44.194) 56(84) bytes of data.
64 bytes from 192.168.44.194: icmp_seq=1 ttl=62 time=1.07 ms
64 bytes from 192.168.44.194: icmp_seq=2 ttl=62 time=1.11 ms
64 bytes from 192.168.44.194: icmp_seq=3 ttl=62 time=0.476 ms
64 bytes from 192.168.44.194: icmp_seq=4 ttl=62 time=0.738 ms
64 bytes from 192.168.44.194: icmp_seq=5 ttl=62 time=0.641 ms
64 bytes from 192.168.44.194: icmp_seq=6 ttl=62 time=0.756 ms
^C
--- 192.168.44.194 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5071ms
rtt min/avg/max/mdev = 0.476/0.797/1.109/0.224 ms
root@Jasmine:/tmp/pycore.35889/Jasmine.conf# ping 192.168.44.228
PING 192.168.44.228 (192.168.44.228) 56(84) bytes of data.
64 bytes from 192.168.44.228: icmp_seq=1 ttl=61 time=0.767 ms
64 bytes from 192.168.44.228: icmp_seq=2 ttl=61 time=0.896 ms
64 bytes from 192.168.44.228: icmp_seq=3 ttl=61 time=0.650 ms
64 bytes from 192.168.44.228: icmp_seq=4 ttl=61 time=0.678 ms
64 bytes from 192.168.44.228: icmp_seq=5 ttl=61 time=0.653 ms
64 bytes from 192.168.44.228: icmp_seq=6 ttl=61 time=0.628 ms
^C64 bytes from 192.168.44.228: icmp_seq=7 ttl=61 time=0.599 ms
64 bytes from 192.168.44.228: icmp_seq=8 ttl=61 time=0.770 ms
^C
--- 192.168.44.228 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7142ms
rtt min/avg/max/mdev = 0.599/0.705/0.896/0.092 ms
root@Jasmine:/tmp/pycore.35889/Jasmine.conf# █

root@Ariel:/tmp/pycore.35889/Ariel.conf# ping 192.168.44.226
PING 192.168.44.226 (192.168.44.226) 56(84) bytes of data.
64 bytes from 192.168.44.226: icmp_seq=1 ttl=62 time=0.801 ms
64 bytes from 192.168.44.226: icmp_seq=2 ttl=62 time=0.475 ms
64 bytes from 192.168.44.226: icmp_seq=3 ttl=62 time=0.672 ms
64 bytes from 192.168.44.226: icmp_seq=4 ttl=62 time=0.611 ms
^C
--- 192.168.44.226 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3060ms
rtt min/avg/max/mdev = 0.475/0.639/0.801/0.117 ms
root@Ariel:/tmp/pycore.35889/Ariel.conf# █

```

Figs. 28 - Confirmação da conectividade de rede.

## Conclusão

Com este trabalho prático conseguimos aplicar os conhecimentos obtidos nas aulas teóricas em situações “reais” e perceber melhor conceitos como o IP, fragmentação, encaminhamento e subnetting. A utilização do software Core e Wireshark também nos ajudaram bastante na realização deste trabalho e serão ferramentas que nos podem vir a ser úteis na nossa vida profissional.