

Protocolo IPv4

Filipa Rebelo and Joana Oliveira

Universidade do Minho, Departamento de Informática, 4710-057 Braga, Portugal
email: {a90234,a87956}@alunos.uminho.pt

1 Parte I

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.

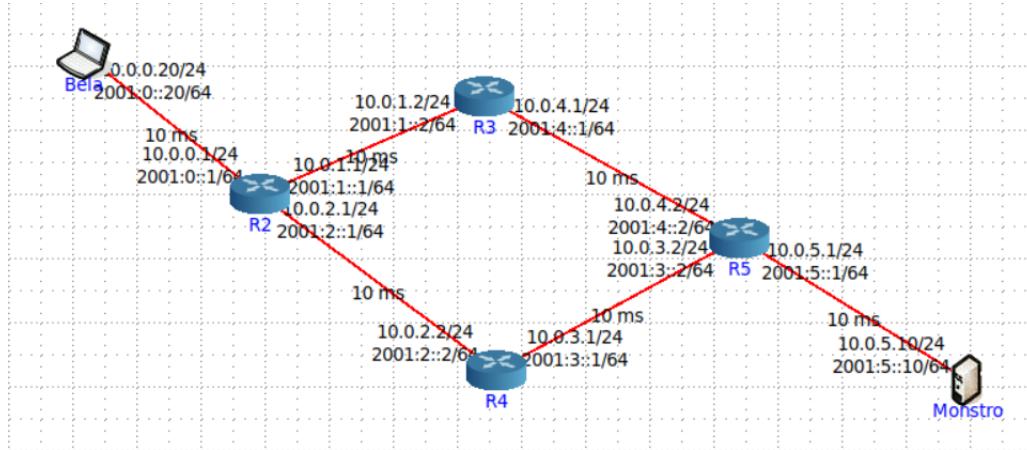


Fig. 1. Topologia Core

Alínea 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.

```
root@Bela:/tmp/pycore.45085/Bela.conf# traceroute -I 10.0.5.10/24
10.0.5.10/24: Name or service not known
Cannot handle "host" cmdline arg `10.0.5.10/24' on position 1 (argc 2)
root@Bela:/tmp/pycore.45085/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)  20.275 ms  20.251 ms  20.248 ms
 2  10.0.1.2 (10.0.1.2)  40.710 ms  40.709 ms  40.707 ms
 3  10.0.3.2 (10.0.3.2)  61.027 ms  61.025 ms  61.020 ms
 4  10.0.5.10 (10.0.5.10)  81.275 ms  81.275 ms  81.270 ms
root@Bela:/tmp/pycore.45085/Bela.conf# █
```

Fig. 2. Traceroute

Alínea 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.

Foi possível observar que o Bela envia diversos pacotes. Estes tinham inicialmente TTL = 1, tendo sido este valor incrementado, até se conseguir estabelecer uma conexão com sucesso.

Os pacotes com TTL = 6 foram descartados por R2, em seguida os pacotes com TTL = 7 foram descartados por R3 e com TTL = 8 foram descartados por R5. Para cada pacote descartado foi recebida a resposta *Time to live exceeded* proveniente do router que o descartou.

A partir de TTL = 9, os pacotes deixam de ser descartados, conseguindo finalmente chegar ao seu destino e passando assim a receber como resposta *Echo (ping) reply*.

No.	Time	Source	Destination	Protocol	Length	Info
313	471.441164196	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=10/2500, ttl=4 (reply in 338)
314	471.441164947	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=11/2810, ttl=4 (reply in 339)
315	471.441165734	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=12/3072, ttl=4 (reply in 340)
316	471.441166508	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=13/3328, ttl=5 (reply in 341)
317	471.441167469	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=14/3500, ttl=5 (reply in 342)
318	471.441168230	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=15/3749, ttl=5 (reply in 343)
319	471.441169164	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=16/4009, ttl=6 (reply in 344)
320	471.461652524	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
321	471.461645441	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
322	471.461647440	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
323	471.466791868	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=17/4352, ttl=6 (reply in 345)
324	471.466817592	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=18/4608, ttl=6 (reply in 346)
325	471.466831354	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=19/4864, ttl=7 (reply in 347)
326	471.482120101	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
327	471.482127648	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
328	471.482129427	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
329	471.485990135	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=20/5120, ttl=7 (reply in 348)
330	471.486024680	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=21/5376, ttl=7 (reply in 349)
331	471.486048147	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=22/5632, ttl=7 (reply in 350)
332	471.504240362	10.0.3.2	10.0.5.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
333	471.504247647	10.0.3.2	10.0.5.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
334	471.504247819	10.0.3.2	10.0.5.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
335	471.504189451	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=23/5888, ttl=8 (reply in 351)
336	471.504211726	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=24/6144, ttl=8 (reply in 352)
337	471.504224884	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x002e, seq=25/6400, ttl=9 (reply in 353)
338	471.542921267	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x002e, seq=10/2500, ttl=6 (request in 31)
339	471.542916784	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x002e, seq=11/2810, ttl=6 (request in 31)
340	471.542917506	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x002e, seq=12/3072, ttl=61 (request in 31)

Fig. 3. Wireshark

Alínea 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.

O valor mínimo de TTL deverá ser 9, tal como verificado na alínea anterior. É a partir deste valor que os pacotes deixam de ser descartados e que o cliente recebe uma resposta do servidor.

Alínea 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.

Tendo em conta os pacotes que conseguiram chegar ao destino e os valores da Figura 2:

$$\frac{81.275+81.275+81.270}{3} = 81,273ms$$

Alínea 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?

Não, uma vez que o número de saltos a ser feito em cada sentido pode variar de acordo com o número de ligações existentes na rota escolhida. Além disso, mesmo que este número seja igual em ambos os sentidos, há ainda a possibilidade de existência de congestionamentos nas travessias que contribuem para o aumento deste atraso.

1.2 Exercício 2

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

Através da imagem abaixo verifica-se que o IP Source é 172.26.78.94.

```

> Frame 7: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlo1, id 0
> Ethernet II, Src: CloudNet_01:0f:77 (48:5f:99:01:0f:77), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
  Internet Protocol Version 4, Src: 172.26.78.94, 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: 0x9fa7 (40871)
    > 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)
    Header checksum: 0x5429 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.78.94
    Destination: 193.136.9.240
  > Internet Control Message Protocol

```

Fig. 4. Informação da primeira mensagem ICMP capturada

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

O valor do campo protocolo é 1, que identifica o ICMP, ou seja, o protocolo utilizado para enviar a mensagem.

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

Através da figura anterior e consultando o campo *Header Length* é possível perceber que o cabeçalho tem 20 bytes.

O cálculo do *payload* resulta da subtração do valor do *Header Length* ao valor que se encontra no *Total Length*. Assim sendo, o *payload* é então de $60 - 20 = 40$ bytes.

Alínea d) O datagrama IP foi fragmentado? Justifique

Como se pode observar na Figura 4, o campo *Fragment offset* tem o valor 0, o que poderá significar que este datagrama não foi fragmentado ou então que este é o primeiro fragmento a ser enviado.

No entanto, e como o campo *More fragments* se encontra como *Not set*, é possível concluir que não existem mais fragmentos, pelo que o datagrama não foi fragmentado.

Alínea 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.

Através da Figura 5 é possível verificar que os campos do cabeçalho IP que variam de pacote para pacote são o *Time to Live*, *Identification* e o *Header Checksum*.

No.	Time	Source	Destination	Protocol	Length Info
1 0.0000000000	172.26.78.94	185.199.9.154	TCP	66 66398 -> 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 Tsvval=3445528525 ...	
3 0.131133933	172.26.78.94	193.137.16.145	DNS	86 Standard query 0xf6a3 A marco.uminho.pt OPT	
4 0.131476648	172.26.78.94	193.137.16.145	DNS	86 Standard query 0xce3a AAAA marco.uminho.pt OPT	
7 0.134571348	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=1/256, ttl=1 (no response..	
8 0.134669522	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=2/512, ttl=1 (no response..	
9 0.134635464	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=3/768, ttl=1 (no response..	
10 0.134663939	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=4/1024, ttl=2 (no response..	
11 0.134670574	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=5/1280, ttl=2 (no response..	
12 0.134715156	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=6/1536, ttl=2 (no response..	
13 0.134739306	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=7/1792, ttl=3 (no response..	
14 0.134761729	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=8/2048, ttl=3 (no response..	
15 0.134787381	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=9/2304, ttl=3 (no response..	
16 0.134815737	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=10/2560, ttl=4 (reply in ..	
17 0.134838849	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=11/2816, ttl=4 (reply in ..	
18 0.134865837	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=12/3072, ttl=4 (reply in ..	
19 0.134897429	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=13/3328, ttl=5 (reply in ..	
20 0.134921292	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=14/3584, ttl=5 (reply in ..	
21 0.134949467	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=15/3840, ttl=5 (reply in ..	
22 0.134978557	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=16/4096, ttl=6 (reply in ..	
24 0.135929169	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=17/4352, ttl=6 (reply in ..	
27 0.137333152	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=18/4608, ttl=7 (reply in ..	
28 0.137333261	172.26.78.94	193.136.9.240	ICMP	74 Echo (ping) request id=0x0005, seq=19/4864, ttl=7 (reply in ..	
43 0.140794583	172.26.78.94	193.137.16.145	DNS	98 Standard query 0x9eaf PTR 284.254.26.172.in-addr.arpa OPT	
47 0.142845905	172.26.78.94	193.137.16.145	DNS	87 Standard query 0x9eaf PTR 284.254.26.172.in-addr.arpa	
49 0.147555108	172.26.78.94	193.137.16.145	DNS	94 Standard query 0xc0d3 PTR 1.2.16.172.in-addr.arpa OPT	
51 0.150237136	172.26.78.94	193.137.16.75	DNS	94 Standard query 0xc0d3 PTR 1.2.16.172.in-addr.arpa OPT	
53 0.153064561	172.26.78.94	193.137.16.65	DNS	94 Standard query 0xc0d3 PTR 1.2.16.172.in-addr.arpa OPT	

Fig. 5. Tráfego ICMP

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

É possível observar que os valores contidos no campo relativo à identificação vão sendo sucessivamente incrementados a cada nova mensagem e que todos eles começam por "0x9f".

O campo TTL incrementa 1 valor por cada 3 mensagens, até que seja estabelecida ligação com o servidor.

Alínea 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?

O valor do TTL varia entre 253 e 255. Isto deve-se ao funcionamento do *traceroute*, uma vez que cada pacote poderá ser rejeitado ao fim de um número de passos diferente, o que significa que cada mensagem de retorno irá também dar números de saltos diferentes.

No.	Time	Source	Destination	Protocol	Length Info
261	20.223961144	172.26.78.94	140.62.114.26	TCP	66 34348 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 Tsval=1029230069 ...
508	34.866245610	172.26.78.94	140.62.114.26	TLSv1.2	95 [TCP Previous segment not captured] Application Data
2	0.015403920	185.199.111.154	172.26.78.94	TCP	66 [TCP ACKed unseen segment] 443 → 60308 [ACK] Seq=1 Ack=2 Win=...
5	0.133797716	193.137.16.145	172.26.78.94	DNS	140 Standard query response 0xce3a AAAA marco.uminho.pt SOA dns.u...
6	0.134026639	193.137.16.145	172.26.78.94	DNS	102 Standard query response 0xf6a3 A marco.uminho.pt A 193.136.9...
23	0.139508850	172.16.2.1	172.26.78.94	ICMP	78 Time-to-live exceeded (Time to live exceeded in transit)
25	0.139508854	172.16.2.1	172.26.78.94	ICMP	78 Time-to-live exceeded (Time to live exceeded in transit)
26	0.139508852	172.16.2.1	172.26.78.94	ICMP	78 Time-to-live exceeded (Time to live exceeded in transit)
29	0.139508856	193.136.9.240	172.26.78.94	ICMP	74 Echo (ping) reply id=0x0005, seq=10/2869, ttl=61 (request ...)
30	0.139508831	193.136.9.240	172.26.78.94	ICMP	74 Echo (ping) reply id=0x0005, seq=11/2816, ttl=61 (request ...)
31	0.139508939	193.136.9.240	172.26.78.94	ICMP	74 Echo (ping) reply id=0x0005, seq=12/3672, ttl=61 (request ...)
32	0.139501624	193.136.9.240	172.26.78.94	ICMP	74 Echo (ping) reply id=0x0005, seq=13/3328, ttl=61 (request ...)
33	0.139501123	193.136.9.240	172.26.78.94	ICMP	74 Echo (ping) reply id=0x0005, seq=14/3584, ttl=61 (request ...)
34	0.139578388	193.136.9.240	172.26.78.94	ICMP	74 Echo (ping) reply id=0x0005, seq=15/3840, ttl=61 (request ...)
35	0.139578517	193.136.9.240	172.26.78.94	ICMP	74 Echo (ping) reply id=0x0005, seq=16/4096, ttl=61 (request ...)
36	0.139578607	172.16.15.252	172.26.78.94	ICMP	78 Time-to-live exceeded (Time to live exceeded in transit)
37	0.139578702	172.16.15.252	172.26.78.94	ICMP	78 Time-to-live exceeded (Time to live exceeded in transit)
38	0.139578805	172.16.15.252	172.26.78.94	ICMP	78 Time-to-live exceeded (Time to live exceeded in transit)
39	0.139846689	193.136.9.240	172.26.78.94	ICMP	74 Echo (ping) reply id=0x0005, seq=17/4352, ttl=61 (request ...)
40	0.139846693	193.136.9.240	172.26.78.94	ICMP	78 Time-to-live exceeded (Time to live exceeded in transit)
41	0.139846373	172.26.254.254	172.26.78.94	ICMP	78 Time-to-live exceeded (Time to live exceeded in transit)
42	0.139886637	172.26.254.254	172.26.78.94	ICMP	78 Time-to-live exceeded (Time to live exceeded in transit)
44	8.141974992	193.136.9.240	172.26.78.94	ICMP	74 Echo (ping) reply id=0x0005, seq=18/4608, ttl=61 (request ...)
45	0.142464477	193.136.9.240	172.26.78.94	ICMP	74 Echo (ping) reply id=0x0005, seq=19/4864, ttl=61 (request ...)
46	0.142786606	193.137.16.145	172.26.78.94	DNS	152 Standard query response 0x9eaf No such name PTR 254.254.26.17...
48	0.145283267	193.137.16.145	172.26.78.94	DNS	141 Standard query response 0x9eaf No such name PTR 254.254.26.17...
50	0.149994998	193.137.16.145	172.26.78.94	DNS	94 Standard query response 0xc0d3 Refused PTR 1.2.16.172.in-addr...
52	0.152793202	193.137.16.75	172.26.78.94	DNS	94 Standard query response 0xd0d3 Refused PTR 1.2.16.172.in-addr...

Fig. 6. Tráfego ICMP

1.3 Exercício 3

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

Foi necessário fragmentar o pacote inicial uma vez que este possuía 4037 bytes e apenas é possível transmitir 1500 bytes de cada vez.

No.	Time	Source	Destination	Protocol	Length Info
263	24.127851185	172.26.78.94	193.137.16.145	DNS	86 Standard query 0x39d0 A marco.uminho.pt OPT
264	24.128244886	172.26.78.94	193.137.16.145	DNS	86 Standard query 0x3e69 AAAA marco.uminho.pt OPT
265	24.131356735	193.137.16.145	172.26.78.94	DNS	102 Standard query response 0x39d0 A marco.uminho.pt A 193.136.9...
266	24.131357373	193.137.16.145	172.26.78.94	DNS	149 Standard query response 0x3e69 AAAA marco.uminho.pt SOA dns.u...
267	24.132601589	172.26.78.94	193.136.9.240	ICMP	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5d7] [Reasse...
268	24.132601885	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5d7] [Rea...
269	24.132601541	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=3/240, ttl=1 (no respons...
270	24.132601154	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5d8] [Reasse...
271	24.132612058	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5d8] [Rea...
272	24.132645859	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=2/240, ttl=1 (no respons...
273	24.132667087	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5d9] [Reasse...
274	24.132676722	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5d9] [Rea...
275	24.132703654	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=3/768, ttl=1 (no respons...
276	24.132743261	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5da] [Reasse...
277	24.132760061	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5da] [Rea...
278	24.132778149	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=4/1924, ttl=2 (no respons...
279	24.132796082	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5db] [Reasse...
280	24.132811624	172.26.78.94	193.136.9.240	ICMP	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5db] [Rea...
281	24.132811647	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=5/1924, ttl=1 (no respons...
282	24.132876882	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5dc] [Reasse...
283	24.132891679	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5dc] [Rea...
284	24.132986933	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=6/1536, ttl=2 (no respons...
285	24.132925819	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5dd] [Reasse...
286	24.132946687	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5dd] [Rea...
287	24.132963761	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=7/1792, ttl=3 (no respons...
288	24.132999661	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5de] [Reasse...
289	24.133028084	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5de] [Rea...
290	24.133038235	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=8/2048, ttl=3 (no respons...
291	24.133052040	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5df] [Reasse...
292	24.133071127	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5df] [Rea...
293	24.133093987	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=9/2364, ttl=3 (no respons...
294	24.133107047	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5e0] [Reasse...
295	24.133153860	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=10/2560, ttl=4 (reply in ...
296	24.133169952	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=10/2560, ttl=4 (reply in ...
297	24.133383818	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5e1] [Reasse...
298	24.133399016	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5e1] [Rea...
299	24.133221777	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=11/2816, ttl=4 (reply in ...
300	24.133258433	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5e2] [Reasse...
301	24.133279289	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5e2] [Rea...
302	24.133299433	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=12/3672, ttl=4 (reply in ...
303	24.133315966	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5e3] [Reasse...
304	24.133331077	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a5e3] [Rea...
305	24.133354260	172.26.78.94	193.136.9.240	ICMP	1514 Echo (ping) request id=8x0006, seq=13/3328, ttl=5 (reply in ...
306	24.133389901	172.26.78.94	193.136.9.240	IPv4	1514 Fragmented IP protocol [proto=ICMP 1, off=0, ID=a5e4] [Reasse...

Fig. 7. Fragmentação do pacote inicial

Alínea 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?

Através da observação da Figura 8 verifica-se que o valor da flag *More Fragments* tem o seu bit a 1, o que indica que existirão mais fragmentos associados ao datagrama original.

O valor do campo *Fragment offset* encontra-se a 0. Isto, juntamente com a informação anteriormente observada, permite concluir que este se trata do primeiro fragmento do pacote.

O tamanho deste datagrama encontra-se no campo *Total Length*, sendo este de 1500 bytes.

```

▶ Frame 267: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface wlo1, id 0
▶ Ethernet II, Src: CloudNet_01:0f:77 (48:5f:99:01:0f:77), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.78.94, 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: 0xa5d7 (42455)
    ▶ 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: 0x2859 [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.78.94
  Destination: 193.136.9.240
  Reassembled IPv4 in frame: 269
▶ Data (1480 bytes)

```

Fig. 8. Primeiro segmento do datagrama IP

Alínea 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?

Através da observação da Figura 9 verificamos que o campo *Fragment offset* tem valor 1488. Isto permite-nos concluir que se não se trata do primeiro fragmento uma vez que, nesse caso, o seu valor seria de 0.

Dado que a flag *More fragments* tem o seu bit a 1, é possível concluir que existem mais fragmentos do datagrama original.

```

▶ Frame 268: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface wlo1, id 0
▶ Ethernet II, Src: CloudNet_01:0f:77 (48:5f:99:01:0f:77), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.78.94, 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: 0xa5d7 (42455)
    ▶ Flags: 0x20b9, More fragments
      0... .... .... = Reserved bit: Not set
      .0.. .... .... = Don't fragment: Not set
      ..1. .... .... = More fragments: Set
    Fragment offset: 1488
  ▶ Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x27a0 [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.78.94
  Destination: 193.136.9.240
  Reassembled IPv4 in frame: 269
▶ Data (1480 bytes)

```

Fig. 9. Segundo segmento do datagrama IP

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

Foram criados 3 fragmentos a partir do datagrama inicial, tal como é possível observar na parte final da Figura 10. O último fragmento deverá conter a flag *More fragments* como *Not Set*, bem como a sua *Identification* igual aos restantes fragmentos, o que se pode observar. Sendo este o 3ºfragmento e uma vez que cumpre com os requisitos anteriores é possível concluir que, de facto, foram criados 3 fragmentos.

```

▶ Frame 269: 1091 bytes on wire (8728 bits), 1091 bytes captured (8728 bits) on interface wlo1, id 0
▶ Ethernet II, Src: CloudNet_01:0f:77 (48:5f:99:01:0f:77), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.78.94, 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: 1077
    Identification: 0xa5d7 (42455)
  ▼ 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: 0x488e [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.78.94
  Destination: 193.136.9.240
  ▶ [3 IPv4 Fragments (4017 bytes): #267(1480), #268(1480), #269(1057)]
  ▶ Internet Control Message Protocol

```

Fig. 10. Último segmento do datagrama IP

Alínea 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.

Os campos do cabeçalho IP que mudam entre os diferentes fragmentos do mesmo datagrama IP são a flag *More fragments* e o campo *Fragment offset*. Um bit a 0 na flag *More Fragments* permite-nos saber que nos encontramos no último fragmento e que, caso contrário, nos encontramos em qualquer um dos restantes. O campo *Fragment offset* terá o seu valor a 0 no caso de nos encontrarmos no primeiro fragmento e o valor correspondente ao número de bytes já enviados, caso nos encontremos em qualquer outro. Ordenando os fragmentos por ordem crescente de *offset* é possível obter o datagrama ordenado pela ordem em que os dados se encontravam no datagrama original.

Alínea f) Verifique o processo de fragmentação através de um processo de cálculo

Sendo o tamanho total do datagrama igual a 4037 e o tamanho máximo de cada fragmento igual a 1500, sendo que 20 são referentes ao cabeçalho, teríamos:

$$\frac{4037}{1480} = 2,73$$

No entanto, e como é necessário considerar que cada fragmento terá o seu cabeçalho, iriam ser necessários, no mínimo 3 fragmentos, uma vez que:

$$1480 * 2 = 2960$$

Ou seja, com 2 fragmentos apenas conseguimos enviar 2960 bytes, pelo que nos faltam ainda 1077 bytes. Como cada fragmento permite enviar 1480 bytes de informação, então são suficientes 3 fragmentos.

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

O último fragmento será aquele cujo campo *Fragment offset* se encontre diferente de 0, o que significa que já foram enviados anteriormente outros fragmentos, e a flag *More Fragments* se encontre com o bit a 0, indicando que não existem mais fragmentos a ser enviados. Além disso, o campo *Identification* deverá ser igual aos restantes fragmentos do datagrama.

Expressão Lógica: $More\ Fragments == 0 \ \&\& \ Fragment\ offset > 0 \ \&\& \ \forall 0 \leq i < n (Identification[i] == Identification[n])$

2 Parte II

2.1 Exercício 1

Alínea 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.

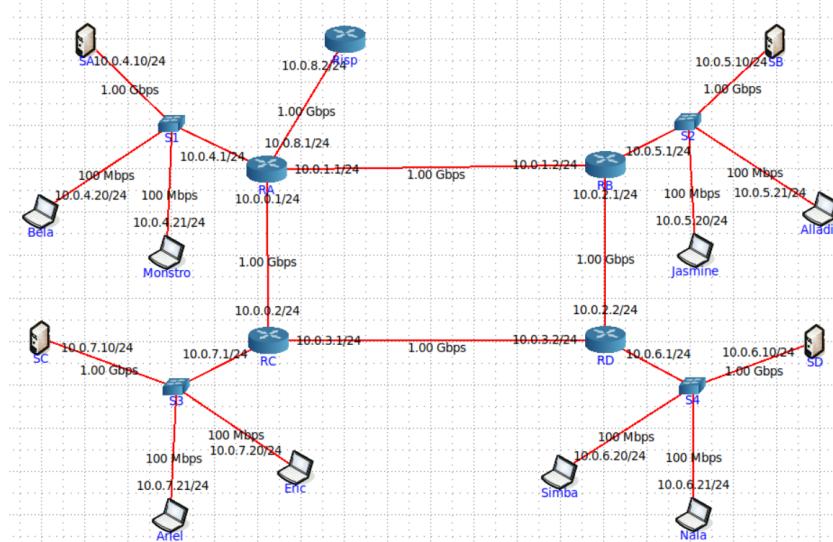


Fig. 11. Topologia Core

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

Tratam-se de endereços privados, uma vez que a gama 10.0.0.0-10.255.255.255/8 faz parte das gamas de endereços privados.

Alínea c) Porque razão não é atribuído um endereço IP aos switches?

Os switches operam uma camada abaixo (camada 2), relativa à ligação de dados, em que são utilizados os endereços MAC, e não os endereços IP, de cada equipamento para fazer a distribuição de pacotes.

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

De forma a verificar a existência de conectividade interna entre cada portátil e o servidor respetivo foi então utilizado o comando *ping* em cada um deles. A imagem abaixo permite confirmar que foi possível estabelecer ligação com cada um dos servidores respetivos.

The figure displays three separate terminal windows, each showing the output of a ping command between two hosts within a department. The first window shows a ping from host 10.0.4.10 to 10.0.4.10. The second window shows a ping from host 10.0.5.10 to 10.0.5.10. The third window shows a ping from host 10.0.6.10 to 10.0.6.10. All pings are successful with low latency.

```

root@Belsx:/tmp/pycore_45391/Belsx.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=0.418 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=64 time=0.352 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=64 time=1.91 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=64 time=0.278 ms

root@Jasmine:/tmp/pycore_45391/Jasmine.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=0.237 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=64 time=0.323 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=64 time=0.151 ms
64 bytes from 10.0.5.10: icmp_seq=4 ttl=64 time=0.194 ms

root@Nebulz:/tmp/pycore_45391/Nebulz.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=0.746 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=64 time=0.294 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=64 time=0.303 ms
64 bytes from 10.0.6.10: icmp_seq=4 ttl=64 time=0.371 ms

```

Fig. 12. Verificação de conectividade interna em todos os departamentos

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

De modo a utilizar o menor número possível de comandos *ping* para verificar a conectividade entre departamentos, apenas foram executados 3 comandos a partir do departamento A, 1 para cada departamento. Daqui é possível concluir que existe conectividade entre A e todos os outros departamentos. De modo a verificar que os restantes departamentos estão também conectados entre si, é necessário analisar a rede. Por exemplo, se quisermos confirmar que existe conectividade de B para D, uma vez que sabemos que existe conectividade de A para B, C e D, sabemos também que de modo a A estabelecer ligação com D teve que percorrer A-B-D ou então A-C-D. Qualquer que tenha sido a travessia é sempre possível estabelecer conexão entre B e D, quer seja diretamente de B para D ou através de B-A-D-C. O mesmo raciocínio pode ser aplicado aos restantes departamentos.

The image shows three separate terminal windows, each titled "vcmd". Each window displays the output of a "ping" command from a host named "Bela" to a specific IP address. The first window shows ping results to 10.0.5.20, the second to 10.0.6.20, and the third to 10.0.7.21. Each ping command sends 56(84) bytes of data and receives three ICMP echo replies with TTL=62 and varying round-trip times (e.g., 0.793 ms, 0.624 ms, 0.524 ms for the first router).

```

root@Bela:/tmp/pycore.45391/Bela.conf# ping 10.0.5.20
PING 10.0.5.20 (10.0.5.20) 56(84) bytes of data.
64 bytes from 10.0.5.20: icmp_seq=1 ttl=62 time=0.793 ms
64 bytes from 10.0.5.20: icmp_seq=2 ttl=62 time=0.624 ms
64 bytes from 10.0.5.20: icmp_seq=3 ttl=62 time=0.524 ms

root@Bela:/tmp/pycore.45391/Bela.conf# ping 10.0.6.20
PING 10.0.6.20 (10.0.6.20) 56(84) bytes of data.
64 bytes from 10.0.6.20: icmp_seq=1 ttl=61 time=4.03 ms
64 bytes from 10.0.6.20: icmp_seq=2 ttl=61 time=0.699 ms
64 bytes from 10.0.6.20: icmp_seq=3 ttl=61 time=0.781 ms

root@Bela:/tmp/pycore.45391/Bela.conf# ping 10.0.7.21
PING 10.0.7.21 (10.0.7.21) 56(84) bytes of data.
64 bytes from 10.0.7.21: icmp_seq=1 ttl=62 time=0.330 ms
64 bytes from 10.0.7.21: icmp_seq=2 ttl=62 time=0.498 ms
64 bytes from 10.0.7.21: icmp_seq=3 ttl=62 time=0.568 ms

```

Fig. 13. Verificação de conectividade IP entre departamentos

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

Tal como na alínea d) foi utilizado o comando *ping* para verificar a existência de conectividade entre o portátil Bela e o router RISP. A imagem seguinte permite confirmar a sua existência.

The image shows a single terminal window titled "vcmd". It displays the output of a "ping" command from a host named "Bela" to the IP address 10.0.8.2. The ping command sends 56(84) bytes of data and receives four ICMP echo replies with TTL=63 and round-trip times ranging from 0.451 ms to 2.33 ms.

```

root@Bela:/tmp/pycore.45391/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=2.33 ms
64 bytes from 10.0.8.2: icmp_seq=2 ttl=63 time=0.461 ms
64 bytes from 10.0.8.2: icmp_seq=3 ttl=63 time=0.451 ms
64 bytes from 10.0.8.2: icmp_seq=4 ttl=63 time=2.07 ms

```

Fig. 14. Verificação de conectividade entre o portátil Bela e o router RISP

2.2 Exercício 2

Para o router RA e o portátil Bela:

Alínea 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`).

Na imagem seguinte é possível observar as tabelas de encaminhamento relativas ao router RA e ao portátil Bela. O campo *Destination* indica o endereço IP da rede de destino, o campo *Gateway* representa o próximo salto a ser dado pelo datagrama para chegar ao seu destino, o campo *Genmask* indica a máscara utilizada, as *Flags* permitem saber se as rotas são válidas (U) e poderão ser utilizadas ou se necessitarão ainda de um salto intermédio, ou seja, um *gateway* (G) e, por fim, o campo *Iface* contém o identificador da interface de saída da máquina local.

```

root@RA:/tmp/pycore.45391/RA.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags MSS Window irtt Iface
0.0.0.0         0.0.0.0       255.255.255.0 U     0 0          0 eth0
10.0.0.0        0.0.0.0       255.255.255.0 U     0 0          0 eth1
10.0.1.0        0.0.0.0       255.255.255.0 U     0 0          0 eth0
10.0.2.0        10.0.1.2      255.255.255.0 UG    0 0          0 eth1
10.0.3.0        10.0.1.2      255.255.255.0 UG    0 0          0 eth0
10.0.4.0        0.0.0.0       255.255.255.0 U     0 0          0 eth2
10.0.5.0        10.0.1.2      255.255.255.0 UG    0 0          0 eth1
10.0.6.0        10.0.0.2      255.255.255.0 UG    0 0          0 eth0
10.0.7.0        10.0.0.2      255.255.255.0 UG    0 0          0 eth0
10.0.8.0        0.0.0.0       255.255.255.0 U     0 0          0 eth3
root@RA:/tmp/pycore.45391/RA.conf# 

root@Bela:/tmp/pycore.45391/Bela.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags MSS Window irtt Iface
0.0.0.0         0.0.0.4.1     0.0.0.0       UG    0 0          0 eth0
10.0.4.0        0.0.0.0       255.255.255.0 U     0 0          0 eth0
root@Bela:/tmp/pycore.45391/Bela.conf# 

```

Fig. 15. Tabelas de encaminhamento do router RA e do portátil Bela

Alínea 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).

Na figura seguinte é possível observar que está a correr um processo `ospfd`, que se trata de um protocolo de encaminhamento dinâmico.

```

root@RA:/tmp/pycore.45391/RA.conf# ps -ax
PID TTY  STAT TIME COMMAND
 1 ? S  0:00 vnodes ~ -c /tmp/pycore.45391/RA -l /tmp/pycore.45
 67 ? Ss  0:00 /usr/local/sbin/zebra -d
 72 ? Ss  0:00 /usr/local/sbin/ospf6d -d
 77 ? Ss  0:00 /usr/local/sbin/ospfd -d
 93 pts/2 Ss  0:00 /bin/bash
100 pts/2 R+   0:00 ps -ax
root@RA:/tmp/pycore.45391/RA.conf# 

root@Bela:/tmp/pycore.45391/Bela.conf# ps -ax
PID TTY  STAT TIME COMMAND
 1 ? S  0:00 vnodes ~ -c /tmp/pycore.45391/Bela -l /tmp/pycore.
41 pts/4 Ss  0:00 /bin/bash
48 pts/4 R+   0:00 ps -ax
root@Bela:/tmp/pycore.45391/Bela.conf# 

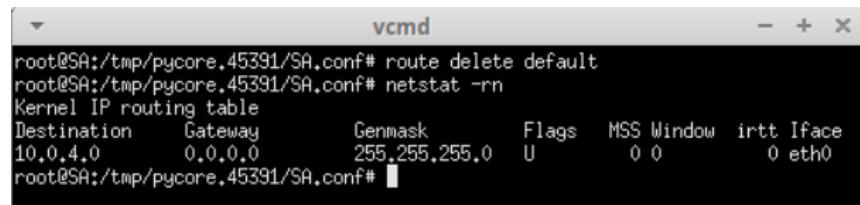
```

Fig. 16. Processos em execução no router RA e no portátil Bela

Alínea 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.

Ao retirar definitivamente a rota por defeito da tabela de encaminhamento de S1, este deixa de conseguir enviar pacotes para fora da sua sub-rede, uma vez que não estão definidas rotas para além dela. Assim sendo não é possível a conexão com outros departamentos.

Tal como é possível observar na Figura 18, o servidor SA tenta estabelecer conexão com um portátil de outro departamento, mas sem sucesso.

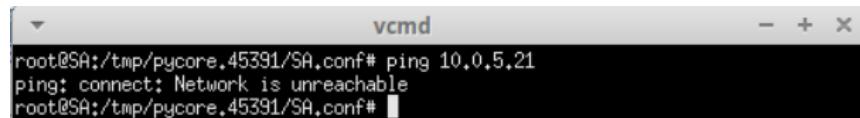


```

root@SA:/tmp/pycore.45391/SA.conf# route delete default
root@SA:/tmp/pycore.45391/SA.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask         Flags   MSS Window irtt Iface
10.0.4.0        0.0.0.0        255.255.255.0    U           0 0          0 eth0
root@SA:/tmp/pycore.45391/SA.conf#

```

Fig. 17. Remoção da rota *default* do servidor SA



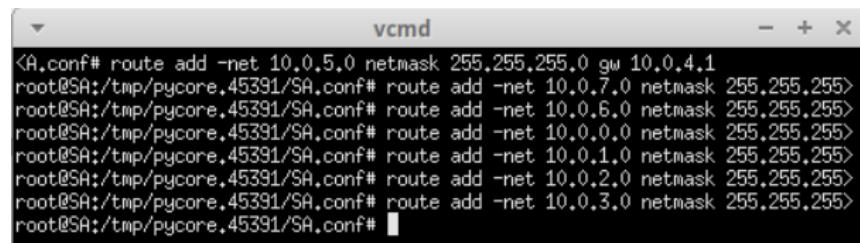
```

root@SA:/tmp/pycore.45391/SA.conf# ping 10.0.5.21
ping: connect: Network is unreachable
root@SA:/tmp/pycore.45391/SA.conf#

```

Fig. 18. Conectividade do servidor SA para o portátil Alladin

Alínea 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

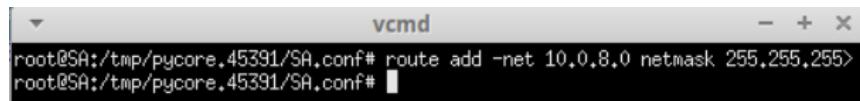


```

<A.conf# route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.4.1
root@SA:/tmp/pycore.45391/SA.conf# route add -net 10.0.7.0 netmask 255.255.255>
root@SA:/tmp/pycore.45391/SA.conf# route add -net 10.0.6.0 netmask 255.255.255>
root@SA:/tmp/pycore.45391/SA.conf# route add -net 10.0.0.0 netmask 255.255.255>
root@SA:/tmp/pycore.45391/SA.conf# route add -net 10.0.1.0 netmask 255.255.255>
root@SA:/tmp/pycore.45391/SA.conf# route add -net 10.0.2.0 netmask 255.255.255>
root@SA:/tmp/pycore.45391/SA.conf# route add -net 10.0.3.0 netmask 255.255.255>
root@SA:/tmp/pycore.45391/SA.conf#

```

Fig. 19. Adição das rotas estáticas



```

root@SA:/tmp/pycore.45391/SA.conf# route add -net 10.0.8.0 netmask 255.255.255>
root@SA:/tmp/pycore.45391/SA.conf#

```

Fig. 20. Adição das rotas estáticas

Alínea 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.

Através da Figura 22 é possível perceber que o servidor já consegue novamente estabelecer conexões para fora da sua sub-rede. Foi utilizado o comando *ping* para um portátil de cada departamento e observamos que não houve perda de pacotes, uma vez que todos eles são enviados e são recebidas as respetivas respostas.

```

root@SA:/tmp/pycore.45391/SA.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
10.0.0.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.1.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.2.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.3.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.4.0        0.0.0.0        255.255.255.0 U          0 0          0 eth0
10.0.5.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.6.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.7.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.8.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
root@SA:/tmp/pycore.45391/SA.conf#

```

Fig. 21. Tabela de encaminhamento de SA

```

root@SA:/tmp/pycore.45391/SA.conf# ping 10.0.5.20
PING 10.0.5.20 (10.0.5.20) 56(84) bytes of data.
64 bytes from 10.0.5.20: icmp_seq=1 ttl=62 time=16.5 ms
64 bytes from 10.0.5.20: icmp_seq=2 ttl=62 time=0.626 ms
64 bytes from 10.0.5.20: icmp_seq=3 ttl=62 time=0.183 ms
64 bytes from 10.0.5.20: icmp_seq=4 ttl=62 time=0.558 ms
^C
--- 10.0.5.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.183/4.466/16.497/6.948 ms
root@SA:/tmp/pycore.45391/SA.conf# ping 10.0.7.20
PING 10.0.7.20 (10.0.7.20) 56(84) bytes of data.
64 bytes from 10.0.7.20: icmp_seq=1 ttl=62 time=2.25 ms
64 bytes from 10.0.7.20: icmp_seq=2 ttl=62 time=0.579 ms
64 bytes from 10.0.7.20: icmp_seq=3 ttl=62 time=1.89 ms
64 bytes from 10.0.7.20: icmp_seq=4 ttl=62 time=0.168 ms
^C
--- 10.0.7.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3017ms
rtt min/avg/max/mdev = 0.168/1.219/2.245/0.867 ms
root@SA:/tmp/pycore.45391/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=61 time=1.27 ms
64 bytes from 10.0.6.21: icmp_seq=2 ttl=61 time=0.743 ms
64 bytes from 10.0.6.21: icmp_seq=3 ttl=61 time=0.690 ms
^C
--- 10.0.6.21 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 0.690/0.900/1.263/0.261 ms
root@SA:/tmp/pycore.45391/SA.conf#

```

Fig. 22. Teste de conectividade entre SA e um portátil de cada departamento

2.3 Exercício 3

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.

Alínea 1) Considere que dispõe apenas do endereço de rede IP 192.168.XXX.128/25, em que XXX é o decimal correspondendo 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.

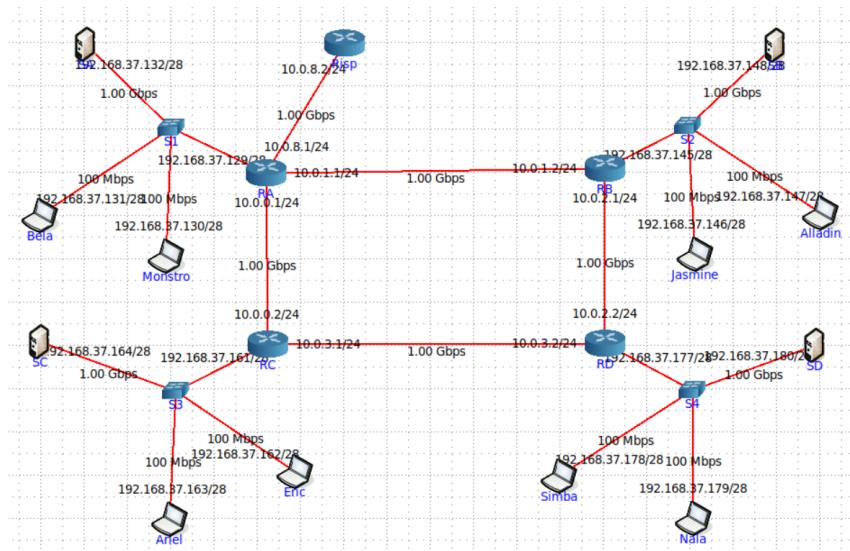


Fig. 23. Topologia Core

Uma vez que o número do nosso grupo é o 37, o endereço de rede IP que nos foi atribuído é o 192.168.37.128/25, cuja representação em binário é 11000000.10101000.00100101.10000000.

Dada a existência de 4 departamentos, são então necessários, pelo menos, 2 bits. No entanto, e como é necessário considerar possíveis expansões futuras em termos de departamentos, 2 bits não chegam pois apenas permitem $2^2 = 4$ combinações.

Assim sendo iremos então reservar 3 bits para a identificação de sub-redes, ficando com uma nova máscara de 28 bits e 4 bits de sobra para hosts.

As sub-redes atribuídas a cada departamento foram então:

Departamento A: 192.168.37.128/28
 Departamento B: 192.168.37.144/28
 Departamento C: 192.168.37.160/28
 Departamento D: 192.168.37.176/28

Alínea 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.

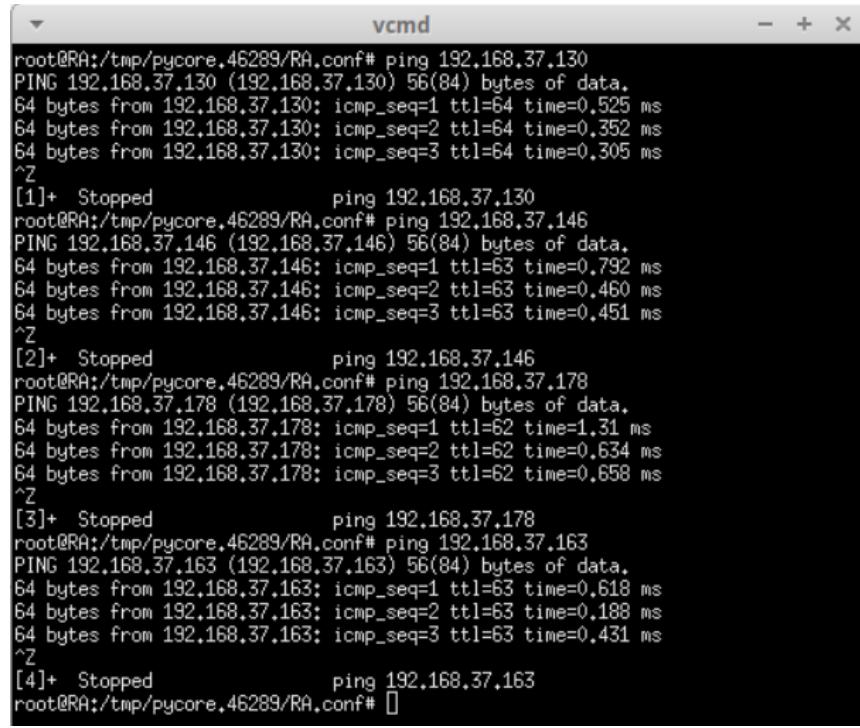
A máscara de rede utilizada foi a /28 (notação CIDR), correspondente a 255.255.255.240. Tal como dito anteriormente, restam 4 bits para a identificação de hosts, o que significa que teremos $2^4 - 2 = 14$ possíveis hosts, uma vez que é necessário remover os endereços correspondentes ao endereço de broadcast e ao endereço da rede.

Uma vez que existem disponíveis $2^3 = 8$ prefixos de sub-rede e já estão a ser utilizados 4 deles, o número de prefixos de sub-rede disponíveis corresponde a 4 ($8-4 = 4$).

Alínea 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.

Para verificar a conectividade IP entre as várias redes utilizamos o router RA, pertencente ao departamento A. Através do comando *ping* tentamos então estabelecer conexão com os portáteis Monstro, Jasmine, Simba e Ariel, que se encontram todos em sub-redes distintas.

A figura seguinte permite confirmar a existência de conectividade entre o departamento A e os restantes departamentos. Para confirmar a conectividade entre os restantes departamentos (por exemplo, entre B e C), pode ser aplicada mesma lógica que foi utilizada na alínea e) do 1º exercício relativo a esta parte.



```
root@RA:/tmp/pycore.46289/RA.conf# ping 192.168.37.130
PING 192.168.37.130 (192.168.37.130) 56(84) bytes of data.
64 bytes from 192.168.37.130: icmp_seq=1 ttl=64 time=0.525 ms
64 bytes from 192.168.37.130: icmp_seq=2 ttl=64 time=0.352 ms
64 bytes from 192.168.37.130: icmp_seq=3 ttl=64 time=0.305 ms
^Z
[1]+  Stopped                  ping 192.168.37.130
root@RA:/tmp/pycore.46289/RA.conf# ping 192.168.37.146
PING 192.168.37.146 (192.168.37.146) 56(84) bytes of data.
64 bytes from 192.168.37.146: icmp_seq=1 ttl=63 time=0.792 ms
64 bytes from 192.168.37.146: icmp_seq=2 ttl=63 time=0.460 ms
64 bytes from 192.168.37.146: icmp_seq=3 ttl=63 time=0.451 ms
^Z
[2]+  Stopped                  ping 192.168.37.146
root@RA:/tmp/pycore.46289/RA.conf# ping 192.168.37.178
PING 192.168.37.178 (192.168.37.178) 56(84) bytes of data.
64 bytes from 192.168.37.178: icmp_seq=1 ttl=62 time=1.31 ms
64 bytes from 192.168.37.178: icmp_seq=2 ttl=62 time=0.634 ms
64 bytes from 192.168.37.178: icmp_seq=3 ttl=62 time=0.658 ms
^Z
[3]+  Stopped                  ping 192.168.37.178
root@RA:/tmp/pycore.46289/RA.conf# ping 192.168.37.163
PING 192.168.37.163 (192.168.37.163) 56(84) bytes of data.
64 bytes from 192.168.37.163: icmp_seq=1 ttl=63 time=0.618 ms
64 bytes from 192.168.37.163: icmp_seq=2 ttl=63 time=0.188 ms
64 bytes from 192.168.37.163: icmp_seq=3 ttl=63 time=0.431 ms
^Z
[4]+  Stopped                  ping 192.168.37.163
root@RA:/tmp/pycore.46289/RA.conf# [ ]
```

Fig. 24. Verificação de conectividade

3 Conclusão

A realização deste trabalho permitiu-nos consolidar os conceitos abordados nas aulas teóricas desta UC, mais especificamente sobre **Datagramas IP e Fragmentação, Endereçamento e Encaminhamento** e ainda, **Subnetting**.

O trabalho encontrava-se dividido em duas partes, sendo que em ambas as partes foram utilizadas as ferramentas de gestão e monitorização de redes, *Core* e *Wireshark*.

Na primeira parte começamos por definir uma topologia de rede simples, recorrendo ao *Core* e focamo-nos principalmente no envio e receção de pacotes, através da análise do tráfego ICMP, bem como na constituição dos datagramas IP e em como funciona a sua fragmentação.

Na segunda parte foi definida uma nova topologia mais complexa, constituída por diferentes sub-redes.

Esta parte centrou-se no redirecionamento de pacotes IP, através da manipulação de tabelas de encaminhamento dos vários routers e da inserção e remoção de rotas. Centrou-se também nos tipos de encaminhamento estático e dinâmico, bem como nos endereços públicos e privados.

Ainda nesta parte foi também abordado o conceito de *subnetting*, nomeadamente as máscaras de redes, os endereços de sub-rede e as respetivas gamas.

Por fim, acreditamos ter cumprido o objetivo deste trabalho e obtido conhecimento importante na área das redes de computadores.