



Trabalho Prático 2 - Protocolo IPv4

Redes de Computadores

Grupo 7 - TP8

Gabriela Santos Ferreira da Cunha - a97393

Nuno Guilherme Cruz Varela - a96455

Simão Jorge da Silva Costa - a95176

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.

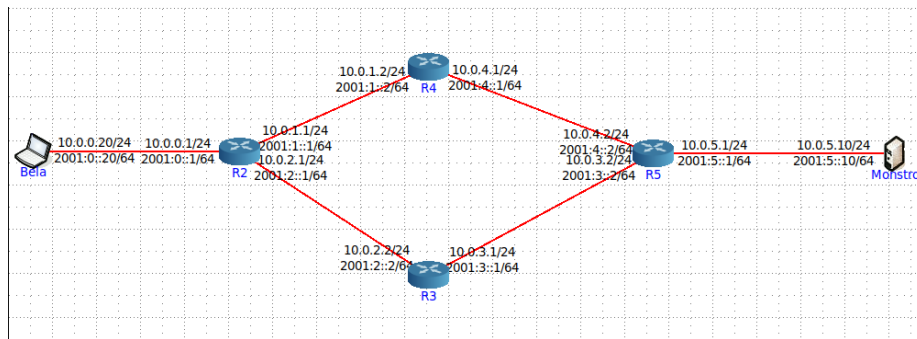


Figura 1: Topologia CORE.

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.42591/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) 0.039 ms 0.015 ms 0.013 ms
 2 10.0.1.2 (10.0.1.2) 0.025 ms 0.021 ms 0.020 ms
 3 10.0.3.2 (10.0.3.2) 0.039 ms 0.027 ms 0.027 ms
 4 10.0.5.10 (10.0.5.10) 0.037 ms 0.032 ms 0.033 ms
root@Bela:/tmp/pycore.42591/Bela.conf#
```

Figura 2: Output do comando “traceroute -I”.

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.

No.	Time	Source	Destination	Protocol	Length	Info
6	0.002292467	10.0.0.1	224.0.0.5	OSPF	78	Hello Packet
7	0.003112697	10.0.0.1	224.0.0.5	OSPF	78	Hello Packet
8	11.888215783	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=1/256, ttl=1 (no response...
9	11.888243370	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
10	11.888256136	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=2/512, ttl=1 (no response...
11	11.888267856	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
12	11.888276456	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=3/768, ttl=1 (no response...
13	11.888286944	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
14	11.888297366	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=4/1024, ttl=2 (no respons...
15	11.888314745	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
16	11.888330907	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=5/1280, ttl=2 (no respons...
17	11.888343444	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
18	11.888357488	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=6/1536, ttl=2 (no respons...
19	11.888374410	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
20	11.888384603	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=7/1792, ttl=3 (no respons...
21	11.888421129	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
22	11.888438876	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=8/2048, ttl=3 (no respons...
23	11.888450911	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
24	11.888464635	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=9/2304, ttl=3 (no respons...
25	11.888488996	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
26	11.888499188	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=10/2560, ttl=4 (reply in ...
27	11.888535558	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0066, seq=10/2560, ttl=61 (request ...
28	11.888545737	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x0066, seq=11/2816, ttl=4 (reply in ...
29	11.888575462	10.0.5.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0066, seq=11/2816, ttl=61 (request ...
▶ Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface veth1.0.37, id 0 ▶ Ethernet II, Src: 00:00:00:aa:00:01 (00:00:00:aa:00:01), Dst: IPv4mcast_05 (01:00:5e:00:00:05) ▶ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 224.0.0.5 ▶ Open Shortest Path First						
0000	01 00 5e 00 00 05 00 00	aa 00 00 01 08 00 45 c0	...A.....E..			
0010	00 40 6a 41 00 00 01 59	64 5e 0a 00 00 01 e0 00	...@JA...Y d^.....			
0020	00 00 02 01 00 00 0a 00	00 01 00 00 00 01 f1 00			
0030	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 02			
0040	02 01 00 00 00 00 00 00	00 00 00 00 00 00 00 00			

Figura 3: Tráfego enviado e recebido pelo sistema Bela.

Neste caso em específico, o cliente Bela envia conjuntos de 3 pacotes, em que cada conjunto leva um valor no campo TTL (time-to-live) incrementado numa unidade, começando em 1. Nesta tipologia, os pacotes deveriam retornar uma mensagem de erro ICMP quando o TTL é inferior a 4. Tal facto, é comprovado com os resultados obtidos na figura 3.

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 inicial mínimo do campo TTL para alcançar o servidor Monstro é 4, tal como explicado na alínea anterior.

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.

Para obtermos uma média mais precisa, alteramos o número de pacotes de prova para 10 e obtivemos os seguintes resultados:

```

root@Bela:/tmp/pycore.42591/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)  0.040 ms  0.015 ms  0.014 ms  0.013 ms  0.014 ms  0.014
ms  * * * *
 2  10.0.1.2 (10.0.1.2)  0.028 ms  0.021 ms  0.021 ms  0.021 ms  0.020 ms  0.021
ms  * * * *
 3  10.0.3.2 (10.0.3.2)  0.083 ms  0.037 ms  0.028 ms  0.048 ms  0.029 ms  0.040
ms  * * * *
 4  10.0.5.10 (10.0.5.10)  0.040 ms  0.033 ms  0.033 ms  0.034 ms  0.079 ms  0.0
47 ms  0.033 ms  0.034 ms  0.045 ms  0.081 ms
root@Bela:/tmp/pycore.42591/Bela.conf# S

```

Figura 4: Output do comando traceroute -I -q 10.

A partir dos valores apresentados na figura 4, conseguimos calcular o valor médio de ida-e-volta.

$$media = \frac{0.040+0.033+0.033+0.034+0.079+0.047+0.033+0.034+0.045+0.081}{10} = 0.0459$$

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?

O valor médio do atraso num sentido não deve ser calculado dividindo o RTT por dois, visto que poderá haver tráfego no sistema que faça com que o pacote não volte pelo mesmo caminho que percorreu na ida, pelo que poderá ter diferentes tempos na ida e na volta, sendo este o principal entrave a obter uma medição precisa desta métrica.

1.2 Exercício 2

Pretende-se agora usar o traceroute na sua máquina nativa, e gerar datagramas IP de diferentes tamanhos. Usando o wireshark capture o tráfego gerado pelo traceroute para o tamanho do pacote de prova por defeito; Utilize como máquina destino o host marco.uminho.pt. Selecione a primeira mensagem ICMP capturada (referente a (i) tamanho por defeito) e centre a análise no nível protocolar IP (expanda o tab correspondente na janela de detalhe do wireshark).

No.	Time	Source	Destination	Protocol	Length	Info
128	15.935783564	172.26.80.233	172.26.80.79	TCP	54	[TCP ACKed unseen segment] 443 - 54150 [ACK] Seq=1 Ack=2 Win=...
129	16.398223039	142.250.200.138	172.26.80.79	TCP	66	[TCP ACKed unseen segment] 443 - 35444 [ACK] Seq=1 Ack=2 Win=...
130	17.002344654	172.26.80.79	193.137.16.65	DNS	86	Standard query 0x9878 A marco.uminho.pt OPT
131	17.002451687	172.26.80.79	193.137.16.65	DNS	86	Standard query 0x863d AAAA marco.uminho.pt OPT
132	17.007340494	193.137.16.65	172.26.80.79	DNS	102	Standard query response 0x9878 A marco.uminho.pt A 193.136.9...
133	17.008552068	193.137.16.65	172.26.80.79	DNS	140	Standard query response 0x863d AAAA marco.uminho.pt SOA dns.u...
134	17.008893556	172.26.80.79	193.136.9.240	ICMP	74	Echo (ping) request id=0x0001, seq=1/256, ttl=1 (no response...
135	17.008899723	172.26.80.79	193.136.9.240	ICMP	74	Echo (ping) request id=0x0001, seq=2/512, ttl=1 (no response...
136	17.008899128	172.26.80.79	193.136.9.240	ICMP	74	Echo (ping) request id=0x0001, seq=3/768, ttl=1 (no response...
137	17.008907693	172.26.80.79	193.136.9.240	ICMP	74	Echo (ping) request id=0x0001, seq=4/1024, ttl=2 (no respons...
138	17.008912645	172.26.80.79	193.136.9.240	ICMP	74	Echo (ping) request id=0x0001, seq=5/1280, ttl=2 (no respons...
139	17.008917697	172.26.80.79	193.136.9.240	ICMP	74	Echo (ping) request id=0x0001, seq=6/1536, ttl=2 (no respons...
140	17.008923769	172.26.80.79	193.136.9.240	ICMP	74	Echo (ping) request id=0x0001, seq=7/1792, ttl=3 (no respons...
141	17.008929989	172.26.80.79	193.136.9.240	ICMP	74	Echo (ping) request id=0x0001, seq=8/2048, ttl=3 (no respons...
142	17.008936402	172.26.80.79	193.136.9.240	ICMP	74	Echo (ping) request id=0x0001, seq=9/2304, ttl=3 (no respons...
143	17.008942461	172.26.80.79	193.136.9.240	ICMP	74	Echo (ping) request id=0x0001, seq=10/2560, ttl=4 (no respons...
Frame 134: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp3s0, id 0						
Ethernet II, Src: LiteonTe_8a:34:ca (ac:b5:7d:8a:34:ca), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)						
Internet Protocol Version 4, Src: 172.26.80.79, 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: 0xd86c (55404)						
Flags: 0x0000						
Fragment offset: 0						
Time to live: 1						
Protocol: ICMP (1)						
Header checksum: 0x1973 [validation disabled]						
[Header checksum status: Unverified]						
Source: 172.26.80.79						
Destination: 193.136.9.240						
Internet Control Message Protocol						

Figura 5: Dados relativos à 1ª mensagem ICMP capturada.

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

O endereço IP é 172.26.80.79, como podemos verificar através da figura 5.

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

Frame 134: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp3s0, id 0
Ethernet II, Src: LiteonTe_8a:34:ca (ac:b5:7d:8a:34:ca), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.80.79, 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: 0xd86c (55404)
Flags: 0x0000
Fragment offset: 0
Time to live: 1
Protocol: ICMP (1)
Header checksum: 0x1973 [validation disabled]
[Header checksum status: Unverified]
Source: 172.26.80.79
Destination: 193.136.9.240
Internet Control Message Protocol

Figura 6: Valor do campo protocolo.

O valor do campo protocolo é ICMP(1), protocolo usado para enviar a mensagem de erro.

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?

```

> Frame 134: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp3s0, id 0
> Ethernet II, Src: LiteonTe_8a:34:ca (ac:b5:7d:8a:34:ca), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
- Internet Protocol Version 4, Src: 172.26.80.79, 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: 0xd86c (55404)
  > Flags: 0x0000
  Fragment offset: 0
  > Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x1973 [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.80.79
  Destination: 193.136.9.240

```

Figura 7: Tamanho do cabeçalho IPv4 e tamanho total.

Através da figura 7, verificamos que o tamanho do cabeçalho IPv4 é 20 *bytes* e o tamanho total é 60 *bytes*. Podemos calcular o tamanho do *payload*, subtraindo o tamanho do cabeçalho IPv4 ao tamanho total do datagrama.

$$\text{tamanho do payload} = 60 - 20 = 40 \text{ bytes}$$

Alínea D O datagrama IP foi fragmentado? Justifique.

```

> Frame 134: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp3s0, id 0
> Ethernet II, Src: LiteonTe_8a:34:ca (ac:b5:7d:8a:34:ca), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
- Internet Protocol Version 4, Src: 172.26.80.79, 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: 0xd86c (55404)
  > 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: 0x1973 [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.80.79
  Destination: 193.136.9.240

```

Figura 8: Campos "more fragments" e "fragment offset".

Pela figura 8, observamos que o campo "Fragment offset" e o campo "More fragments" têm ambos valor 0, ou seja, 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.

134	17.008863856	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=1/256, ttl=1 (no response...
135	17.008890723	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=2/512, ttl=1 (no response...
136	17.008899128	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=3/768, ttl=1 (no response...
137	17.008907693	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=4/1024, ttl=2 (no respons...
138	17.008912645	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=5/1280, ttl=2 (no respons...
139	17.008917697	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=6/1536, ttl=2 (no respons...
140	17.008923769	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=7/1792, ttl=3 (no respons...
141	17.008929989	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=8/2048, ttl=3 (no respons...
142	17.008936402	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=9/2304, ttl=3 (no respons...
143	17.008943461	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=10/2560, ttl=4 (reply in ...
144	17.008949948	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=11/2816, ttl=4 (reply in ...
145	17.008954131	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=12/3072, ttl=4 (reply in ...
146	17.008958678	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=13/3328, ttl=5 (reply in ...
147	17.008963280	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=14/3584, ttl=5 (reply in ...
148	17.008968029	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=15/3840, ttl=5 (reply in ...
149	17.008971259	172.26.80.79	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0001, seq=16/4096, ttl=6 (reply in ...

Figura 9: Sequência de tráfego ICMP ordenado por IP.

Os campos do cabeçalho IP que variam de pacote para pacote são “Identification“, “Header checksum“, “Checksum“ e “Time to live“.

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

O valor do campo “Identification“ incrementa uma unidade por cada pacote e o valor do campo “Time to live“ incrementa uma unidade a cada três pacotes.

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?

150	17.016791757	172.26.254.254	172.26.80.79	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
152	17.020816608	172.26.254.254	172.26.80.79	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
153	17.020832399	172.16.2.1	172.26.80.79	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
154	17.020838261	172.16.2.1	172.26.80.79	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
155	17.020843379	172.26.254.254	172.26.80.79	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
156	17.020849597	172.16.2.1	172.26.80.79	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
157	17.020853162	193.136.9.240	172.26.80.79	ICMP	74 Echo (ping) reply	id=0x0001, seq=10/2560, ttl=4 (request ...
158	17.020858616	193.136.9.240	172.26.80.79	ICMP	74 Echo (ping) reply	id=0x0001, seq=13/3328, ttl=6 (request ...
159	17.020862898	193.136.9.240	172.26.80.79	ICMP	74 Echo (ping) reply	id=0x0001, seq=11/2816, ttl=6 (request ...
160	17.020867585	193.136.9.240	172.26.80.79	ICMP	74 Echo (ping) reply	id=0x0001, seq=14/3584, ttl=6 (request ...
161	17.020872565	193.136.9.240	172.26.80.79	ICMP	74 Echo (ping) reply	id=0x0001, seq=12/3072, ttl=6 (request ...
162	17.020876826	193.136.9.240	172.26.80.79	ICMP	74 Echo (ping) reply	id=0x0001, seq=15/3840, ttl=6 (request ...
163	17.020881091	193.136.9.240	172.26.80.79	ICMP	74 Echo (ping) reply	id=0x0001, seq=16/4096, ttl=6 (request ...
164	17.020885488	172.16.115.252	172.26.80.79	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
165	17.020889432	172.16.115.252	172.26.80.79	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
166	17.020897088	172.16.115.252	172.26.80.79	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
167	17.021346067	193.137.16.65	172.26.80.79	DNS	152 Standard query response 0xa5f6 No such name PTR 254.254.26.17...	
169	17.027459042	193.137.16.65	172.26.80.79	DNS	141 Standard query response 0xa5f6 No such name PTR 254.254.26.17...	
172	17.030194074	193.136.9.240	172.26.80.79	ICMP	74 Echo (ping) reply	id=0x0001, seq=17/4352, ttl=6 (request ...

Figura 10: Sequência de tráfego ICMP ordenado por endereço destino.

O valor do campo “Time to live“ varia de 255 a 253, pelo que não permanece constante. Isto deve-se ao decremento que o valor sofre a cada *router* que passa até chegar ao *host* de origem.

1.3 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 4087 *bytes*.

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

7 0.992681171	172.26.80.79	193.136.9.240	ICMP	1514 Echo (ping) request id=0x0002, seq=1/256, ttl=1 (no response..
8 0.992708113	172.26.80.79	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=f1480, ID=f2b9)
9 0.992716980	172.26.80.79	193.136.9.240	IPv4	1141 Fragmented IP protocol (proto=ICMP 1, off=f2960, ID=f2b9)
10 0.992741226	172.26.80.79	193.136.9.240	ICMP	1514 Echo (ping) request id=0x0002, seq=2/512, ttl=1 (no response..
11 0.992748804	172.26.80.79	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=f1480, ID=f2b9)
12 0.992751833	172.26.80.79	193.136.9.240	IPv4	1141 Fragmented IP protocol (proto=ICMP 1, off=f2960, ID=f2ba)
13 0.992771225	172.26.80.79	193.136.9.240	ICMP	1514 Echo (ping) request id=0x0002, seq=3/768, ttl=1 (no response..
14 0.992775546	172.26.80.79	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=f1480, ID=f2bb)
15 0.992778943	172.26.80.79	193.136.9.240	IPv4	1141 Fragmented IP protocol (proto=ICMP 1, off=f2960, ID=f2bb)
16 0.992795953	172.26.80.79	193.136.9.240	ICMP	1514 Echo (ping) request id=0x0002, seq=4/1024, ttl=2 (no respons..
17 0.992799352	172.26.80.79	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=f1480, ID=f2bc)
18 0.992801476	172.26.80.79	193.136.9.240	IPv4	1141 Fragmented IP protocol (proto=ICMP 1, off=f2960, ID=f2bc)

Figura 11: Sequência de tráfego ICMP ordenado por endereço destino.

O pacote inicial foi fragmentado porque o seu tamanho excede o valor máximo suportado. Apenas é possível transmitir 1500 *bytes* por cada mensagem e o pacote tem 4087 *bytes*.

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?

```
▶ Frame 7: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface wlp3s0, id 0
▶ Ethernet II, Src: LiteonTe_8a:34:ca (ac:b5:7d:8a:34:ca), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.80.79, 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: 0xf2b9 (62137)
  ▼ 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: 0xd985 [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.80.79
  Destination: 193.136.9.240
```

Figura 12: Informação relativa ao primeiro fragmento do datagrama IP.

Como o campo “More fragments” tem o valor 1, podemos concluir que o datagrama foi fragmentado. O “Fragment offset” está com o valor a 0, logo estamos perante o primeiro fragmento, com um tamanho de 1500 *bytes*.

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?

```

> Frame 8: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface wlp3s0, id 0
> Ethernet II, Src: LiteonTe_8a:34:ca (ac:b5:7d:8a:34:ca), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
- Internet Protocol Version 4, Src: 172.26.80.79, 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: 0xf2b9 (62137)
  - Flags: 0x20b9, More fragments
    0... .. = Reserved bit: Not set
    .0... .. = Don't fragment: Not set
    ..1... .. = More fragments: Set
  - Fragment offset: 1480
  > Time to live: 1
  > Protocol: ICMP (1)
  Header checksum: 0xd8cc [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.80.79
  Destination: 193.136.9.240

```

Figura 13: Informação relativa ao segundo fragmento do datagrama IP original.

Como o “Fragment offset” está a 1480, concluímos que não se trata do primeiro fragmento. O bit da *flag* “More fragments” está a 1, pelo que existem mais fragmentos.

Alínea D Quantos fragmentos foram criados a partir do datagrama original?

```

> Frame 9: 1141 bytes on wire (9128 bits), 1141 bytes captured (9128 bits) on interface wlp3s0, id 0
> Ethernet II, Src: LiteonTe_8a:34:ca (ac:b5:7d:8a:34:ca), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
- Internet Protocol Version 4, Src: 172.26.80.79, 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: 1127
    Identification: 0xf2b9 (62137)
  - 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: 0xf988 [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.80.79
  Destination: 193.136.9.240

```

Figura 14: Informação relativa ao último fragmento do datagrama IP original.

Foram criados 3 fragmentos a partir do datagrama original, uma vez que este fragmento da figura 14 possui o bit da *flag* “More fragments” a 0 e é o terceiro fragmento. Outra maneira de os contar seria verificar quantos fragmentos têm o campo “Identification” igual.

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 que mudam entre os diferentes fragmentos são o “Fragment offset“, a *flag* “More fragments“, o “Total Length“ e o “Header checksum“. Isto permite-nos saber a posição de cada fragmento no datagrama original e o seu respetivo tamanho, o que nos ajuda na reconstrução do datagrama original.

Alínea F Verifique o processo de fragmentação através de um processo de cálculo.

$length_total = 4087 \text{ bytes}$
 $length_max_fragmento = 1500 \text{ bytes}$
 $id_fragmento = 0xf2b9$
 $tll_fragmento = 1$
 $dados_datagrama_original = 4087 - 20 = 4067 \text{ bytes}$
 $dados_por_fragmento = 1500 - 20 = 1480 \text{ bytes}$

- **Fragmento 1**

$dados_frag1 = 1480 \text{ bytes}$
 $header_frag1 = 20 \text{ bytes}$
 $length_frag1 = 1480 + 20 = 1500 \text{ bytes}$
 $dados_datagrama_original = 4067 - 1480 = 2587 \text{ bytes}$
 $morefragments_frag1 = 1$
 $offset_frag1 = 0$

- **Fragmento 2**

$dados_frag2 = 1480 \text{ bytes}$
 $header_frag2 = 20 \text{ bytes}$
 $length_frag2 = 1480 + 20 = 1500 \text{ bytes}$
 $dados_datagrama_original = 2587 - 1480 = 1107 \text{ bytes}$
 $morefragments_frag2 = 1$
 $offset_frag2 = 1480$

- **Fragmento 3**

$dados_frag3 = 1107 \text{ bytes}$

$header_frago3 = 20 \text{ bytes}$

$length_frag3 = 1107 + 20 = 1127 \text{ bytes}$

$dados_datagrama_original = 1107 - 1107 = 0 \text{ bytes}$

$morefragments_frag3 = 0$

$offset_frag3 = 2960$

$dados_final = dados_frag1 + dados_frag2 + dados_frag3 = 4067 \text{ bytes}$

$length_1pacote = 4087 \text{ bytes}$

$length_3pacotes = length_frag1 + length_frag2 + length_frag3 = 4127 \text{ bytes}$

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

Considerando que a expressão contempla apenas os pacotes fragmentados, podemos definir o último fragmento pela seguinte expressão lógica:

$$morefragments = 0 \wedge offset \neq 0 \wedge id_fragment = id_original$$

2 Parte II

Considere que a topologia de rede LEI-RC é distribuída por quatro departamentos (A, B, C e D) e cada departamento possui um router de acesso à sua rede local. Estes routers de acesso (RA, RB, RC e RD) estão interligados entre si por ligações Ethernet a 1Gbps, formando um anel. Por sua vez, existe um servidor por departamento (SA, SB, SC, SD) e dois portáteis (pc) por departamento (A - Bela, Monstro; B - Jasmine, Alladin; C - Ariel, Eric; D - Simba e Nala), todos interligados ao router respetivo através de um comutador (switch). Cada servidor S tem uma ligação a 1Gbps e os laptops ligações a 100Mbps. Considere apenas a existência de um comutador por departamento. A conectividade IP externa da organização é assegurada através de um router de acesso RISP conectado a RA por uma ligação ponto-a-ponto a 1 Gbps. Construa uma topologia CORE que reflita a rede local da organização.

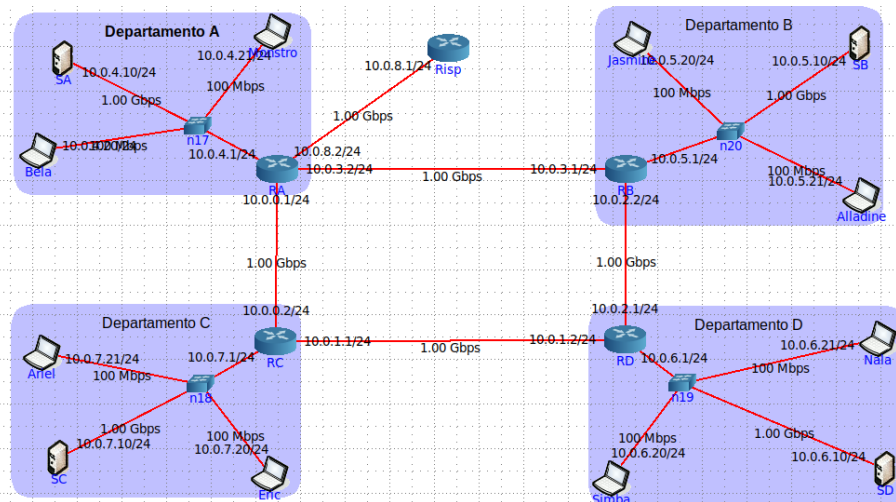


Figura 15: Topologia CORE

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.

Os endereços IP atribuídos utilizam 24 bits para o endereço de rede. A máscara de rede correspondente é 255.255.255.0, como podemos ver na figura 15.

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

Os endereços são privados, uma vez que todos os endereços IP atribuídos pertencem ao intervalo de 10.0.0.0 a 10.255.255.255 - intervalo que define a classe A de endereços privados.

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

Estes equipamentos servem apenas para reencaminhar informação e funcionam exclusivamente através dos endereços físicos de rede - endereços MAC. Assim, não possuem endereço IP - endereço lógico de rede.

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).

```
root@Bela:/tmp/pycore.41785/Bela.conf# ping 10.0.4.21
PING 10.0.4.21 (10.0.4.21) 56(84) bytes of data.
64 bytes from 10.0.4.21: icmp_seq=1 ttl=64 time=0,271 ms
64 bytes from 10.0.4.21: icmp_seq=2 ttl=64 time=0,155 ms
64 bytes from 10.0.4.21: icmp_seq=3 ttl=64 time=0,145 ms
64 bytes from 10.0.4.21: icmp_seq=4 ttl=64 time=0,145 ms
64 bytes from 10.0.4.21: icmp_seq=5 ttl=64 time=0,141 ms
^C
--- 10.0.4.21 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4068ms
rtt min/avg/max/mdev = 0,141/0,171/0,271/0,050 ms
```

Figura 16: Ping do *laptop* Bela para o *laptop* Monstro.

```
root@Jasmine:/tmp/pycore.41785/Jasmine.conf# ping 10.0.5.21
PING 10.0.5.21 (10.0.5.21) 56(84) bytes of data.
64 bytes from 10.0.5.21: icmp_seq=1 ttl=64 time=0,632 ms
64 bytes from 10.0.5.21: icmp_seq=2 ttl=64 time=0,154 ms
64 bytes from 10.0.5.21: icmp_seq=3 ttl=64 time=0,182 ms
64 bytes from 10.0.5.21: icmp_seq=4 ttl=64 time=0,149 ms
64 bytes from 10.0.5.21: icmp_seq=5 ttl=64 time=0,261 ms
^C
--- 10.0.5.21 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4091ms
rtt min/avg/max/mdev = 0,149/0,275/0,632/0,182 ms
```

Figura 17: Ping do *laptop* Jasmine para o *laptop* Alladine.

```

root@Ariel:/tmp/pycore.41785/Ariel.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=64 time=0.778 ms
64 bytes from 10.0.7.20: icmp_seq=2 ttl=64 time=1.17 ms
64 bytes from 10.0.7.20: icmp_seq=3 ttl=64 time=0.149 ms
64 bytes from 10.0.7.20: icmp_seq=4 ttl=64 time=0.160 ms
64 bytes from 10.0.7.20: icmp_seq=5 ttl=64 time=0.155 ms
^C
--- 10.0.7.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4064ms
rtt min/avg/max/mdev = 0.149/0.482/1.170/0.420 ms

```

Figura 18: Ping do *laptop* Ariel para o *laptop* Eric.

```

root@Simba:/tmp/pycore.41785/Simba.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=64 time=0.597 ms
64 bytes from 10.0.6.21: icmp_seq=2 ttl=64 time=0.147 ms
64 bytes from 10.0.6.21: icmp_seq=3 ttl=64 time=0.150 ms
64 bytes from 10.0.6.21: icmp_seq=4 ttl=64 time=0.958 ms
64 bytes from 10.0.6.21: icmp_seq=5 ttl=64 time=0.142 ms
^C
--- 10.0.6.21 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4059ms
rtt min/avg/max/mdev = 0.142/0.398/0.958/0.329 ms

```

Figura 19: Ping do *laptop* Simba para o *laptop* Nala.

Como podemos verificar nas 4 figuras anteriores, foram transmitidos e recebidos 5 pacotes em cada uma e o *packet loss* é 0%, ou seja, concluímos que existe conectividade IP 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.

O número mínimo de comandos “ping” é 6. Em relação ao departamento A, começamos por provar as conectividades $A \rightarrow B$, $A \rightarrow C$ e $A \rightarrow D$, sendo necessários 3 “pings”. Ao provar estas, provamos também as inversas, isto é, $B \rightarrow A$, $C \rightarrow A$ e $D \rightarrow A$. Em relação ao departamento B, falta-nos provar $B \rightarrow C$ ($C \rightarrow B$) e $B \rightarrow D$ ($D \rightarrow B$) e são, portanto, necessários mais 2 “pings”. Por último, falta-nos apenas provar $C \rightarrow D$ ($D \rightarrow C$), sendo preciso apenas 1 “ping”.

```
root@Bela:/tmp/pycore.44099/Bela.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=62 time=0.572 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=62 time=0.240 ms
^C
--- 10.0.5.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.240/0.406/0.572/0.166 ms
root@Bela:/tmp/pycore.44099/Bela.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=61 time=1.22 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=61 time=0.348 ms
^C
--- 10.0.6.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.348/0.784/1.220/0.436 ms
root@Bela:/tmp/pycore.44099/Bela.conf# ping 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data.
64 bytes from 10.0.7.10: icmp_seq=1 ttl=62 time=1.06 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=62 time=0.263 ms
^C
--- 10.0.7.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.263/0.659/1.055/0.396 ms
```

Figura 20: Pings com origem no departamento A para os restantes departamentos.

```
root@Alladine:/tmp/pycore.44099/Alladine.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=62 time=0.986 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=62 time=0.253 ms
^C
--- 10.0.6.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.253/0.619/0.986/0.366 ms
root@Alladine:/tmp/pycore.44099/Alladine.conf# ping 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data.
64 bytes from 10.0.7.10: icmp_seq=1 ttl=61 time=1.11 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=61 time=0.350 ms
^C
--- 10.0.7.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.350/0.728/1.106/0.378 ms
```

Figura 21: Pings com origem no departamento B para os departamentos C e D.

```

root@Simba:/tmp/pycore.44099/Simba.conf# ping 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data:
64 bytes from 10.0.7.10: icmp_seq=1 ttl=62 time=1.06 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=62 time=0.270 ms
^C
--- 10.0.7.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.270/0.666/1.062/0.396 ms

```

Figura 22: Pings com origem no departamento D para o departamento C.

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

```

root@Bela:/tmp/pycore.41785/Bela.conf# ping 10.0.8.1
PING 10.0.8.1 (10.0.8.1) 56(84) bytes of data:
64 bytes from 10.0.8.1: icmp_seq=1 ttl=63 time=0.913 ms
64 bytes from 10.0.8.1: icmp_seq=2 ttl=63 time=0.202 ms
64 bytes from 10.0.8.1: icmp_seq=3 ttl=63 time=0.208 ms
64 bytes from 10.0.8.1: icmp_seq=4 ttl=63 time=0.230 ms
64 bytes from 10.0.8.1: icmp_seq=5 ttl=63 time=0.212 ms
^C
--- 10.0.8.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4069ms
rtt min/avg/max/mdev = 0.202/0.353/0.913/0.280 ms

```

Figura 23: Ping do *laptop* Bela para o *router* RISP.

Como podemos verificar na figura 23, foram enviados 5 pacotes e recebidos 5 pacotes, pelo que concluímos que existe conectividade entre o portátil Bela e o *router* RISP.

2.2 Exercício 2

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`).

```

root@Bela:/tmp/pycore.43509/Bela.conf# netstat -rn
Kernel IP routing table

```

Destination	Gateway	Genmask	Flags	MSS Window	irrtt	Iface
0.0.0.0	10.0.4.1	0.0.0.0	UG	0 0		eth0
10.0.4.0	0.0.0.0	255.255.255.0	U	0 0		eth0

Figura 24: Tabela de encaminhamento do *laptop* Bela.

Na tabela de encaminhamento do *laptop* Bela, verificamos que o *laptop* utiliza o destino *default* (0.0.0.0) para envio de pacotes para fora do departamento, passando, deste modo, pelo *router* RA (10.0.4.1). Todos os pacotes com destino na própria sub-rede (10.0.4.0) são encaminhados diretamente pelo host sem necessidade de passar no *router*.

```
root@RA:/tmp/pycore.41785/RA.conf# netstat -rn
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.0.0	0.0.0.0	255.255.255.0	U	0 0		0	eth0
10.0.1.0	10.0.0.2	255.255.255.0	UG	0 0		0	eth0
10.0.2.0	10.0.3.1	255.255.255.0	UG	0 0		0	eth1
10.0.3.0	0.0.0.0	255.255.255.0	U	0 0		0	eth1
10.0.4.0	0.0.0.0	255.255.255.0	U	0 0		0	eth3
10.0.5.0	10.0.3.1	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	eth2

Figura 25: Tabela de encaminhamento do *router* RA.

Relativamente à tabela de encaminhamento do *router* RA, os pacotes destinados às sub-redes 10.0.1.0, 10.0.6.0 e 10.0.7.0 têm como próximo salto o IP 10.0.0.2, correspondente ao *router* RC, saindo pela interface *Ethernet* 0. Os pacotes com destino às sub-redes 10.0.2.0 e 10.0.5.0 são encaminhados para o endereço IP 10.0.3.1, referente ao *router* RB. Quanto às restantes sub-redes, estas possuem um *gateway* de 0.0.0.0, pelo que o encaminhamento de pacotes será tratado pelo próprio *router* RA. Isto acontece, visto que o *router* é adjacente às sub-redes.

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).

```
root@Bela:/tmp/pycore.43509/Bela.conf# ps -ax
```

PID	TTY	STAT	TIME	COMMAND
1	?	S	0:00	vnode -v -c /tmp/pycore.43509/Bela -l /tmp/pycore.
20	pts/3	Ss+	0:00	/bin/bash
35	pts/4	Ss	0:00	/bin/bash
42	pts/4	R+	0:00	ps -ax

Figura 26: Processos do *laptop* Bela.

```

root@RA:/tmp/pycore.43509/RA.conf# ps -ax
  PID TTY          STAT TIME COMMAND
    1 ?           S      0:00 vncnode -v -c /tmp/pycore.43509/RA -l /tmp/pycore.43
   75 ?           Ss     0:00 /usr/local/sbin/zebra -d
   81 ?           Ss     0:00 /usr/local/sbin/ospf6d -d
   85 ?           Ss     0:00 /usr/local/sbin/ospfd -d
   93 pts/6       Ss     0:00 /bin/bash
  100 pts/6       R+     0:00 ps -ax

```

Figura 27: Processos do *router* RA.

Como no *laptop* Bela não existem processos a correr para além dos do sistema, concluímos que o encaminhamento é estático. Já no *router* RA é usado o encaminhamento dinâmico pois podemos visualizar processos, como os de identificador 75, 81, 85, os quais são protocolos de encaminhamento dinâmico.

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.

```

root@SA:/tmp/pycore.43509/SA.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0          10.0.4.1        0.0.0.0         UG        0 0          0 eth0
10.0.4.0         0.0.0.0         255.255.255.0   U        0 0          0 eth0
root@SA:/tmp/pycore.43509/SA.conf# route delete default
root@SA:/tmp/pycore.43509/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.43509/SA.conf#

```

Figura 28: Tabelas de encaminhamento do *router* RA antes e após o comando “route delete”.

Com a utilização do comando “route delete” retiramos a rota entre o servidor SA e o *router* RA, pelo que a conexão para fora do departamento A foi cortada. Ainda assim, o servidor SA consegue estabelecer conexões a nível de rede local (departamento A), uma vez que mantém o destino 10.0.4.0 .

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.

```

root@SA:/tmp/pycore.36427/SA.conf# route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.4.1
root@SA:/tmp/pycore.36427/SA.conf# route add -net 10.0.6.0 netmask 255.255.255.0 gw 10.0.4.1
root@SA:/tmp/pycore.36427/SA.conf# route add -net 10.0.7.0 netmask 255.255.255.0 gw 10.0.4.1

```

Figura 29: Comandos para restaurar a conectividade para o servidor SA.

Como podemos averiguar na figura 29, foram adicionadas rotas com o comando “route add” para as sub-redes dos departamentos B, C e D.

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 das figuras 30, 31 e 32, podemos confirmar que foi restabelecida a conectividade para os restantes departamentos. Na figura 33 encontra-se a nova tabela de encaminhamento do servidor SA.

```

root@Bela:/tmp/pycore.43509/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=0.594 ms
64 bytes from 10.0.5.20: icmp_seq=2 ttl=62 time=0.254 ms
64 bytes from 10.0.5.20: icmp_seq=3 ttl=62 time=0.254 ms
64 bytes from 10.0.5.20: icmp_seq=4 ttl=62 time=0.261 ms
64 bytes from 10.0.5.20: icmp_seq=5 ttl=62 time=0.250 ms
^C
--- 10.0.5.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4055ms
rtt min/avg/max/mdev = 0.250/0.322/0.594/0.135 ms

```

Figura 30: Ping do *laptop* Bela para o *laptop* Jasmine.

```

root@Bela:/tmp/pycore.43509/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=1.15 ms
64 bytes from 10.0.6.20: icmp_seq=2 ttl=61 time=0.364 ms
64 bytes from 10.0.6.20: icmp_seq=3 ttl=61 time=0.368 ms
64 bytes from 10.0.6.20: icmp_seq=4 ttl=61 time=0.350 ms
64 bytes from 10.0.6.20: icmp_seq=5 ttl=61 time=0.338 ms
^C
--- 10.0.6.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4050ms
rtt min/avg/max/mdev = 0.338/0.513/1.148/0.317 ms

```

Figura 31: Ping do *laptop* Bela para o *laptop* Simba.

```

root@Bela:/tmp/pycore.43509/Bela.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=1.36 ms
64 bytes from 10.0.7.20: icmp_seq=2 ttl=62 time=0.269 ms
64 bytes from 10.0.7.20: icmp_seq=3 ttl=62 time=0.260 ms
64 bytes from 10.0.7.20: icmp_seq=4 ttl=62 time=0.252 ms
64 bytes from 10.0.7.20: icmp_seq=5 ttl=62 time=0.855 ms
^C
--- 10.0.7.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4056ms
rtt min/avg/max/mdev = 0.252/0.598/1.357/0.443 ms

```

Figura 32: Ping do *laptop* Bela para o *laptop* Eric.

```

root@SA:/tmp/pycore.36427/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
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

```

Figura 33: Tabela de encaminhamento do servidor SA.

2.3 Exercício 3

Alínea A 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 planejamento.

O decimal do nosso grupo é 87, sendo o nosso endereço de rede IP 192.168.87.128/25. Inicialmente temos uma máscara de rede de 25 bits, o que permite representar os endereços entre 192.168.87.128 e 192.168.87.254.

Cada departamento é uma sub-rede e temos, assim, 4 sub-redes no nosso esquema de endereçamento, visto que as redes de acesso externo e *backbone* se mantêm inalteradas. Para representar estas sub-redes, utilizaremos uma parte dos 7 bits que sobram dos 32, devido aos 25 bits inalteráveis da máscara de rede. Uma vez que todos os endereços de sub-redes devem ser usáveis poderíamos apenas utilizar 2 bits. Todavia, temos de ter em conta a criação de novos departamentos futuros e, com isto, optamos por alocar 3 bits para a representação das sub-redes. Deste modo, os 25 bits iniciais para a máscara passam a ser 28 bits, pelo que dispomos de $32 - 28 = 4$ bits para gerir *host interfaces*.

Departamento	Sub-Rede	Endereço IP	Gama de Endereços
A	000	192.168.87.128	192.168.87.129 - 192.168.87.142
B	001	192.168.87.144	192.168.87.145 - 192.168.87.158
C	010	192.168.87.160	192.168.87.161 - 192.168.87.174
D	011	192.168.87.176	192.168.87.177 - 192.168.87.190
Livre	100	192.168.87.192	192.168.87.193 - 192.168.87.206
Livre	101	192.168.87.208	192.168.87.209 - 192.168.87.222
Livre	110	192.168.87.224	192.168.87.225 - 192.168.87.238
Livre	111	192.168.87.240	192.168.87.241 - 192.168.87.254

Como podemos observar a partir da tabela, sobram 4 sub-redes que estão livres para departamentos futuros. Apresentamos na figura seguinte a topologia CORE com o novo endereçamento.

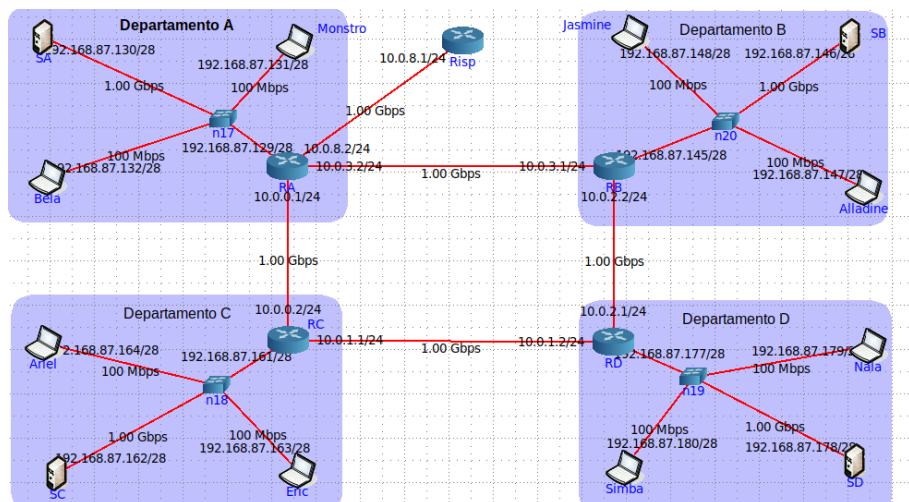


Figura 34: Topologia core com *subnetting*.

Alínea B 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 para as sub-redes foi 255.255.255.248. Como distribuímos os 7 bits em 3 bits para representar as sub-redes e 4 bits para gerir *host interfaces*, podemos adicionar mais $2^3 - 4 = 4$ departamentos futuros e a capacidade de interligação é de $2^4 - 2 = 14$ hosts IP por departamento, tendo em conta que o endereço com os bits todos a 0 e 1 estão reservados para o *host* e *broadcast*, respetivamente.

Alínea C 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.

A conectividade IP interna na rede local LEI-RC é mantida. Para tal, utilizamos o comando “ping” e enviamos pacotes através do *laptop* Bela para cada um dos *laptops* de cada departamento que foram entregues com sucesso, como demonstrado na figura 35.

```
root@Bela:/tmp/pycore.36427/Bela.conf# ping 192.168.87.148
PING 192.168.87.148 (192.168.87.148) 56(84) bytes of data.
64 bytes from 192.168.87.148: icmp_seq=1 ttl=62 time=1.11 ms
64 bytes from 192.168.87.148: icmp_seq=2 ttl=62 time=0.288 ms
^C
--- 192.168.87.148 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.288/0.697/1.106/0.409 ms
root@Bela:/tmp/pycore.36427/Bela.conf# ping 192.168.87.180
PING 192.168.87.180 (192.168.87.180) 56(84) bytes of data.
64 bytes from 192.168.87.180: icmp_seq=1 ttl=61 time=1.82 ms
64 bytes from 192.168.87.180: icmp_seq=2 ttl=61 time=0.370 ms
^C
--- 192.168.87.180 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.370/1.095/1.820/0.725 ms
root@Bela:/tmp/pycore.36427/Bela.conf# ping 192.168.87.163
PING 192.168.87.163 (192.168.87.163) 56(84) bytes of data.
64 bytes from 192.168.87.163: icmp_seq=1 ttl=62 time=1.40 ms
64 bytes from 192.168.87.163: icmp_seq=2 ttl=62 time=0.316 ms
^C
--- 192.168.87.163 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.316/0.855/1.395/0.539 ms
```

Figura 35: Conetividade com o novo endereçamento.

3 Conclusão

De modo a concluir este trabalho, consideramos que conseguimos, de um modo geral, cumprir os objetivos.

Na primeira parte do trabalho abordamos a transmissão de pacotes entre vários equipamentos ligados a uma mesma rede, em específico, usando o protocolo IPv4. Pudemos realizar uma topologia CORE, analisar o tráfego correspondente ao envio de pacotes e estudar o formato de um datagrama IP e o processo de fragmentação de pacotes realizado pelo protocolo IP.

Já na segunda parte, abordamos o endereçamento e encaminhamento de pacotes entre departamentos distintos. Construimos mais 2 topologias CORE, ambas com 4 departamentos distintos, uma de modo a verificar a conectividade entre departamentos e a analisar e modificar o encaminhamento dos pacotes quando enviados para outros departamentos e outra para implementar o *subnetting*, manipulando os endereços IP de modo a criar sub-redes.

Assim, este trabalho promoveu a consolidação dos conceitos teóricos. Destacamos, também, a importância da aplicação prática, isto é, a aprendizagem da utilização de *softwares* e comandos relativos à monitorização de redes, algo que consideramos que será útil na vida profissional.