

UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

TP2: Protocolo IP (Parte I e II)
Grupo Nº 7

João Teixeira (A85504)

José Ferreira (A83683)

Miguel Solino (A86435)

17 de Novembro de 2019

Conteúdo

1	Parte 1	3
1.1	Exercício 1	3
1.1.1	Alínea a	3
1.1.2	Alínea b	4
1.1.3	Alínea c	4
1.1.4	Alínea d	5
1.2	Exercício 2	5
1.2.1	Alínea a	5
1.2.2	Alínea b	5
1.2.3	Alínea c	6
1.2.4	Alínea d	6
1.2.5	Alínea e	6
1.2.6	Alínea f	6
1.2.7	Alínea g	7
1.3	Exercício 3	7
1.3.1	Alínea a	8
1.3.2	Alínea b	8
1.3.3	Alínea c	9
1.3.4	Alínea d	9
1.3.5	Alínea e	9
2	Parte 2	10
2.1	Exercício 1	10
2.1.1	Alínea a	10
2.1.2	Alínea b	10
2.1.3	Alínea c	11
2.1.4	Alínea d	11
2.1.5	Alínea e	11
2.2	Exercício 2	12
2.2.1	Alínea a	12
2.2.2	Alínea b	13
2.2.3	Alínea c	13
2.2.4	Alínea d	14
2.2.5	Alínea e	15
2.3	Exercício 3	15
2.3.1	Alínea 1	15
2.3.2	Alínea 2	16
2.3.3	Alínea 3	16
3	Conclusão	19

Capítulo 1

Parte 1

1.1 Exercício 1

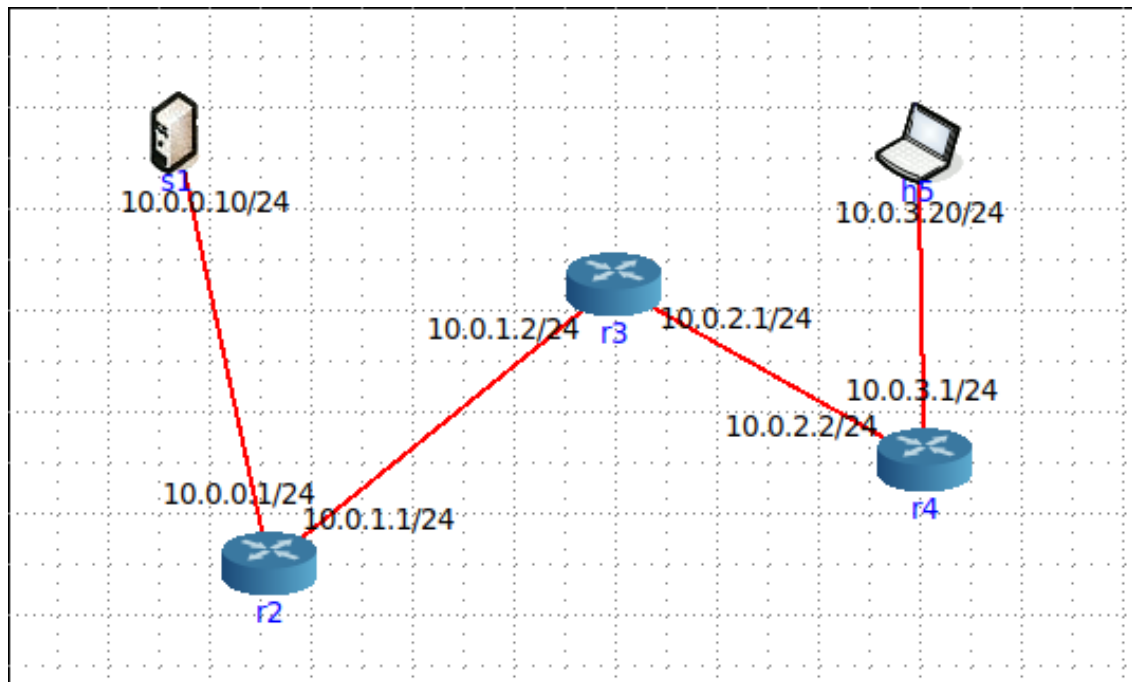


Figura 1.1: Core

1.1.1 Alínea a

Active o wireshark ou o tcpdump no pc s1. Numa shell de s1, execute o comando `tracert -l` para o endereço IP do host h5.

```
root@s1:/tmp/pycore.37745/s1.conf# traceroute -I 10.0.3.20
traceroute to 10.0.3.20 (10.0.3.20), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.075 ms 0.013 ms 0.010 ms
 2 10.0.1.2 (10.0.1.2) 0.022 ms 0.012 ms 0.012 ms
 3 10.0.2.2 (10.0.2.2) 0.025 ms 0.014 ms 0.013 ms
 4 10.0.3.20 (10.0.3.20) 0.041 ms 0.016 ms 0.015 ms
root@s1:/tmp/pycore.37745/s1.conf#
```

Figura 1.2: Traceroute

1.1.2 Alínea b

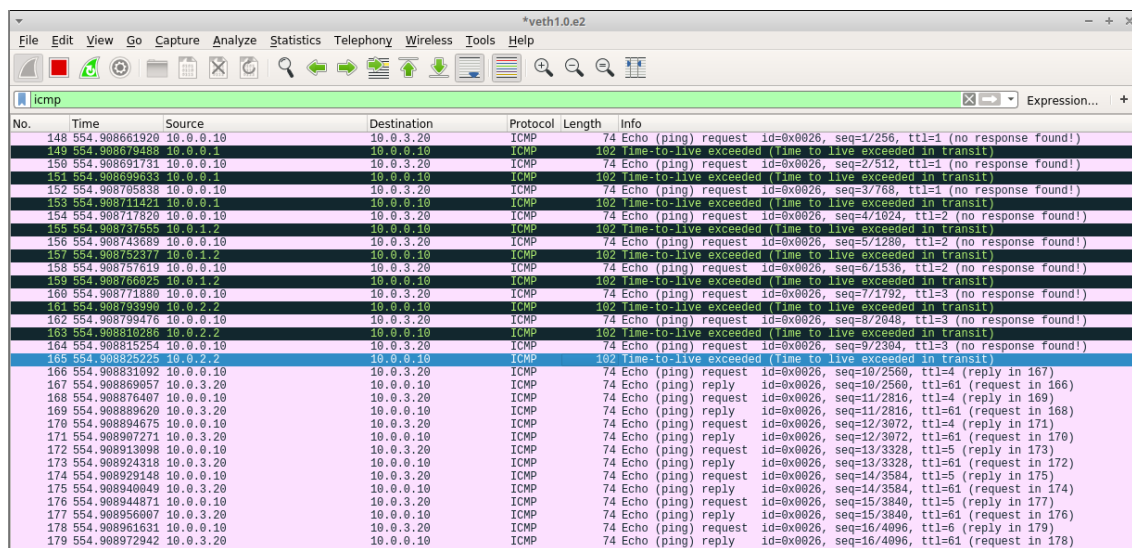
Registe e analise o tráfego ICMP enviado enviado por s1 e tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.

Analisando os resultados obtidos, constatamos que o envio de pacotes teve duas fases.

As fases estão divididas entre os pacotes com TTL abaixo de 4 e os com TTL acima de 4.

Na primeira fase, os pacotes com TTL = 1, TTL = 2 e TTL = 3 foram descartados pelos routers r1, r2 e r3, respetivamente. Para cada um destes foi recebido um pacote *Time-to-live exceeded*.

Na segunda fase, ao contrário dos pacotes anteriores, nenhum deles foi descartado tendo como resposta pacotes *Echo (ping) reply*.



No.	Time	Source	Destination	Protocol	Length	Info
148	554.908661920	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=1/256, ttl=1 (no response found!)
149	554.908679488	10.0.0.1	10.0.0.10	ICMP	102	time-to-live exceeded (time to live exceeded in transit)
150	554.908691761	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=2/512, ttl=1 (no response found!)
151	554.908699933	10.0.0.1	10.0.0.10	ICMP	102	time-to-live exceeded (time to live exceeded in transit)
152	554.908705838	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=3/768, ttl=1 (no response found!)
153	554.908711421	10.0.0.1	10.0.0.10	ICMP	102	time-to-live exceeded (time to live exceeded in transit)
154	554.908717820	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=4/1024, ttl=2 (no response found!)
155	554.908723655	10.0.0.1	10.0.0.10	ICMP	102	time-to-live exceeded (time to live exceeded in transit)
156	554.908735889	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=5/1280, ttl=2 (no response found!)
157	554.908752377	10.0.1.2	10.0.0.10	ICMP	102	time-to-live exceeded (time to live exceeded in transit)
158	554.908757619	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=6/1536, ttl=2 (no response found!)
159	554.908766025	10.0.1.2	10.0.0.10	ICMP	102	time-to-live exceeded (time to live exceeded in transit)
160	554.908771880	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=7/1792, ttl=3 (no response found!)
161	554.908783900	10.0.0.1	10.0.0.10	ICMP	102	time-to-live exceeded (time to live exceeded in transit)
162	554.908799476	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=8/2048, ttl=3 (no response found!)
163	554.908810286	10.0.0.2	10.0.0.10	ICMP	102	time-to-live exceeded (time to live exceeded in transit)
164	554.908815254	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=9/2304, ttl=3 (no response found!)
165	554.908823225	10.0.0.2	10.0.0.10	ICMP	102	time-to-live exceeded (time to live exceeded in transit)
166	554.908831092	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=10/2560, ttl=4 (reply in 167)
167	554.908869957	10.0.3.20	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0026, seq=10/2560, ttl=4 (request in 166)
168	554.908876407	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=11/2816, ttl=4 (reply in 169)
169	554.908889620	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0026, seq=11/2816, ttl=4 (request in 168)
170	554.908894675	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=12/3072, ttl=4 (reply in 171)
171	554.908907271	10.0.3.20	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0026, seq=12/3072, ttl=4 (request in 170)
172	554.908913898	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=13/3328, ttl=5 (reply in 173)
173	554.908924318	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0026, seq=13/3328, ttl=5 (request in 172)
174	554.908929148	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=14/3584, ttl=5 (reply in 175)
175	554.908940949	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0026, seq=14/3584, ttl=5 (request in 174)
176	554.908944871	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=15/3840, ttl=5 (reply in 177)
177	554.908956807	10.0.3.20	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0026, seq=15/3840, ttl=5 (request in 176)
178	554.908961631	10.0.0.10	10.0.3.20	ICMP	74	Echo (ping) request id=0x0026, seq=16/4096, ttl=6 (reply in 179)
179	554.908972942	10.0.3.20	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0026, seq=16/4096, ttl=6 (request in 178)

Figura 1.3: Wireshark

1.1.3 Alínea c

Qual deve ser o valor inicial mínimo do campo TTL para alcançar o destino h5? Verifique na prática que a sua resposta está correta.

Face aos resultados analisados na questão anterior, verifica-se que a partir de TTL = 4 os pacotes deixam de receber mensagem de erro como resposta, logo o valor inicial mínimo para alcançar o destino h5 será 4.

1.1.4 Alínea d

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

O valor médio é obtido calculando a seguinte equação: $RTT = ((0.075 + 0.013 + 0.010)/3 + (0.022 + 0.012 + 0.012)/3 + (0.025 + 0.014 + 0.013)/3 + (0.041 + 0.016 + 0.015)/3) * 2 = 0.178$

1.2 Exercício 2

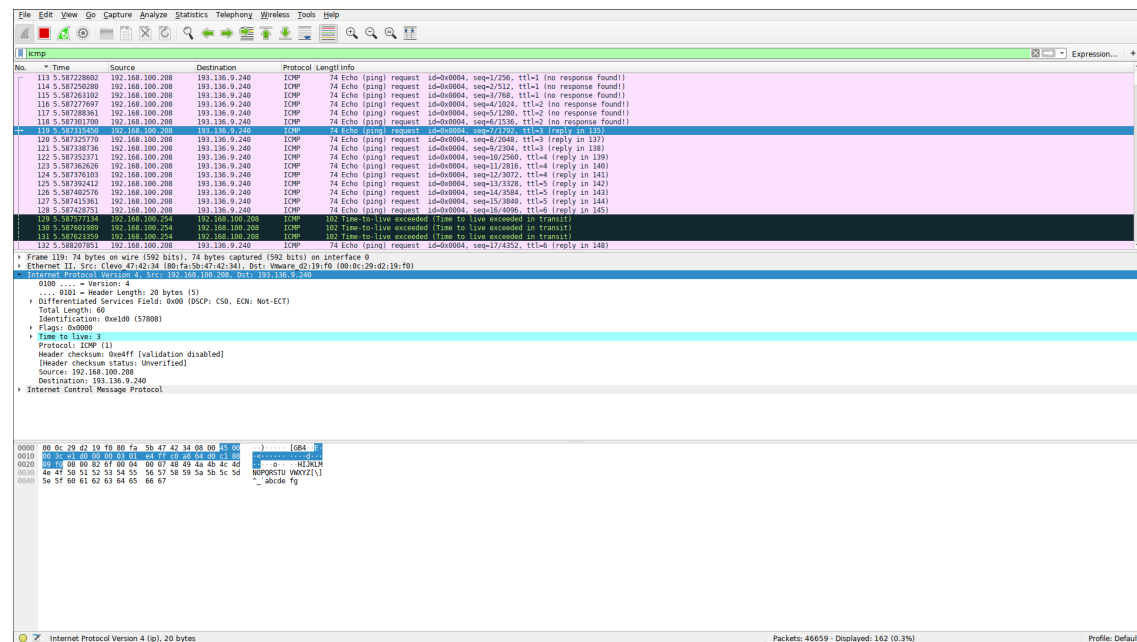


Figura 1.4: Wireshark

1.2.1 Alínea a

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

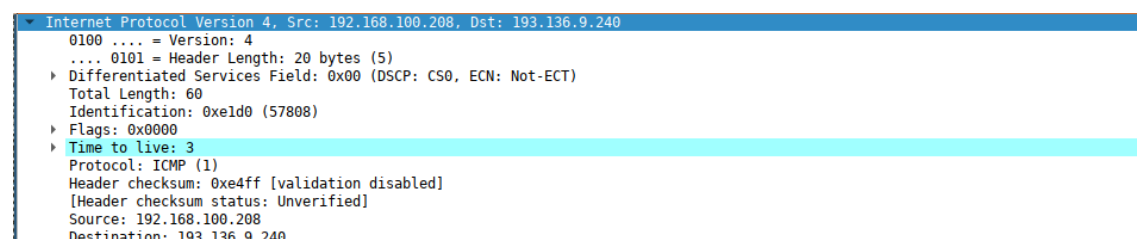


Figura 1.5: Cabeçalho IP

192.168.100.208

1.2.2 Alínea b

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

ICMP (1).

O valor do campo protocolo é 1, ou seja, identifica o protocolo ICMP.

1.2.3 Alínea c

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

Analisando o campo Header length na figura 1.5, conclui-se que o cabeçalho IP tem 20 bytes.

O tamanho do campo de dados (payload) é a diferença entre o número total de bytes e o tamanho do cabeçalho do datagrama. Logo, $Payload = 60 - 20 = 40$ bytes.

1.2.4 Alínea d

O datagrama IP foi fragmentado?

```
▼ Flags: 0x0000
  0.... = Reserved bit: Not set
 .0... = Don't fragment: Not set
 ..0. = More fragments: Not set
...0 0000 0000 0000 = Fragment offset: 0
```

Figura 1.6: Flags do Cabeçalho IP

Observando a figura 1.6 reparamos que no campo Flags, tudo está a 0, logo o *Fragment offset* tem valor 0, o que tendo em atenção agora a flag *More Fragments* concluímos que não existem mais fragmentos pois, se o valor for 1 existem, caso contrário é 0. Logo, se estamos no primeiro fragmento e não existem mais então este é o datagrama original.

1.2.5 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.

Após ordenarmos, concluímos que os campos do cabeçalho IP que variam de pacote para pacote são o *TTL*, *Header Checksum* e o identificador.

113	5.587228602	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=1/256, ttl=1 (no response found!)
114	5.587250280	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=2/512, ttl=1 (no response found!)
115	5.587263102	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=3/768, ttl=1 (no response found!)
116	5.587277697	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=4/1024, ttl=2 (no response found!)
117	5.587288361	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=5/1280, ttl=2 (no response found!)
118	5.587301709	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=6/1536, ttl=2 (no response found!)
119	5.587315450	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=7/1792, ttl=3 (reply in 135)
120	5.587325770	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=8/2048, ttl=3 (reply in 137)
121	5.587338736	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=9/2304, ttl=3 (reply in 138)
122	5.587352371	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=10/2560, ttl=4 (reply in 139)
123	5.587362626	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=11/2816, ttl=4 (reply in 140)
124	5.587376103	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=12/3072, ttl=4 (reply in 141)
125	5.587392412	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=13/3328, ttl=5 (reply in 142)
126	5.587402576	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=14/3584, ttl=5 (reply in 143)
127	5.587415361	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=15/3840, ttl=5 (reply in 144)
128	5.587428751	192.168.100.208	193.136.9.240	ICMP	74 Echo (ping) request	id=0x0004, seq=16/4096, ttl=6 (reply in 145)

Figura 1.7: Tráfego Wireshark ordenado por endereço fonte

1.2.6 Alínea f

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

Ao analisar o datagrama IP, verificamos que este conserva os primeiros 8 bits em todos os casos apresentados e os restantes são incrementados sequencialmente.

Também observamos que o TTL é incrementado sequencialmente

1.2.7 Alínea g

Ordene o tráfego capturado por endereço destino e encontre a série de repostas 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 exceed enviados ao seu host? Porquê?

O valor do campo TTL é 62.

Para todas as mensagens de resposta *ICMP TTL exceeded* recebidas no nosso host esse valor manteve-se constante.

Apesar do TTL observado (pré-definido pelo destino) ser 64, quando o pacote chega ao destino o TTL é de 62. Tal deve-se ao facto do pacote em questão ter passado por 2 routers antes de chegar e, por isso, ter sido decrementado 2 vezes.

129	5.587577134	192.168.100.254	192.168.100.208	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
130	5.587601989	192.168.100.254	192.168.100.208	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
131	5.587623359	192.168.100.254	192.168.100.208	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
135	5.588278204	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=7/1792, ttl=62 (request in 119)
136	5.588300434	193.136.9.240	192.168.100.208	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
137	5.588323420	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=8/2048, ttl=62 (request in 120)
138	5.588359878	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=9/2304, ttl=62 (request in 121)
139	5.588389816	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=10/2560, ttl=62 (request in 122)
140	5.588413102	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=11/2816, ttl=62 (request in 123)
141	5.588437721	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=12/3072, ttl=62 (request in 124)
142	5.588497187	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=13/3328, ttl=62 (request in 125)
143	5.588517888	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=14/3584, ttl=62 (request in 126)
144	5.588538867	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=15/3840, ttl=62 (request in 127)
145	5.588607440	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=16/4096, ttl=62 (request in 128)
146	5.588629822	193.136.9.240	192.168.100.208	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
147	5.588797736	193.136.9.240	192.168.100.208	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
148	5.588895070	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=17/4352, ttl=62 (request in 132)
149	5.588960919	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=18/4608, ttl=62 (request in 133)
150	5.588984539	193.136.9.240	192.168.100.208	ICMP	74 Echo (ping) reply id=0x0004, seq=19/4864, ttl=62 (request in 134)

Figura 1.8: Tráfego Wireshark ordenado por endereço de destino

1.3 Exercício 3

214	20.369379994	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d73) [Reassembled in #216]
215	20.369382681	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d73) [Reassembled in #216]
216	20.369383290	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=1/256, ttl=1 (no response found!)
217	20.369387633	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d74) [Reassembled in #219]
218	20.369388412	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d74) [Reassembled in #219]
219	20.369412752	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=2/512, ttl=1 (no response found!)
220	20.369413977	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d75) [Reassembled in #222]
221	20.369414480	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d75) [Reassembled in #222]
222	20.369599300	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=3/768, ttl=1 (no response found!)
223	20.369881973	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d76) [Reassembled in #225]
224	20.369882975	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d76) [Reassembled in #225]
225	20.370065798	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=4/1024, ttl=2 (no response found!)
226	20.370107504	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d77) [Reassembled in #229]
227	20.370291021	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d77) [Reassembled in #229]
228	20.370406029	192.168.100.254	192.168.100.208	ICMP	590 Time-to-live exceeded (Time to live exceeded in transit)
229	20.370423065	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=5/1280, ttl=2 (no response found!)
230	20.370495057	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d78) [Reassembled in #233]
231	20.370580891	192.168.100.254	192.168.100.208	ICMP	590 Time-to-live exceeded (Time to live exceeded in transit)
232	20.370586895	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d78) [Reassembled in #233]
233	20.370725534	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=6/1536, ttl=2 (no response found!)
234	20.370869689	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d79) [Reassembled in #237]
235	20.370889308	192.168.100.254	192.168.100.208	ICMP	590 Time-to-live exceeded (Time to live exceeded in transit)
236	20.370928410	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d79) [Reassembled in #237]
237	20.371048114	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=7/1792, ttl=3 (reply in 273)
238	20.371158980	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d7a) [Reassembled in #240]
239	20.371277023	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d7a) [Reassembled in #240]
240	20.371396287	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=8/2048, ttl=3 (reply in 279)
241	20.371562016	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d7b) [Reassembled in #243]
242	20.371678427	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d7b) [Reassembled in #243]
243	20.371797435	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=9/2304, ttl=3 (reply in 286)
244	20.371852034	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d7c) [Reassembled in #246]
245	20.371976918	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d7c) [Reassembled in #246]
246	20.372115902	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=10/2560, ttl=4 (reply in 292)
247	20.372261104	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d7d) [Reassembled in #250]
248	20.372282431	193.136.9.254	192.168.100.208	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
249	20.372324071	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d7d) [Reassembled in #250]
250	20.372443354	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=11/2816, ttl=4 (reply in 297)
251	20.372550036	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=5d7e) [Reassembled in #254]
252	20.372623875	193.136.9.254	192.168.100.208	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
253	20.372672507	192.168.100.208	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=5d7e) [Reassembled in #254]
254	20.372794424	192.168.100.208	193.136.9.240	ICMP	1261 Echo (ping) request id=0x0005, seq=12/3072, ttl=4 (reply in 300)

Figura 1.9: Fragmentos do datagrama IP

1.3.1 Alínea a

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

Analizando a figura, vemos que a primeira mensagem ICMP é a 214.

Visto que o tamanho permitido pelo protocolo é inferior ao tamanho do PDU é necessário que o pacote inicial seja fragmentado para poder circular na rede.

1.3.2 Alínea b

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

```

> Frame 214: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
> Ethernet II, Src: Clevo_47:42:34 (80:fa:5b:47:42:34), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
▼ Internet Protocol Version 4, Src: 192.168.100.208, 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: 0x5d73 (23923)
    ▼ Flags: 0x2000, More fragments
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..1. .... = More fragments: Set
        ...0 0000 0000 0000 = Fragment offset: 0
    > Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x45bd [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.100.208
    Destination: 193.136.9.240
    Reassembled IPv4 in frame: 216
> Data (1480 bytes)
```

Figura 1.10: Cabeçalho do primeiro fragmento do datagrama IP

Podemos observar na figura que, no campo Flags, o *More fragments* tem valor 1, isso indica que o datagrama foi fragmentado, existindo então mais fragmentos.

Seguidamente, podemos ver que o *Fragment offset* é 0, provando de que se trata do primeiro fragmento.

A *Total Length* é igual a 1500 bytes.

1.3.3 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 215: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▶ Ethernet II, Src: Clevo_47:42:34 (80:fa:5b:47:42:34), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
▼ Internet Protocol Version 4, Src: 192.168.100.208, 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: 0x5d73 (23923)
    ▼ Flags: 0x20b9, More fragments
      0... .. = Reserved bit: Not set
      .0... .. = Don't fragment: Not set
      ..1... .. = More fragments: Set
      ...0 0101 1100 1000 = Fragment offset: 1480
    ▶ Time to live: 1
      Protocol: ICMP (1)
      Header checksum: 0x4504 [validation disabled]
      [Header checksum status: Unverified]
      Source: 192.168.100.208
      Destination: 193.136.9.240
      Reassembled IPv4 in frame: 216
    ▶ Data (1480 bytes)
```

Figura 1.11: Cabeçalho do segundo fragmento do datagrama IP

Tal como já foi referido anteriormente, para se verificar se um fragmento é o primeiro basta ter em atenção o valor que está no *Fragment offset*. Se esse valor for 0 então podemos concluir que se trata do primeiro. Como o valor apresentado é diferente de 0 pudemos concluir que o fragmento em questão não se trata do primeiro.

Podemos concluir que existem mais fragmentos pois o bit correspondente ao **More fragments** é igual a 1.

1.3.4 Alínea d

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

Como está mostrado na imagem, o terceiro fragmento do datagrama original tem o bit correspondente a *More fragments* a 0, ou seja, não há mais fragmentos a seguir a este. Concluindo assim que foram criados 3 fragmentos (214, 215 e 216).

1.3.5 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.

Ao longo dos diferentes fragmentos, os campos do *Fragment offset* e do *More Fragments* são alterados no cabeçalho IP.

O primeiro permite identificar a posição do fragmento no datagrama original. O segundo indica se existem mais fragmentos do datagrama original para além do próprio.

Capítulo 2

Parte 2

2.1 Exercício 1

2.1.1 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.

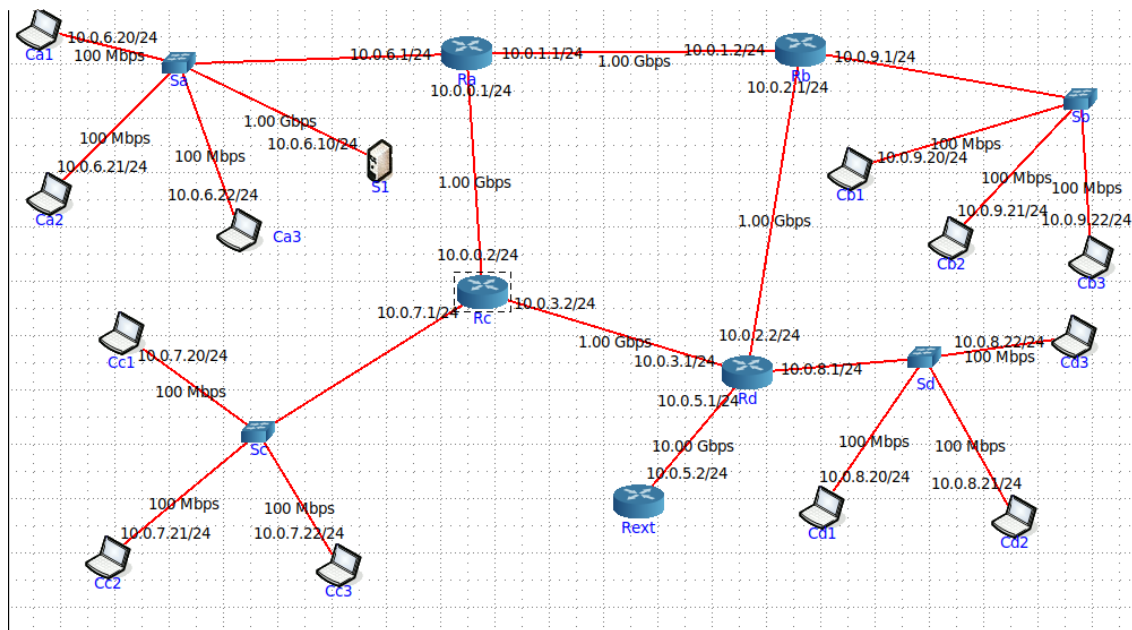


Figura 2.1: Topologia Core

Na figura 2.1, é possível verificar os endereços atribuídos a cada equipamento.

2.1.2 Alínea b

Trata-se de endereços públicos ou privados? Porquê?

Uma vez que todos os endereços utilizam um dos blocos reservados a endereços privados: "10.0.0.0 - 10.255.255.255 / 8", concluímos que se tratam de endereços privados.

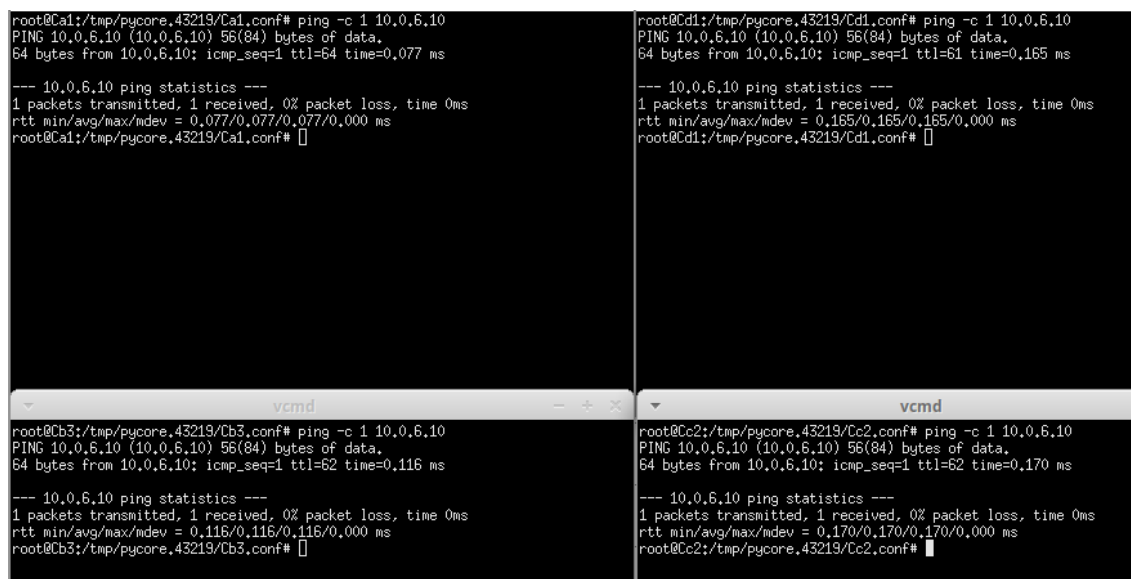
2.1.3 Alínea c

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

Não é necessário a atribuição de endereços IP aos switches porque são intervenientes na camada de ligação 2 e por sua vez transparentes à camada de ligação 3. Estes encaminham os pacotes apenas tendo em atenção os endereços MAC dos equipamentos.

2.1.4 Alínea d

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



```
root@Ca1:/tmp/pycore.43219/Ca1.conf# ping -c 1 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=0.077 ms

--- 10.0.6.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.077/0.077/0.077/0.000 ms
root@Ca1:/tmp/pycore.43219/Ca1.conf#

root@Cd1:/tmp/pycore.43219/Cd1.conf# ping -c 1 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=0.165 ms

--- 10.0.6.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.165/0.165/0.165/0.000 ms
root@Cd1:/tmp/pycore.43219/Cd1.conf#

root@Cb3:/tmp/pycore.43219/Cb3.conf# ping -c 1 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=0.116 ms

--- 10.0.6.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.116/0.116/0.116/0.000 ms
root@Cb3:/tmp/pycore.43219/Cb3.conf#

root@Cc2:/tmp/pycore.43219/Cc2.conf# ping -c 1 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=0.170 ms

--- 10.0.6.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.170/0.170/0.170/0.000 ms
root@Cc2:/tmp/pycore.43219/Cc2.conf#
```

Figura 2.2: Conectividade entre os laptops de cada departamento e o servidor

Como podemos observar pela figura 2.2, para verificar se existia conectividade foi utilizado o comando ping em pelo menos um laptop de cada departamento. Sendo que todos obtiveram resposta do servidor após terem enviado pacotes, concluímos que existe conectividade em todos os departamentos.

2.1.5 Alínea e

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

```

root@Rext:/tmp/pycore.43219/Rext.conf# ping -c 1 10.0.6.1
PING 10.0.6.1 (10.0.6.1) 56(84) bytes of data:
64 bytes from 10.0.6.1: icmp_seq=1 ttl=62 time=0.142 ms

--- 10.0.6.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdew = 0.142/0.142/0.142/0.000 ms
root@Rext:/tmp/pycore.43219/Rext.conf# █

```

Figura 2.3: Conectividade S1 - Router Rext

Observando a figura 2.3 e seguindo o raciocínio da alínea anterior, verificamos que existe conectividade do router Rext para o servidor S1.

2.2 Exercício 2

2.2.1 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@Cb2:/tmp/pycore.37745/Cb2.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.9.1 0.0.0.0 UG 0 0 0 eth0
10.0.9.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@Cb2:/tmp/pycore.37745/Cb2.conf# █

```

Figura 2.4: Tabela de encaminhamento de um laptop do Departamento B

```

vcmd
root@Rb:/tmp/pycore.37745/Rb.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 10.0.1.1 255.255.255.0 UG 0 0 0 eth0
10.0.1.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
10.0.3.0 10.0.2.2 255.255.255.0 UG 0 0 0 eth1
10.0.4.0 10.0.1.1 255.255.255.0 UG 0 0 0 eth0
10.0.5.0 10.0.2.2 255.255.255.0 UG 0 0 0 eth1
10.0.6.0 10.0.1.1 255.255.255.0 UG 0 0 0 eth0
10.0.7.0 10.0.1.1 255.255.255.0 UG 0 0 0 eth0
10.0.8.0 10.0.2.2 255.255.255.0 UG 0 0 0 eth1
10.0.9.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2

```

Figura 2.5: Tabela de encaminhamento do router Departamento B

Na figura 2.4 está representada a tabela de encaminhamento de onde podemos retirar informações relativa à rota que irá ser feita pelo pacote. Na coluna "Destination" é nos indicado a sub-rede

destino, na "Gateway" a informação do equipamento pelo qual irá passar o pacote e na "Genmask" o tipo da máscara.

Analisando a primeira e a segunda entrada da figura 2.4 reparamos que as duas diferem no Gateway. Sendo que os dois endereços estão ligados diretamente entre si, o Gateway não precisa de ser definido. Caso contrário, seria necessário para se saber o próximo salto.

Relativamente à coluna Flags, estas apenas servem para acrescentar informações adicionais. A flag "UG" é utilizada quando o gateway está definido e a "U" caso contrário. Sendo que a primeira tabela e a segunda tabela são idênticas, não é necessário fazer uma análise de todas as entradas da segunda.

2.2.2 Alínea b

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

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
157	root	20	0	34932	4752	3548	R	0.0	0.2	0:00.03	htop
74	quagga	20	0	29808	2856	2000	S	0.0	0.1	0:00.11	/usr/sbin/ospfd -d
59	quagga	20	0	27548	3212	2216	S	0.0	0.2	0:00.00	/usr/sbin/zebra -d
70	quagga	20	0	27336	2800	2036	S	0.0	0.1	0:00.01	/usr/sbin/ospf6d -d
82	root	20	0	29044	4032	3548	S	0.0	0.2	0:00.01	/bin/bash

Figura 2.6: Processos a serem executados em Ra

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
17	root	20	0	29044	3944	3456	S	0.0	0.2	0:00.01	/bin/bash
29	root	20	0	34184	4108	3644	R	0.0	0.2	0:00.00	htop

Figura 2.7: Processos a serem executados em Ca1

Se apenas analisássemos a figura 2.6 reparávamos que é usado pelo router o protocolo ospfd (este permite que o pacote siga diferentes caminhos quando um não é possível) e assim concluíamos que se tratava de um encaminhamento dinâmico. No entanto, após analisar a figura 2.7 verificamos que não é usado esse protocolo, concluindo então que se trata de um encaminhamento estático.

2.2.3 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 S1 localizado no departamento A. Use o comando `route delete` para o efeito. Que implicação tem esta medida para os utilizadores da empresa que acedem ao servidor? Justifique.

```

root@S1:/tmp/pycore.43219/S1.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0          10.0.6.1        0.0.0.0         UG      0 0        0 eth0
10.0.6.0         0.0.0.0         255.255.255.0   U        0 0        0 eth0
root@S1:/tmp/pycore.43219/S1.conf# route del default
root@S1:/tmp/pycore.43219/S1.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.6.0         0.0.0.0         255.255.255.0   U        0 0        0 eth0
root@S1:/tmp/pycore.43219/S1.conf# █

```

Figura 2.8: Remoção da rota default

```

root@Rext:/tmp/pycore.43219/Rext.conf# ping -c 1 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data.

--- 10.0.6.10 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

```

Figura 2.9: Conectividade do Rext para S1

```

root@S1:/tmp/pycore.43219/S1.conf# ping -c 1 10.0.5.2
connect: Network is unreachable

```

Figura 2.10: Conectividade do S1 para Rext

Removendo a rota por defeito é perdida a conectividade entre o servidor S1 e os restantes hosts existentes fora do departamento onde se encontra. Isto acontece porque o servidor S1 não tendo definida a rota de envio de tráfego para redes não locais faz com que não saiba para onde enviar de volta o que recebeu dos utilizadores. Tal verifica-se nas figuras 2.9 e 2.10 em que o router consegue enviar para o servidor pacotes mas não os recebe de volta e este não consegue enviar pacotes para o router.

2.2.4 Alínea d

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

```

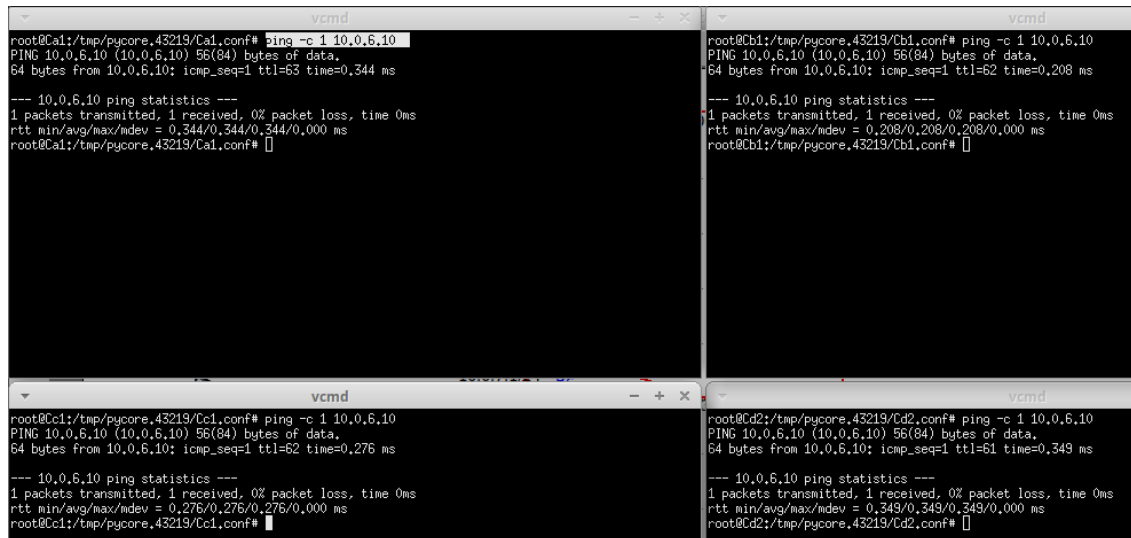
root@S1:/tmp/pycore.43219/S1.conf# route add -net 10.0.0.0 netmask 255.255.255.0 gw 10.0.6.1
root@S1:/tmp/pycore.43219/S1.conf# route add -net 10.0.1.0 netmask 255.255.255.0 gw 10.0.6.1
root@S1:/tmp/pycore.43219/S1.conf# route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.6.1
root@S1:/tmp/pycore.43219/S1.conf# route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.6.1
root@S1:/tmp/pycore.43219/S1.conf# route add -net 10.0.6.0 netmask 255.255.255.0 gw 10.0.6.1
root@S1:/tmp/pycore.43219/S1.conf# route add -net 10.0.7.0 netmask 255.255.255.0 gw 10.0.6.1
root@S1:/tmp/pycore.43219/S1.conf# route add -net 10.0.8.0 netmask 255.255.255.0 gw 10.0.6.1
root@S1:/tmp/pycore.43219/S1.conf# route add -net 10.0.9.0 netmask 255.255.255.0 gw 10.0.6.1
root@S1:/tmp/pycore.43219/S1.conf# █

```

Figura 2.11: Restauração da conectividade adicionando rotas estáticas

2.2.5 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.



The image shows four terminal windows, each representing a different department (Ca1, Cb1, Cc1, Cd2) performing a ping test to the server S1 (10.0.6.10). Each window displays the command used, the raw ping output, and a summary of the ping statistics.

```
root@Ca1:/tmp/pycore.43219/Ca1.conf# ping -c 1 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=63 time=0.344 ms

--- 10.0.6.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.344/0.344/0.344/0.000 ms
root@Ca1:/tmp/pycore.43219/Ca1.conf#

root@Cb1:/tmp/pycore.43219/Cb1.conf# ping -c 1 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=0.208 ms

--- 10.0.6.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.208/0.208/0.208/0.000 ms
root@Cb1:/tmp/pycore.43219/Cb1.conf#

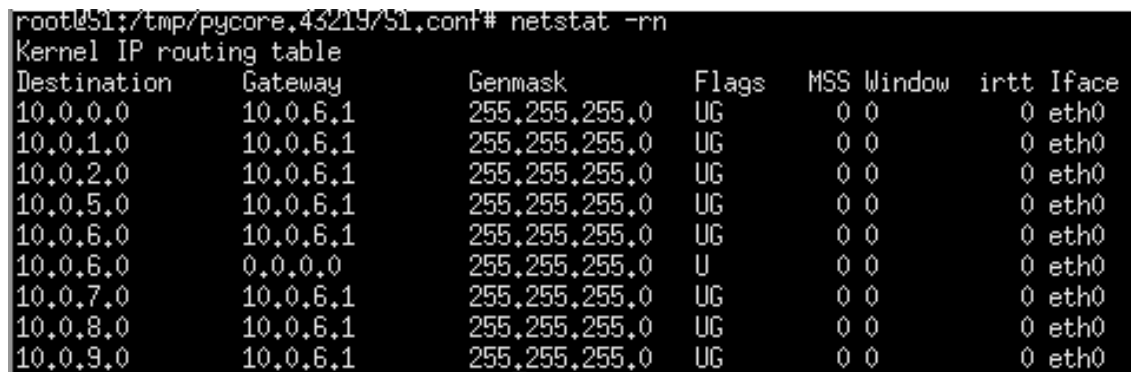
root@Cc1:/tmp/pycore.43219/Cc1.conf# ping -c 1 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data:
64 bytes from 10.0.6.10: icmp_seq=1 ttl=62 time=0.276 ms

--- 10.0.6.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.276/0.276/0.276/0.000 ms
root@Cc1:/tmp/pycore.43219/Cc1.conf#

root@Cd2:/tmp/pycore.43219/Cd2.conf# ping -c 1 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=0.349 ms

--- 10.0.6.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.349/0.349/0.349/0.000 ms
root@Cd2:/tmp/pycore.43219/Cd2.conf#
```

Figura 2.12: Conectividade entre os departamentos e S1



The image shows a terminal window on server S1 displaying the output of the 'netstat -rn' command, which shows the kernel IP routing table.

```
root@S1:/tmp/pycore.43219/S1.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask         Flags        MSS Window  irtt Iface
10.0.0.0          10.0.6.1         255.255.255.0   UG           0 0        0 eth0
10.0.1.0          10.0.6.1         255.255.255.0   UG           0 0        0 eth0
10.0.2.0          10.0.6.1         255.255.255.0   UG           0 0        0 eth0
10.0.5.0          10.0.6.1         255.255.255.0   UG           0 0        0 eth0
10.0.6.0          10.0.6.1         255.255.255.0   UG           0 0        0 eth0
10.0.6.0          0.0.0.0          255.255.255.0   U            0 0        0 eth0
10.0.7.0          10.0.6.1         255.255.255.0   UG           0 0        0 eth0
10.0.8.0          10.0.6.1         255.255.255.0   UG           0 0        0 eth0
10.0.9.0          10.0.6.1         255.255.255.0   UG           0 0        0 eth0
```

Figura 2.13: Tabela de encaminhamento

Analisando as figuras 2.12 e 2.13 podemos verificar que existe conectividade entre o servidor S1 e os laptops de todos os departamentos.

2.3 Exercício 3

2.3.1 Alínea 1

Considere que dispõe apenas do endereço de rede IP 172.yyx.32.0/20, em que “yy” são os dígitos correspondendo ao seu número de grupo (Gyy) e “x” é o dígito correspondente ao seu turno prático (PLx). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo a rede de acesso e core inalteradas) e atribua endereços às interfaces dos vários sistemas envolvidos. Deve justificar as opções usadas.

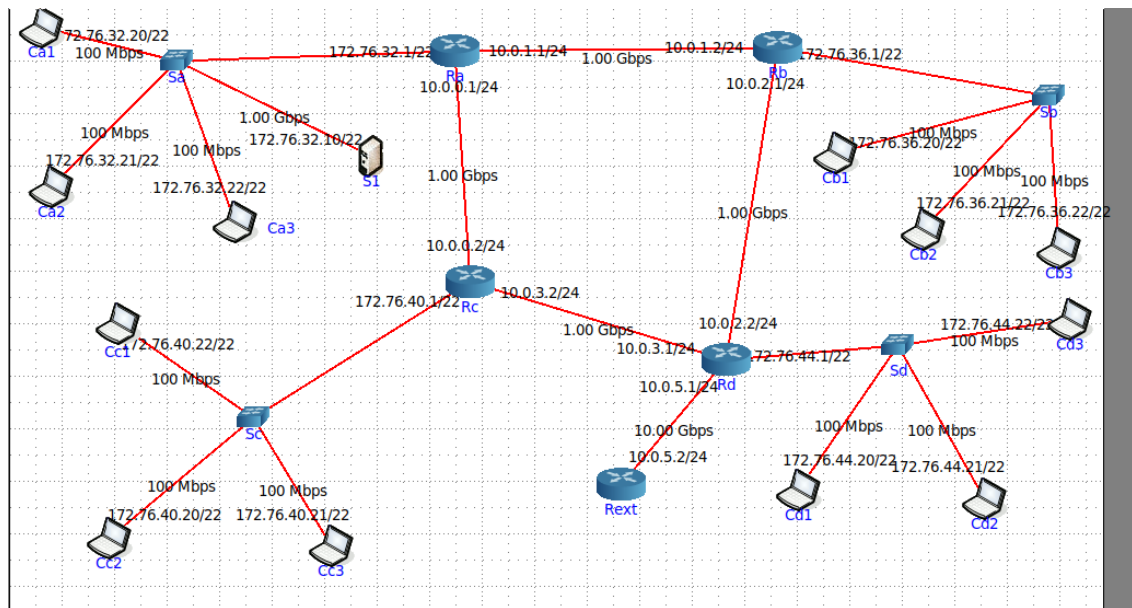


Figura 2.14: Topologia do novo endereçamento

O nosso grupo é o 7 e pertencemos ao PL6, logo o nosso endereço IP é 172.76.32.0/20. Sendo que existem 4 departamentos então vão ser necessárias 4 sub-redes e vai ser preciso usar 22 bits para suportar a topologia atual. Assim, para cada departamento foram atribuídas as sub-redes:

- Departamento A: 172.76.32.0/22
- Departamento B: 172.76.36.0/22
- Departamento C: 172.76.40.0/22
- Departamento D: 172.76.44.0/22

2.3.2 Alínea 2

Qual a máscara de rede que usou (em notação decimal)? Quantos interfaces IP pode interligar em cada departamento? Justifique.

A máscara de rede que decidimos utilizar foi /22 o que corresponde a 255.255.252.0. Como a máscara é /22 então temos 10 bits disponíveis para hosts, o que significa que temos $2^{10} - 2$ hosts para cada departamento, ou seja, 1022 hosts.

2.3.3 Alínea 3

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


```

root@Ca1:/tmp/pycore.43219/Ca1.conf# ping -c 1 172.76.36.20
PING 172.76.36.20 (172.76.36.20) 56(84) bytes of data.
64 bytes from 172.76.36.20: icmp_seq=1 ttl=62 time=0.206 ms

--- 172.76.36.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.206/0.206/0.206/0.000 ms
root@Ca1:/tmp/pycore.43219/Ca1.conf# ping -c 1 172.76.40.20
PING 172.76.40.20 (172.76.40.20) 56(84) bytes of data.
64 bytes from 172.76.40.20: icmp_seq=1 ttl=62 time=0.206 ms

--- 172.76.40.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.206/0.206/0.206/0.000 ms
root@Ca1:/tmp/pycore.43219/Ca1.conf# ping -c 1 172.76.44.20
PING 172.76.44.20 (172.76.44.20) 56(84) bytes of data.
64 bytes from 172.76.44.20: icmp_seq=1 ttl=61 time=0.339 ms

--- 172.76.44.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.339/0.339/0.339/0.000 ms
root@Ca1:/tmp/pycore.43219/Ca1.conf# ping -c 1 10.0.5.2
PING 10.0.5.2 (10.0.5.2) 56(84) bytes of data.
64 bytes from 10.0.5.2: icmp_seq=1 ttl=61 time=0.190 ms

--- 10.0.5.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.190/0.190/0.190/0.000 ms
root@Ca1:/tmp/pycore.43219/Ca1.conf# █

```

Figura 2.15: Conectividade a partir do Departamento A

```

root@Cb1:/tmp/pycore.43219/Cb1.conf# ping -c 1 172.76.32.20
PING 172.76.32.20 (172.76.32.20) 56(84) bytes of data.
64 bytes from 172.76.32.20: icmp_seq=1 ttl=62 time=0.133 ms

--- 172.76.32.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.133/0.133/0.133/0.000 ms
root@Cb1:/tmp/pycore.43219/Cb1.conf# ping -c 1 172.76.40.20
PING 172.76.40.20 (172.76.40.20) 56(84) bytes of data.
64 bytes from 172.76.40.20: icmp_seq=1 ttl=61 time=0.184 ms

--- 172.76.40.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.184/0.184/0.184/0.000 ms
root@Cb1:/tmp/pycore.43219/Cb1.conf# ping -c 1 172.76.44.20
PING 172.76.44.20 (172.76.44.20) 56(84) bytes of data.
64 bytes from 172.76.44.20: icmp_seq=1 ttl=62 time=0.122 ms

--- 172.76.44.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.122/0.122/0.122/0.000 ms
root@Cb1:/tmp/pycore.43219/Cb1.conf# ping -c 1 10.0.5.2
PING 10.0.5.2 (10.0.5.2) 56(84) bytes of data.
64 bytes from 10.0.5.2: icmp_seq=1 ttl=62 time=0.127 ms

--- 10.0.5.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.127/0.127/0.127/0.000 ms
root@Cb1:/tmp/pycore.43219/Cb1.conf# █

```

Figura 2.16: Conectividade a partir do Departamento B

```

root@Cc2:/tmp/pycore.43219/Cc2.conf# ping -c 1 172.76.32.20
PING 172.76.32.20 (172.76.32.20) 56(84) bytes of data.
64 bytes from 172.76.32.20: icmp_seq=1 ttl=62 time=0.158 ms

--- 172.76.32.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.158/0.158/0.158/0.000 ms
root@Cc2:/tmp/pycore.43219/Cc2.conf# ping -c 1 172.76.36.20
PING 172.76.36.20 (172.76.36.20) 56(84) bytes of data.
64 bytes from 172.76.36.20: icmp_seq=1 ttl=61 time=0.147 ms

--- 172.76.36.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.147/0.147/0.147/0.000 ms
root@Cc2:/tmp/pycore.43219/Cc2.conf# ping -c 1 172.76.44.20
PING 172.76.44.20 (172.76.44.20) 56(84) bytes of data.
64 bytes from 172.76.44.20: icmp_seq=1 ttl=62 time=0.184 ms

--- 172.76.44.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.184/0.184/0.184/0.000 ms
root@Cc2:/tmp/pycore.43219/Cc2.conf# ping -c 1 10.0.5.2
PING 10.0.5.2 (10.0.5.2) 56(84) bytes of data.
64 bytes from 10.0.5.2: icmp_seq=1 ttl=62 time=0.267 ms

--- 10.0.5.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.267/0.267/0.267/0.000 ms
root@Cc2:/tmp/pycore.43219/Cc2.conf# █

```

Figura 2.17: Conectividade a partir do Departamento C

```

root@Cd1:/tmp/pycore.43219/Cd1.conf# ping -c 1 172.76.32.20
PING 172.76.32.20 (172.76.32.20) 56(84) bytes of data.
64 bytes from 172.76.32.20: icmp_seq=1 ttl=61 time=0.144 ms

--- 172.76.32.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.144/0.144/0.144/0.000 ms
root@Cd1:/tmp/pycore.43219/Cd1.conf# ping -c 1 172.76.36.20
PING 172.76.36.20 (172.76.36.20) 56(84) bytes of data.
64 bytes from 172.76.36.20: icmp_seq=1 ttl=62 time=0.222 ms

--- 172.76.36.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.222/0.222/0.222/0.000 ms
root@Cd1:/tmp/pycore.43219/Cd1.conf# ping -c 1 172.76.40.20
PING 172.76.40.20 (172.76.40.20) 56(84) bytes of data.
64 bytes from 172.76.40.20: icmp_seq=1 ttl=62 time=0.117 ms

--- 172.76.40.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.117/0.117/0.117/0.000 ms
root@Cd1:/tmp/pycore.43219/Cd1.conf# ping -c 1 10.0.5.2
PING 10.0.5.2 (10.0.5.2) 56(84) bytes of data.
64 bytes from 10.0.5.2: icmp_seq=1 ttl=63 time=0.107 ms

--- 10.0.5.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.107/0.107/0.107/0.000 ms
root@Cd1:/tmp/pycore.43219/Cd1.conf# █

```

Figura 2.18: Conectividade a partir do Departamento D

Analisando as figuras 2.15, 2.16, 2.17 e 2.18 reparamos que existe conectividade entre todos os departamentos.

Capítulo 3

Conclusão

Neste trabalho prático foi nos dada a oportunidade de enriquecer o nosso conhecimento relativamente a redes. Para isso foram usadas duas ferramentas: Core, para simulação de redes, e Wireshark, para captura de tráfego.

O trabalho está dividido em duas partes e cada uma delas dividida em 3 questões. Cada parte teve como foco um tema e o mesmo acontece para cada questão facilitando a nossa percepção em detalhes que mesmo não parecendo são relevantes.

Na primeira parte, o objetivo principal foi analisar o IP (Internet Protocol) e para isso analisamos o formato dos pacotes/datagramas IP e fragmentação dos mesmos.

A segunda parte tinha como foco o estudo e percepção do endereçamento e encaminhamento do que foi estudado na parte anterior. Construímos topologias na tentativa de simular o máximo do que se passa na realidade e nas mesmas remover, alterar e adicionar endereços para compreender como são feitos os transportes de pacotes numa rede.

Resumindo, consideramos que conseguimos atingir com sucesso todos os desafios apresentados no enunciado obtendo assim um conhecimento mais avançado sobre o protocolo IPv4 e *sub-netting*.