



Redes de Computadores 2019/2020

TP2: Protocolo IPv4 Grupo 05



Ana Afonso
A85762



João Diogo Mota
A80791



Márcia Teixeira
A80943

1. Parte I: Datagramas IP e Fragmentação

1. Questões e Respostas

Questão 1

Prepare uma topologia no CORE para verificar o comportamento do traceroute. Ligue um host (servidor) s1 a um router r2; o router r2 a um router r3, o router r3 a um router r4, que por sua vez, se liga a um host (pc) h5. (Note que pode não existir conectividade IP imediata entre s1 e h5 até que o routing estabilize). Ajuste o nome dos equipamentos atribuídos por defeito para a topologia do enunciado.

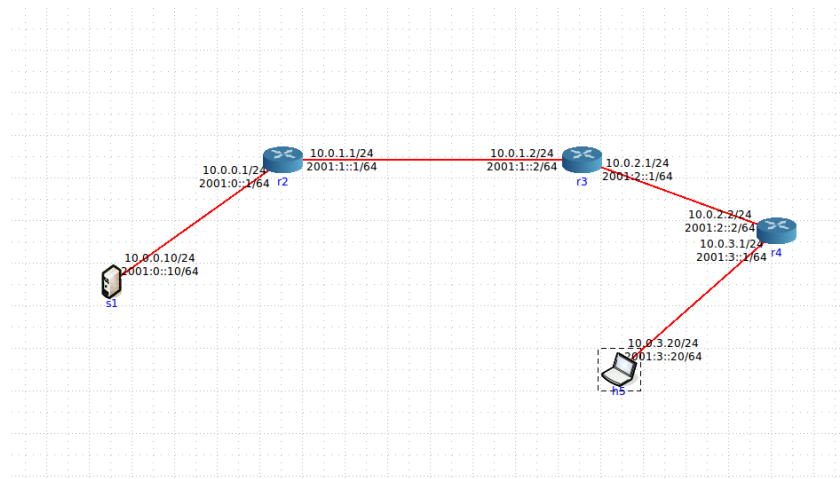


Figura 1: Implementação da topologia no CORE

- a. Ative o wireshark ou o tcpdump no pc s1. Numa shell de s1, execute o comando traceroute -I para o endereço IP do host h5.

```
root@s1:/tmp/pycore.33223/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  A0 (10.0.0.1) 0.045 ms 0.007 ms 0.007 ms
 2  10.0.1.2 (10.0.1.2) 0.021 ms 0.012 ms 0.011 ms
 3  * * *
 4  10.0.3.20 (10.0.3.20) 0.033 ms 0.019 ms 0.020 ms
root@s1:/tmp/pycore.33223/s1.conf#
```

Figura 2: Execução do comando "traceroute -I" para o endereço IP do host h5



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

Através do comando *traceroute -I*, espera-se que sejam enviados um ou mais datagramas com o campo TTL (*Time to live*) = 1. Posteriormente, prevê-se que o mesmo suceda para valores incrementais do TTL na tentativa de obter a identificação dos routers durante o percurso até ao *host* h5.

Com a análise do tráfego ICMP enviado pelo servidor s1, foram enviados três datagramas com a mensagem “*Echo Request*” com TTL = 1, recebendo como resposta a mensagem “*Time To Live Exceeded*”, o que indica que o servidor não conseguiu que o datagrama chegasse até ao host. Seguidamente, este processo repetiu-se para TTL = 2 e para TTL = 3, sendo que neste último, não foi obtido um “*Echo Reply*” para os respetivos “*Echo Request*”, sendo este ignorado por aconselhamento do docente. Por fim, a partir de TTL = 4, não é recebida a mensagem de controlo “*Time To Live Exceeded*”.

No.	Time	Source	Destination	Protocol	Length	Info
56	158.530567	00:00:00_aa:00:01	00:00:00_aa:00:00	ARP	42	10.0.0.1 is at 00:00:00_aa:00:01
57	158.530570	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=1/256, ttl=1
58	158.530583	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
59	158.530590	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=2/512, ttl=1
60	158.530595	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
61	158.530599	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=3/768, ttl=1
62	158.530604	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
63	158.530608	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=4/1024, ttl=2
64	158.530627	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
65	158.530633	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=5/1280, ttl=2
66	158.530642	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
67	158.530646	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=6/1536, ttl=2
68	158.530655	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
69	158.530660	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=7/1792, ttl=3
70	158.530685	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=8/2048, ttl=3
71	158.530695	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=9/2304, ttl=3
72	158.530704	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=10/2560, ttl=4
73	158.530736	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0055, seq=10/2560, ttl=61
74	158.530741	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=11/2816, ttl=4
75	158.530759	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0055, seq=11/2816, ttl=61
76	158.530763	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0055, seq=12/3072, ttl=4
77	158.530780	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0055, seq=12/3072, ttl=61

Figura 3: Tráfego ICMP (1/2)

No.	Time	Source	Destination	Protocol	Length	Info
702	2558.37454	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
703	2558.37454	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=6/1536, ttl=2
704	2558.37455	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
705	2558.37456	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=7/1792, ttl=3
706	2558.37457	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=8/2048, ttl=3
707	2558.37458	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=9/2304, ttl=3
708	2558.37459	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=10/2560, ttl=4
709	2558.37461	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x005b, seq=10/2560, ttl=61
710	2558.37462	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=11/2816, ttl=4
711	2558.37463	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x005b, seq=11/2816, ttl=61
712	2558.37464	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=12/3072, ttl=4
713	2558.37465	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x005b, seq=12/3072, ttl=61
714	2558.37466	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=13/3328, ttl=5
715	2558.37468	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x005b, seq=13/3328, ttl=61
716	2558.37468	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=14/3584, ttl=5
717	2558.37470	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x005b, seq=14/3584, ttl=61
718	2558.37470	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=15/3840, ttl=5
719	2558.37472	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x005b, seq=15/3840, ttl=61
720	2558.37472	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=16/4096, ttl=6
721	2558.37474	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x005b, seq=16/4096, ttl=61
722	2558.37491	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x005b, seq=17/4352, ttl=6
723	2558.37493	10.0.0.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x005b, seq=17/4352, ttl=61
724	2563.38390	00:00:00_aa:00:00	00:00:00_aa:00:01	ARP	42	who has 10.0.0.12 Tell 10.0.0.10

Figura 4: Tráfego ICMP (2/2)



- 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.

O valor inicial mínimo do campo TTL para que seja possível alcançar o destino h5 deve ser $TTL = 4$, o que se verifica uma vez que, a partir deste valor de TTL, são recebidas mensagens de controlo “*Echo Reply*”.

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

O valor médio de ida-e-volta obtido foi de 0.024ms, sendo este alcançado a partir do cálculo da média aritmética entre os três valores obtidos.

```
root@s1:/tmp/pycore.33223/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  A0 (10.0.0.1)  0.045 ms  0.007 ms  0.007 ms
 2  10.0.1.2 (10.0.1.2)  0.021 ms  0.012 ms  0.011 ms
 3  * * *
 4  10.0.3.20 (10.0.3.20)  0.033 ms  0.019 ms  0.020 ms
root@s1:/tmp/pycore.33223/s1.conf#
```

Figura 5: Round-Trip-Time

Questão 2

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

```
core@XubunCORE:~$ sudo traceroute -I marco.uminho.pt
[sudo] password for core:
traceroute to marco.uminho.pt (193.136.9.240), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.208 ms  0.180 ms *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * marco.uminho.pt (193.136.9.240)  1.831 ms  1.626 ms
core@XubunCORE:~$
```

Figura 6: Comando “traceroute” para a máquina nativa



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

O endereço IP da interface ativa do computador usado é 10.0.2.15

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.2.15	192.168.100.254	DNS	75	Standard query AAAA marco.uminho.pt
2	0.002067	192.168.100.254	10.0.2.15	DNS	129	Standard query response
3	0.002205	10.0.2.15	192.168.100.254	DNS	91	Standard query AAAA marco.uminho.pt.sa.di.uminho.pt
4	0.003737	192.168.100.254	10.0.2.15	DNS	136	Standard query response, No such name
5	0.003971	10.0.2.15	192.168.100.254	DNS	75	Standard query A marco.uminho.pt
6	0.005573	192.168.100.254	10.0.2.15	DNS	319	Standard query response A 193.136.9.240
7	0.006208	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=1/256, ttl=1
8	0.006410	10.0.2.15	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
9	0.006455	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=2/512, ttl=1
10	0.006531	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=3/768, ttl=1
11	0.006680	10.0.2.15	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
12	0.006682	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=4/1024, ttl=2
13	0.006747	10.0.2.15	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
14	0.006779	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=5/1280, ttl=2
15	0.006852	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=6/1536, ttl=2
16	0.006940	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=7/1792, ttl=3
17	0.007014	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=8/2048, ttl=3
18	0.007084	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=9/2304, ttl=3
19	0.007170	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=10/2560, ttl=4
20	0.007240	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=11/2816, ttl=4
21	0.007309	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=12/3072, ttl=4
22	0.007379	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=13/3328, ttl=5

Figura 7: Tráfego capturado com o Wireshark

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

O valor do campo protocolo é “ICMP (1)”. Este valor permite identificar de que protocolo procede o datagrama.

Frame 7: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
Ethernet II, Src: CadmusCo_78:e5:64 (08:00:27:78:e5:64), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 193.136.9.240 (193.136.9.240)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 60
Identification: 0x25e8 (9704)
Flags: 0x00
Fragment offset: 0
Time to live: 1
Protocol: ICMP (1)
Header checksum: 0xb5c2 [correct]
Source: 10.0.2.15 (10.0.2.15)
Destination: 193.136.9.240 (193.136.9.240)
Internet Control Message Protocol

Figura 8: Informações sobre o datagrama-Campo Protocolo



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?

O cabeçalho IP(v4) tem 20 bytes.

O tamanho do campo de dados (payload), foi calculado através da diferença entre o tamanho total do datagrama (60 bytes) e o tamanho do cabeçalho (20 bytes), concluindo-se assim que tem como tamanho 40 bytes.

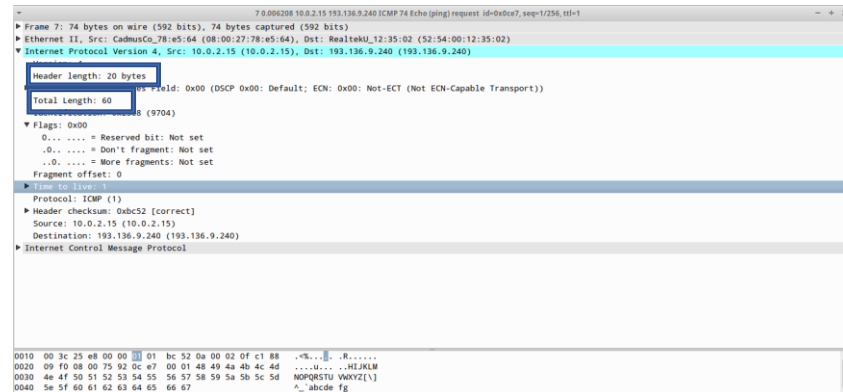


Figura 9: Informações sobre o datagrama-Campos Header length & Total Length

d. O datagrama IP foi fragmentado? Justifique.

Através da análise das *flags* concluiu-se que o datagrama IP não foi fragmentado, dado que “*Fragment offset=0*” e “*More fragments = Not set*”.

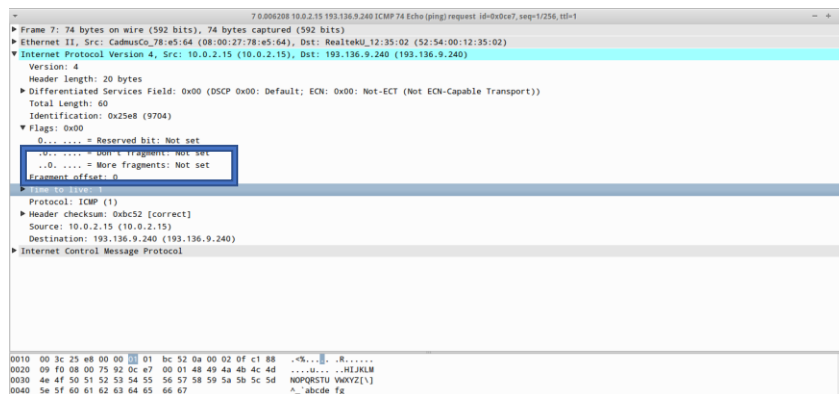


Figura 10: Informações sobre o datagrama-Análise das Flags



- 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.

Com a análise de três datagramas aleatórios de mensagens ICMP enviadas pelo computador em que foi feita a análise, foi verificado que, além da sua identificação e do *header checksum*, o campo TTL varia entre eles.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.2.15	192.168.100.254	DNS	75	Standard query AAAA marco.uninho.pt
3	0.002205	10.0.2.15	192.168.100.254	DNS	91	Standard query AAAA marco.uninho.pt-sa.di.uninho.pt
5	0.003971	10.0.2.15	192.168.100.254	DNS	75	Standard query A marco.uninho.pt
8	0.005555	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=2/512, ttl=1
9	0.006455	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=3/768, ttl=1
10	0.006531	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=4/1024, ttl=1
12	0.006682	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=5/1280, ttl=2
14	0.006779	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=6/1536, ttl=2
15	0.006852	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=7/1792, ttl=3
16	0.006940	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=8/2048, ttl=3
17	0.007014	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=9/2304, ttl=3
18	0.007084	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=10/2560, ttl=4
19	0.007170	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=11/2816, ttl=4
20	0.007240	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=12/3072, ttl=4
21	0.007309	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=13/3328, ttl=5
22	0.007379	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=14/3584, ttl=5
23	0.007449	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=15/3840, ttl=5
24	0.007518	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=16/4096, ttl=6
25	0.007616	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=17/4352, ttl=6
26	0.007888	10.0.2.15	192.168.100.254	DNS	81	Standard query PTR 2.2.0.10.in-addr.arpa
42	0.110998	10.0.2.15	224.0.0.251	MNCS	81	Standard query PTR 2.2.0.10.in-addr.arpa, "Qr" question
44	0.114651	10.0.2.15	224.0.0.251	MNCS	81	Standard query PTR 2.2.0.10.in-addr.arpa, "Qr" question

Figura 11: Análise do tráfego ordenado por Source

7.0.000000 10.0.2.15 193.136.9.240 ICMP 74 Echo (ping) request id=0x0ce7, seq=2/512, ttl=1

Ethernet II, Src: CadmusCo_78:e5:b4 (08:00:27:78:e5:b4), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)

Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 193.136.9.240 (193.136.9.240)

Version: 4

Header Length: 20 bytes

Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECT-Capable Transport))

Total Length: 60

Identification: 0x70b9 (27039)

Flags: 0x00

0... .. Reserved bit: Not set

0... .. Don't Fragment: Not set

0... .. More Fragments: Not set

Time to live: 1

Header checksum: 0x70b9 [correct]

Source: 10.0.2.15 (10.0.2.15)

Destination: 193.136.9.240 (193.136.9.240)

Internet Control Message Protocol

Figura 12: Informações sobre o datagrama (1/3)

18.0.000000 10.0.2.15 193.136.9.240 ICMP 74 Echo (ping) request id=0x0ce7, seq=3/768, ttl=1

Ethernet II, Src: CadmusCo_78:e5:b4 (08:00:27:78:e5:b4), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)

Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 193.136.9.240 (193.136.9.240)

Version: 4

Header Length: 20 bytes

Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECT-Capable Transport))

Total Length: 60

Identification: 0x70b9 (27039)

Flags: 0x00

0... .. Reserved bit: Not set

0... .. Don't Fragment: Not set

0... .. More Fragments: Not set

Time to live: 1

Header checksum: 0x70b9 [correct]

Source: 10.0.2.15 (10.0.2.15)

Destination: 193.136.9.240 (193.136.9.240)

Internet Control Message Protocol

Figura 13. : Informações sobre o datagrama (2/3)

21.0.007309 10.0.2.15 193.136.9.240 ICMP 74 Echo (ping) request id=0x0ce7, seq=12/3072, ttl=4

Ethernet II, Src: CadmusCo_78:e5:b4 (08:00:27:78:e5:b4), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)

Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 193.136.9.240 (193.136.9.240)

Version: 4

Header Length: 20 bytes

Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECT-Capable Transport))

Total Length: 60

Identification: 0x70b9 (27039)

Flags: 0x00

0... .. Reserved bit: Not set

0... .. Don't Fragment: Not set

0... .. More Fragments: Not set

Time to live: 4

Header checksum: 0x70b9 [correct]

Source: 10.0.2.15 (10.0.2.15)

Destination: 193.136.9.240 (193.136.9.240)

Internet Control Message Protocol

Figura 14. : Informações sobre o datagrama (3/3)



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

Ao ser feita a análise do tráfego no Wireshark, concluiu-se que o TTL incrementa de três em três datagramas e o campo de identificação incrementa a cada datagrama.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.2.15	192.168.100.254	DNS	75	Standard query AAAA marco.uninho.pt
3	0.002205	10.0.2.15	192.168.100.254	DNS	91	Standard query AAAA marco.uninho.pt.sa.dl.uninho.pt
5	0.003971	10.0.2.15	192.168.100.254	DNS	75	Standard query A marco.uninho.pt
7	0.006208	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=1/256, ttl=1
9	0.006455	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=2/512, ttl=1
10	0.006531	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=3/768, ttl=1
12	0.006482	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=4/1024, ttl=2
14	0.006779	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=5/1280, ttl=2
15	0.006852	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=6/1536, ttl=2
16	0.006940	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=7/1792, ttl=3
17	0.007014	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=8/2048, ttl=3
18	0.007084	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=9/2304, ttl=3
19	0.007170	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=10/2560, ttl=4
20	0.007240	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=11/2816, ttl=4
21	0.007259	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=12/3072, ttl=4
22	0.007379	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=13/3328, ttl=5
23	0.007449	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=14/3584, ttl=5
24	0.007518	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=15/3840, ttl=5
25	0.007616	10.0.2.15	193.136.9.240	ICMP	74	Echo (ping) request id=0x0ce7, seq=16/4096, ttl=6
26	0.007686	10.0.2.15	192.168.100.254	DNS	81	Standard query PTR 2.2.0.10.in-addr.arpa
42	0.113998	10.0.2.15	224.0.0.251	MDNS	81	Standard query PTR 2.2.0.10.in-addr.arpa, "QM" question
44	1.114851	10.0.2.15	224.0.0.251	MDNS	81	Standard query PTR 2.2.0.10.in-addr.arpa, "QM" question

Figura 15: Análise de Tráfego

g. Ordene o tráfego capturado por endereço destino e encontre a série de respostas ICMP TTL exceeded enviadas ao seu computador. Qual é o valor do campo TTL? Esse valor permanece constante para todas as mensagens de resposta ICMP TTL exceeded enviados ao seu host? Porquê?

O valor do campo TTL varia de acordo com a *source* da qual foi recebido o datagrama, logo se os datagramas corresponderem à mesma *source* o campo TTL permanecerá constante. Caso contrário, esse valor não permanecerá constante, irá variar.

No.	Time	Source	Destination	Protocol	Length	Info
2	0.002067	192.168.100.254	10.0.2.15	DNS	129	Standard query response
4	0.003737	192.168.100.254	10.0.2.15	DNS	136	Standard query response, No such name
6	0.005373	192.168.100.254	10.0.2.15	DNS	319	Standard query response A 193.136.9.240
8	0.006410	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
11	0.006830	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
13	0.006747	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
15	0.006925	192.168.100.254	10.0.2.15	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
18	0.008033	192.168.100.254	10.0.2.15	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
28	0.008366	192.168.100.254	10.0.2.15	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
30	0.008538	193.136.9.240	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0ce7, seq=10/2560, ttl=61
32	0.011400	193.136.9.240	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0ce7, seq=11/2816, ttl=61
33	0.011434	193.136.9.240	10.0.2.15	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
34	0.011416	193.136.9.240	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0ce7, seq=12/3072, ttl=61
35	0.011418	193.136.9.240	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0ce7, seq=13/3328, ttl=61
36	0.011421	193.136.9.240	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0ce7, seq=14/3584, ttl=61
37	0.011422	193.136.9.240	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0ce7, seq=15/3840, ttl=61
38	0.011424	193.136.9.240	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0ce7, seq=16/4096, ttl=61
39	0.011426	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
40	0.011428	192.168.100.254	10.0.2.15	DNS	116	Standard query response, No such name
64	5.017319	193.136.9.240	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0ce7, seq=17/4352, ttl=61
65	5.017326	193.136.9.240	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0ce7, seq=18/4608, ttl=61

Figura 16: Tráfego ordenado por Destination

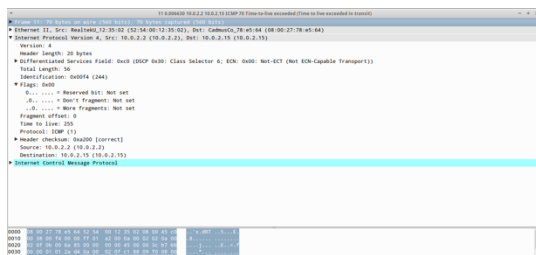


Figura 17: Informações sobre o datagrama (1/3)



Figura 18: Informações sobre o datagrama (2/3)

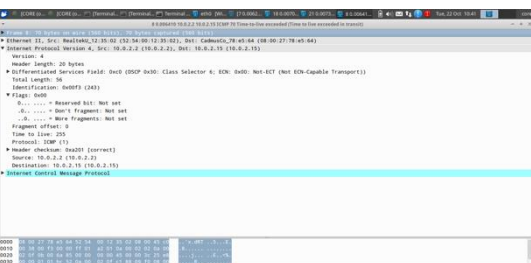


Figura 19: Informações sobre o datagrama (3/3)

Questão 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 42XX bytes.

```
File Edit View Terminal Go Help
core@XubuntuCORE:~$ sudo traceroute -I marco.uminho.pt 4205
[sudo] password for core:
traceroute to marco.uminho.pt (193.136.9.240), 30 hops max, 4205 byte packets
 1 10.0.2.2 (10.0.2.2) 0.248 ms 0.072 ms *
 2 * * *
 3 * * *
 4 * * *
 5 * * *
 6 * marco.uminho.pt (193.136.9.240) 3.695 ms 4.146 ms
core@XubuntuCORE:~$
```

Figura 20: Comando "traceroute" para tamanho de pacote = 4205 bytes



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

Houve necessidade de fragmentar o pacote inicial pois o tamanho original do pacote é mais pequeno do que o *maximum transmission unit* (MTU), o que não permite que o pacote original, como um todo, consiga ser enviado.

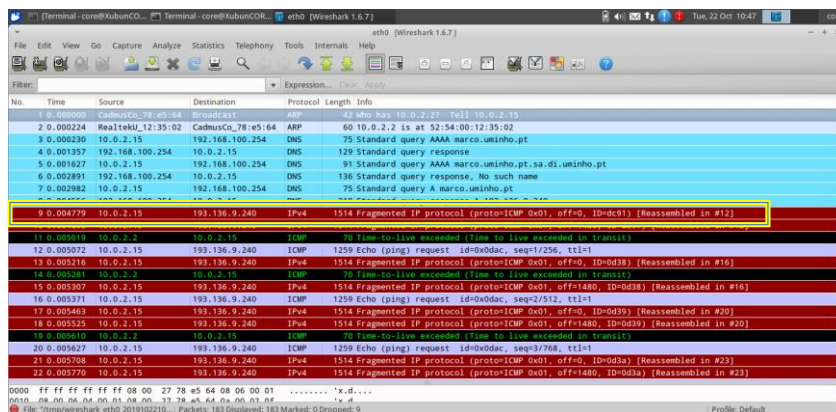


Figura 21: Verificação da fragmentação da primeira mensagem ICMP

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?

A informação presente no cabeçalho permitiu a conclusão de que o datagrama foi fragmentado, uma vez que a *flag* “More fragments” se encontra a “Set”, o que indica que existem outros fragmentos desse datagrama. É possível afirmar que se trata do primeiro fragmento, visto que o valor de “Fragment offset” é zero.

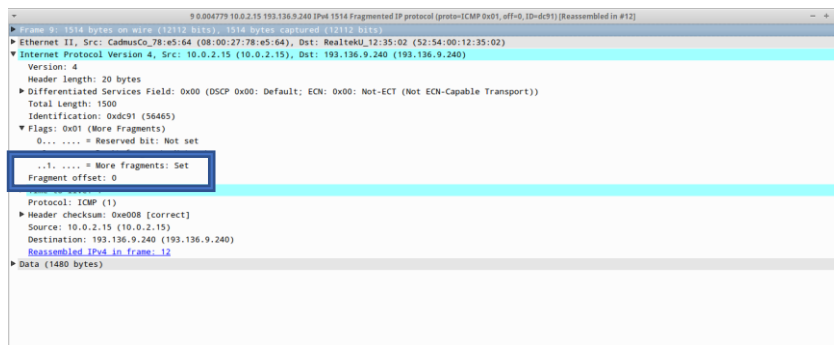


Figura 22: Análise do primeiro fragmento do datagrama



- 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?

O que permite concluir que não se trata do primeiro fragmento é o campo “*Fragment offset*” se encontrar diferente de zero, neste caso, 1480. Existem mais fragmentos, uma vez que o campo “*More fragments*” se encontra a “*Set*”.

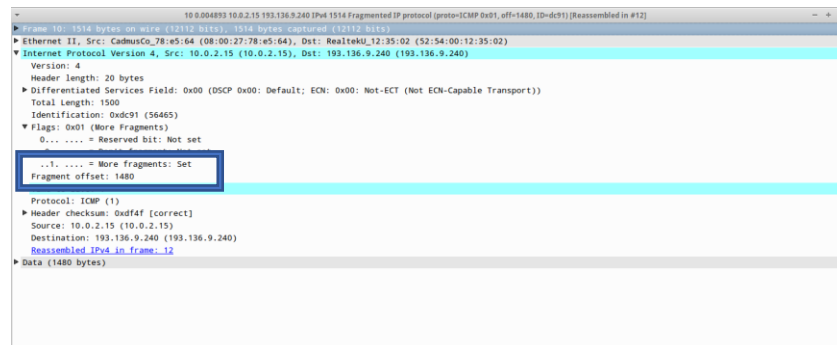


Figura 23: Análise do segundo fragmento do datagrama

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

A partir do datagrama original foram criados três fragmentos. Além dos especificados nas figuras 22 e 23, também o fragmento impresso na figura 24. Neste é possível concluir que é o último fragmento, pois os campos “*Fragments offset*” é diferente de zero, indicando que não é o primeiro fragmento e “*More fragments*” se encontra a “*Not set*”, indicando que não existem mais fragmentos após este.

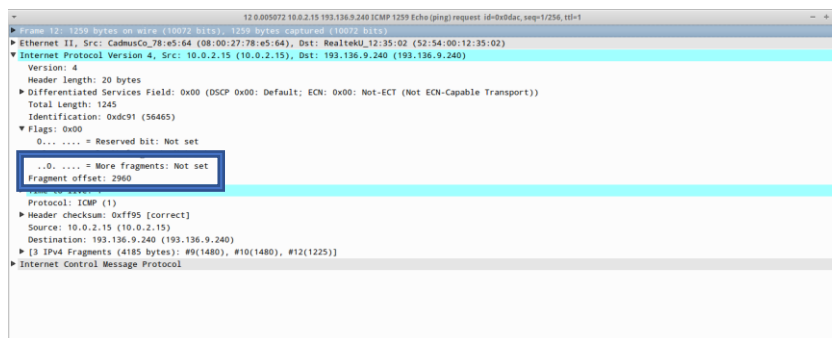


Figura 24: Análise do último fragmento do datagrama



- 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.**

Para que seja possível a reconstrução do datagrama original é preciso ter em atenção a ordem dos fragmentos. Essa ordem pode ser desvendada através da análise das *flags*.

Quanto ao primeiro fragmento, observa-se que o campo “*More fragments*” se encontra a “*Set*”, indicando que existem mais fragmentos para além do atual e o “*Fragment offset*” toma o valor de zero, mostrando que é o primeiro fragmento.

Quanto ao último fragmento, verifica-se que o campo “*More fragments*” se encontra a “*Not set*”, o que indica que não há mais fragmentos após este, e “*Fragment offset*” diferente de zero.

Entre estes dois limites, podem ser encontrados mais fragmentos, cujos valores do campo “*Fragment offset*” são progressivamente maiores, entre zero e “*Fragment offset*” do último fragmento e o campo “*More Fragments*” encontra-se a “*Set*”.

2. Parte II: Endereçamento e Encaminhamento IP

2. Questões e Respostas

Questão 1

Atenda aos endereços IP atribuídos automaticamente pelo CORE aos diversos equipamentos da topologia.

- a. Indique que endereços IP e máscaras de rede foram atribuídos pelo CORE a cada equipamento. Para simplificar, pode incluir uma imagem que ilustre de forma clara a topologia definida e o endereçamento usado.

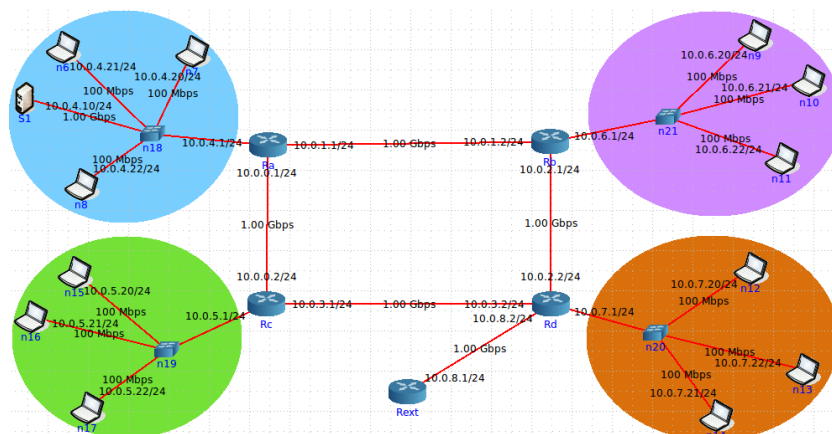


Figura 25: Endereços IP - Topologia Core

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

Tratam-se de endereços privados, uma vez que se encontram no intervalo 10.0.0.0 a 10.255.255.255.



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

Não são atribuídos endereços IP aos switches dado que se tratam de *ethernet switches* pelo que estes, ao se encontrarem na camada “*Link Layer*”, utilizam os endereços MAC dos datagramas recebidos em vez dos endereços IP.

d. Usando o comando ping certifique-se que existe conectividade IP entre os laptops dos vários departamentos e o servidor do departamento A.

Com a utilização do comando “*ping*” foi possível comprovar que existe conectividade entre os laptops dos quatro departamentos com o servidor do departamento A.

```
root@n6:/tmp/pycore.60298/n6.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_req=1 ttl=64 time=0.025 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=64 time=0.051 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=64 time=0.053 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=64 time=0.053 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=64 time=0.052 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=64 time=0.049 ms
64 bytes from 10.0.4.10: icmp_req=7 ttl=64 time=0.059 ms
64 bytes from 10.0.4.10: icmp_req=8 ttl=64 time=0.051 ms
64 bytes from 10.0.4.10: icmp_req=9 ttl=64 time=0.053 ms
64 bytes from 10.0.4.10: icmp_req=10 ttl=64 time=0.053 ms
64 bytes from 10.0.4.10: icmp_req=11 ttl=64 time=0.059 ms
64 bytes from 10.0.4.10: icmp_req=12 ttl=64 time=0.052 ms
64 bytes from 10.0.4.10: icmp_req=13 ttl=64 time=0.047 ms
64 bytes from 10.0.4.10: icmp_req=14 ttl=64 time=0.053 ms
^C
--- 10.0.4.10 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 1239ms
rtt min/avg/max/mdev = 0.025/0.050/0.059/0.011 ms
root@n6:/tmp/pycore.60298/n6.conf#
```

Figura 26: Execução do comando ping (Laptop n6-Servidor)

```
root@n10:/tmp/pycore.60298/n10.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_req=1 ttl=62 time=0.080 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=62 time=0.091 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=62 time=0.099 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=62 time=0.085 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=62 time=0.083 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=62 time=0.085 ms
64 bytes from 10.0.4.10: icmp_req=7 ttl=62 time=0.084 ms
64 bytes from 10.0.4.10: icmp_req=8 ttl=62 time=0.084 ms
64 bytes from 10.0.4.10: icmp_req=9 ttl=62 time=0.086 ms
64 bytes from 10.0.4.10: icmp_req=10 ttl=62 time=0.084 ms
64 bytes from 10.0.4.10: icmp_req=11 ttl=62 time=0.085 ms
64 bytes from 10.0.4.10: icmp_req=12 ttl=62 time=0.094 ms
64 bytes from 10.0.4.10: icmp_req=13 ttl=62 time=0.102 ms
^C
--- 10.0.4.10 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 1199ms
rtt min/avg/max/mdev = 0.080/0.087/0.102/0.011 ms
root@n10:/tmp/pycore.60298/n10.conf#
```

Figura 27: Execução do comando ping (Laptop n10-Servidor)

```
root@n14:/tmp/pycore.60298/n14.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_req=1 ttl=61 time=0.102 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=61 time=0.124 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=61 time=0.142 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=61 time=0.127 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=61 time=0.095 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=61 time=0.145 ms
64 bytes from 10.0.4.10: icmp_req=7 ttl=61 time=0.125 ms
64 bytes from 10.0.4.10: icmp_req=8 ttl=61 time=0.141 ms
64 bytes from 10.0.4.10: icmp_req=9 ttl=61 time=0.066 ms
64 bytes from 10.0.4.10: icmp_req=10 ttl=61 time=0.122 ms
64 bytes from 10.0.4.10: icmp_req=11 ttl=61 time=0.143 ms
^C
--- 10.0.4.10 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.066/0.121/0.145/0.023 ms
root@n14:/tmp/pycore.60298/n14.conf#
```

Figura 28: Execução do comando ping (Laptop n14-Servidor)

```
root@n16:/tmp/pycore.60298/n16.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_req=1 ttl=62 time=0.071 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=62 time=0.086 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=62 time=0.037 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=62 time=0.079 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=62 time=0.078 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=62 time=0.051 ms
64 bytes from 10.0.4.10: icmp_req=7 ttl=62 time=0.085 ms
64 bytes from 10.0.4.10: icmp_req=8 ttl=62 time=0.083 ms
64 bytes from 10.0.4.10: icmp_req=9 ttl=62 time=0.085 ms
64 bytes from 10.0.4.10: icmp_req=10 ttl=62 time=0.084 ms
64 bytes from 10.0.4.10: icmp_req=11 ttl=62 time=0.085 ms
64 bytes from 10.0.4.10: icmp_req=12 ttl=62 time=0.084 ms
^C
--- 10.0.4.10 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 1100ms
rtt min/avg/max/mdev = 0.037/0.076/0.086/0.016 ms
root@n16:/tmp/pycore.60298/n16.conf#
```

Figura 29: Execução do comando ping (Laptop n16-Servidor)



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

Através do comando “ping” foi possível verificar a conectividade IP do router de acesso Rext para o servidor S1.

```
root@Rext:/tmp/pycore.60322/Rext.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_req=1 ttl=61 time=0.084 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=61 time=0.178 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=61 time=0.143 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=61 time=0.123 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=61 time=0.143 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=61 time=0.125 ms
64 bytes from 10.0.4.10: icmp_req=7 ttl=61 time=0.142 ms
64 bytes from 10.0.4.10: icmp_req=8 ttl=61 time=0.142 ms
64 bytes from 10.0.4.10: icmp_req=9 ttl=61 time=0.144 ms
64 bytes from 10.0.4.10: icmp_req=10 ttl=61 time=0.144 ms
64 bytes from 10.0.4.10: icmp_req=11 ttl=61 time=0.124 ms
64 bytes from 10.0.4.10: icmp_req=12 ttl=61 time=0.110 ms
64 bytes from 10.0.4.10: icmp_req=13 ttl=61 time=0.118 ms
64 bytes from 10.0.4.10: icmp_req=14 ttl=61 time=0.118 ms
64 bytes from 10.0.4.10: icmp_req=15 ttl=61 time=0.112 ms
--- 10.0.4.10 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 1399ms
rtt min/avg/max/mdev = 0.084/0.123/0.178/0.025 ms
root@Rext:/tmp/pycore.60322/Rext.conf#
```

Figura 26: Execução do comando ping (Router de acesso Rext-Servidor)

Questão 2

Para o router e um laptop do departamento B:

- 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@Rb:/tmp/pycore.60322/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 10.0.1.1 255.255.255.0 UG 0 0 0 eth0
10.0.6.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
10.0.7.0 10.0.2.2 255.255.255.0 UG 0 0 0 eth1
10.0.8.0 10.0.2.2 255.255.255.0 UG 0 0 0 eth1
root@Rb:/tmp/pycore.60322/Rb.conf#
```

Figura 31: Tabela de encaminhamento - Router Rb

```
root@n10:/tmp/pycore.60322/n10.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@n10:/tmp/pycore.60322/n10.conf#
```

Figura 32: Tabela de encaminhamento - Laptop n10



- **Destination:** Indica a sub-rede de destino.
- **Gateway:** Indica o caminho a percorrer pelo tráfego até este chegar ao destino.
- **Genmask:** Indica o tipo de máscara da sub-rede utilizada.
- **Flags:**
 - **U:** Flag que indica rota válida.
 - **UG:** Flag que indica rota válida e ligação a um *Gateway*.
- **MSS:** Sigla para “*Maximum Segment Size*”, parâmetro que especifica a maior quantidade de dados (em *bytes*) que um dispositivo pode receber num único segmento TCP.
- **Iface:** Tipo de interface para onde os pacotes irão ser enviados.

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

Através da análise dos processos obtidos através do comando “ps ax” foi possível concluir-se que está a ser usado encaminhamento dinâmico, uma vez que é possível encontrar o protocolo de roteamento OSPF (“*Open Shortest Path First*”). Este protocolo indica que as rotas são atualizadas ao longo do tempo.

```
root@Rb: /tmp/pycore.60322/Rb.conf# ps ax
PID TTY STAT TIME COMMAND
1 ? S 0:00 /usr/sbin/vncnode -v -c /tmp/pycore.60322/Rb -l /tmp/p
41 ? Ss 0:00 /usr/lib/quagga/zebra -u root -g root -d
47 ? Ss 0:00 /usr/lib/quagga/ospfd -u root -g root -d
53 ? Ss 0:00 /usr/lib/quagga/ospf6d -u root -g root -d
180 pts/5 Ss 0:00 /bin/bash
234 pts/5 R+ 0:00 ps ax
root@Rb: /tmp/pycore.60322/Rb.conf#
```

Figura 27: Execução do comando "ps ax" para análise de processos



- 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.

De modo a eliminar a rota por defeito executou-se o comando “`route delete default`”. Para testar as implicações que esta ação traz, realizamos novamente o comando “`ping`” para testar a conectividade entre o servidor S1 e um laptop de cada departamento.

Com a análise dos resultados concluiu-se que para o laptop que se encontra na mesma sub-rede do servidor S1 continuou a existir conectividade entre ambos, enquanto que laptops fora dessa rede local deixaram de conseguir comunicar com S1. Essa impossibilidade deve-se à ausência da rota a seguir no caso de não existir uma entrada específica na tabela para a rede destino.

```
root@S1: /tmp/pycore.60355/S1.conf
root@S1:/tmp/pycore.60355/S1.conf# route delete default
root@S1:/tmp/pycore.60355/S1.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@S1:/tmp/pycore.60355/S1.conf#
```

Figura 28: Execução do comando "Route delete" para eliminar a rota por defeito

```
root@n8: /tmp/pycore.60355/n8.conf
root@n8:/tmp/pycore.60355/n8.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_req=1 ttl=64 time=0.775 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=64 time=0.061 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=64 time=0.053 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=64 time=0.081 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=64 time=0.067 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=64 time=0.066 ms
^C
--- 10.0.4.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 499ms
rtt min/avg/max/mdev = 0.053/0.180/0.775/0.266 ms
root@n8:/tmp/pycore.60355/n8.conf#
```

Figura 35: Comando ping (Laptop n8 - Servidor S1)

```
root@n10: /tmp/pycore.60355/n10.conf
root@n10:/tmp/pycore.60355/n10.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
^C
--- 10.0.4.10 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8000ms
root@n10:/tmp/pycore.60355/n10.conf#
```

Figura 36: Comando ping (Laptop n10 - Servidor S1)

```
root@n16: /tmp/pycore.60355/n16.conf
root@n16:/tmp/pycore.60355/n16.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
^C
--- 10.0.4.10 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 705ms
root@n16:/tmp/pycore.60355/n16.conf#
```

Figura 37: Comando ping (Laptop n16 - Servidor S1)

```
root@n14: /tmp/pycore.60355/n14.conf
root@n14:/tmp/pycore.60355/n14.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
^C
--- 10.0.4.10 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 599ms
root@n14:/tmp/pycore.60355/n14.conf#
```

Figura 38: Comando ping (Laptop n14 - Servidor S1)



- 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.60355/S1.conf
root@S1: /tmp/pycore.60355/S1.conf# route add -net 10.0.5.0 netmask 255.255.255.0
dev eth0 gw 10.0.4.1
SIOCADDRT: File exists
root@S1: /tmp/pycore.60355/S1.conf# route add -net 10.0.6.0 netmask 255.255.255.0 dev eth0 gw 10.0.4.1
root@S1: /tmp/pycore.60355/S1.conf# route add -net 10.0.7.0 netmask 255.255.255.0 dev eth0 gw 10.0.4.1
root@S1: /tmp/pycore.60355/S1.conf#
```

Figura 39: Execução do comando “route add” para os diferentes sub-redes

- e. Teste a nova política de encaminhamento garantindo que o servidor está novamente acessível utilizando para o efeito o comando `ping`. Registe a nova tabela de encaminhamento do servidor.

```
root@n8: /tmp/pycore.60355/n8.conf
root@n8: /tmp/pycore.60355/n8.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_req=1 ttl=64 time=0.053 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=64 time=0.071 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=64 time=0.052 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=64 time=0.070 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=64 time=0.069 ms
^C
--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3996ms
rtt min/avg/max/mdev = 0.052/0.063/0.071/0.008 ms
root@n8: /tmp/pycore.60355/n8.conf#
```

Figura 40: Comando ping (Laptop n8 - Servidor S1)

```
root@n16: /tmp/pycore.60355/n16.conf
root@n16: /tmp/pycore.60355/n16.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_req=1 ttl=62 time=0.070 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=62 time=0.117 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=62 time=0.116 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=62 time=0.123 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=62 time=0.144 ms
^C
--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3937ms
rtt min/avg/max/mdev = 0.070/0.114/0.144/0.024 ms
root@n16: /tmp/pycore.60355/n16.conf#
```

Figura 41: Comando ping (Laptop n16 - Servidor S1)

```
root@n10: /tmp/pycore.60355/n10.conf
root@n10: /tmp/pycore.60355/n10.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_req=1 ttl=62 time=0.080 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=62 time=0.106 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=62 time=0.095 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=62 time=0.094 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=62 time=0.094 ms
^C
--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.084/0.097/0.106/0.009 ms
root@n10: /tmp/pycore.60355/n10.conf#
```

Figura 42: Comando ping (Laptop n10 - Servidor S1)

```
root@n14: /tmp/pycore.60355/n14.conf
root@n14: /tmp/pycore.60355/n14.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_req=1 ttl=61 time=0.084 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=61 time=0.140 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=61 time=0.142 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=61 time=0.192 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=61 time=0.141 ms
^C
--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3936ms
rtt min/avg/max/mdev = 0.084/0.141/0.192/0.034 ms
root@n14: /tmp/pycore.60355/n14.conf#
```

Figura 43: Comando ping (Laptop n14 - Servidor S1)

```
root@S1: /tmp/pycore.60355/S1.conf
root@S1: /tmp/pycore.60355/S1.conf# route add -net 10.0.5.0 netmask 255.255.255.0
dev eth0 gw 10.0.4.1
SIOCADDRT: File exists
root@S1: /tmp/pycore.60355/S1.conf# route add -net 10.0.6.0 netmask 255.255.255.0 dev eth0 gw 10.0.4.1
root@S1: /tmp/pycore.60355/S1.conf# route add -net 10.0.7.0 netmask 255.255.255.0 dev eth0 gw 10.0.4.1
root@S1: /tmp/pycore.60355/S1.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
10.0.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.0.6.0 10.0.4.1 255.255.255.0 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 10.0.4.1 255.255.255.0 UG 0 0 0 eth0
10.0.7.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1: /tmp/pycore.60355/S1.conf#
```

Figura 44: Nova tabela de encaminhamento do servidor S1

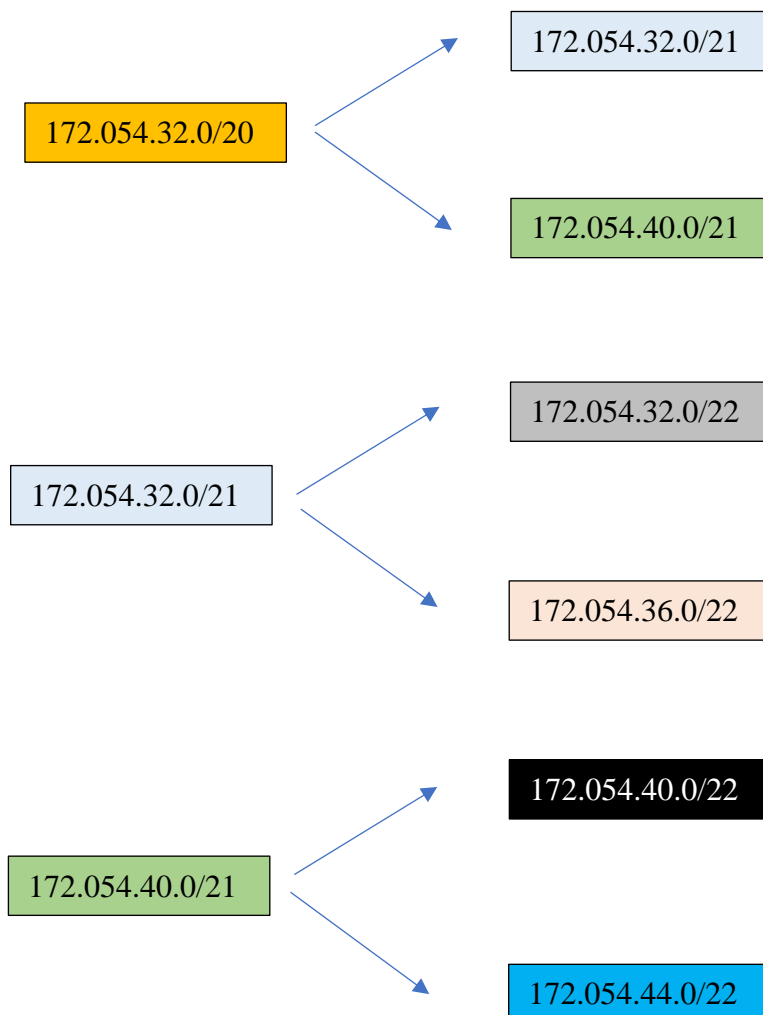


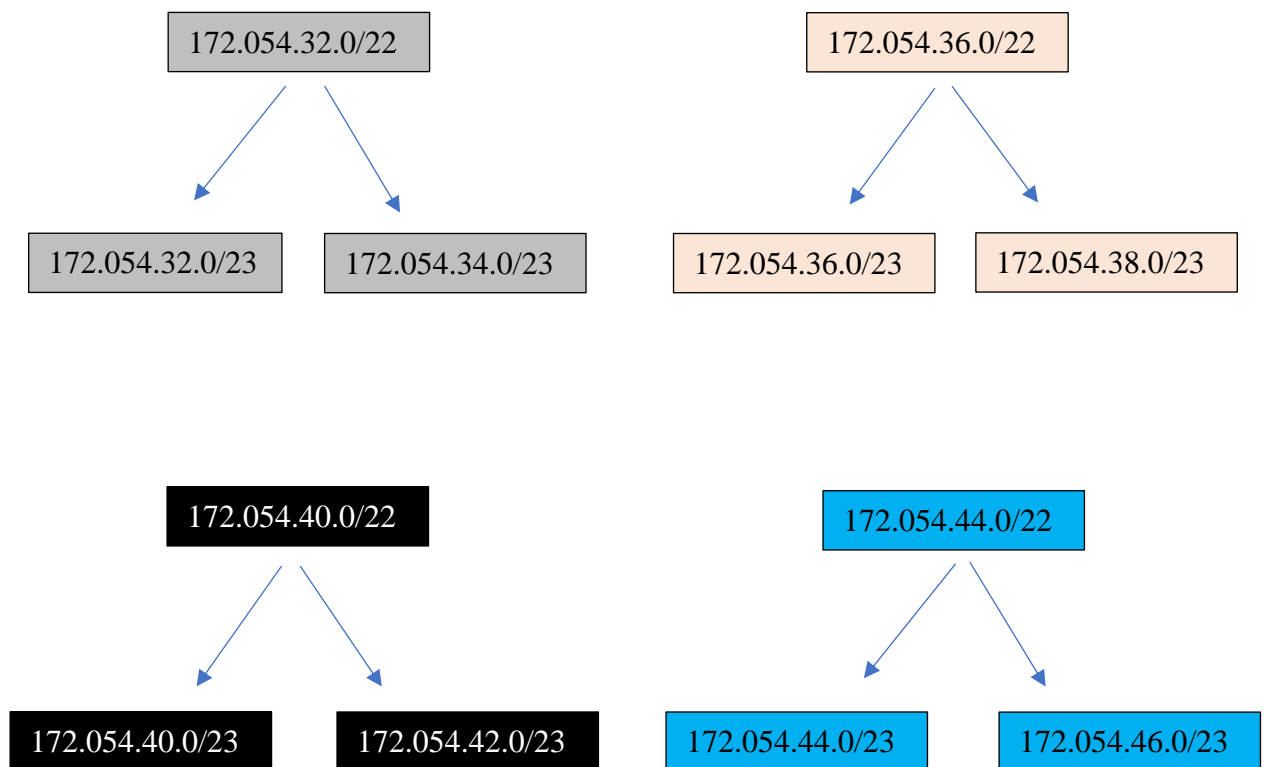
2.1. Parte II: Definição de Sub-redes

Questão 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.

Sendo o endereço inicial 172.054.32.0/20, tem-se:





O processo realizado anteriormente deve-se ao facto de ser necessário atribuir uma sub-rede a cada um dos quatro departamentos, ou seja, uma totalidade de quatro sub-redes.

Inicialmente, dividiu-se o endereço de rede IP inicial (172.054.32.0/20) em dois endereços, visto que ainda não era suficiente para endereçar os quatro departamentos, continuou-se a divisão e ficou-se com quatro endereços. No entanto, como os endereços tudo a 1's e o tudo a 0's são reservados, continua-se com a problemática referida anteriormente, por isso houve necessidade de se voltar a dividir.

Por fim, obteve-se oito endereços e apesar de dois destes se encontrarem reservados, ainda se consegue usufruir de seis, embora só seja preciso quatro para endereçar os departamentos (escolheu-se aleatoriamente o endereço 172.054.36.0/23 para o departamento A, o 172.054.44.0/23 para o departamento B, o 172.054.40.0/23 para o departamento C e 172.054.48.0/23 para o departamento D).

Relativamente à parte do host do endereço foram mantidos os valores atribuídos na topologia inicial realizada no CORE.

A topologia CORE após as alterações referidas encontra-se representada na figura 45.

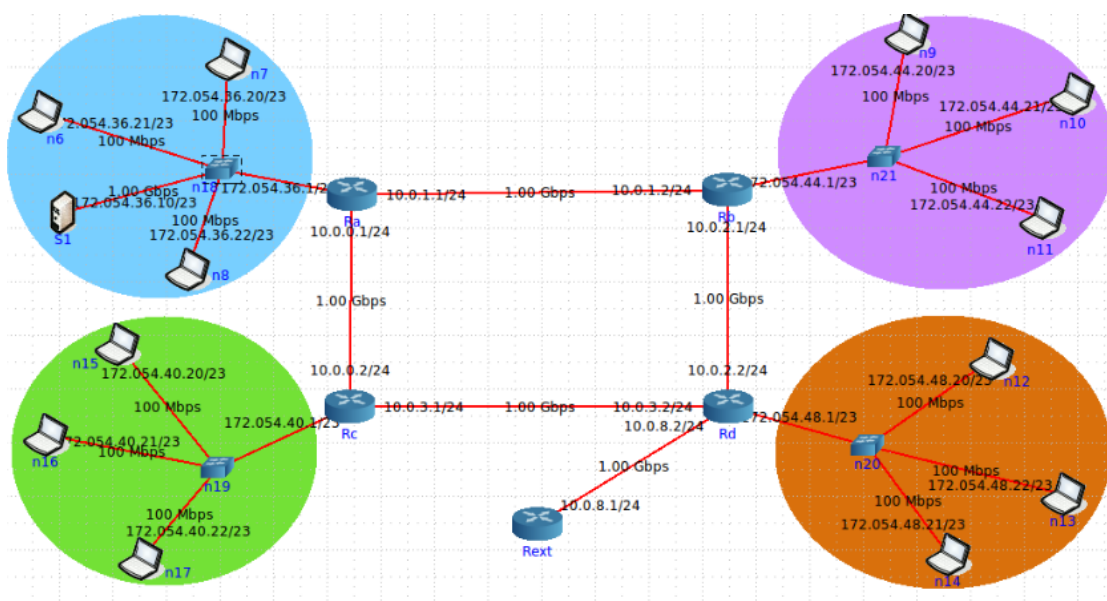


Figura 45: Topologia CORE atualizada com subnetting

Questão 2

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

A máscara de rede usada (em notação decimal) foi 255.255.254.0.

11111111

11111111

11111110

00000000

Dado que em notação CIDR: /23



Questão 3

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

```
root@n6:/tmp/pycore.60997/n6.conf# ping 172.054.36.10
PING 172.054.36.10 (172.44.36.10) 56(84) bytes of data:
64 bytes from 172.44.36.10: icmp_req=1 ttl=64 time=0.051 ms
64 bytes from 172.44.36.10: icmp_req=2 ttl=64 time=0.049 ms
64 bytes from 172.44.36.10: icmp_req=3 ttl=64 time=0.072 ms
64 bytes from 172.44.36.10: icmp_req=4 ttl=64 time=0.072 ms
64 bytes from 172.44.36.10: icmp_req=5 ttl=64 time=0.073 ms
64 bytes from 172.44.36.10: icmp_req=6 ttl=64 time=0.075 ms
^C
--- 172.054.36.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4996ms
rtt min/avg/max/mdev = 0.049/0.065/0.075/0.012 ms
root@n6:/tmp/pycore.60997/n6.conf#
```

Figure 46: Comando ping (Laptop n6 - Servidor S1)

```
root@n10:/tmp/pycore.60997/n10.conf# ping 172.054.36.10
PING 172.054.36.10 (172.44.36.10) 56(84) bytes of data:
64 bytes from 172.44.36.10: icmp_req=1 ttl=62 time=0.595 ms
64 bytes from 172.44.36.10: icmp_req=2 ttl=62 time=0.049 ms
64 bytes from 172.44.36.10: icmp_req=3 ttl=62 time=0.039 ms
64 bytes from 172.44.36.10: icmp_req=4 ttl=62 time=0.043 ms
64 bytes from 172.44.36.10: icmp_req=5 ttl=62 time=0.118 ms
64 bytes from 172.44.36.10: icmp_req=6 ttl=62 time=0.096 ms
^C
--- 172.054.36.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5000ms
rtt min/avg/max/mdev = 0.039/0.156/0.595/0.198 ms
root@n10:/tmp/pycore.60997/n10.conf#
```

Figure 47: Comando ping (Laptop n10 - Servidor S1)

```
root@n16:/tmp/pycore.60997/n16.conf# ping 172.054.36.10
PING 172.054.36.10 (172.44.36.10) 56(84) bytes of data:
64 bytes from 172.44.36.10: icmp_req=1 ttl=62 time=0.088 ms
64 bytes from 172.44.36.10: icmp_req=2 ttl=62 time=0.116 ms
64 bytes from 172.44.36.10: icmp_req=3 ttl=62 time=0.174 ms
64 bytes from 172.44.36.10: icmp_req=4 ttl=62 time=0.115 ms
64 bytes from 172.44.36.10: icmp_req=5 ttl=62 time=0.064 ms
64 bytes from 172.44.36.10: icmp_req=6 ttl=62 time=0.121 ms
^C
--- 172.054.36.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4996ms
rtt min/avg/max/mdev = 0.064/0.113/0.174/0.033 ms
root@n16:/tmp/pycore.60997/n16.conf#
```

Figure 48: Comando ping (Laptop n16 - Servidor S1)

```
root@n14:/tmp/pycore.60997/n14.conf# ping 172.054.36.10
PING 172.054.36.10 (172.44.36.10) 56(84) bytes of data:
64 bytes from 172.44.36.10: icmp_req=1 ttl=61 time=0.106 ms
64 bytes from 172.44.36.10: icmp_req=2 ttl=61 time=0.144 ms
64 bytes from 172.44.36.10: icmp_req=3 ttl=61 time=0.145 ms
64 bytes from 172.44.36.10: icmp_req=4 ttl=61 time=0.147 ms
64 bytes from 172.44.36.10: icmp_req=5 ttl=61 time=0.147 ms
64 bytes from 172.44.36.10: icmp_req=6 ttl=61 time=0.147 ms
^C
--- 172.054.36.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4998ms
rtt min/avg/max/mdev = 0.106/0.139/0.147/0.019 ms
root@n14:/tmp/pycore.60997/n14.conf#
```

Figure 49: Comando ping (Laptop n14 - Servidor S1)

```
root@Rext:/tmp/pycore.60997/Rext.conf# ping 172.054.36.10
PING 172.054.36.10 (172.44.36.10) 56(84) bytes of data:
64 bytes from 172.44.36.10: icmp_req=1 ttl=61 time=0.048 ms
64 bytes from 172.44.36.10: icmp_req=2 ttl=61 time=0.145 ms
64 bytes from 172.44.36.10: icmp_req=3 ttl=61 time=0.147 ms
64 bytes from 172.44.36.10: icmp_req=4 ttl=61 time=0.152 ms
64 bytes from 172.44.36.10: icmp_req=5 ttl=61 time=0.151 ms
64 bytes from 172.44.36.10: icmp_req=6 ttl=61 time=0.151 ms
^C
--- 172.054.36.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4997ms
rtt min/avg/max/mdev = 0.048/0.132/0.152/0.038 ms
root@Rext:/tmp/pycore.60997/Rext.conf#
```

Figure 50: Comando ping (Router Rext - Servidor S1)



3. Conclusões

A realização do presente trabalho permitiu ao grupo aprender a utilizar ferramentas de simulação, nomeadamente o CORE e de captura de tráfego, o WIRESHARK.

Numa primeira etapa, analisou-se o tráfego de rede utilizando o comando “traceroute” de forma a conseguir descobrir a rota desde uma origem IP até um destino IP, em que para isto foram dados vários valores de TTL para a análise. Seguidamente observou-se as várias características dos datagramas, como por exemplo, o seu tamanho total e do cabeçalho IP (em bytes), o protocolo a ele associado e ainda se verificou o funcionamento da fragmentação nestes pacotes.

Numa segunda etapa, foi realizado um protótipo no CORE constituído por routers ligados a alguns departamentos com diversos laptops, um servidor localizado num dos departamentos, switches e um router externo que assegura a conectividade externa. A partir desta topologia, observou-se tabelas de encaminhamento, nas quais foi possível verificar as diferentes rotas e o tipo de encaminhamento associado. Foi também constatado o impacto de remover a rota existente por defeito na tabela, para contornar esta situação, adicionou-se as rotas necessárias para repor a conectividade. Para definir o novo esquema de endereçamento aplicaram-se os conceitos de subnetting lecionados, o que inicialmente se mostrou um desafio para os integrantes do grupo, no entanto com a ajuda do docente e posteriores pesquisas tornou-se uma barreira ultrapassada com sucesso.

Por fim, o grupo adquiriu conhecimentos análogos ao funcionamento do protocolo IP e a forma como estes endereços estão atribuídos em todo o protótipo, concluindo assim, que estes foram alcançados e o conteúdo programático bem cimentado e compreendido.