

UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

TP2: Protocolo IP (Parte I e II)
Grupo Nº 51

Bruno Carvalho (A89476)

João Correia (A84414)

Rúben Cerqueira (A89593)

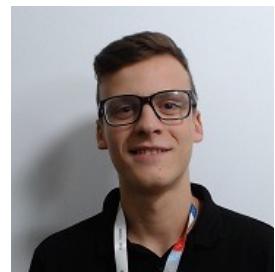
25 de novembro de 2020



Bruno



João



Rúben

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	5
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	7
1.2.7 Alínea g	7
1.3 Exercício 3	9
1.3.1 Alínea a	9
1.3.2 Alínea b	9
1.3.3 Alínea c	10
1.3.4 Alínea d	10
1.3.5 Alínea e	11
2 Parte 2	12
2.1 Exercício 1	12
2.1.1 Alínea a	12
2.1.2 Alínea b	13
2.1.3 Alínea c	13
2.1.4 Alínea d	13
2.1.5 Alínea e	13
2.2 Exercício 2	14
2.2.1 Alínea a	14
2.2.2 Alínea b	15
2.2.3 Alínea c	15
2.2.4 Alínea d	16
2.2.5 Alínea e	17
2.3 Exercício 3	17
2.3.1 Alínea 1	17
2.3.2 Alínea 2	18
2.3.3 Alínea 3	18
3 Conclusão	20

Capítulo 1

Parte 1

1.1 Exercício 1

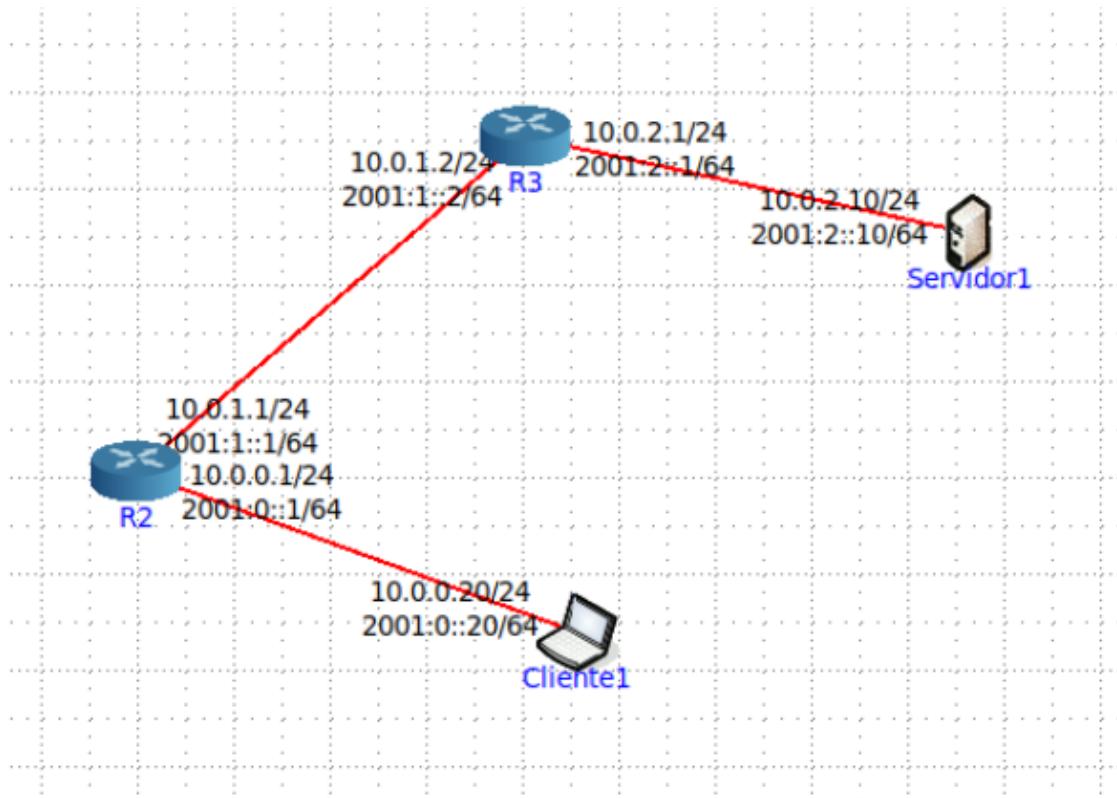


Figura 1.1: Core

1.1.1 Alínea a

Active o *wireshark* ou o *tcpdump* no *Cliente1*. Numa shell do *Cliente1*, execute o comando traceroute -l para o endereço IP do *Servidor1*.

```

root@Cliente1:/tmp/pycore.40365/Cliente1.conf
root@Cliente1:/tmp/pycore.40365/Cliente1.conf# traceroute -I 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.101 ms  0.019 ms  0.015 ms
 2  10.0.1.2 (10.0.1.2)  0.033 ms  0.019 ms  0.016 ms
 3  10.0.2.10 (10.0.2.10)  0.114 ms  0.029 ms  0.020 ms
root@Cliente1:/tmp/pycore.40365/Cliente1.conf# S

```

Figura 1.2: Traceroute

1.1.2 Alínea b

Registe e analise o tráfego ICMP enviado pelo *Cliente1* e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.

O comportamento esperado verificou-se. Foram enviados foram com o campo TTL inicialmente a 1 e que foi incrementado até se conseguir estabelecer uma conexão com sucesso.

Os pacotes com o campo TTL 1 e 2 excederam o seu TTL ainda em trânsito, no router *R2* e *R3*, respetivamente, tendo sido recebido um pacote *Time-to-live exceeded* vindo desses routers, por cada pacote que neles excedeu o seu TTL.

A partir de TTL=3, os pacotes deixaram de ser descartados, passando-se a receber *Echo (ping) reply* como resposta, pelo que se conclui que é este o número mínimo de saltos que o pacote terá de dar na rede para chegar ao destino.

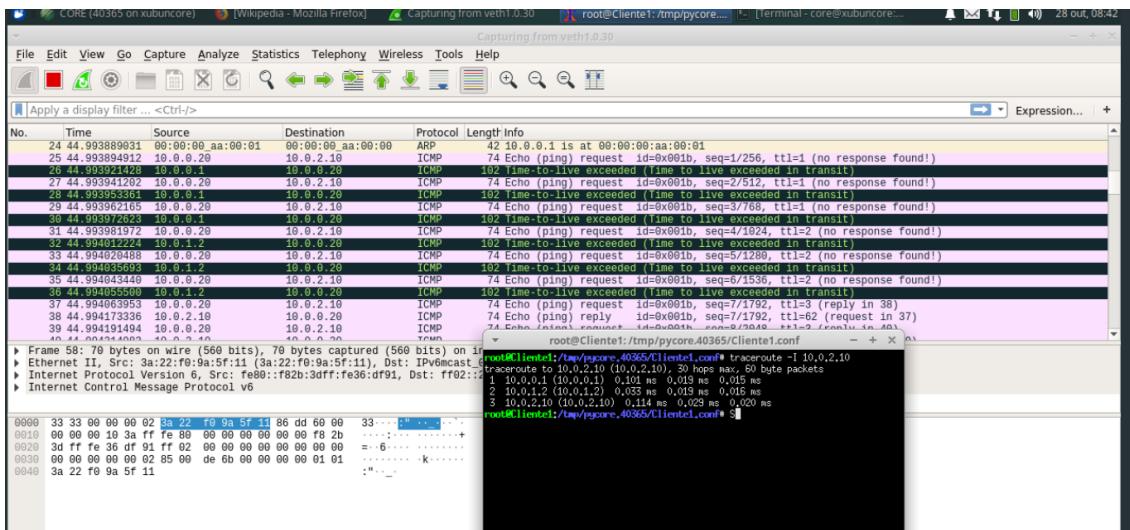


Figura 1.3: Wireshark

1.1.3 Alínea c

Qual deve ser o valor inicial mínimo do campo TTL para alcançar o *Servidor1*? Verifique na prática que a sua resposta está correta.

Observando a topologia da rede, o valor mínimo esperado no campo TTL para um pacote a atravessar com sucesso será 3.

Como verificado na **alínea b**, é exatamente a partir deste valor que os pacotes deixam de ser descartados e conseguem completar a ligação entre o cliente e o servidor.

1.1.4 Alínea d

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

Assumiu-se que esta questão apenas se relacionava com os pacotes que conseguiram completar a travessia. Como tal, o tempo médio é calculado da seguinte forma:

$$RTT = (0.114 + 0.093 + 0.020)/3 = 0.543ms$$

1.2 Exercício 2

1.2.1 Alínea a

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

Observando o tráfego capturado, registado na figura 1.4, a *source* dos pacotes enviados corresponderá ao endereço IP da interface ativa do computador, sendo este 172.26.47.109.

No.	Time	Source	Destination	Protocol	Length	Info
8	0.412518	172.16.2.1	172.26.47.109	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
10	0.485590	172.16.2.1	172.26.47.109	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
12	0.633881	172.16.2.1	172.26.47.109	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
2	0.155742	172.26.254.254	172.26.47.109	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
4	0.202799	172.26.254.254	172.26.47.109	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
6	0.342279	172.26.254.254	172.26.47.109	ICMP	78	Time-to-live exceeded (Time to live exceeded in transit)
1	0.080000	172.26.47.109	193.136.9.254	ICMP	86	Echo (ping) request id=0xaeae0, seq=1/256, ttl=1 (no response found!)
3	0.157331	172.26.47.109	193.136.9.254	ICMP	86	Echo (ping) request id=0xaeae0, seq=2/512, ttl=1 (no response found!)
5	0.203030	172.26.47.109	193.136.9.254	ICMP	86	Echo (ping) request id=0xaeae0, seq=3/768, ttl=1 (no response found!)
7	0.342569	172.26.47.109	193.136.9.254	ICMP	86	Echo (ping) request id=0xaeae0, seq=4/1024, ttl=2 (no response found!)
9	0.413475	172.26.47.109	193.136.9.254	ICMP	86	Echo (ping) request id=0xaeae0, seq=5/1280, ttl=2 (no response found!)
11	0.485798	172.26.47.109	193.136.9.254	ICMP	86	Echo (ping) request id=0xaeae0, seq=6/1536, ttl=2 (no response found!)
13	0.634184	172.26.47.109	193.136.9.254	ICMP	86	Echo (ping) request id=0xaeae0, seq=7/1792, ttl=3 (Reply in 14)
15	0.805865	172.26.47.109	193.136.9.254	ICMP	86	Echo (ping) request id=0xaeae0, seq=8/2048, ttl=3 (Reply in 16)
17	0.949080	172.26.47.109	193.136.9.254	ICMP	86	Echo (ping) request id=0xaeae0, seq=9/2304, ttl=3 (Reply in 18)
14	0.485865	193.136.9.254	172.26.47.109	ICMP	86	Echo (ping) reply id=0xaeae0, seq=7/1792, ttl=253 (request in 13)
16	0.948892	193.136.9.254	172.26.47.109	ICMP	86	Echo (ping) reply id=0xaeae0, seq=8/2048, ttl=253 (request in 15)
18	0.987817	193.136.9.254	172.26.47.109	ICMP	86	Echo (ping) reply id=0xaeae0, seq=9/2304, ttl=253 (request in 17)

Figura 1.4: Tráfego capturado

1.2.2 Alínea b

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

Analizando o pacote apresentado na figura 1.5, verifica-se que o valor do campo protocolo é 1, identificando, assim, o protocolo ICMP.

```

> Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface en0, id 0
> Ethernet II, Src: Apple_d1:29:eb (38:f9:d3:d1:29:eb), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
  ▾ Internet Protocol Version 4, Src: 172.26.47.109, Dst: 193.136.9.254
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 72
    Identification: 0xae0f (44559)
    Flags: 0x00
    Fragment Offset: 0
    ▶ Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x6498 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.47.109
    Destination Address: 193.136.9.254
  ▶ Internet Control Message Protocol

```

Figura 1.5: Endereço IP da interface

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*?

Voltando a analisar o pacote apresentado na imagem 1.5, verifica-se que o campo *Header Length* tem o valor de 20 bytes. O campo de dados terá 52 bytes. Este valor é obtido subtraindo o comprimento do *header* ao valor do campo *Total Length*.

1.2.4 Alínea d

O datagrama IP foi fragmentado?

Expandi-se o campo relativo às flags do protocolo IP, obtendo-se a figura 1.6. Nesta, verifica-se que o campo *More Fragments* tem o valor 0. Isto poderá representar 2 cenários: ou o datagrama não foi fragmentado ou este é o último fragmento do datagrama original.

Analizando o campo *Fragment Offset*, verifica-se que este tem o valor 0, o que confirma o primeiro dos dois cenários, ou seja, **o datagrama IP não foi fragmentado**, pois, no segundo cenário, este campo não poderia ter o valor 0 ao mesmo tempo que a flag anteriormente mencionada se encontrava também a 0, pois o fragmento final teria um *offset* obrigatoriamente.

```

> Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface en0, id 0
> Ethernet II, Src: Apple_d1:29:eb (38:f9:d3:d1:29:eb), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
  ▾ Internet Protocol Version 4, Src: 172.26.47.109, Dst: 193.136.9.254
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 72
    Identification: 0xae0f (44559)
    Flags: 0x00
      0... .... = Reserved bit: Not set
      .0... .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    Fragment Offset: 0
    ▶ Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x6498 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.47.109
    Destination Address: 193.136.9.254
  ▶ Internet Control Message Protocol

```

Figura 1.6: Flags do protocolo IP

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.

Ordenaram-se os pacotes de acordo com a coluna *Source* e observaram-se dois dos pacotes, estando a análise destes representada na figura 1.7.

Da sua observação, resultou a conclusão que existem 3 campos cujos valores variam consoante o pacote. Estes são o campo *Identification*, único a cada pacote uno, o campo *Header Checksum* e, devido à natureza do funcionamento do *traceroute*, o campo *Time to live*.

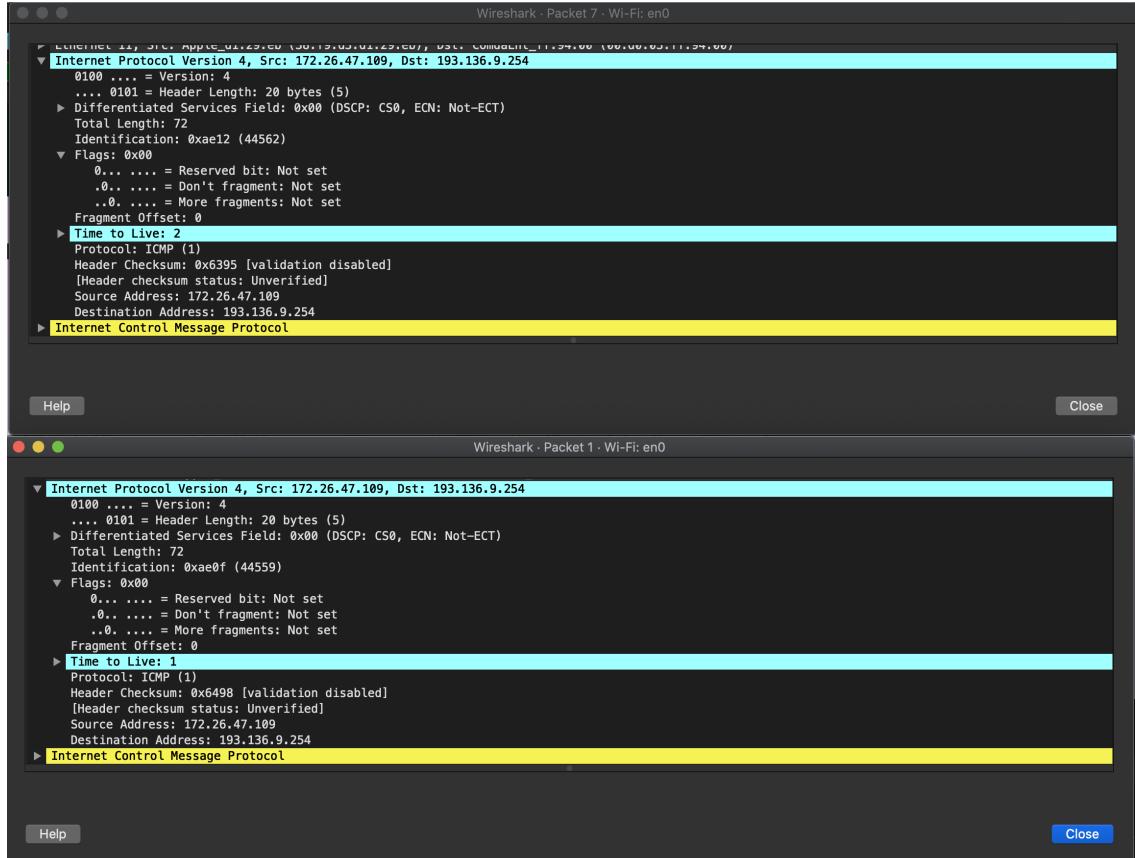


Figura 1.7: Pacotes ordenados por Source

1.2.6 Alínea f

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

Com base no tráfego observado, conclui-se que a atribuição dos valores no campo identificativo é sequencialmente incremental.

No que concerne o valor do campo TTL, verificamos que estes se alinham com o funcionamento do comando *traceroute*. Existem conjuntos de 3 pacotes com TTLs iguais e futuros conjuntos incrementam o seu TTL por 1 até ser estabelecida ligação com o destinatário pretendido.

1.2.7 Alínea g

Ordene o tráfego capturado por endereço destino e encontre a série de respostas ICMP TTL exceeded enviadas ao seu computador. Qual é o valor do campo TTL?

Esse valor permanece constante para todas as mensagens de resposta ICMP TTL exceeded enviados ao seu host? Porquê?

Foram observadas várias respostas TTL *exceeded* enviadas como resposta do comando *traceroute*. Duas destas respostas foram analisadas, como indica a figura 1.8.

Como se consegue verificar, o valor do campo TTL não é constante, observando-se tanto o valor 255 como o 254.

Isto deve-se ao funcionamento do comando *traceroute*. Os pacotes que são enviados com TTL=1 serão rejeitados ao fim do primeiro passo, sendo enviada uma mensagem de retorno que apenas dará um salto e, como tal, terá como TTL o valor 255, sendo 256 o valor atribuído à mensagem pelo router onde o pacote excedeu o seu TTL. Pacotes que sejam rejeitados ao fim do segundo salto retornarão mensagens com TTL=254, visto que essas mensagens terão de executar dois saltos antes de retornarem à interface que enviou o pacote original.

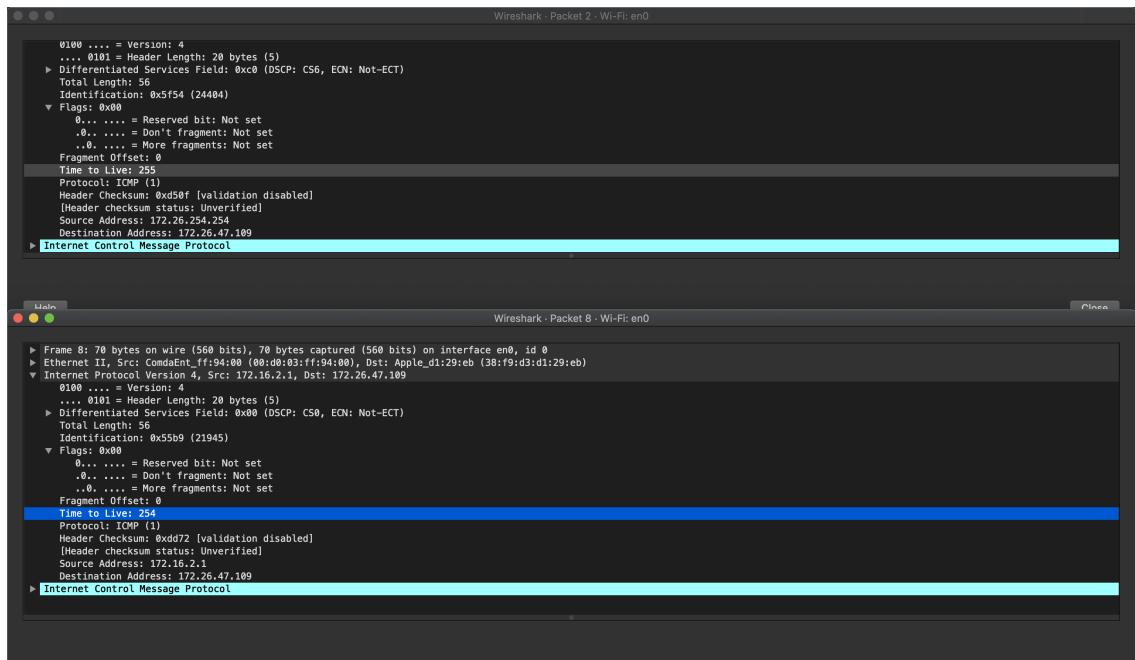


Figura 1.8: Respostas TTL exceeded

1.3 Exercício 3

11	2.287274	172.26.47.109	193.136.9.254	ICMP	1514	Echo (ping) request id=0xaf18, seq=1/256, ttl=1 (no response found!)
12	2.287275	172.26.47.109	193.136.9.254	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=faf19)
13	2.287276	172.26.47.109	193.136.9.254	IPv4	385	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=faf19)
14	2.289585	172.26.47.109	172.26.47.109	TLSv1.2	1	Application Data
15	2.289587	172.26.47.109	172.26.47.109	TCP	66	31915 > 443 (ACK) Seq=36 Ack=32 Win=2047 Len=0 TStamp=34785472 TSecr=1211767016
16	2.290371	172.26.47.109	193.136.9.254	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
17	2.292865	172.26.47.109	193.136.9.254	ICMP	1514	Echo (ping) request id=0xaf18, seq=2/512, ttl=1 (no response found!)
18	2.292866	172.26.47.109	193.136.9.254	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=faf1a)
19	2.292867	172.26.47.109	193.136.9.254	IPv4	385	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=faf1a)
20	2.294958	172.26.254.254	172.26.47.109	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
21	2.295071	172.26.47.109	193.136.9.254	ICMP	1514	Echo (ping) request id=0xaf18, seq=3/768, ttl=1 (no response found!)
22	2.295072	172.26.47.109	193.136.9.254	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=faf1b)
23	2.295073	172.26.47.109	193.136.9.254	IPv4	385	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=faf1b)
24	2.298187	172.26.254.254	172.26.47.109	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
25	2.298215	172.26.47.109	193.136.9.254	ICMP	1514	Echo (ping) request id=0xaf18, seq=4/1024, ttl=2 (no response found!)
26	2.298216	172.26.47.109	193.136.9.254	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=faf1c)
27	2.298217	172.26.47.109	193.136.9.254	IPv4	385	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=faf1c)
28	2.298218	172.26.47.109	172.26.47.109	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
29	2.301558	172.26.47.109	193.136.9.254	DNS	83	Standard query 0xd6d6 PTR 1.2.16.172.in-addr.arpa
30	2.303569	193.137.16.65	172.26.47.109	DNS	83	Standard query response 0x9dd6 Refused PTR 1.2.16.172.in-addr.arpa
31	2.304288	172.26.47.109	193.137.16.145	DNS	83	Standard query 0xd6d6 PTR 1.2.16.172.in-addr.arpa
32	2.307070	193.137.16.145	172.26.47.109	DNS	83	Standard query response 0x9dd6 Refused PTR 1.2.16.172.in-addr.arpa
33	2.307345	172.26.47.109	193.137.16.75	DNS	83	Standard query 0xd6d6 PTR 1.2.16.172.in-addr.arpa
34	2.309751	193.137.16.75	172.26.47.109	DNS	83	Standard query response 0x9dd6 Refused PTR 1.2.16.172.in-addr.arpa
35	3.047818	172.26.47.109	239.255.255.250	SSDP	212	M-SEARCH * HTTP/1.1
36	3.09234	172.26.47.109	193.136.9.254	ICMP	1514	Echo (ping) request id=0xaf18, seq=5/1280, ttl=2 (no response found!)
37	3.09235	172.26.47.109	193.136.9.254	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=faf1d)
38	3.09236	172.26.47.109	193.136.9.254	IPv4	385	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=faf1d)
39	3.320021	172.16.2.1	172.26.47.109	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
40	3.320246	172.26.47.109	193.136.9.254	ICMP	1514	Echo (ping) request id=0xaf18, seq=6/1536, ttl=2 (no response found!)
41	3.320447	172.26.47.109	193.136.9.254	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=faf1e)
42	3.320448	172.26.47.109	193.136.9.254	IPv4	385	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=faf1e)
43	3.320355	172.16.2.1	172.26.47.109	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
44	3.323493	172.26.47.109	193.136.9.254	ICMP	1514	Echo (ping) request id=0xaf18, seq=7/1792, ttl=2 (reply in 47)
45	3.323538	172.26.47.109	193.136.9.254	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=faf1f)
46	3.323539	172.26.47.109	193.136.9.254	IPv4	385	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=faf1f)
47	3.338598	193.136.9.254	172.26.47.109	ICMP	1514	Echo (ping) reply id=0xaf18, seq=7/1792, ttl=253 (request in 44)
48	3.331446	193.136.9.254	172.26.47.109	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=faf1f)
49	3.331463	193.136.9.254	172.26.47.109	IPv4	385	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=faf1f)
50	3.332723	172.26.47.109	193.136.9.254	ICMP	1514	Echo (ping) request id=0xaf18, seq=8/2048, ttl=3 (reply in 53)
51	3.332724	172.26.47.109	193.136.9.254	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=faf20)
52	3.332726	172.26.47.109	193.136.9.254	IPv4	385	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=faf20)

Figura 1.9: Fragmentos de datagramas detetados

1.3.1 Alínea a

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

Analisando a figura 1.9, verificamos que a primeira mensagem ICMP é a 11. Uma vez que o tamanho permitido pelo protocolo (1514 bytes) é inferior ao tamanho do PDU(protocol data unit) (3251 bytes) torna-se necessário fragmentar o pacote para que seja possível fazer este atravessar a 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?

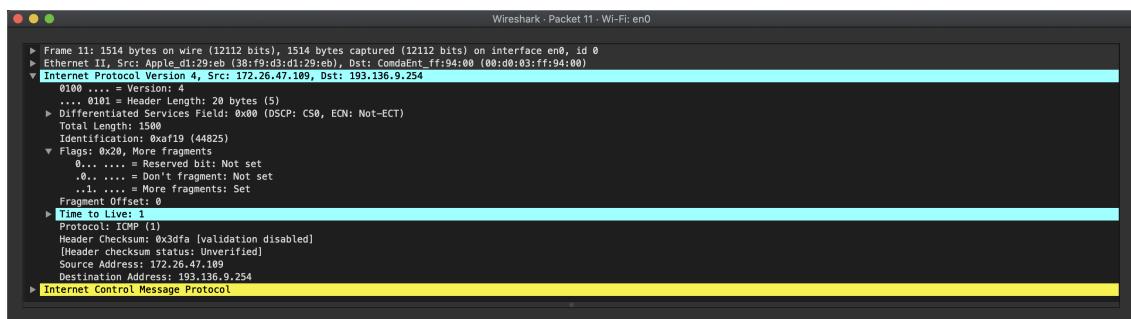


Figura 1.10: Cabeçalho do primeiro fragmento do datagrama IP

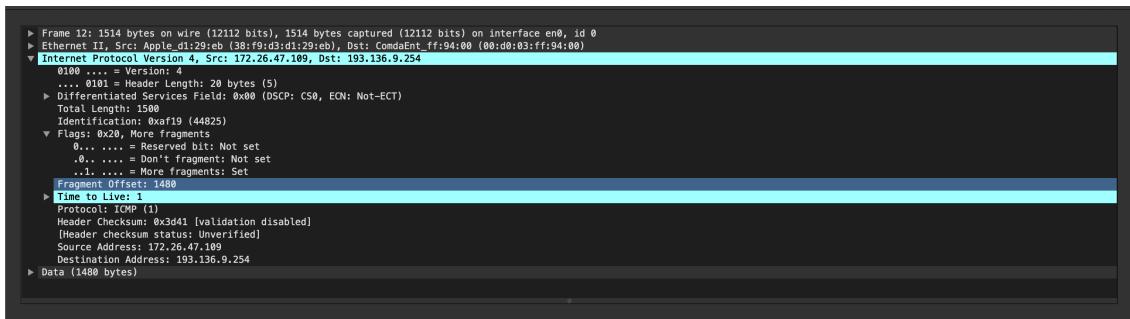
Observando o pacote exposto na figura 1.10, denota-se que o valor da flag *More fragments*, se encontra a um, indicando que existirão mais fragmentos relativos ao mesmo pacote original.

O valor do campo *Fragment offset*, encontra-se a 0. Esta informação, conjugada com a informação retirada da flag anterior, indica que este é o primeiro fragmento do datagrama original.

O tananho deste fragmento é de 1500 bytes, conforme observado no campo *Total Length*.

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 12: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface en0, id 0
Ethernet II, Src: Apple_d1:29:eb (38:f9:d3:d1:29:eb), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.47.109, Dst: 193.136.9.254
    ... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSFP: CS0, ECN: Not-ECT)
        Total Length: 1500
        Identification: 0xa19 (44825)
    > Flags: 0x20, More fragments
        0... .... = Reserved bit: Not set
        .0... .... = Don't fragment: Not set
        ...1.... = More fragments: Set
        Fragment offset: 1480
    > Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0xd4d1 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.47.109
    Destination Address: 193.136.9.254
    > Data (1480 bytes)

```

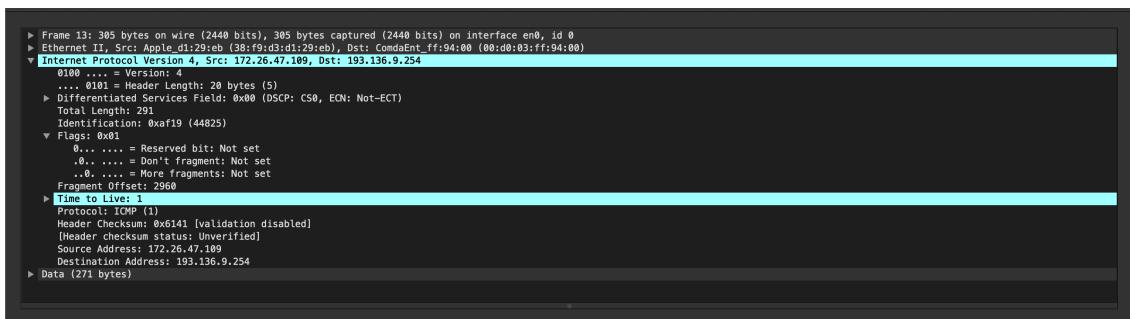
Figura 1.11: Cabeçalho do segundo fragmento do datagrama IP

Observando o campo *Fragment Offset*, verificamos que o seu valor se encontra a 1480. Isto denota imediatamente que se trata de um fragmento de um datagrama IP e que obrigatoriamente não é o primeiro, pois, nesse caso, o valor do campo deveria ser 0.

Visto que a flag *More fragments* continua ativa, conclui-se que existem ainda mais fragmentos do datagrama original.

1.3.4 Alínea d

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



```

Frame 13: 305 bytes on wire (2440 bits), 305 bytes captured (2440 bits) on interface en0, id 0
Ethernet II, Src: Apple_d1:29:eb (38:f9:d3:d1:29:eb), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.47.109, Dst: 193.136.9.254
    ... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSFP: CS0, ECN: Not-ECT)
        Total Length: 291
        Identification: 0xa19 (44825)
    > Flags: 0x01
        0... .... = Reserved bit: Not set
        .0... .... = Don't fragment: Not set
        ...1.... = More fragments: Not set
        Fragment offset: 2960
    > Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x6141 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.47.109
    Destination Address: 193.136.9.254
    > Data (271 bytes)

```

Figura 1.12: Cabeçalho do último fragmento do datagrama IP

O fragmento final de um datagrama deverá conter a flag *More fragments* com o valor 0, retendo, no entanto, a mesma identificação que os fragmentos anteriores.

É esse o comportamento observado no datagrama da figura 1.12. O datagrama original foi, portanto, fragmentado em 3: os pacotes 11,12 e 13.

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.

Os dois campos do cabeçalho IP que variam entre fragmentos do mesmo datagrama IP serão a flag *More fragments* e o *Fragment offset*. O *Header Checksum* também variará, porém não é um campo relevante para a ordenação dos vários fragmentos.

O primeiro fragmento será aquele cujo valor do campo *Fragment Offset* se encontre a 0. Enquanto a flag *More fragments* se encontre ativa, dever-se-á procurar por mais fragmentos, cujo valor do campo *Fragment Offset* será incremental, tendo em conta o tamanho de fragmentos anteriores.

O fragmento final será aquele cujo valor da flag *More Fragments* se encontre a 0.

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.

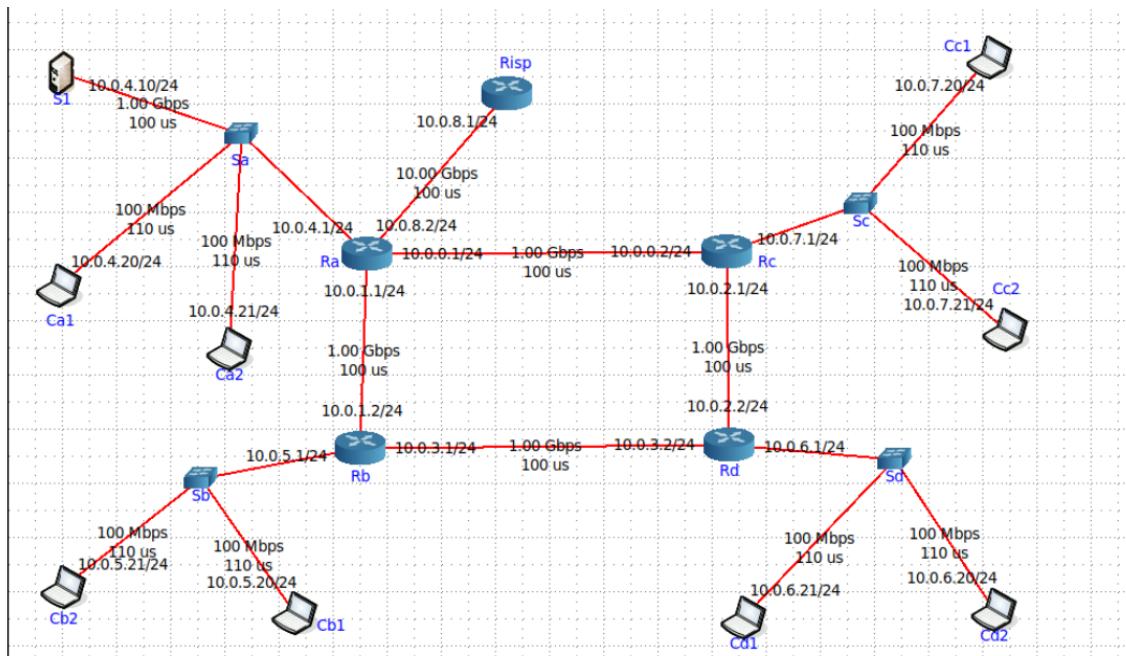


Figura 2.1: Topologia da rede

A figura 2.1 representa a topologia da rede proposta, definida no core. Nesta, podem-se observar os endereços IP e máscaras atribuídos a cada elemento da topologia.

2.1.2 Alínea b

Trata-se de endereços públicos ou privados? Porquê?

Para a atribuição de endereços privados, são utilizadas as gamas $10.0.0.0 - 10.255.255.255 /8$, $172.16.0.0 - 172.31.255.255 /12$ e $192.168.0.0 - 192.168.255.255 /16$.

Visto que os endereços atribuídos pelo Core se encontram dentro da primeira gama, conclui-se que estes são privados.

2.1.3 Alínea c

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

Os switches Ethernet não utilizam endereços IP visto a sua operação se centra na camada 2 do modelo *OSI*, distribuindo os pacotes segundo os *MAC addresses* de cada equipamento, ao invés do seu endereço IP, que são processados na camada 3 do modelo *OSI*.

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).

The figure consists of four separate terminal windows, each showing the output of a ping command. The windows are arranged in a 2x2 grid. Top-left window: root@Ca1:/tmp/pycore.40365/Ca1.conf# ping 10.0.4.10. Top-right window: root@Cc1:/tmp/pycore.40365/Cc1.conf# ping 10.0.4.10. Bottom-left window: root@Cb2:/tmp/pycore.40365/Cb2.conf# ping 10.0.4.10. Bottom-right window: root@Cd1:/tmp/pycore.40365/Cd1.conf# ping 10.0.4.10. Each window displays the ping results with various time values (e.g., 2.21 ms, 1.32 ms, etc.) and sequence numbers (seq=1, seq=2, seq=3).

Figura 2.2: Pings entre computadores dos vários departamentos e o servidor

De forma a verificar a conectividade entre os computadores dos vários departamentos e o servidor no departamento A, utilizou-se o comando *ping* a partir de um computador de cada departamento e tentou-se estabelecer uma conexão com o servidor. Como ilustrado na figura 2.2, foi possível estabelecer estas relações, certificando que existe conectividade IP.

2.1.5 Alínea e

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

A single terminal window titled 'root@Risp:/tmp/pycore.40365/Risp.conf'. It shows the command 'root@Risp:/tmp/pycore.40365/Risp.conf# ping 10.0.4.10' followed by the ping results. The results show three successful packets sent to the address 10.0.4.10, with times ranging from 2.56 ms to 2.19 ms and sequence numbers seq=1, seq=2, and seq=3.

Figura 2.3: Conectividade entre router RISP e servidor S1

Utilizou-se novamente o comando *ping*, desta vez a partir do router de acesso e endereçado ao endereço IP do servidor S1. Como se observa da figura 2.3, foi possível estabelecer uma ligação, confirmando-se a conectividade entre os dois.

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*).

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.0.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
10.0.1.0	10.0.0.1	255.255.255.0	UG	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	10.0.0.1	255.255.255.0	UG	0	0	0	eth0
10.0.5.0	10.0.0.1	255.255.255.0	UG	0	0	0	eth0
10.0.6.0	10.0.2.2	255.255.255.0	UG	0	0	0	eth1
10.0.7.0	0.0.0.0	255.255.255.0	U	0	0	0	eth2
10.0.8.0	10.0.0.1	255.255.255.0	UG	0	0	0	eth0

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
0.0.0.0	10.0.7.1	0.0.0.0	UG	0	0	0	eth0
10.0.7.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0

Figura 2.4: Tabela de encaminhamento do router do Departamento C (esquerda) e de um computador do Departamento C (direita)

Através da Figura 2.4 podemos observar duas tabelas de encaminhamento, uma de um router de um departamento e outra de um computador do mesmo departamento. As informações importantes de retirar são das colunas *Destination*, *Gateway*, *GenMask*, *Flags* e *Iface*.

A coluna *Destination* representa um endereço possível ao qual o pacote poderá ser dirigido, seguido pela coluna *Gateway*, que representa o próximo nó que o pacote terá de ser enviado para chegar ao seu destino. A coluna *Genmask* será a máscara da subrede usada para identificar a subrede em causa. Em seguida temos a coluna *Flags* que representam as observações do routing em causa, neste caso, a flag U indica que a rota é "Usable" ou seja, está disponível para uso, e a flag G indica que a rota requere um intermediário para ser completa ("forwarding"). Por fim, a coluna *Iface* indica a interface à qual o pacote se irá dirigir para retomar a sua rota.

É possível observar que pode haver linhas em que a destination seria 0.0.0.0, isto consiste na rota *Default* que indica um retorno do pacote de dados ao router de serviço.

Também deteta-se que há rotas em que o Gateway seria 0.0.0.0, isto indica que como IP de destino está diretamente ligado ao IP de fonte, não será necessário alcançar um próximo nó para chegar ao seu destino.

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, por exemplo, ps -ax).

The figure consists of two terminal windows side-by-side. The left window is titled 'root@Rc:/tmp/pycore.40365/Rc.conf' and shows the command 'ps -ax'. The output is as follows:

PID	TTY	STAT	TIME	COMMAND
1	?	S	0:00	/usr/local/bin/vnoded -v -c /tmp/pycore.40365/Rc -l /
58	?	Ss	0:00	/usr/sbin/zebra -d
64	?	Ss	0:00	/usr/sbin/ospf6d -d
68	?	Ss	0:00	/usr/sbin/ospfd -d
76	pts/5	Ss	0:00	/bin/bash
85	pts/5	R+	0:00	ps -ax

The right window is titled 'root@Cc1:/tmp/pycore.40365/Cc1.conf' and shows the command 'ps -ax'. The output is as follows:

PID	TTY	STAT	TIME	COMMAND
1	?	S	0:00	/usr/local/bin/vnoded -v -c /tmp/pycore.40365/Cc1 -l
45	pts/3	Ss	0:00	/bin/bash
55	pts/3	R+	0:00	ps -ax

Figura 2.5: Processos a serem executados em Rc e Cc1

Através da análise da figura 2.5 observa-se que no router do departamento C corre um processo *ospfd* que consiste num protocolo de reencaminhamento. Para os computadores esse protocolo não se encontra em execução uma vez que não é necessário pois estes apenas se precisam de preocupar de estabelecer rotas locais, ou seja, na mesma subrede, logo, podemos concluir que se trata de um encaminhamento dinâmico.

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.

The figure shows a single terminal window titled 'root@S1:/tmp/pycore.40365/S1.conf'. It contains the following commands and their outputs:

```
root@S1:/tmp/pycore.40365/S1.conf# route del default
root@S1:/tmp/pycore.40365/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
```

Figura 2.6: Remoção da rota default do servidor S1

```

root@S1:/tmp/pycore.40365/S1.conf# ping 10.0.8.1
connect: Network is unreachable
root@S1:/tmp/pycore.40365/S1.conf# 

root@Cb1:/tmp/pycore.40365/Cb1.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 ---
2 packets transmitted, 0 received, 100% packet loss, time 1029ms
root@Cb1:/tmp/pycore.40365/Cb1.conf# 

```

Figura 2.7: Conectividade de S1 para o Risp (janela de cima) e conectividade de Cb1 para S1 (janela de baixo)

Ao remover a rota default do Servidor S1, este não terá qualquer conexão para fora do seu departamento uma vez que não está definida qualquer rota para além da sua subrede. Logo, todos os pacotes que chegam ao servidor S1 vão ser recebidos mas não respondidos, daí, na janela de baixo da figura 2.7 se conseguir observar que o pacote foi transmitido mas não foi recebido, resultando em 100% de perda de pacote. Por outro lado, o servidor S1 não poderá transmitir qualquer pacote de dados para fora da sua subrede, como se pode ver na janela superior da figura 2.7, o servidor S1 tenta comunicar um pacote para o Risp sem sucesso.

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.40365/S1.conf# route add -net 10.0.0.0 netmask 255.255.255.0 gw 10.0.4.1
root@S1:/tmp/pycore.40365/S1.conf# route add -net 10.0.1.0 netmask 255.255.255.0 gw 10.0.4.1
root@S1:/tmp/pycore.40365/S1.conf# route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.4.1
root@S1:/tmp/pycore.40365/S1.conf# route add -net 10.0.3.0 netmask 255.255.255.0 gw 10.0.4.1
root@S1:/tmp/pycore.40365/S1.conf# route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.4.1
root@S1:/tmp/pycore.40365/S1.conf# route add -net 10.0.6.0 netmask 255.255.255.0 gw 10.0.4.1
root@S1:/tmp/pycore.40365/S1.conf# route add -net 10.0.7.0 netmask 255.255.255.0 gw 10.0.4.1
root@S1:/tmp/pycore.40365/S1.conf# route add -net 10.0.8.0 netmask 255.255.255.0 gw 10.0.4.1
root@S1:/tmp/pycore.40365/S1.conf# 

```

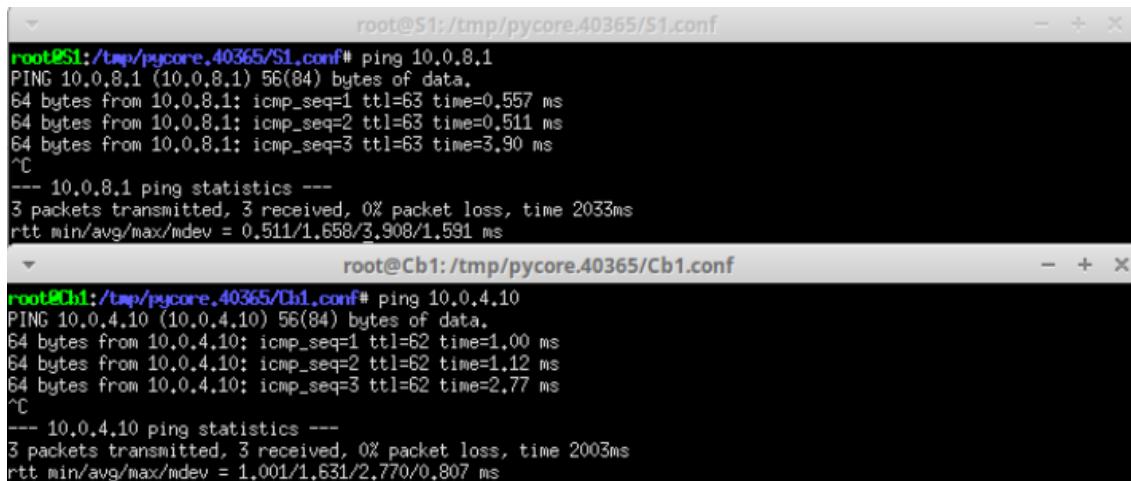
Figura 2.8: Adição manual das rotas estáticas

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.

```
root@S1:/tmp/pycore.40365/S1.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask        Flags MSS Window irtt Iface
10.0.0.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.1.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.2.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.3.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.4.0        0.0.0.0        255.255.255.0 U          0 0          0 eth0
10.0.5.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.6.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.7.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
10.0.8.0        10.0.4.1       255.255.255.0 UG        0 0          0 eth0
```

Figura 2.9: Tabela de encaminhamento de S1



The figure consists of two terminal windows. The top window, titled 'root@S1:/tmp/pycore.40365/S1.conf', shows the command 'ping 10.0.8.1' being run. The output indicates three packets were transmitted to 10.0.8.1, with no packet loss and a round-trip time (RTT) of 2033ms. The bottom window, titled 'root@Cb1:/tmp/pycore.40365/Cb1.conf', shows the command 'ping 10.0.4.10' being run. The output indicates three packets were transmitted to 10.0.4.10, with no packet loss and an RTT of 2003ms.

```
root@S1:/tmp/pycore.40365/S1.conf# ping 10.0.8.1
PING 10.0.8.1 (10.0.8.1) 56(84) bytes of data.
64 bytes from 10.0.8.1: icmp_seq=1 ttl=63 time=0.557 ms
64 bytes from 10.0.8.1: icmp_seq=2 ttl=63 time=0.511 ms
64 bytes from 10.0.8.1: icmp_seq=3 ttl=63 time=3.90 ms
^C
--- 10.0.8.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.511/1.658/3.908/1.591 ms

root@Cb1:/tmp/pycore.40365/Cb1.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=1.00 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=1.12 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=2.77 ms
^C
--- 10.0.4.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.001/1.631/2.770/0.807 ms
```

Figura 2.10: Conectividade entre S1 e Risp (janela de cima) e conectividade entre Cb1 e S1 (janela de baixo)

Observando a figura 2.10 verifica-se que o servidor S1 volta a estar devidamente conectado com os vários departamentos uma vez que é detetada uma *packet loss* de 0%, ou seja, todos os pacotes de dados que foram transmitidos para o servidor conseguiram dar resposta à fonte e todos os pacotes que foram transmitidos do próprio servidor deram também resposta a este.

2.3 Exercício 3

2.3.1 Alínea 1

Considere que dispõe apenas do endereço de rede IP 130.XX.96.0/19, em que XX é o decimal correspondendo ao seu número de grupo (PLXX). 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. Assuma que todos os endereços de sub-redes são usáveis. Deve justificar as opções usadas.

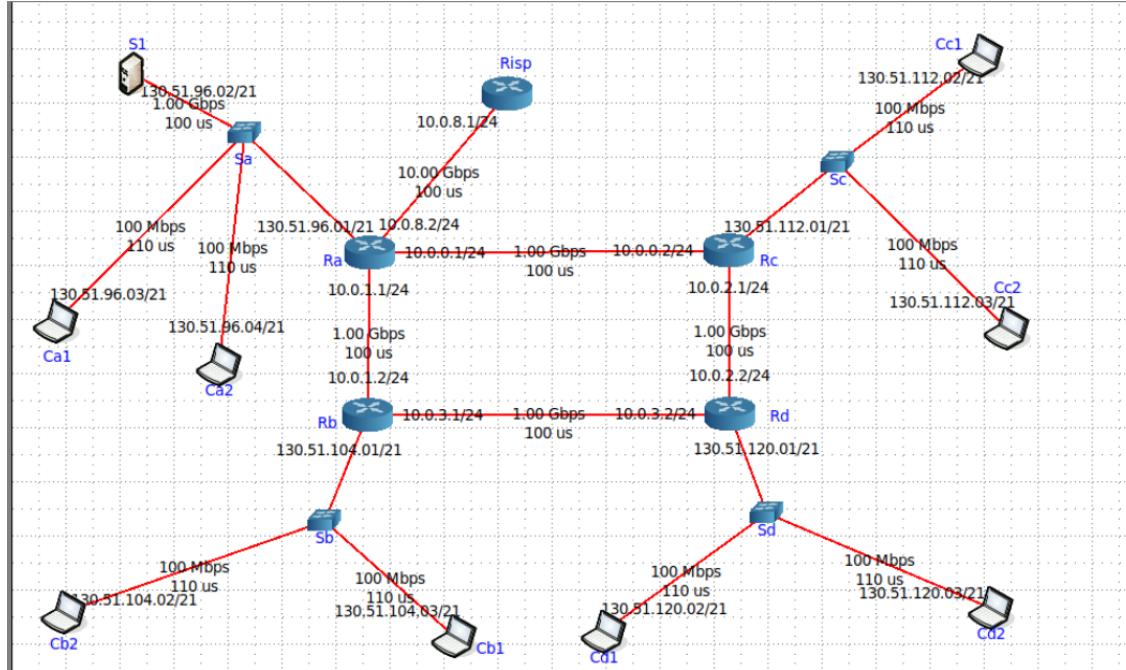


Figura 2.11: Novo esquema de endereçamento

Sendo o nosso grupo o número 51, é-nos atribuído o endereço de rede IP 130.51.96.0/19

A existência de 4 departamentos exige a definição de uma subrede para cada um, o que implica a necessidade de utilizar, no mínimo, 2 bits para a definição destas. Visto não ser indicada a intenção de expandir o número de departamentos da organização, opta-se pela utilização de apenas 2 bits para o endereçamento das subredes.

O endereçamento regir-se-á pelo esquema: **10000010 00110011 011—yyxxxx xxxx xxxx/21**, em que a secção *y* corresponderá à subrede atribuída ao departamento e os restantes 11 bits de *x*, à identificação do host dentro da subrede.

Sendo assim, explicitam-se as subredes atribuídas a cada departamento:

- Departamento A: 130.51.96.0/21
- Departamento B: 130.51.104.0/21
- Departamento C: 130.51.112.0/21
- Departamento D: 130.51.120.0/21

2.3.2 Alínea 2

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

A máscara utilizada será a /21 em notação CIDR, correspondente a 255.255.248.0.

Como referido na alínea anterior, restarão 11 bits para a identificação de hosts dentro de cada subrede, o que resulta em **2046** possíveis hosts dentro de cada departamento, visto removerem-se os dois endereços reservados do total de 2^{11} hosts possíveis.

2.3.3 Alínea 3

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

```

root@Cb1:/tmp/pycore.40365/Cb1.conf# ping 130.51.96.03
PING 130.51.96.03 (130.51.96.3) 56(84) bytes of data.
64 bytes from 130.51.96.3: icmp_seq=1 ttl=62 time=5.37 ms
64 bytes from 130.51.96.3: icmp_seq=2 ttl=62 time=1.97 ms
64 bytes from 130.51.96.3: icmp_seq=3 ttl=62 time=4.13 ms
^C
--- 130.51.96.03 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.970/3.825/5.374/1.406 ms
root@Cb1:/tmp/pycore.40365/Cb1.conf# ping 130.51.112.02
PING 130.51.112.02 (130.51.112.2) 56(84) bytes of data.
64 bytes from 130.51.112.2: icmp_seq=1 ttl=61 time=4.73 ms
64 bytes from 130.51.112.2: icmp_seq=2 ttl=61 time=2.97 ms
64 bytes from 130.51.112.2: icmp_seq=3 ttl=61 time=2.76 ms
^C
--- 130.51.112.02 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.769/3.491/4.733/0.883 ms
root@Cb1:/tmp/pycore.40365/Cb1.conf# ping 130.51.120.02
PING 130.51.120.02 (130.51.120.2) 56(84) bytes of data.
64 bytes from 130.51.120.2: icmp_seq=1 ttl=62 time=0.689 ms
64 bytes from 130.51.120.2: icmp_seq=2 ttl=62 time=0.732 ms
64 bytes from 130.51.120.2: icmp_seq=3 ttl=62 time=0.875 ms
^C
--- 130.51.120.02 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.689/0.765/0.875/0.082 ms
root@Cb1:/tmp/pycore.40365/Cb1.conf# ping 10.0.8.1
PING 10.0.8.1 (10.0.8.1) 56(84) bytes of data.
64 bytes from 10.0.8.1: icmp_seq=1 ttl=62 time=1.72 ms
64 bytes from 10.0.8.1: icmp_seq=2 ttl=62 time=3.36 ms
64 bytes from 10.0.8.1: icmp_seq=3 ttl=62 time=2.46 ms
^C
--- 10.0.8.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.727/2.520/3.389/0.671 ms
root@Cb1:/tmp/pycore.40365/Cb1.conf#

```

Figura 2.12: Teste da conectividade

Para testar a conectividade IP entre as várias redes, apenas é necessário verificar que um host inserido numa das subredes tem conectividade com um host inserido em cada uma das outras subredes.

Para tal fim, foi escolhido o PC **Cb1**, pertencente ao **departamento B**. Testou-se a conectividade com **Ca1**, **Cc1** e **Cd1**, assim como com o router **Risp**.

A figura 2.12 apresenta o resultado do vos vários comandos *ping*, que confirmam a conectividade pretendida.

Visto que as tabelas de endereçamento foram geradas automaticamente pelo CORE e estando provada a correção da lógica de endereçamento, não se julgou necessário repetir o teste tendo como origem hosts nos outros departamentos.

Capítulo 3

Conclusão

Este trabalho permitiu a exploração de dois temas muito relevantes na área de redes de computadores: **Datagramas IP e Fragmentação** e **Endereçamento e Encaminhamento IP**

Na primeira parte, correspondente ao tema **Datagramas IP e Fragmentação**, a utilização da ferramenta de emulação de redes Core, associada à utilização também do Wireshark, uma ferramenta de captação e análise de pacotes, explorou-se a constituição dos datagramas IP e o funcionamento do comando **traceroute**.

Alterando o tamanho dos pacotes enviados, foi também explorada a mecânica de fragmentação, através da verificação das flags e campos associados nos cabeçalhos dos datagramas.

Na segunda parte, correspondente ao tema **Endereçamento e Encaminhamento IP**, foi formulada uma nova topologia mais complexa, relativamente à utilizada na primeira parte. Este topologia mimicava uma situação habitual de uma organização com vários departamentos, cada um com uma subrede associada.

Explorou-se a mecânica de redirecionamento de pacotes IP, através da inspeção e manipulação das tabelas de encaminhamento dos vários routers presentes na topologia.

A definição manual do esquema de endereçamento permitiu consolidar os conhecimentos acerca da mecânica de *subnetting*, associando a exploração de máscaras de rede e subrede.

Em conclusão, julgamos ter alcançado o aproveitamento desejável, o que permitiu aprofundar e consolidar o conhecimento nas duas temáticas supramencionadas.