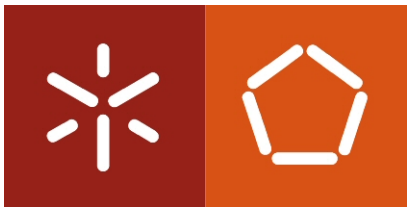


UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO DE ENGENHARIA INFORMÁTICA

Redes de Computadores

TP2: Protocolo IP

Autores:

Frederico Pinto	A73639	
Pedro Silva	A78434	
Ricardo Leal	A75411	

1 de Abril de 2018

1 Introdução

Este relatório diz respeito ao segundo trabalho prático da Unidade Curricular Redes de Computadores, que se baseia nas experiências propostas no enunciado.

Com o auxílio de ferramentas fornecidas (CORE e WIRESHARK) pretende-se explorar e compreender o funcionamento do protocolo IP (Internet Protocol) e das suas vertentes. A primeira parte do trabalho é direcionada para a análise de datagramas enviados e recebidos com recurso ao programa traceroute e uma ligação á rede Ethernet da sala. A segunda parte recorre à emulação de uma arquitetura de uma rede local de topologia CORE para estudar endereçamento e encaminhamento IP entre outras técnicas mais complexas. Para além dos principais temas, são alvo de estudo outros tópicos tais como máscaras de rede (sub-rede), fragmentação de IP, subnetting (sub-redes), entre outras.

2 Parte I - Datagramas IP e Fragmentação

2.1 Questão 1

Definimos a topologia da seguinte forma:

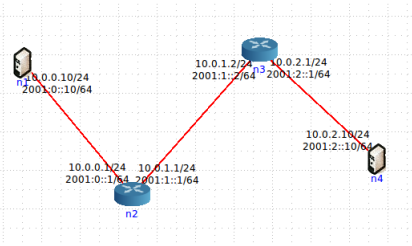


Figura 1. Topologia de rede.

Ativamos o wireshark em *n1* e executamos **traceroute –I 10.0.2.10** (IPv4 de *n4*), obtendo:

```
root@n1:/tmp/pycore.41287/n1.conf# traceroute -I 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  A0 (10.0.0.1)  0.106 ms  0.016 ms  0.014 ms
 2  10.0.1.2 (10.0.1.2)  0.085 ms  0.022 ms  0.021 ms
 3  10.0.2.10 (10.0.2.10)  0.065 ms  0.029 ms  0.028 ms
```

Figura 2. traceroute -I 10.0.2.10 em *n1*.

permitindo estimar o valor do *Round-Trip Time*. Seja μ_{ij} o tempo médio desde o nodo i até j. Ora, assumindo que $\mu_{14} \approx \mu_{41}$, vamos ter:

$RTT \approx 2(\mu_{12} + \mu_{23} + \mu_{34}) = 2((0.106 + 0.016 + 0.014)/3 + (0.085 + 0.022 + 0.021)/3 + (0.062 + 0.029 + 0.028)/3) \approx 0.2553ms$

No.	Time	Source	Destination	Protocol	Length	Info
5	18.270710	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=1/256, ttl=1
6	18.270736	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
7	18.270736	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=2/512, ttl=1
8	18.270736	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
9	18.270736	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=3/768, ttl=1
10	18.270736	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
11	18.270736	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=4/1024, ttl=2
12	18.270753	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
13	18.270770	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=5/1280, ttl=2
14	18.270790	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
15	18.270835	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=6/1536, ttl=2
16	18.270858	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
17	18.270872	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=7/1792, ttl=3
18	18.270907	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0057, seq=7/1792, ttl=62
19	18.270919	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=8/2048, ttl=3
20	18.270944	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0057, seq=8/2048, ttl=62
21	18.270953	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=9/2304, ttl=3
22	18.270978	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0057, seq=9/2304, ttl=62
23	18.270988	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=10/2560, ttl=4
24	18.271013	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0057, seq=10/2560, ttl=62
25	18.271022	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=11/2816, ttl=4
26	18.271047	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0057, seq=11/2816, ttl=62
27	18.271056	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=12/3072, ttl=4
28	18.271082	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0057, seq=12/3072, ttl=62
29	18.271092	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=13/3328, ttl=5
30	18.271117	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0057, seq=13/3328, ttl=62
31	18.271126	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=14/3584, ttl=5
32	18.271151	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0057, seq=14/3584, ttl=62
33	18.271160	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=15/3840, ttl=5
34	18.271185	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x0057, seq=15/3840, ttl=62
35	18.271195	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=16/4096, ttl=6
36	18.273375	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x0057, seq=17/4352, ttl=6

Figura 3. Tráfego capturado pelo Wireshark.

Dado que o comando **traceroute**, para descobrir o caminho entre dois endereços IP, envia datagramas com TTL inicialmente a 1 e incrementando sucessivamente, permite-nos observar qual o TTL mínimo necessário de um datagrama para chegar desde o IP origem até ao IP destino. Sendo o TTL decrementado uma unidade por cada router que passa, TTL = 0 resulta no envio de uma mensagem de controlo à origem, sendo o datagrama descartado. Ora, para esta topologia, estando o nodo origem e o nodo destino separados por 2 routers (3 saltos), podemos intuir que o TTL_{min} inicial para que um datagrama chegue até ao destino é 3, o que é confirmado na Figura 3.

2.2 Questão 2

Fazendo *traceroute* recorrendo ao pingplotter obtemos:

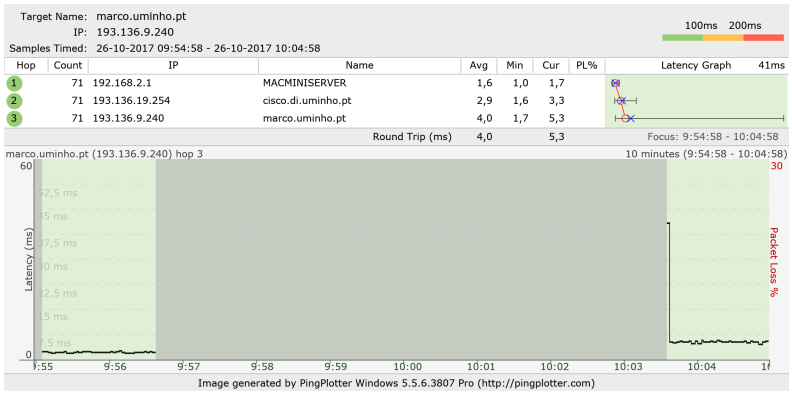
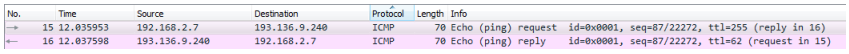


Figura 4. Pingplotter.

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

192.168.2.7



No.	Time	Source	Destination	Protocol	Length	Info
→ 15	12.035953	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=87/22272, ttl=255 (reply in 16)
← 16	12.037598	193.136.9.240	192.168.2.7	ICMP	70	Echo (ping) reply id=0x0001, seq=87/22272, ttl=62 (request in 15)

Figura 5. Primeira mensagem ICMP capturada.

A partir da Figura 6 sabemos que:
b) Qual é o valor do campo protocolo? O que identifica?

O campo *Protocol* vale 1 e identifica ICMP;

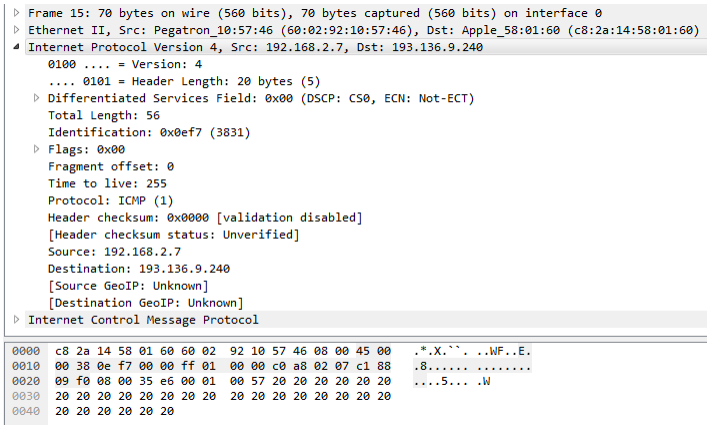
c) Quantos bytes tem o cabeçalho IP(v4)? Quantos bytes tem o campo de dados(payload) do datagrama?

Como se calcula o tamanho do payload?
Por *Header Length*, temos que o cabeçalho IPv4 tem 20 octetos. No total, pelo campo *Total Length*, temos 56 bytes, o que significa que o *payload* tem 36 bytes, sendo

$$payload = Total\ Length - Header\ Length$$

d) O datagrama IP foi fragmentado?Justifique.

Podemos constatar que o datagrama não foi fragmentado, pois *Flags* = 0 e *Fragment offset* = 0.



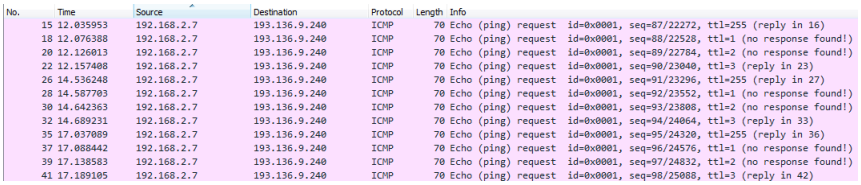
```
▷ Frame 15: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▷ Ethernet II, Src: Pegatron_10:57:46 (60:02:92:10:57:46), Dst: Apple_58:01:60 (c8:2a:14:58:01:60)
◢ Internet Protocol Version 4, Src: 192.168.2.7, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▷ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x0ef7 (3831)
  ▷ Flags: 0x00
    Fragment offset: 0
    Time to live: 255
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.2.7
    Destination: 193.136.9.240
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▷ Internet Control Message Protocol

0000  c8 2a 14 58 01 60 60 02  92 10 57 46 08 00 45 00   .*.X.``. ..WF..E.
0010  00 38 0e f7 00 00 ff 01  00 00 c0 a8 02 07 c1 88   .8...... ........
0020  09 f0 08 00 35 e6 00 01  00 57 20 20 20 20 20 20   ....5... .W
0030  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
0040  20 20 20 20 20 20
```

Figura 6. Primeira mensagem ICMP capturada.

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 atribuí do à sua máquina. Para a sequência de mensagens ICMP enviadas pelo seu computador, indique que campos do cabeçalho IP variam de pacote para pacote.

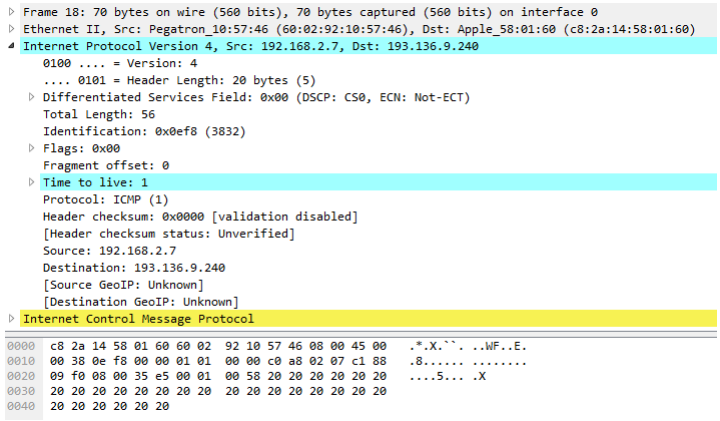
Ordenando os pacotes de acordo com o endereço IP fonte, observando o tráfego ICMP obtemos:



No.	Time	Source	Destination	Protocol	Length	Info
15	12.035953	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=87/22272, ttl=255 (reply in 16)
18	12.076388	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=88/22528, ttl=1 (no response found!)
20	12.126013	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=89/22784, ttl=2 (no response found!)
22	12.157408	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=90/23040, ttl=3 (reply in 23)
26	14.536248	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=91/23296, ttl=255 (reply in 27)
28	14.587703	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=92/23552, ttl=1 (no response found!)
30	14.642363	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=93/23808, ttl=2 (no response found!)
32	14.689231	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=94/24064, ttl=3 (reply in 33)
35	17.037089	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=95/24320, ttl=255 (reply in 36)
37	17.088442	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=96/24576, ttl=1 (no response found!)
39	17.138583	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=97/24832, ttl=2 (no response found!)
41	17.189105	192.168.2.7	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=98/25088, ttl=3 (reply in 42)

Figura 7. Tráfego ICMP de acordo com as indicações da alínea e).

Ora, comparando a Figura 6 com a Figura 8 verificamos que os campos *Identification* e *Time To Live* variam. Através da Figura 10, podemos constatar que o *Header checksum* também se altera.



```
▷ Frame 18: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▷ Ethernet II, Src: Pegatron_10:57:46 (60:02:92:10:57:46), Dst: Apple_58:01:60 (c8:2a:14:58:01:60)
◢ Internet Protocol Version 4, Src: 192.168.2.7, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▷ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x0ef8 (3832)
  ▷ Flags: 0x00
    Fragment offset: 0
  ▷ Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.2.7
    Destination: 193.136.9.240
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▷ Internet Control Message Protocol

0000  c8 2a 14 58 01 60 60 02  92 10 57 46 08 00 45 00   .*.X.``. ..WF..E.
0010  00 38 0e f8 00 00 01 01  00 00 c0 a8 02 07 c1 88   .8...... ........
0020  09 f0 08 00 35 e5 00 01  00 58 20 20 20 20 20 20   ....5... .X
0030  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
0040  20 20 20 20 20 20
```

Figura 8. Frame 18.

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

Para cada $datagrama_{i+1}$ vamos ter:

$$Identification_{i+1} = Identification_i + 1$$

$$\text{TTL}_{i+1} = \begin{cases} 1 & \text{se TTL}_i = 255 \\ 255 & \text{se TTL}_i = 3 \\ \text{TTL}_i + 1 & \text{caso contrário} \end{cases}$$

onde $datagrama_1$ corresponde ao representado na Figura 6.

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?

Ordenando o tráfego por endereço destino e procurando a série de TTL *exceeded*, temos:

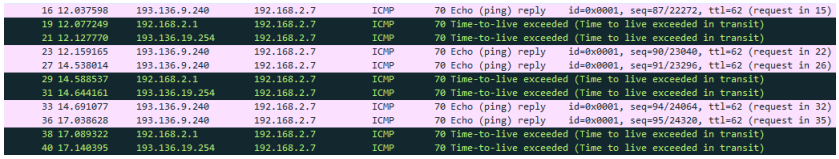


Figura 9. Tráfego ICMP de acordo com as indicações da alínea g).

Tomando como exemplo 192.168.2.1 (MACMINISERVER) e 193.136.19.254 (cisco.di.uminho.pt), obtemos, respetivamente, 64 e 254 valores para o TTL, com observado nas figuras 10 e 11.

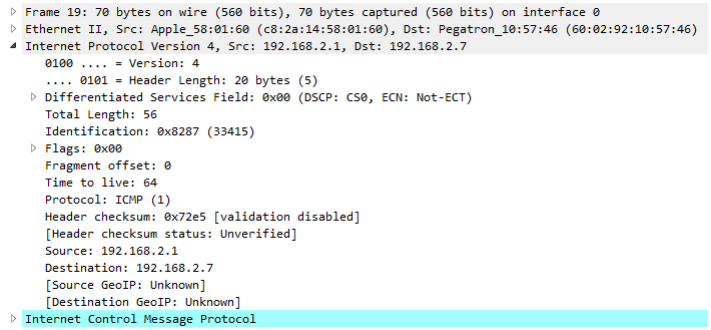


Figura 10. Exemplo 192.168.2.1.

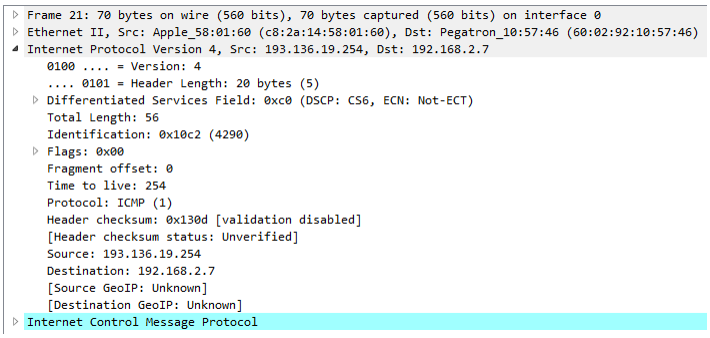


Figura 11. Exemplo 193.136.18.254.

Esses valores de TTL são constantes por endereço IP e relativamente grandes, pois, como não existe conexão na rede IP, apenas têm informação da localização dos nodos adjacentes, assim sendo, esse valor do TTL é pré-definido consoante determinadas variantes (por exemplo, depende do sistema operativo).

2.3 Questão 3

Pretende-se agora analisar a fragmentação de pacotes IP. Reponha a ordem do tráfego capturado usando a coluna do tempode captura. Observe o tráfego depois do tamanho depacote ter sido definido para 30XX bytes.

Consideramos agora pacotes de tamanho 3068 bytes.

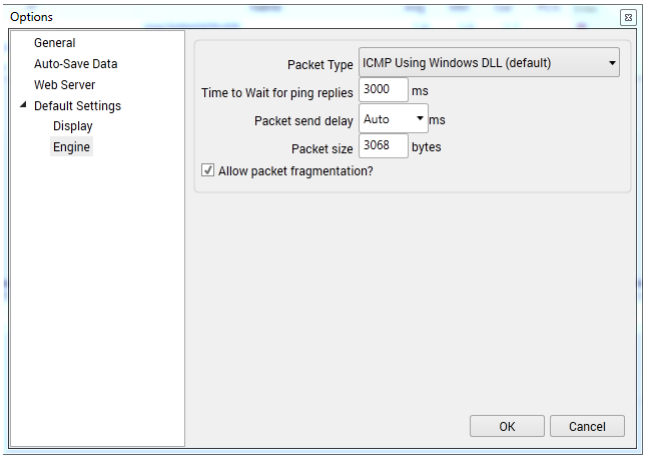


Figura 12. Alteração do tamanho do pacote.

a) Localize a primeira mensagem ICMP. Porque é que houve necessidade de fragmentar o pacote inicial?
Pois o tamanho do pacote excede a dimensão suportada pela rede onde transita.

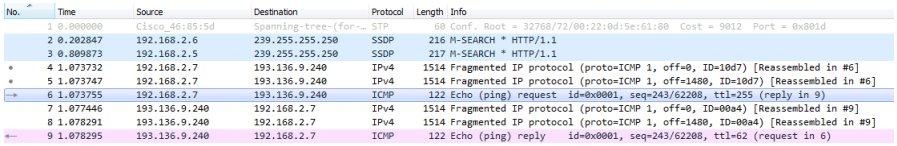


Figura 13. Primeira mensagem ICMP.

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?
Como *Flags* = 1, concluímos que o datagrama foi fragmentado. Dado que o *Fragment offset* = 0, sabemos que se trata do primeiro fragmento. Pelo campo *Total length*, constatamos que o tamanho deste fragmento é de 1500 bytes (20 para o cabeçalho, mais 1480 para o *payload*).

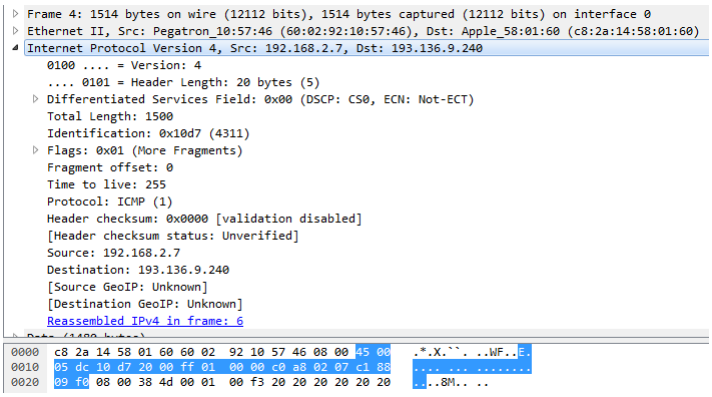


Figura 14. Primeiro fragmento do datagrama.

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?

Fragment offset $\neq 0$, logo, não se trata do primeiro fragmento. Como *Flags* = 1, concluímos que se trata de um fragmento intermédio.

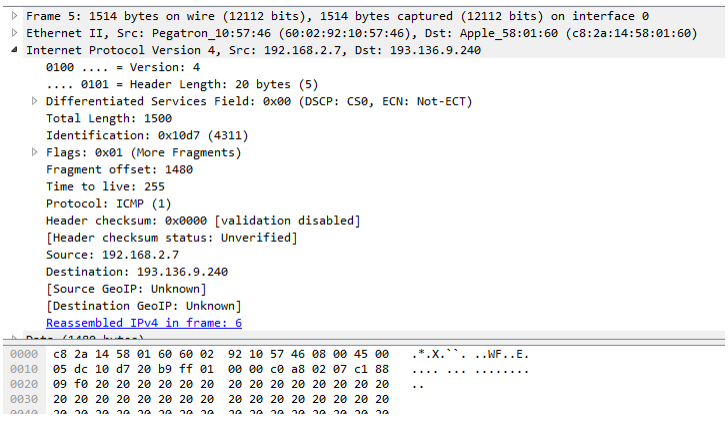


Figura 15. Segundo fragmento do datagrama.

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

Concluimos que foram criados 3 fragmentos, a partir do datagrama inicial, a partir do campo *Fragment Count* da Figura 17.

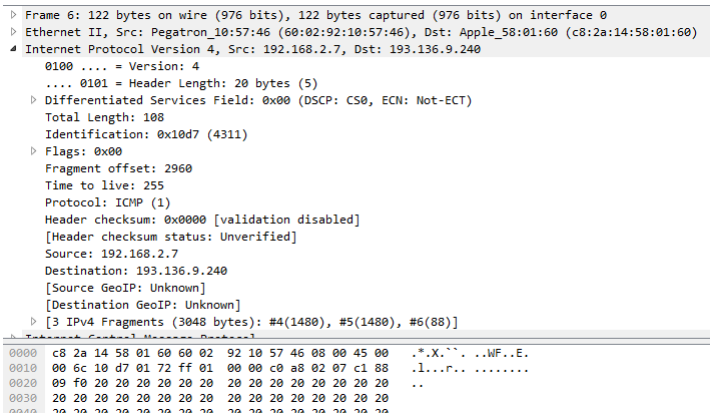


Figura 16. Terceiro fragmento do datagrama.

Tratando-se este do último fragmento:

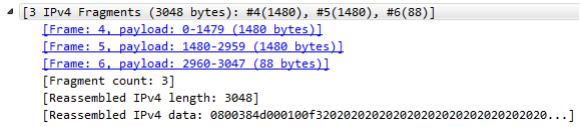


Figura 17. Detalhes da fragmentação do datagrama.

dado que *Flags* = 0 e *Fragment offset* $\neq 0$.

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.

Tamanho, *Flags* e *Fragment offset* variam de fragmento para fragmento do mesmo datagrama, sendo a partir dos dois últimos que conseguimos obter informação sobre a posição do fragmento no datagrama original, daí a possibilidade de o reconstruir posteriormente.

3 Parte II - Endereçamento e Encaminhamento IP

3.1 Questão 1

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.

Definimos a topologia MIEI-RC conforme o descrito:

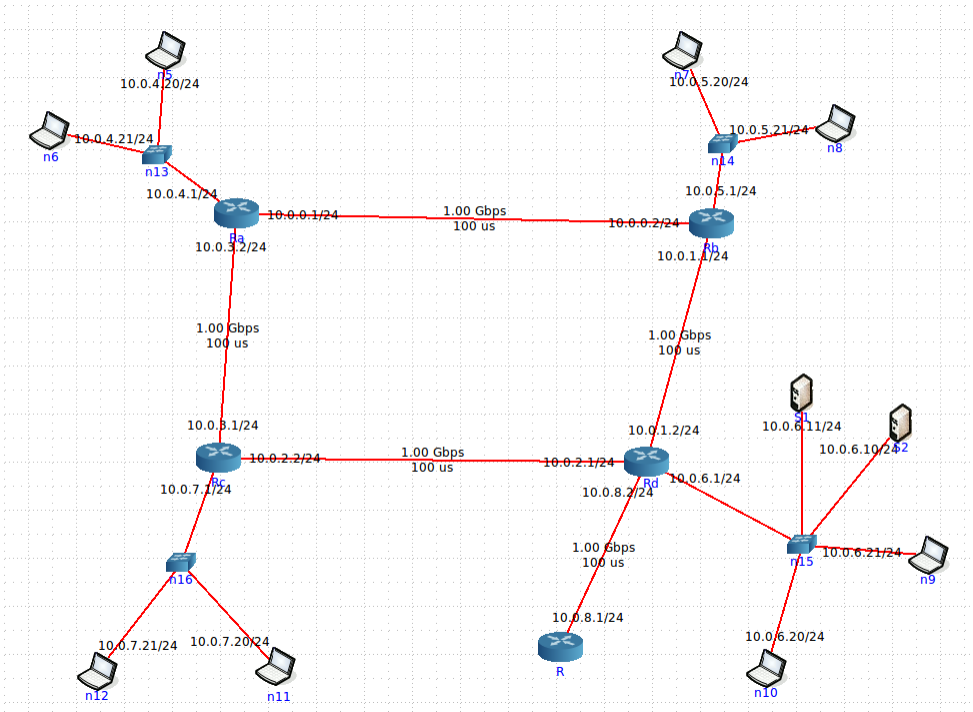


Figura 18. Topologia MIEI-RC.

De notar que não foi possível definir a ligação de R a R_d a 10 Gbps, sendo o limite 1 Gbps na versão do CORE utilizada por nós.
Podemos constatar que é usada a máscara de rede para **/24** $\equiv$ 255.255.255.0 .

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

Sabemos quem um endereço privado pertence a uma das seguintes gamas de valores:

- $[192.168.0.0, 192.168.255.255]$
- $[172.16.0.0, 172.31.255.255]$
- $[10.0.0.0, 10.255.255.255]$

De imediato constatamos que qualquer endereço IP da rede pertence ao terceiro intervalo, pelo que se tratam de endereços privados.

c) Porque razão não é atribuído um endereço IP aos switches?
Os *switches* atuam na camada de ligação de dados (L2), enquanto que o endereço IP encontra-se na camada de rede (L3), sendo os elementos de L1 e L2 transparentes a L3.

d) Usando o comando ping certifique-se que existe conectividade IP entre os laptops dos vários departamentos e os servidores do departamento D.
Podemos observar que existe conetividade IP entre os *laptops* do departamento A, B e C aos servidores do departamento D, através das figuras 19, 20 e 21, respetivamente.

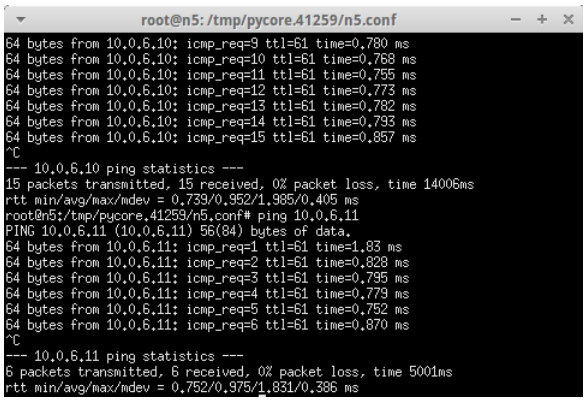


Figura 19. Conetividade entre n5 e S_1 / S_2.

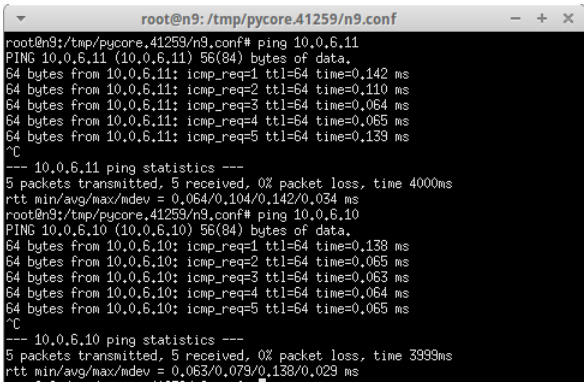


Figura 20. Conetividade entre n9 e S_1 / S_2.

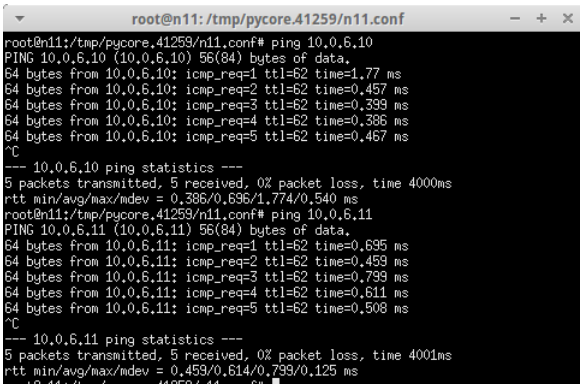


Figura 21. Conetividade entre n11 e S_1 / S_2.

e) Verifique se existe conectividade IP do router de acesso para os servidores S1 e S2
Fazendo *ping* aos servidores a partir do router de acesso verificamos que existe conetividade IP entre eles, como se comprova na Figura 22.

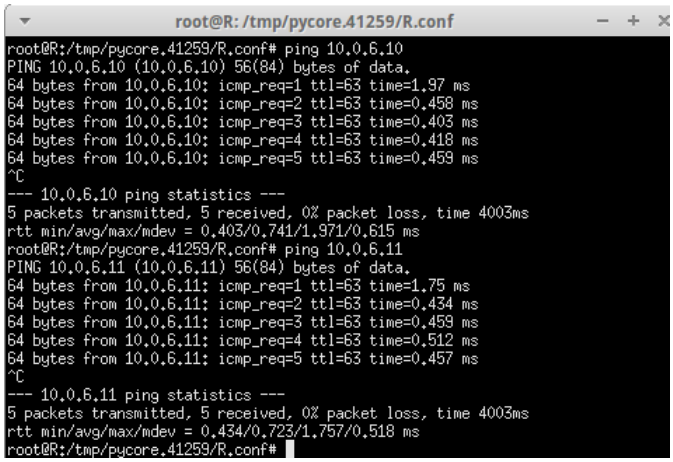


Figura 22. Conetividade entre R e S_1 / S_2.

3.2 Questão 2

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).
Obtemos as seguintes tabelas de encaminhamento:

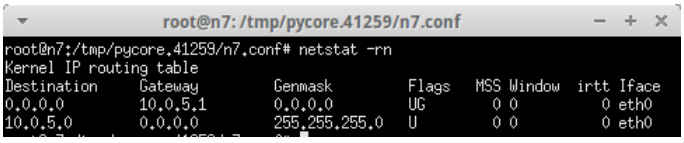


Figura 23. Tabela de encaminhamento IP do *laptop* n7.

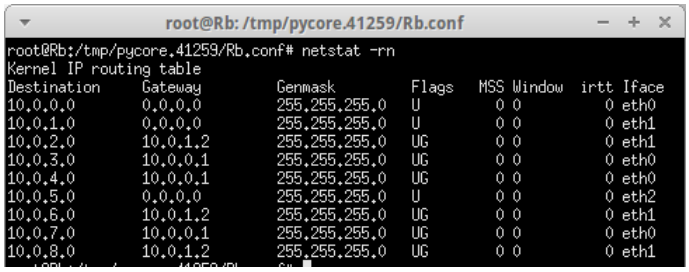


Figura 24. Tabela de encaminhamento IP do *router* do departamento B.

Quando se pretende enviar datagramas IP desde um nodo até um $\text{IP}_{destino}$, para cada linha da tabela de encaminhamento desse nodo é mapeado o resultado de $\text{IP}_{destino}$ && *Netmask* com *Destination*. Se o mapeamento for sucedido, o proximo salto é realizado para o endereço IP indicado na coluna *Gateaway*, senão, repete-se a tentativa de mapeamento nas restantes entradas da tabela.

b) Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico (sugestão: analise que processos estão a correrem cada sistema).
Recorrendo ao comando **ps ax**, constatamos que:

- para o *router*, pela Figura 25, os processos 57 e 63 mostram que está a ser usado uma forma de encaminhamento dinâmico, *Open Shortest Path First*, respetivamente para IPv4 e IPv6;
- quanto ao *laptop*, está a ser usado encaminhamento estático, não sendo observáveis na Figura 26 processos relativos a encaminhamento dinâmico.

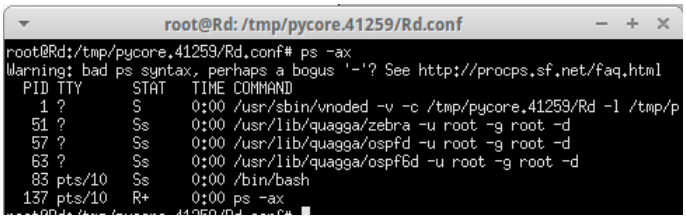


Figura 25. Output do comando *ps ax* no *router* do departamento B.

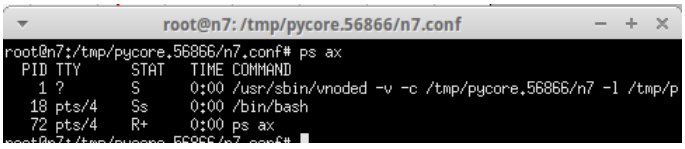


Figura 26. Output do comando *ps ax* no *laptop* n7.

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 D. Use o comando route delete para o efeito.Que implicações tem esta medida para os utilizadores da empresa que acedem aos servidores. Justifique
Removemos a rota por defeito recorrendo ao comando **route delete default** e obtemos a tabela de encaminhamento:

```
root@S1:/tmp/pycore.50563/S1.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.6.1        0.0.0.0         UG        0 0          0 eth0
10.0.6.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
root@S1:/tmp/pycore.50563/S1.conf# route delete default
root@S1:/tmp/pycore.50563/S1.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.6.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
root@S1:/tmp/pycore.50563/S1.conf#
```

Figura 27. Tabela de encaminhamento IP inicial e depois de ser removida a rota por defeito.

Ora, a tabela fica apenas com uma entrada, a que garante a conetividade interna do departamento D. Assim sendo, os utilizadores da empresa que acedem ao servidor a partir do departamento D não são afetados, como podemos ver, por exemplo, na Figura 29. No entanto, fora do departamento D, os nodos que enviam datagramas IP para esse servidor não obtêm resposta, pois este último deixou de ter informação sobre para onde enviar datagramas IP para destinos que não sejam da forma 10.0.6.x, $x \in]0, 255[$, como verificamos pela Figura 28.

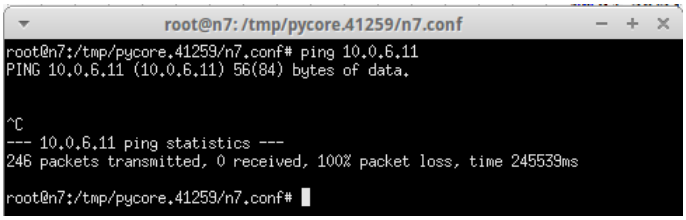


Figura 28. *ping* do *laptop* n7 ao servidor S_1.

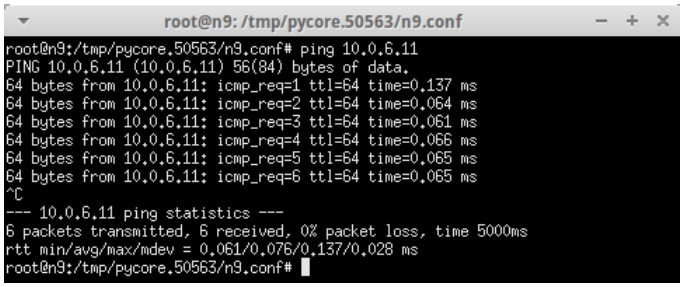


Figura 29. *ping* do *laptop* n9 ao servidor S_1.

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.
Para restaurar a conetividade entre S_1 e os restantes departamentos é necessário incluir uma entrada na sua tabela de encaminhamento por departamento, por exemplo, para o departamento A, fazemos **route add -net 10.0.4.0 gw 10.0.6.1 netmask 255.255.255.0**. No fim, obtemos a tabela de encaminhamento representada na Figura 30.

```
root@S1:/tmp/pycore.50563/S1.conf# route add -net 10.0.5.0 gw 10.0.6.1 netmask 2
55.255.255.0
root@S1:/tmp/pycore.50563/S1.conf# route add -net 10.0.4.0 gw 10.0.6.1 netmask 2
55.255.255.0
root@S1:/tmp/pycore.50563/S1.conf# route add -net 10.0.7.0 gw 10.0.6.1 netmask 2
55.255.255.0
root@S1:/tmp/pycore.50563/S1.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.4.0        10.0.6.1        255.255.255.0   UG        0 0          0 eth0
10.0.5.0        10.0.6.1        255.255.255.0   UG        0 0          0 eth0
10.0.6.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.7.0        10.0.6.1        255.255.255.0   UG        0 0          0 eth0
root@S1:/tmp/pycore.50563/S1.conf#
```

Figura 30. Tabela de encaminhamento IP do servidor S_1.

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
Recorrendo ao comando *ping*, comprovamos que a conetividade para o servidor S_1 está novamente garantida:

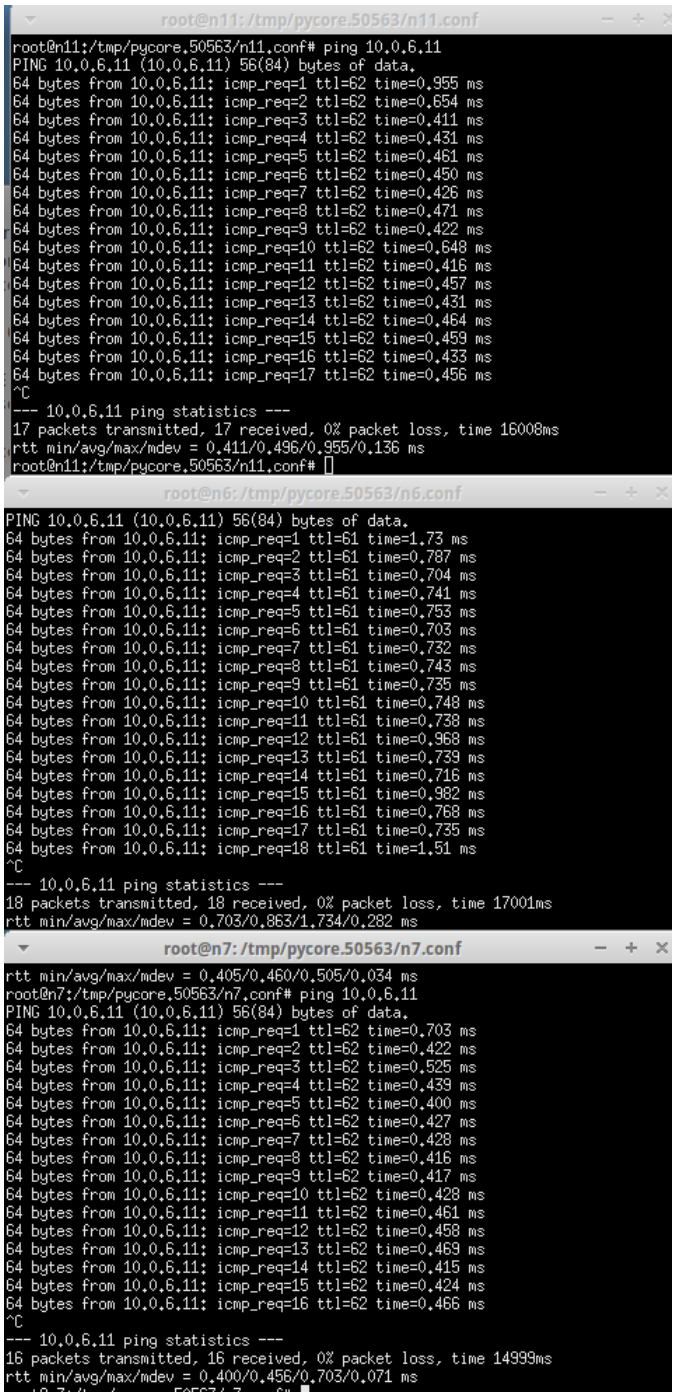


Figura 31. Verificação da acessibilidade do servidor S_1.

Estando a sua nova tabela de encaminhamento definida na Figura 30.

3.3 Questão 3

Partindo do endereço de rede IP 172.16.0.0/16, restam-nos 16 bits para gerir as sub-redes. Ora, reservando n bits para subnetting vamos ter $2^n - 2$ sub-redes e $2^{16-n} - 2$ hosts.
O número mínimo de bits para sub-redes para suportar a topolgia atual são 3. No entanto, se no futuro for desejável introduzir novos departamentos na rede, é interessante deixar espaço de endereçamento suficiente para evitar a necessidade de alterar a configuração da rede na sua totalidade.
Assim sendo, decidimos utilizar 5 bits para sub-rede, restando 11 bits para host, permitindo introduzir até 30 departamentos na rede e para cada sub-rede suportar 2046 equipamentos.
No estado atual, para os quatro departamentos optamos por utilizar os seguintes endereços IP:

Dep A - 172.16.16.0/21

Dep B - 172.16.32.0/21

Dep C - 172.16.64.0/21

Dep D - 172.16.128.0/21

onde utilizamos a máscara de rede para **/21** $\equiv$ 255.255.248.0, visto termos reservado 5 bits para sub-redes, resultando na seguinte topologia CORE:

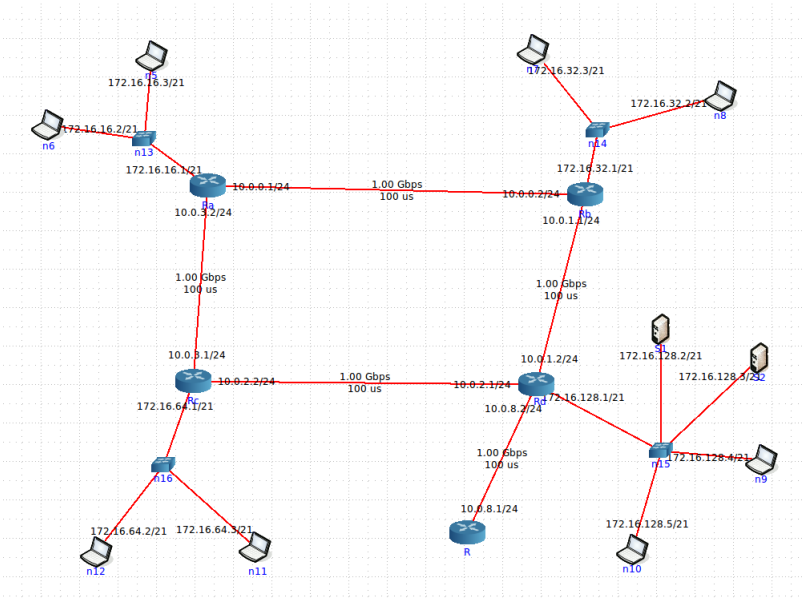


Figura 32. Topologia CORE da organização MIEI-RC recorrendo a sub-redes.

cuja conetividade tanto local ao departamento como externa é garantida, como mostramos em alguns exemplos, nas figuras 33, 34, 35 e 36.

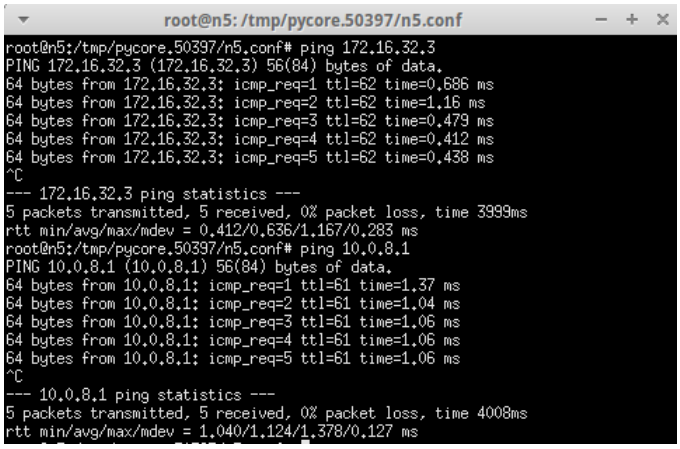


Figura 33. *ping* de n5 do departamento A a n7 e a R.

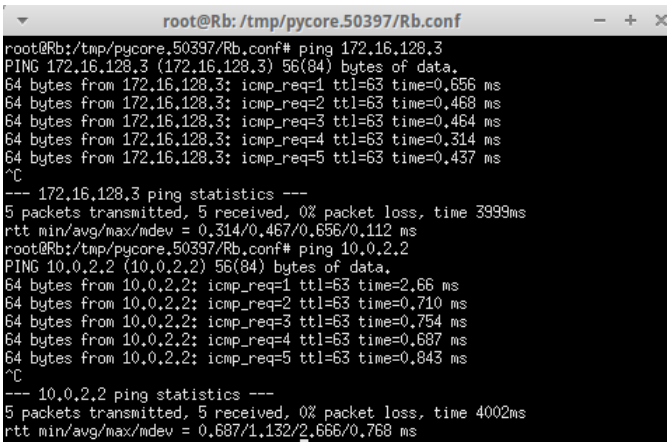


Figura 34. *ping* do *router* do departamento B a S_2 e ao *router* do departamento C.

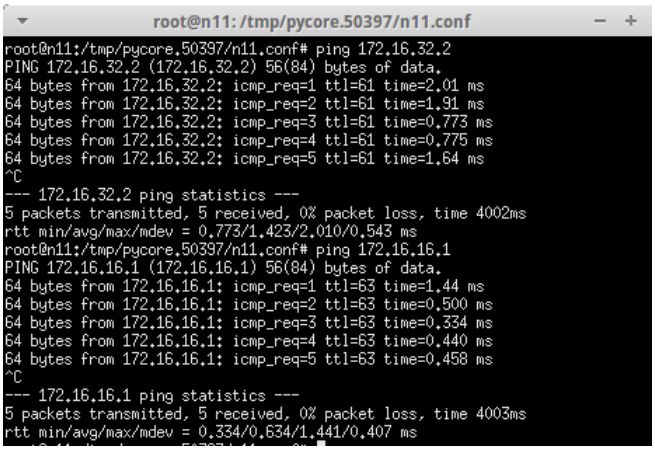


Figura 35. *ping* de n11 do departamento C a n8 e ao *router* do departamento A.

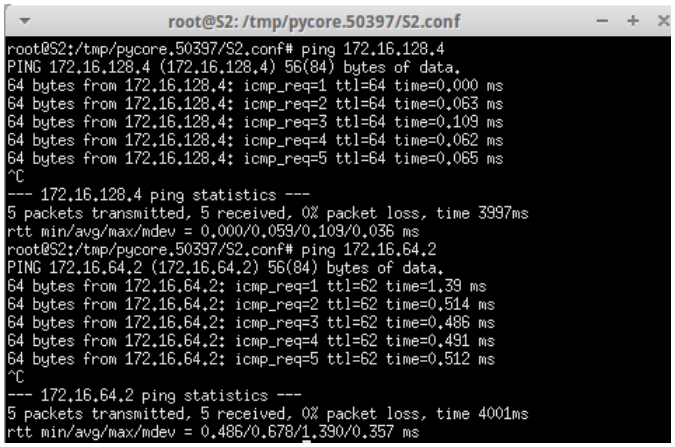


Figura 36. *ping* de S_2 do departamento D a n9 e n12.

4 Conclusão

Este trabalho prático permitiu-nos aprofundar os conhecimentos adquiridos até agora na unidade curricular, utilizar ferramentas de simulação de redes (CORE) e de captura e análise de tramas (Wireshark). Permitiu nos também perceber como funciona o comando traceroute e a forma de utilizá-lo para medir o tempo desde o pedido até à resposta, o número de saltos até ao destino, como calcular o tamanho dos Datagramas IP, a fragmentação dos pacotes e identificar as flags de fragmentação para determinar se o pacote foi ou não fragmentado. Vimos também a consequência de eliminar o endereço de destino default do servidor e efetuamos ainda o encaminhamento manual do servidor de modo a perceber a comunicação do mesmo com os restantes departamentos. No final, foi-nos pedido para a partir de uma máscara de rede com ip 172.16.0.0, endereçar os 4 departamentos. Para além de resolver o problema, tivemos em a atenção a possibilidade de a rede ser expandida.