TP2: Protocolo IPv4

Eduardo Cunha A98980, Gonçalo Magalhães A100084, Fábio Ribeiro A100058

Universidade do Minho, Escola de Ciências, Ciências da Computação

PL1, Grupo 1

1. Perguntas e Respostas – Parte 1

1.1 O RFC 791 diz que um router deve decrementar o TTL de pelo menos uma unidade. Para verificar o comportamento do *traceroute*, implemente no CORE uma topologia retangular com um router em cada vértice. Ligue a cada um dos routers um *host* (*PC*) e atribua à rede de cada *host* os endereços 192. < nº do grupo + N >. < nº do grupo + N >. X/24, com N=0, 1, 2, 3. Atribua ao decimal X um valor adequado. Atribua a este *host* o nome PC1. Ao *host* que está ligado ao router diametralmente oposto do do PC1 atribua o nome PC2. Coloque esses nomes nos respetivos *hosts* da topologia e arranque a rede. Active o *wireshark* no *host* PC1. Numa *shell* do PC1, execute o comando *traceroute –I* para o endereço IP do PC2. (*Note que pode não existir conectividade IP imediata entre os hosts até que o anúncio de rotas estabilize*).

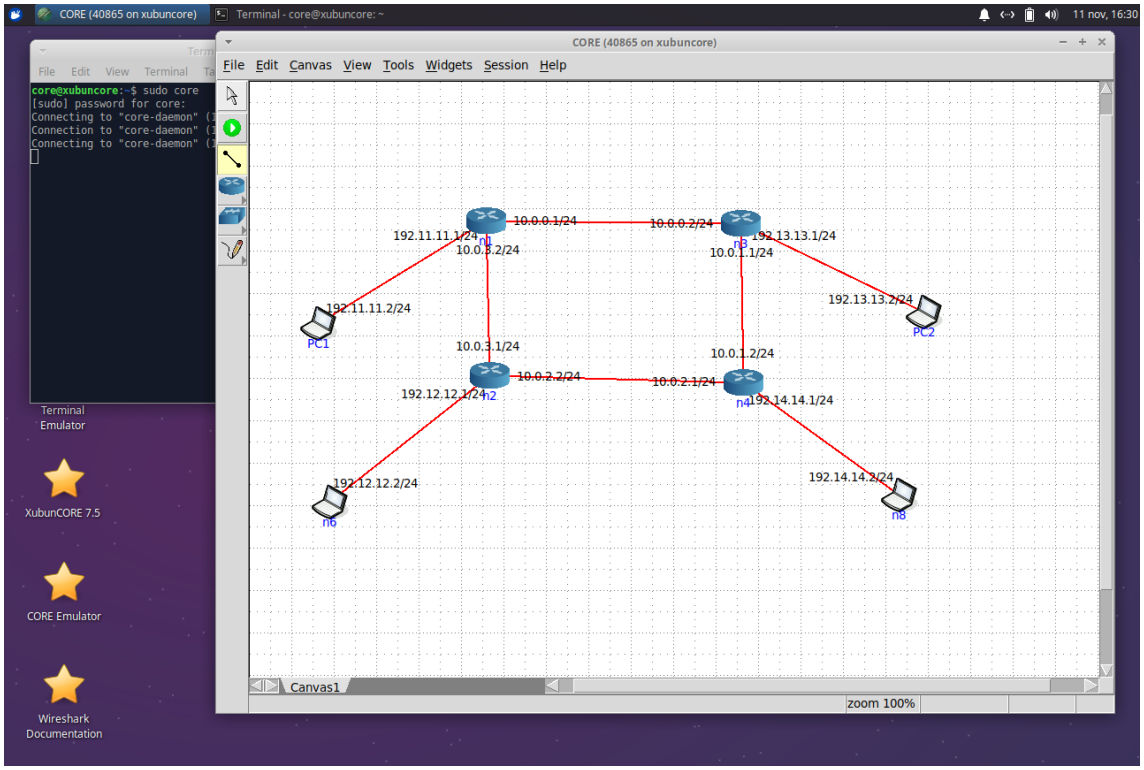


Figura 1

O pc1 não é o que está diametralmente oposto, mas sim o adjacente, por erro de leitura do enunciado.

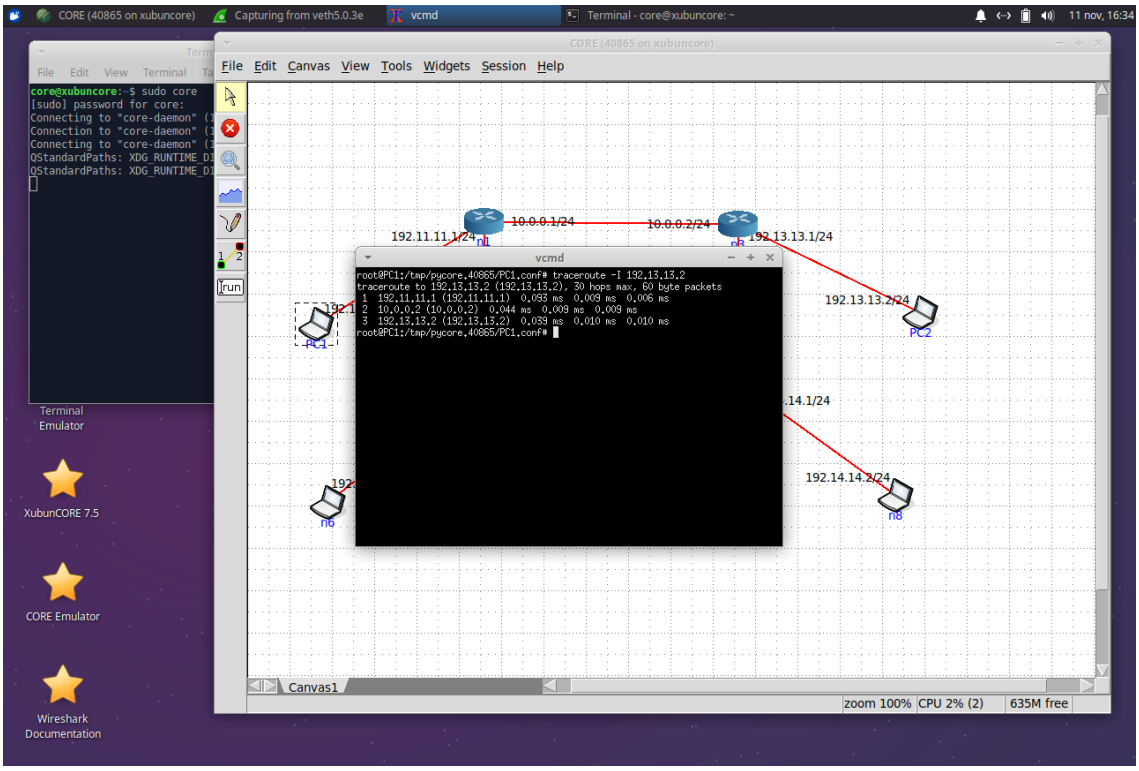


Figura 2

a) Registe e analise o tráfego ICMP enviado pelo PC1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.

```
    21 15.482470750  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=1/256, ttl=1 (no response…
    22 15.482490850  192.11.11.1     192.11.11.2     ICMP     102 Time-to-live exceeded (Time to live exceeded in transit)
    23 15.482502307  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=2/512, ttl=1 (no response…
    24 15.482507886  192.11.11.1     192.11.11.2     ICMP     102 Time-to-live exceeded (Time to live exceeded in transit)
    25 15.482512312  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=3/768, ttl=1 (no response…
    26 15.482516318  192.11.11.1     192.11.11.2     ICMP     102 Time-to-live exceeded (Time to live exceeded in transit)
    27 15.482520615  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=4/1024, ttl=2 (no respons…
    28 15.482561987  10.0.0.2        192.11.11.2     ICMP     102 Time-to-live exceeded (Time to live exceeded in transit)
    29 15.482566574  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=5/1280, ttl=2 (no respons…
    30 15.482573695  10.0.0.2        192.11.11.2     ICMP     102 Time-to-live exceeded (Time to live exceeded in transit)
    31 15.482577521  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=6/1536, ttl=2 (no respons…
    32 15.482584461  10.0.0.2        192.11.11.2     ICMP     102 Time-to-live exceeded (Time to live exceeded in transit)
    33 15.482588838  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=7/1792, ttl=3 (reply in 3…
    34 15.482625063  192.13.13.2     192.11.11.2     ICMP      74 Echo (ping) reply    id=0x001b, seq=7/1792, ttl=62 (request i…
    35 15.482631062  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=8/2048, ttl=3 (reply in 3…
    36 15.482639635  192.13.13.2     192.11.11.2     ICMP      74 Echo (ping) reply    id=0x001b, seq=8/2048, ttl=62 (request i…
    37 15.482643320  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=9/2304, ttl=3 (reply in 3…
    38 15.482651463  192.13.13.2     192.11.11.2     ICMP      74 Echo (ping) reply    id=0x001b, seq=9/2304, ttl=62 (request i…
    39 15.482655369  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=10/2560, ttl=4 (reply in …
    40 15.482663391  192.13.13.2     192.11.11.2     ICMP      74 Echo (ping) reply    id=0x001b, seq=10/2560, ttl=62 (request …
    41 15.482666896  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=11/2816, ttl=4 (reply in …
    42 15.482674968  192.13.13.2     192.11.11.2     ICMP      74 Echo (ping) reply    id=0x001b, seq=11/2816, ttl=62 (request …
    43 15.482678534  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=12/3072, ttl=4 (reply in …
    44 15.482687307  192.13.13.2     192.11.11.2     ICMP      74 Echo (ping) reply    id=0x001b, seq=12/3072, ttl=62 (request …
```

Figura 3

A primeira tentativa de comunicação entre o PC1 e o router é falhada porque o TTL é igual a 1. O pacote ao chegar ao R2 será descartado, porque o TTL é decrementado e ficará igual a 0. Nas tentativas seguintes, o TTL é incrementado até atingir o valor de TTL = 3, que é o valor mínimo para esta comunicação ser bem sucedida.

b) Qual deve ser o valor inicial mínimo do campo TTL para alcançar o PC2? Verifique na prática que a sua resposta está correta.

```
    33 15.482588838  192.11.11.2     192.13.13.2     ICMP      74 Echo (ping) request  id=0x001b, seq=7/1792, ttl=3 (reply in 3…
    34 15.482625063  192.13.13.2     192.11.11.2     ICMP      74 Echo (ping) reply    id=0x001b, seq=7/1792, ttl=62 (request i…
```

Figura 4

Como podemos ver na Figura o TTL inicial mínimo tem de ser 3.

c) Calcule o valor médio do tempo de ida-e-volta (*Round-Trip Time*) obtido? (Para melhorar a média, convém alterar o número de *probe packets* com a opção -q).

```
root@PC1:/tmp/pycore.40865/PC1.conf# traceroute -1 192.13.13.2/24
Bad option `-1' (argc 1)
root@PC1:/tmp/pycore.40865/PC1.conf# traceroute -I 192.13.13.2
traceroute to 192.13.13.2 (192.13.13.2), 30 hops max, 60 byte packets
 1  192.11.11.1 (192.11.11.1)  0.054 ms  0.008 ms  0.006 ms
 2  10.0.0.2 (10.0.0.2)  0.029 ms  0.009 ms  0.009 ms
 3  192.13.13.2 (192.13.13.2)  0.026 ms  0.010 ms  0.010 ms
root@PC1:/tmp/pycore.40865/PC1.conf#
```

Figura 5

O valor médio é a media dos últimos 3 valores obtidos.

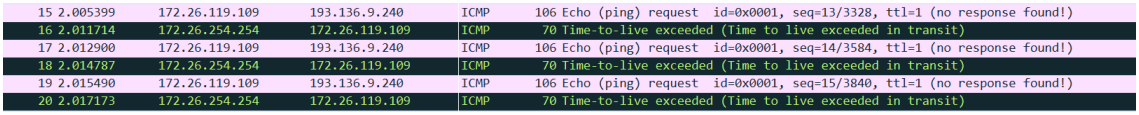
(0.026 + 0.010 + 0.010) / 3 = 0,015 (3)

1.2 Pretende-se agora usar o *traceroute* na sua máquina nativa. (Nota: o *tracert* disponibilizado no Windows não permite mudar o tamanho das mensagens a enviar. Porém, no Linux/Unix, o *traceroute* permite indicar o tamanho do pacote ICMP através da linha de comando, a seguir ao *host* de destino (ver *man traceroute*). Por exemplo, traceroute -I router-di.uminho.pt 512.)
Documente as suas respostas com a impressão do(s) output(s) (e.g. pacote(s)) que as suportam. Para esse feito use, por exemplo, File -> Print, selecione *packet only*. Coloque apenas o detalhe necessário para sustentar a resposta e identificar o seu computador.
Usando o *wireshark*, capture o tráfego gerado pelo *traceroute -I/tracert* usando como máquina destino o *host* marco.uminho.pt. Pare a captura. Com base no tráfego capturado, identifique os pedidos ICMP *Echo Request* e o conjunto de mensagens devolvidas como resposta.
Selecione a primeira mensagem ICMP capturada e centre a análise no nível protocolar IP (expanda o *tab* correspondente na janela de detalhe do *wireshark*).
Através da análise do cabeçalho IP diga:

```
PS C:\Users\eduar> tracert  marco.uminho.pt

Tracing route to marco.uminho.pt [193.136.9.240]
over a maximum of 30 hops:

  1     6 ms     2 ms     1 ms  172.26.254.254
  2     5 ms     1 ms     1 ms  172.16.2.1
  3     6 ms     1 ms     1 ms  172.16.115.252
  4     6 ms     1 ms     1 ms  marco.uminho.pt [193.136.9.240]

Trace complete.
PS C:\Users\eduar>
```

Figura 6

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

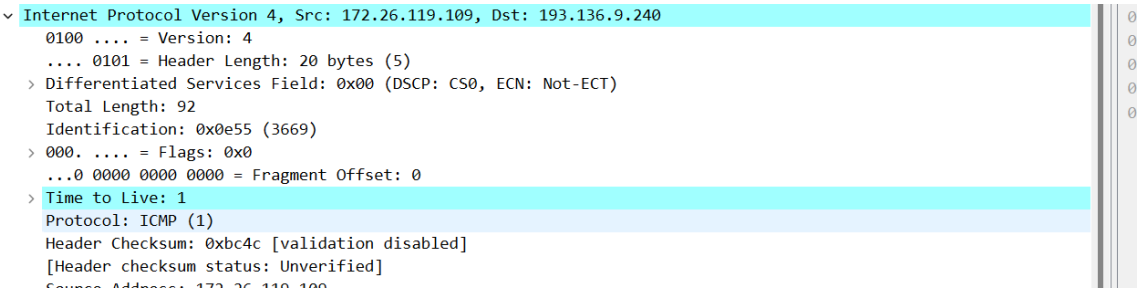


15	2.005399	172.26.119.109	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=13/3328, ttl=1 (no response found!)
16	2.011714	172.26.254.254	172.26.119.109	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
17	2.012900	172.26.119.109	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=14/3584, ttl=1 (no response found!)
18	2.014787	172.26.254.254	172.26.119.109	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
19	2.015490	172.26.119.109	193.136.9.240	ICMP	106 Echo (ping) request id=0x0001, seq=15/3840, ttl=1 (no response found!)
20	2.017173	172.26.254.254	172.26.119.109	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

Figura 7

O IP da interface ativa da nossa máquina é 172.26.119.109

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

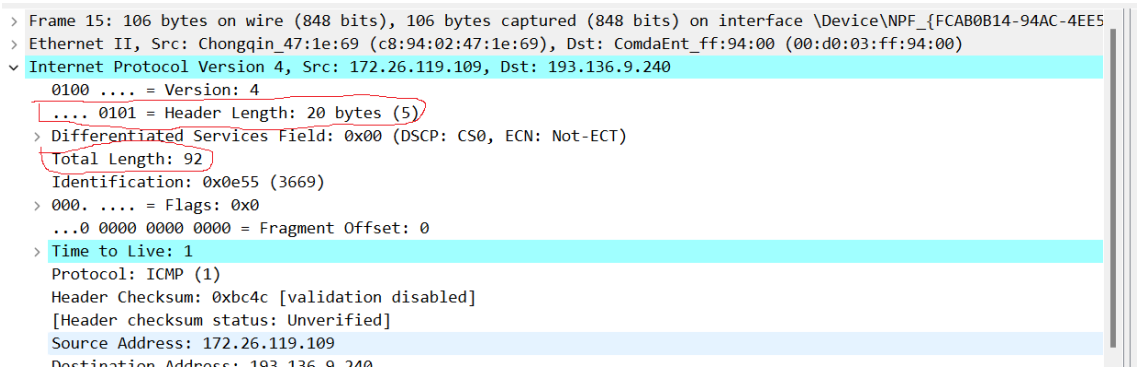


```
Internet Protocol Version 4, Src: 172.26.119.109, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 92
    Identification: 0x0e55 (3669)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
  > Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0xbc4c [validation disabled]
    [Header checksum status: Unverified]
```

Figura 8

Como podemos observar, o valor do campo protocolo é IMCP (1)

c) Quantos *bytes* tem o cabeçalho IPv4? Quantos *bytes* tem o campo de dados (*payload*) do datagrama? Como se calcula o tamanho do *payload*?



```
> Frame 15: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface \Device\NPF_{FCAB0B14-94AC-4EE5
> Ethernet II, Src: Chongqin_47:1e:69 (c8:94:02:47:1e:69), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
v Internet Protocol Version 4, Src: 172.26.119.109, 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: 92
    Identification: 0x0e55 (3669)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
  > Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0xbc4c [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.119.109
    Destination Address: 193.136.9.240
```

Figura 9

Como se pode ver o cabeçalho IPv4 tem 20 bytes. A partir da diferença do tamanho total (92 bytes) com esse valor obtemos o tamanho do campo de dados, que neste caso é 72 bytes.

d) O datagrama IP foi fragmentado? Justifique.

```
▾ 000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
  ...0 0000 0000 0000 = Fragment Offset: 0
```

Figura 10

Como o valor das flags e do offset é 0 então conclui-se que o pacote não foi fragmentado.

e) Ordene os pacotes capturados de acordo com o endereço IP fonte, 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.

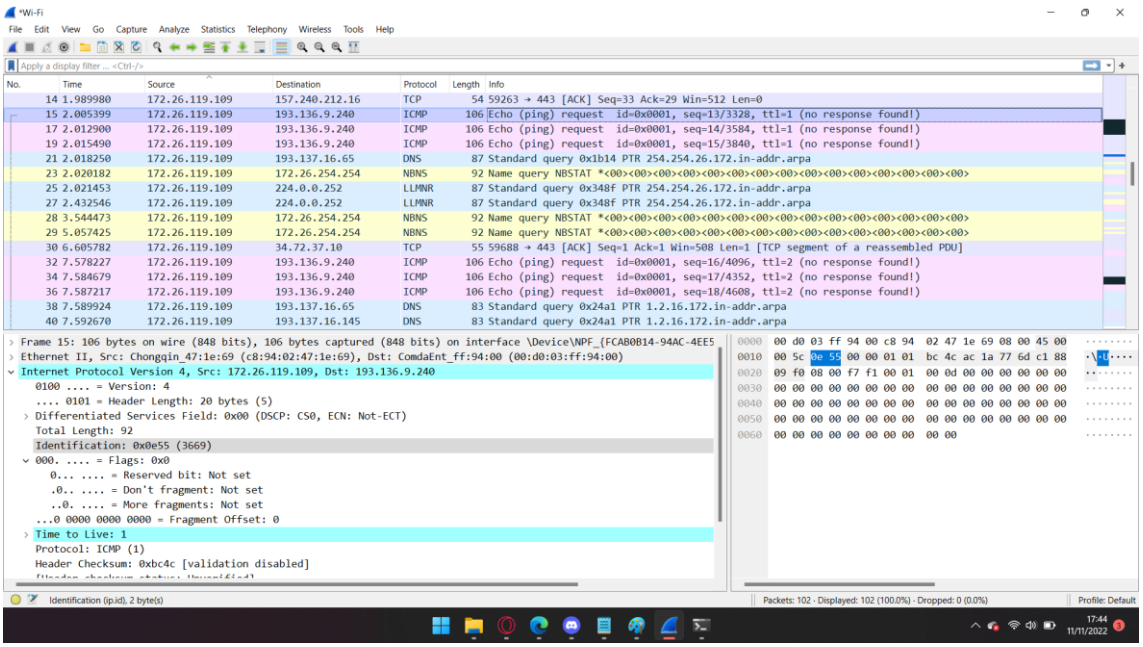


Figura 11

Como podemos ver pelas imagens os campos Time to live variam.

f) Indique o padrão observado nos valores do campo de Identificação do datagrama IP e TTL.

Tendo em conta as imagens da alínea anterior podemos verificar que o campo de identificação é incrementado a cada pacote enviado tal como o campo TTL

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

16 2.011714	172.26.254.254	172.26.119.109	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
18 2.014787	172.26.254.254	172.26.119.109	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
20 2.017173	172.26.254.254	172.26.119.109	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

Figura 12

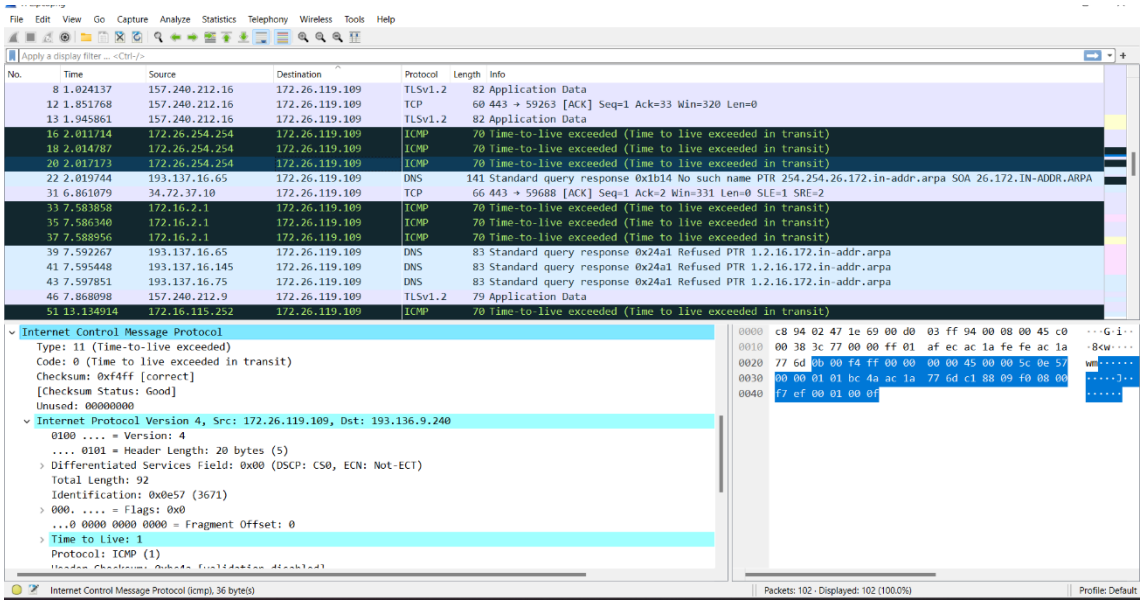


Figura 12

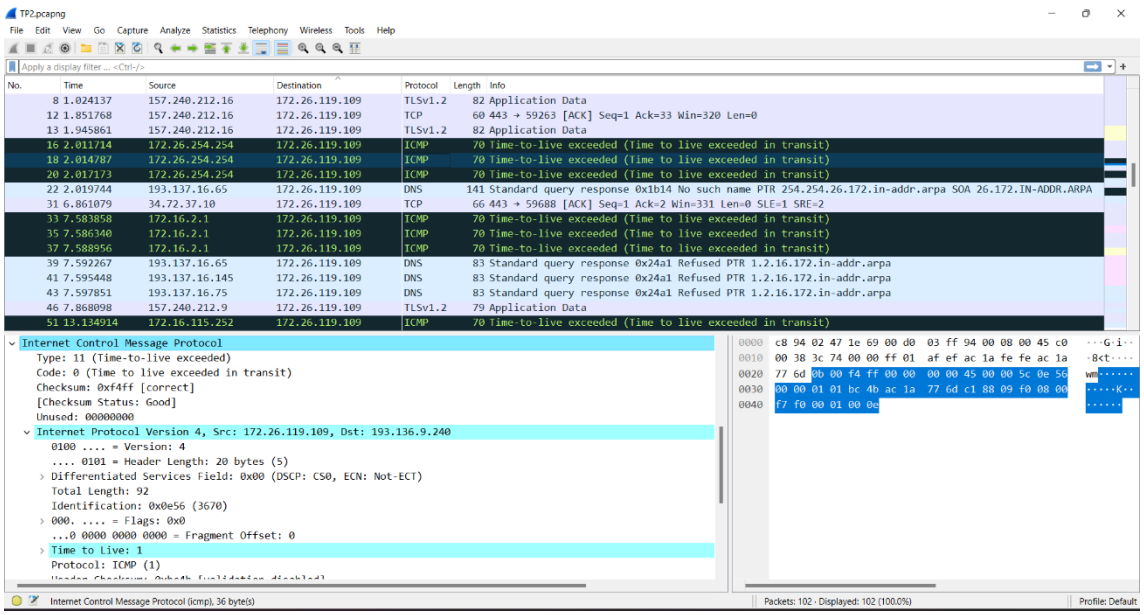


Figura 14

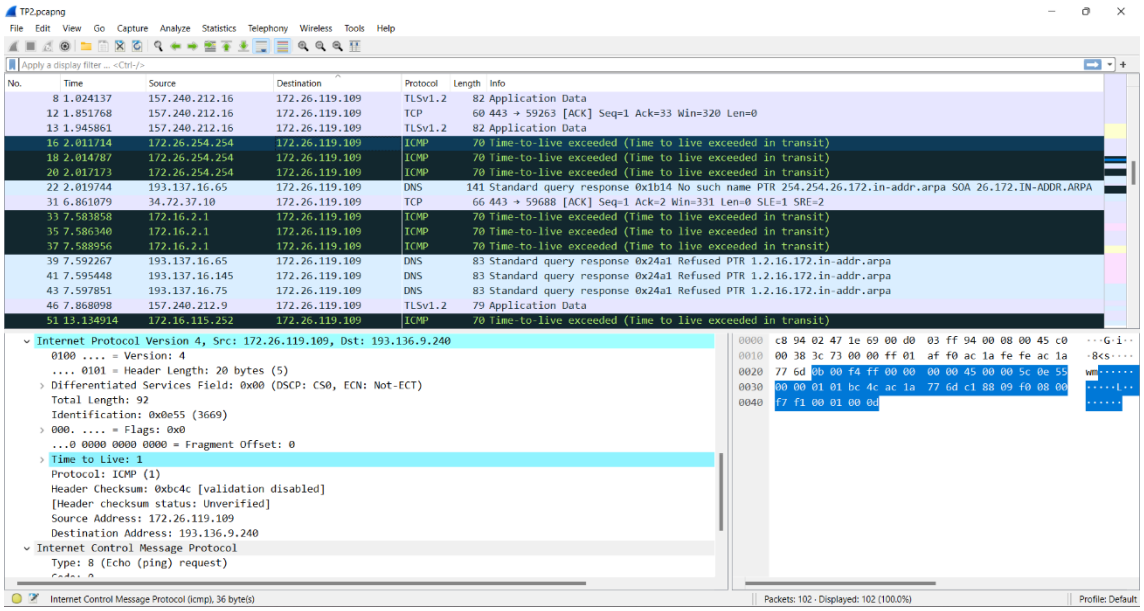


Figura 15

Pelas imagens anteriores podemos verificar que os valores do TTL permanecem constantes (1). Isto acontece porque o TTL já se encontra na origem do nosso pacote.

1.3 Pretende-se agora analisar a fragmentação de pacotes IP. Capture com o *Wireshark* o tráfego gerado pelo comando *ping <opção> 52XY marco.uminho.pt*, onde a opção *–l* (Windows) ou *–s* (Linux) define o número de bytes enviados no campo de dados do pacote ICMP, X representa as dezenas e Y as unidades do número do grupo. Por exemplo, X=2 e Y=4 para os grupos G24 e G124. Observe o tráfego capturado.

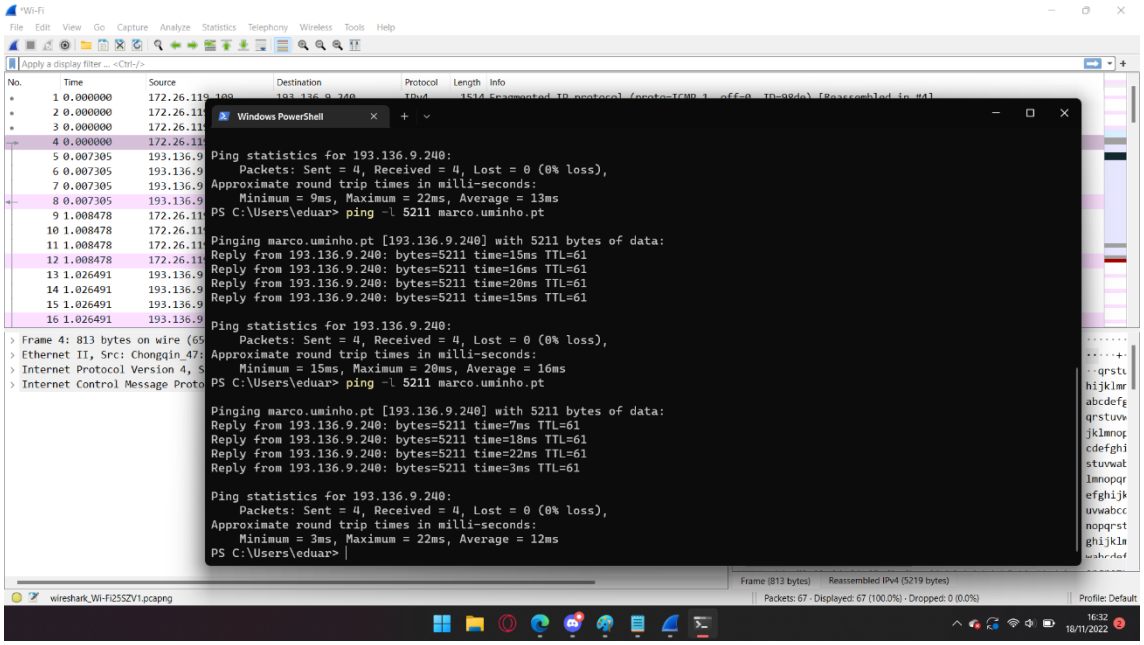


Figura 16

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

1	0.000000	172.26.119.109	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=98de) [Reassembled in #4]
2	0.000000	172.26.119.109	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=98de) [Reassembled in #4]
3	0.000000	172.26.119.109	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=98de) [Reassembled in #4]
4	0.000000	172.26.119.109	193.136.9.240	ICMP	813	Echo (ping) request id=0x0001, seq=41/10496, ttl=128 (reply in 8)

Figura 17

```
Pinging marco.uminho.pt [193.136.9.240] with 5211 bytes of data:
Reply from 193.136.9.240: bytes=5211 time=7ms TTL=61
Reply from 193.136.9.240: bytes=5211 time=18ms TTL=61
Reply from 193.136.9.240: bytes=5211 time=22ms TTL=61
Reply from 193.136.9.240: bytes=5211 time=3ms TTL=61
```

Figura 18

Como a MTU tem apenas capacidade máxima de 1500 bytes e o pacote enviado tem 5211 bytes então é necessário fragmentá-lo.

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?

```
Internet Protocol Version 4, Src: 172.26.119.109, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x98de (39134)
  > 001. .... = Flags: 0x1, More fragments
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x8d42 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.119.109
    Destination Address: 193.136.9.240
    [Reassembled IPv4 in frame: 4]
> Data (1480 bytes)
```

Figura 19

Como o valor do offset é 0 e o valor da Flag More Fragments é 1, podemos concluir que este é o primeiro fragmento. Como referimos anteriormente como a MTU apenas suporta 1500 bytes então o tamanho deste datagrama é de 1500 bytes.

c) Imprima o segundo fragmento do datagrama IP original. Que informação do cabeçalho IP indica que não se trata do primeiro fragmento? Há mais fragmentos? O que nos permite afirmar isso?

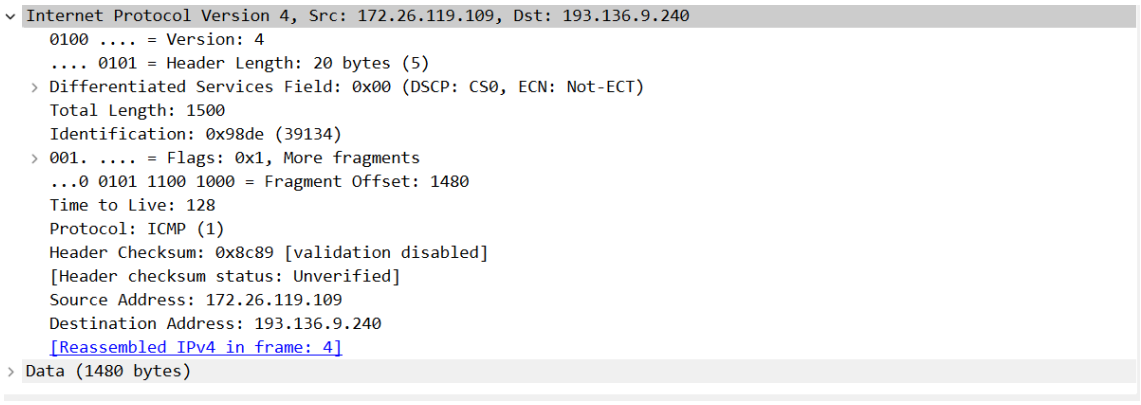


Figura 20

Como o offset é diferente de 0 (é 1480) então sabemos que não se trata do primeiro fragmento. Também é possível concluir que há mais fragmentos porque a Flag More Fragments é igual a 1.

d) Quantos fragmentos foram criados a partir do datagrama original? Como se deteta o último fragmento correspondente ao datagrama original? Estabeleça um filtro no *Wireshark* que permita listar o último fragmento do datagrama IP segmentado.

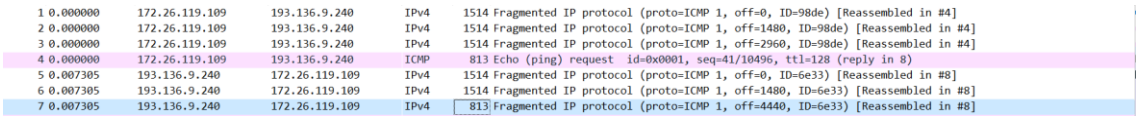


Figura 21

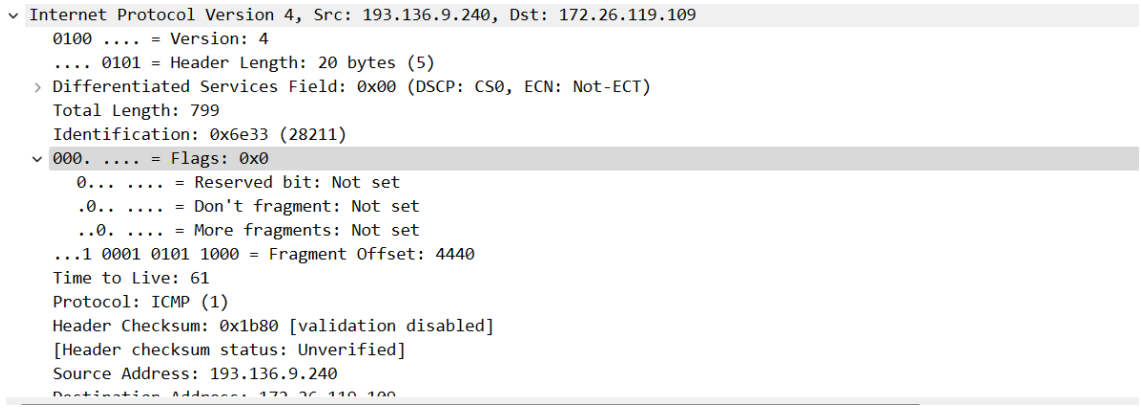
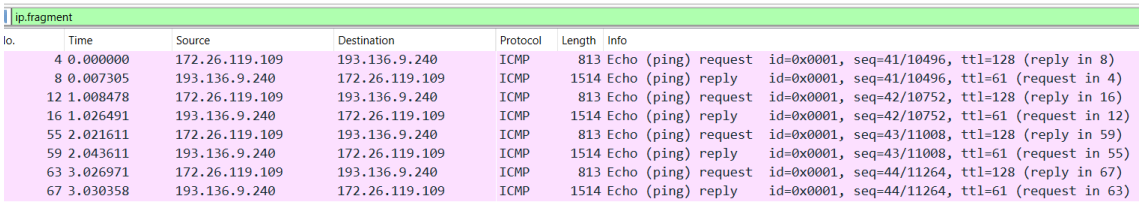


Figura 22

O último fragmento é detetado quando o valor da Flag More Fragments é igual a 0 e o offset também. Contando os datagramas anteriores concluímos que foram criados 4 fragmentos (até ao ICMP).



ip.fragment

lo.	Time	Source	Destination	Protocol	Length	Info
4	0.000000	172.26.119.109	193.136.9.240	ICMP	813	Echo (ping) request id=0x0001, seq=41/10496, ttl=128 (reply in 8)
8	0.007305	193.136.9.240	172.26.119.109	ICMP	1514	Echo (ping) reply id=0x0001, seq=41/10496, ttl=61 (request in 4)
12	1.008478	172.26.119.109	193.136.9.240	ICMP	813	Echo (ping) request id=0x0001, seq=42/10752, ttl=128 (reply in 16)
16	1.026491	193.136.9.240	172.26.119.109	ICMP	1514	Echo (ping) reply id=0x0001, seq=42/10752, ttl=61 (request in 12)
55	2.021611	172.26.119.109	193.136.9.240	ICMP	813	Echo (ping) request id=0x0001, seq=43/11008, ttl=128 (reply in 59)
59	2.043611	193.136.9.240	172.26.119.109	ICMP	1514	Echo (ping) reply id=0x0001, seq=43/11008, ttl=61 (request in 55)
63	3.026971	172.26.119.109	193.136.9.240	ICMP	813	Echo (ping) request id=0x0001, seq=44/11264, ttl=128 (reply in 67)
67	3.030358	193.136.9.240	172.26.119.109	ICMP	1514	Echo (ping) reply id=0x0001, seq=44/11264, ttl=61 (request in 63)

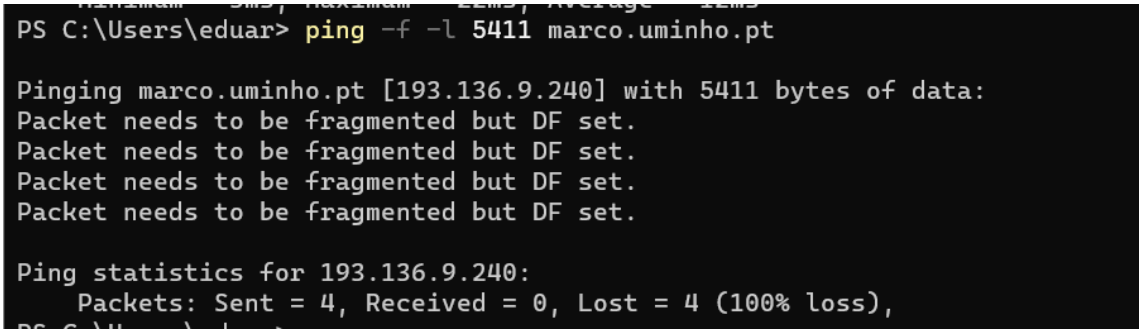
Figura 23

Usamos o filtro ip.fragment para encontrar mais facilmente os fragmentos do datagrama original.

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 valores das Flags Offset e More Fragments vão variando entre os fragmentos. O valor da Flag Offset organiza os fragmentos por ordem, já o valor da Flag More Fragments apenas indica se existem mais fragmentos. A partir dos valores da Flag offset é possível reconstruir o datagrama original.

f) Sabendo que a opção -f (Windows) ou -M do (Linux) ativa a flag "Don't Fragment" (DF) no cabeçalho do IPv4, usando *ping <opção DF> <opção pkt size> SIZE marco.uminho.pt*, (opção *pkt size* = *–l* (Windows) ou *–s* (Linux)), determine o valor máximo de *SIZE* sem que ocorra fragmentação do pacote? Justifique o valor obtido.



```
PS C:\Users\eduar> ping -f -l 5411 marco.uminho.pt

Pinging marco.uminho.pt [193.136.9.240] with 5411 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.

Ping statistics for 193.136.9.240:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Figura 24

O pacote não é enviado porque o seu tamanho excede o limite máximo que a MTU suporta.

O valor máximo de Size sem que ocorra a fragmentação é 1500 bytes (limite máximo da MTU).

2. Perguntas e Respostas – Parte 2

2.1 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.

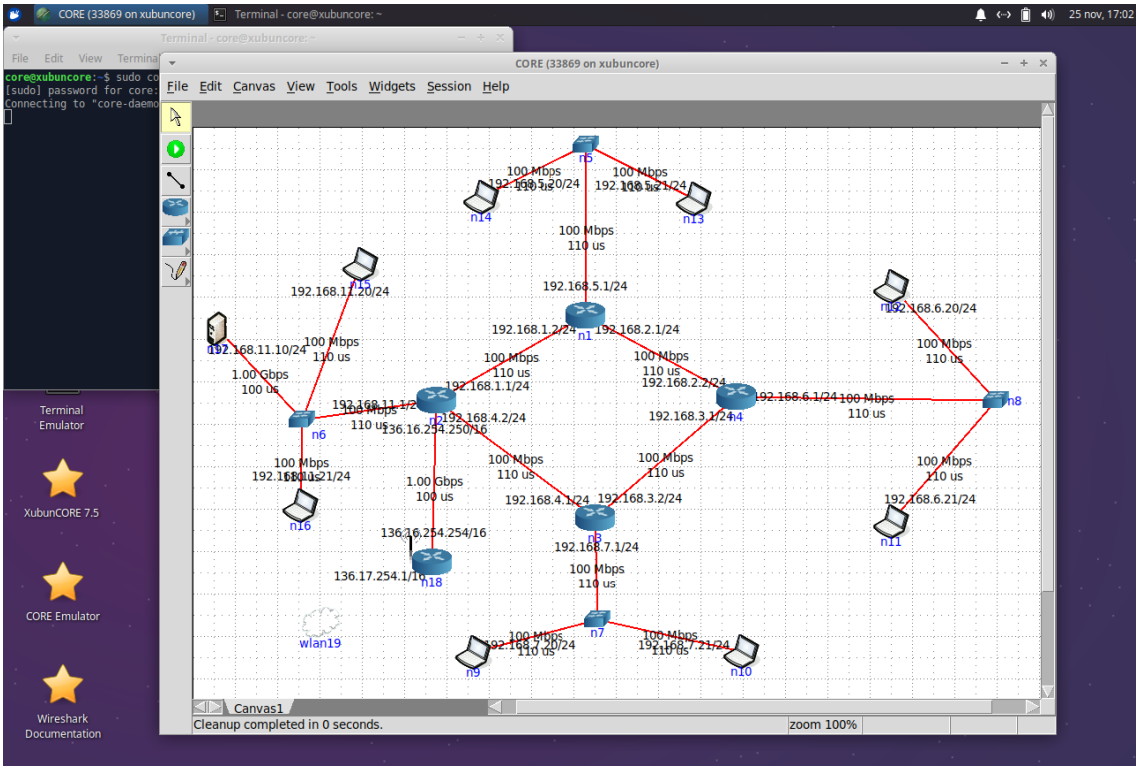


Figura 25

Como podemos observar na imagem a maioria dos endereços possui máscara /24 ou seja 255.255.255.0. Porém algumas ligações possuem máscara /16 ou seja 255.255.0.0.

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

Os endereços entre 192.168.0.0 e 192.168.255.255 são endereços privados, uma vez que são classe C, como todos os endereços (exceto 2 - 136...) estão entre este intervalo então são endereços privados.

c) Porque razão o CORE não atribui um endereço IP aos *switches*? Faz sentido na prática um *switch* ter um endereço IP? Justifique.

O switch regista os endereços MAC dos dispositivos conectados e tem como função fazer a ligação entre equipamentos de uma rede. Após a análise dos endereços MAC, o switch associa as máquinas às respetivas entradas fisicas do equipamento.
Por este motivo não há necessidade de atribuição de um endereço IP ao switch, porque ele não os usa.

2.2 Para o *router* RA e o servidor S1 do departamento A:

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*).

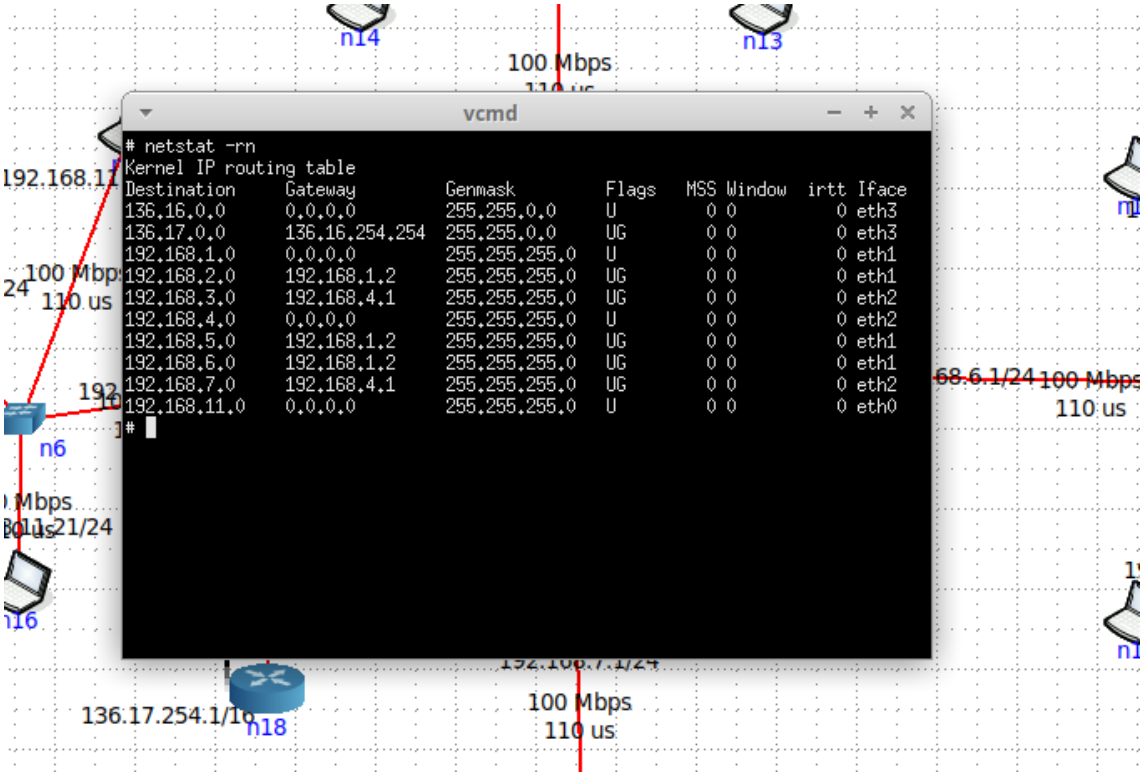


Figura 26 - Tabela de encaminhamento do router 2

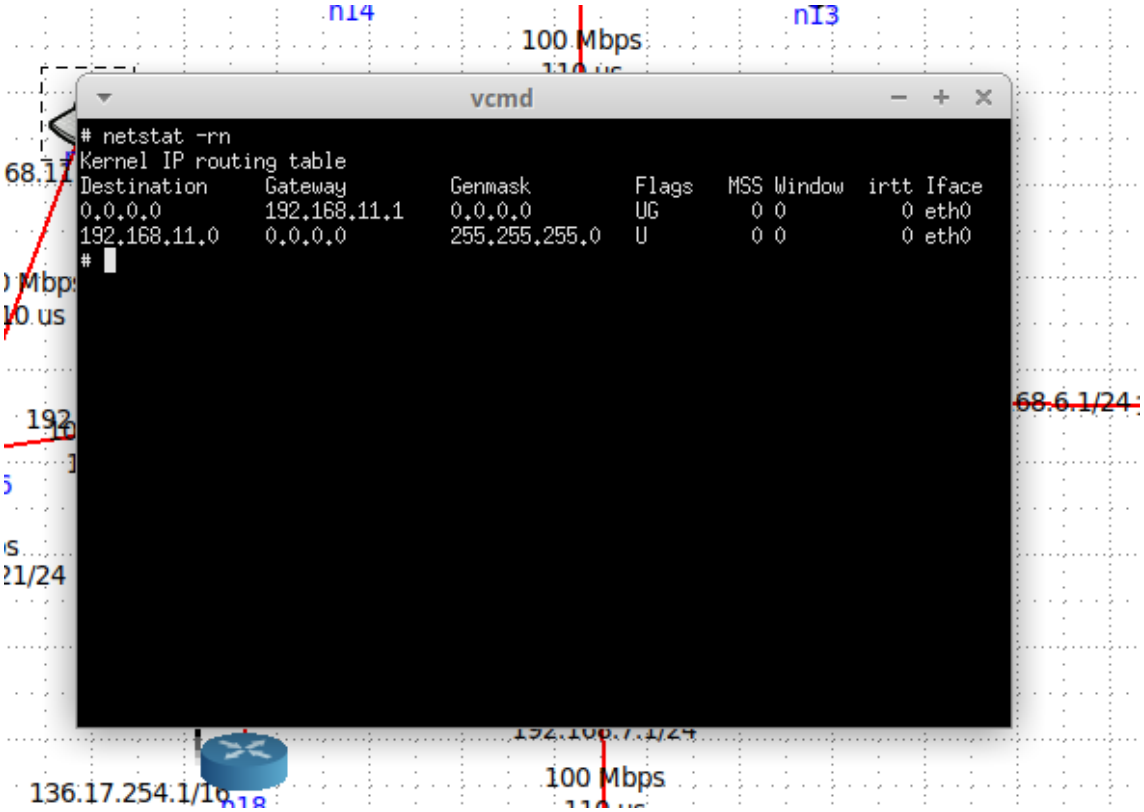


Figura 27 - Tabela de encaminhamento PC15

A partir do comando "netstat -rn", é possível observar as tabelas de encaminhamento do Router1 e do PC15.
As tabelas de encaminhamento apresentam as rotas que os pacotes efetuam dado um certo destino.
A coluna "Destination" indica a sub-rede destino. A coluna "Gateway" indica o próximo equipamento destino. A coluna "Genmask" indica a máscara utilizada.

b) Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico (sugestão: analise os processos que estão a correr em cada sistema (*hosts*, *routers*) usando, por exemplo, o comando *ps -ax*).

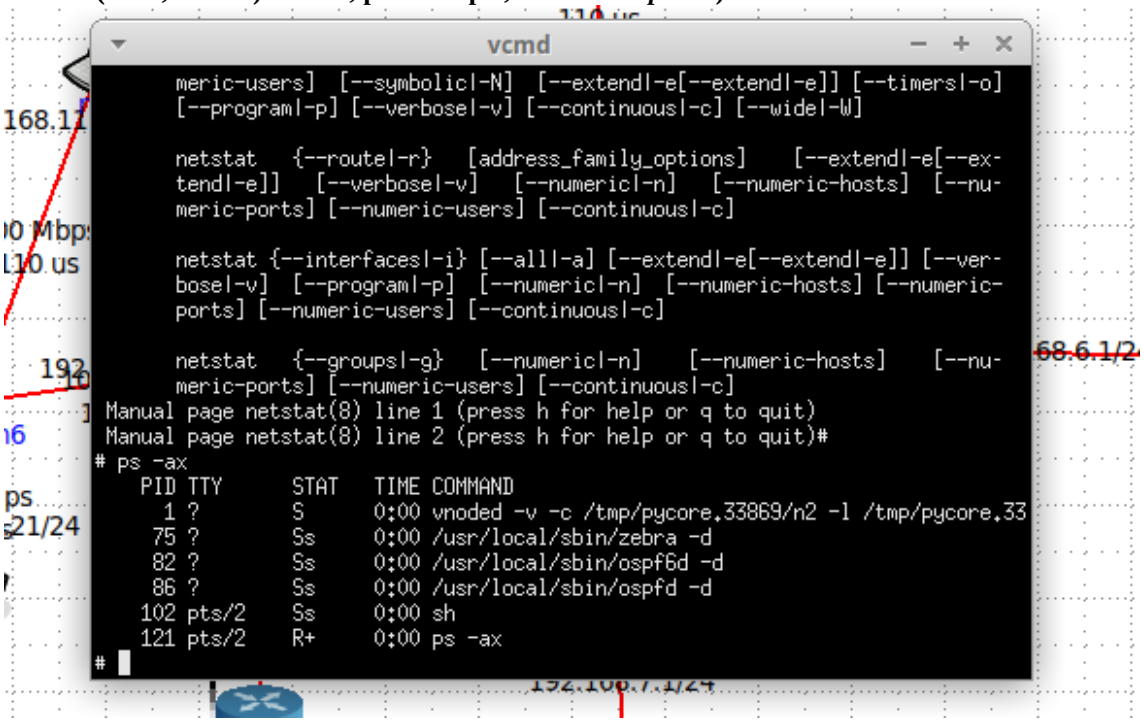


Figura 28 - ps -ax no PC15

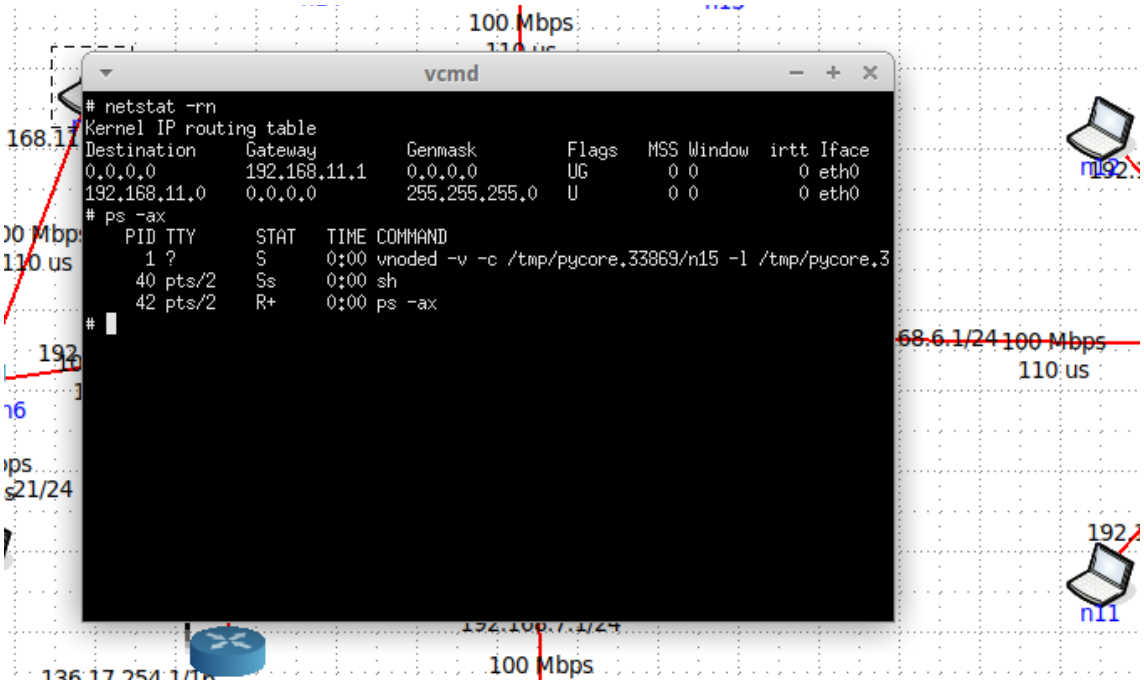


Figura 29

Tal como podemos observar nas figuras anteriores. Através do comando “px -ax” podemos observar que existem protocolos a correr, logo concluímos que o router está a utilizar endereçamento dinâmico. Por outro lado, o PC utiliza endereçamento estático.

c) Escolha um *PC* da organização REDES que esteja o mais distante possível em termos de saltos IP do servidor S1. Recorrendo apenas ao comando *traceroute –I*, responda às seguintes questões, justificando:

```
root@n17:/tmp/pycore.46111/n17.conf# traceroute -I 192.168.6.21
traceroute to 192.168.6.21 (192.168.6.21), 30 hops max, 60 byte packets
 1  192.168.11.10 (192.168.11.10)  3075.369 ms !H  3075.326 ms !H  3075.326 ms !
H
root@n17:/tmp/pycore.46111/n17.conf#
```

Figura 30 - PC selecionado - PC11

i. Existe conectividade IP desse *PC* para o servidor S1? A quantos saltos IP está o *PC* selecionado do servidor S1?

Existe sim conectividade e como podemos observar pela imagem acontecem no máximo 30 "hops".

ii. As rotas dos pacotes ICMP *echo reply* são as mesmas, mas em sentido inverso, das rotas dos pacotes ICMP *echo request* trocadas entre esse *PC* e o servidor S1? (Sugestão: trace a rota de S1 para o PC selecionado).

```
vcmd
root@n11:/tmp/pycore.46111/n11.conf# traceroute -I 192.168.11.10
traceroute to 192.168.11.10 (192.168.11.10), 30 hops max, 60 byte packets
 1  192.168.6.1 (192.168.6.1)  0.769 ms  0.713 ms *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *^C
root@n11:/tmp/pycore.46111/n11.conf#
```

Figura 31

Utilizando o comando "traceroute -I" do PC destino para o Servidor verificamos que é diferente.

d) 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. Se necessário, consulte o manual respetivo (*man route*). Que implicações tem esta medida para os utilizadores da organização REDES que acedem ao servidor. Justifique.

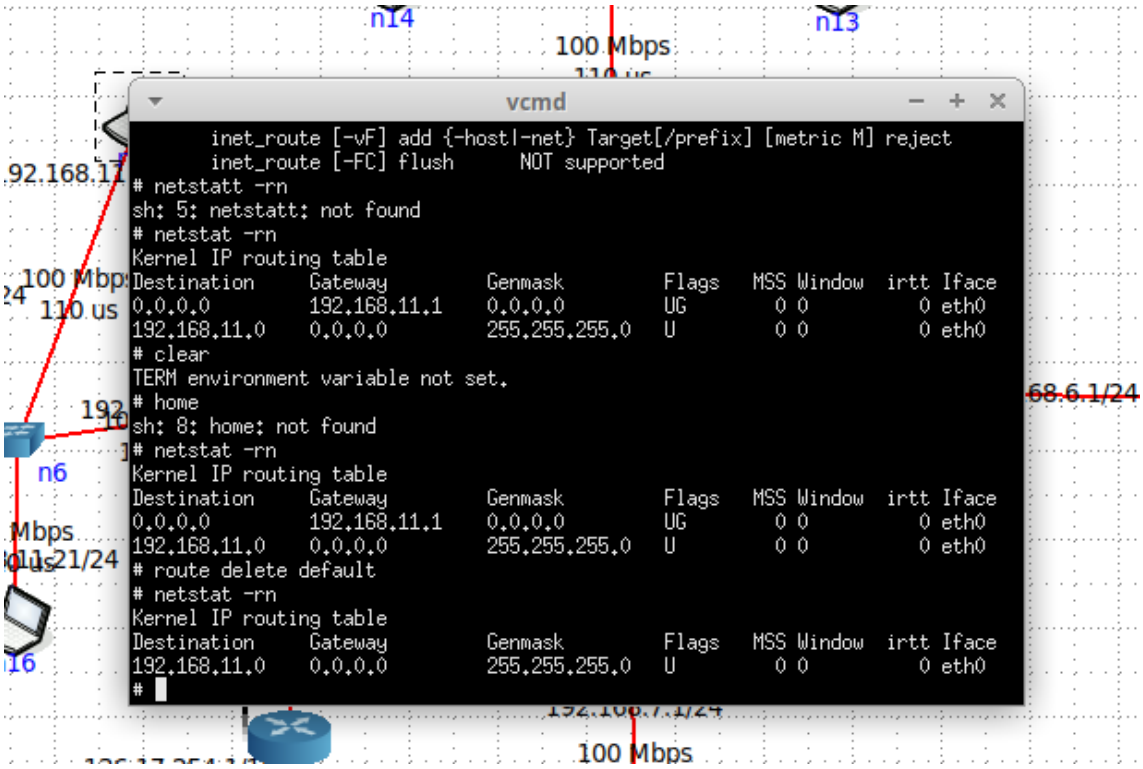


Figura 32

Ao eliminar a rota default tornamos impossível qualquer rota que se situe noutro departamento senão o próprio.

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

```
<n17.conf# route add -net 192.168.11.0 netmask 255.255.255.0 gw 192.168.11.10
<192.168.11.0 netmask 255.255.255.0 gw 192.168.11.10
SIOCADDRT: File exists
root@n17:/tmp/pycore.46111/n17.conf# route add -net 192.168.5.0 netmask 255.25>
root@n17:/tmp/pycore.46111/n17.conf# route add -net 192.168.6.0 netmask 255.25>
root@n17:/tmp/pycore.46111/n17.conf# route add -net 192.168.7.0 netmask 255.25>
<n17.conf# route add -net 136.17.254.0 netmask 255.255.255.0 gw 192.168.11.10
root@n17:/tmp/pycore.46111/n17.conf#
```

Figura 33

f) 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 S1.

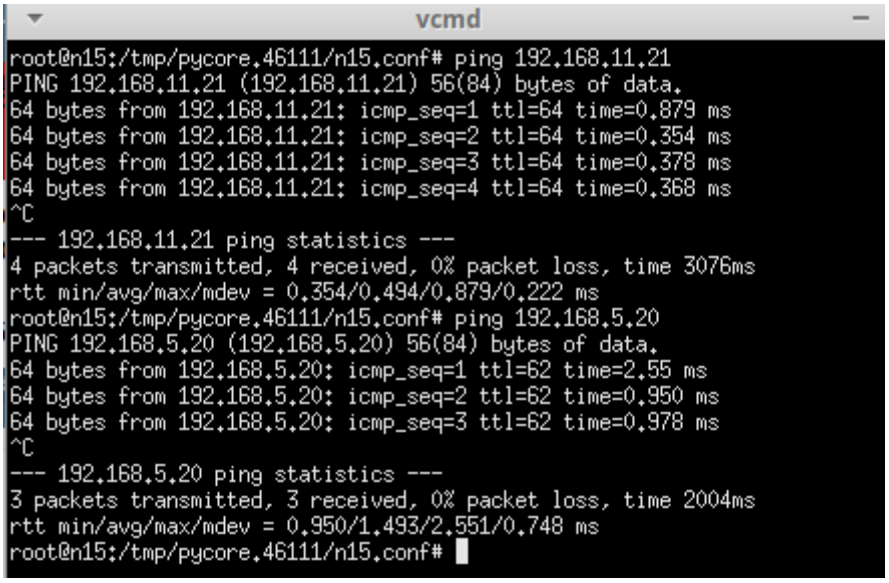


Figura 34 - PC15 para PC16 e PC15 para PC 14

Figura 35 - PC15 para PC11 e PC15 para PC9

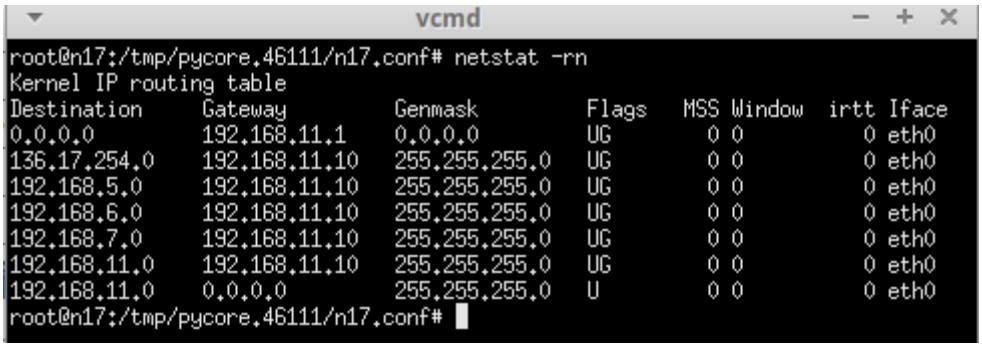


Figura 36 - Tabela de encaminhamento do N17

g) Apague nas tabelas de encaminhamento dos *routers* RA, RB, RC e RD a rede a que pertence a interface *wireless* do *router* RISP. Usando rotas por defeito, altere as tabelas de encaminhamento dos equipamentos da organização REDES de forma a possibilitar aos seus utilizadores o acesso genérico à Internet através do *router* RISP.

i. Apresente as novas tabelas de encaminhamento.

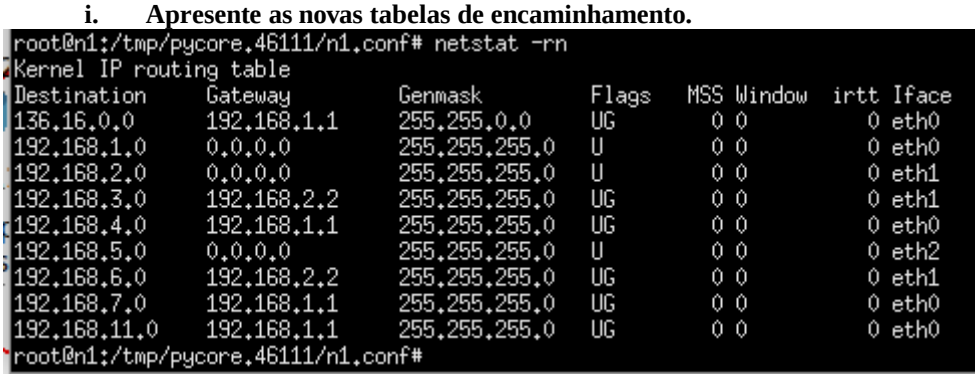


Figura 37 - Tabela de encaminhamento do router N1

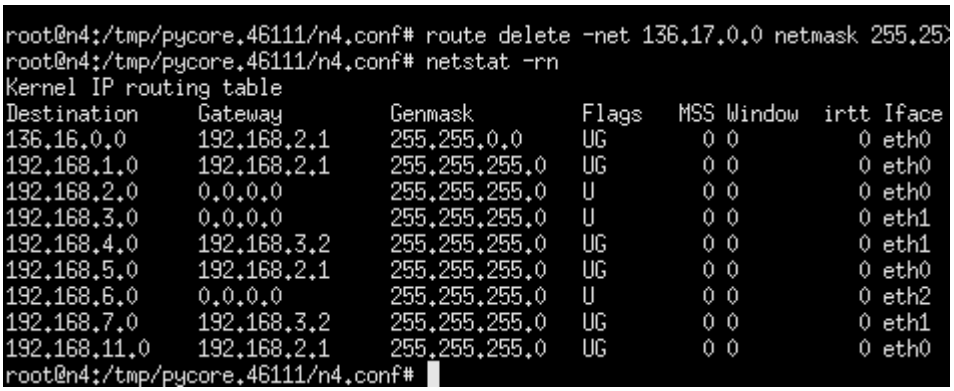


Figura 38 - Tabela de encaminhamento do router N4

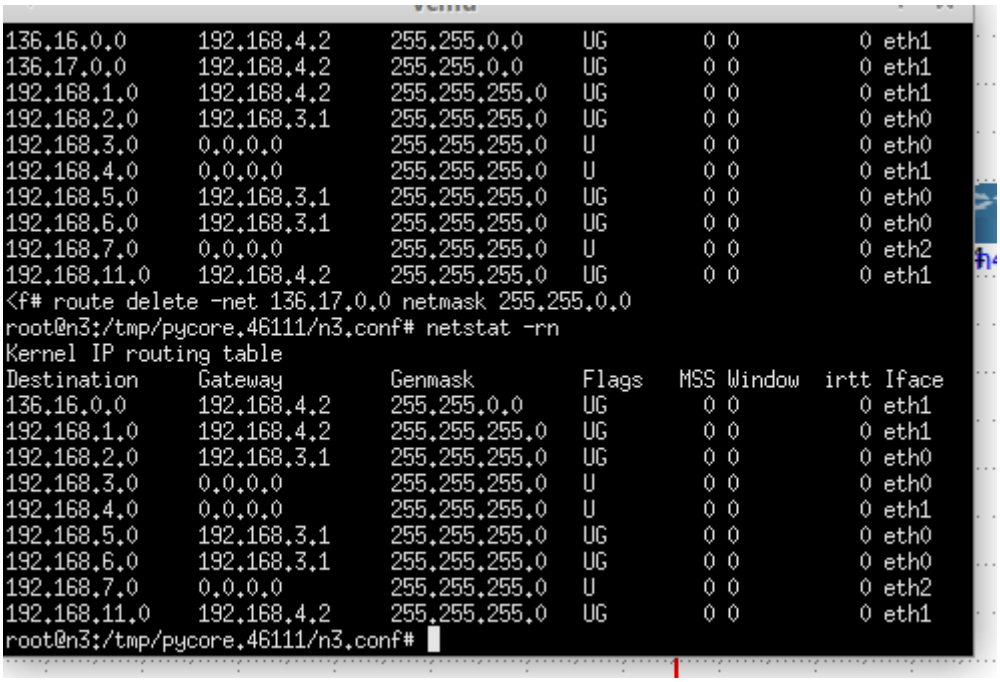


Figura 39- Tabela de encaminhamento do router N3

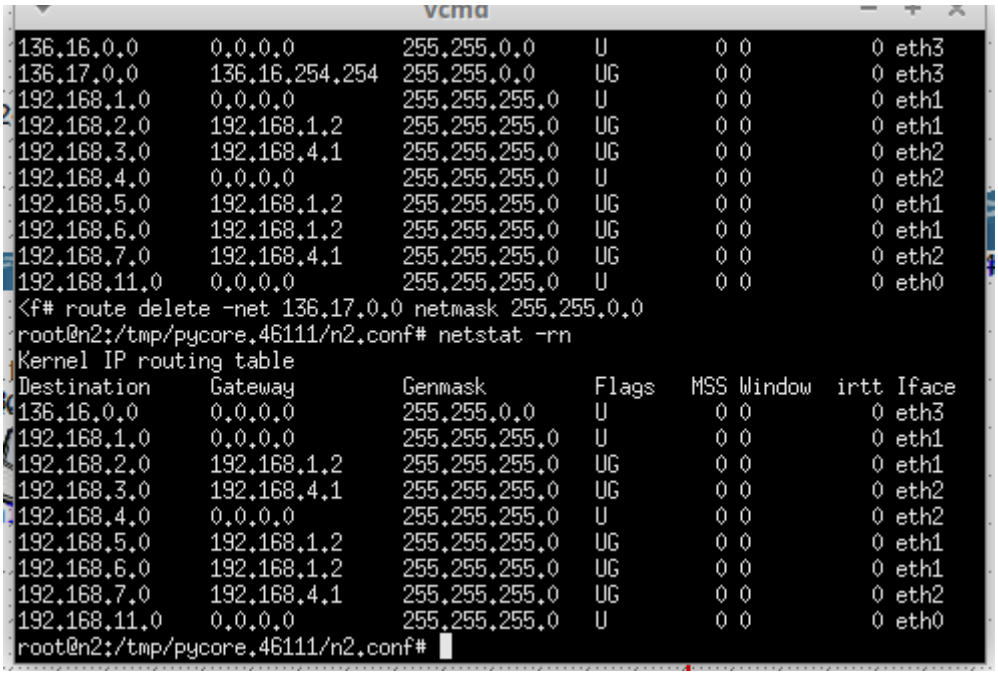


Figura 40 - Tabela de encaminhamento do router N1

ii. Teste a solução verificando a conetividade entre um *PC* de cada departamento e a interface *wireless* do *router* RISP (basta usar um *PC* por departamento).

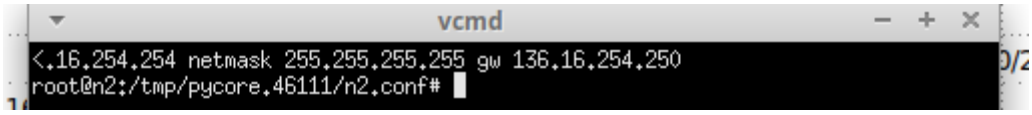


Figura 41 - Router n2

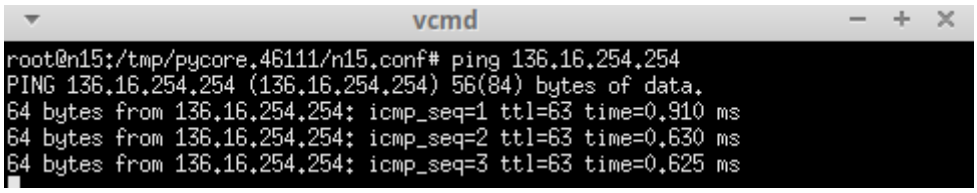


Figura 42 - Router N2

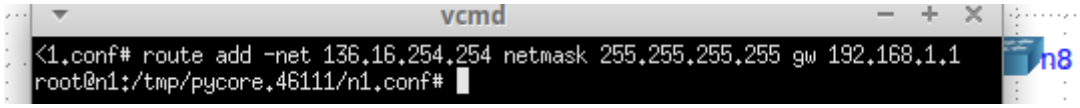


Figura 43 - Router N1

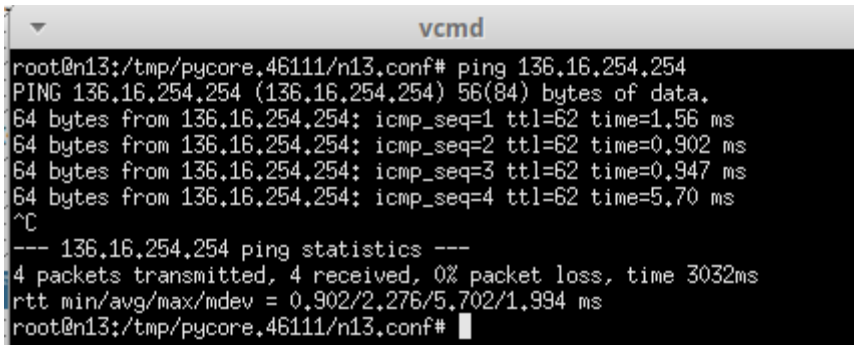


Figura 44 - Router N1

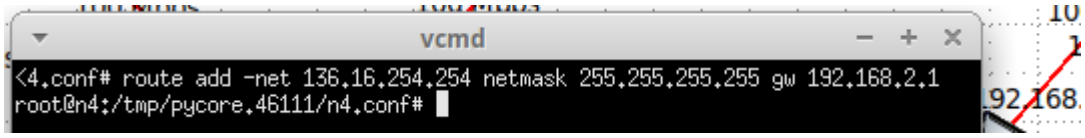


Figura 45 - Router N4

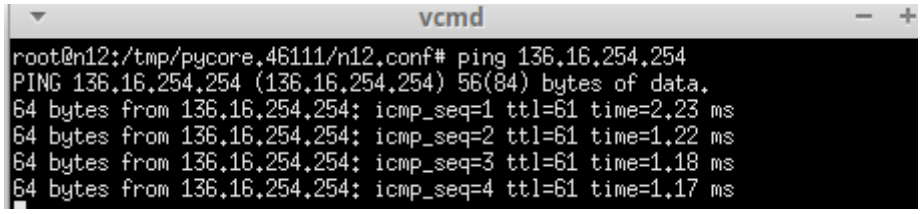


Figura 46 - Router N4

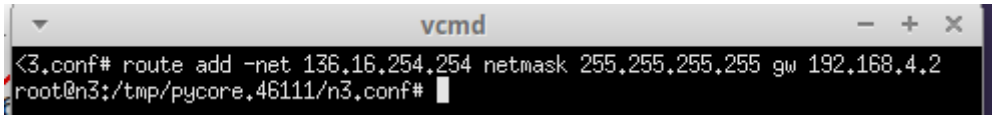


Figura 47 - Router N3

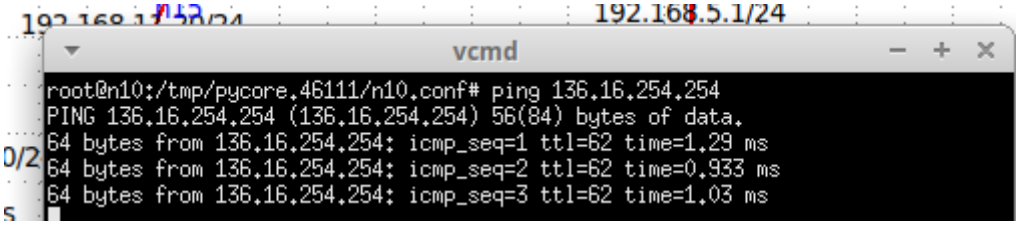


Figura 48 - Router N3

3. Por forma a minimizar a falta de endereços IPv4 é comum a utilização de sub-redes. Além disso, a definição de sub-redes permite uma melhor organização do espaço de endereçamento das redes em questão.
Para definir endereços de sub-rede é necessário usar a parte prevista para endereçamento de *host*, não sendo possível alterar o endereço de rede original. Recordase que o *subnetting*, ao recorrer ao espaço de endereçamento para *host*, implica que possam ser endereçados menos *hosts*.
Considere a topologia definida anteriormente. Assuma que o endereçamento entre os *routers* se mantém inalterado. Contudo, o endereçamento em cada departamento deve ser redefinido.

3.1 Considere que dispõe apenas do endereço de rede IP 150.XX.80.0/20, em que XX é o decimal correspondendo ao seu número de grupo (GXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo as redes de acesso e central (*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.

150.11.80.0/20 (G11)
Temos de criar 4 sub-redes, uma para cada departamento, por este motivo necessitamos de 3 bits pois 2^3 = 8 e 8-2 (endereços reservados) = 6.
A máscara passará a ter 23 bits (20+3 = 23).

Sub-redes:

mascara: 255.255.254.0

11111111.11111111.11111110.0(sub-rede) 00000 (host)
10010110.00001011.010100/0.(sub-rede) 0000000 (host)

Departamento A (001) - 150.11.80.32
Departamento B (010) - 150.11.80.64
Departamento C (100) - 150.11.81.0
Departamento D (101) - 150.11.81.32

Hosts:

10010110.00001011.010100/0.(sub-rede) 00/00000 (host)

Departamento A - 150.11.80.32/23
IP início: 150.11.80.32
IP fim: 150.11.80.62
Departamento B - 150.11.80.64/23
IP início: 150.11.80.64
IP fim: 150.11.80.94
Departamento C - 150.11.81.0/23
IP início: 150.11.81.0
IP fim: 150.11.81.30
Departamento D - 150.11.81.32/23
IP início: 150.11.81.32
IP fim: 150.11.81.62

Departamento A
PC1: 150.11.80.33
PC2: 150.11.80.34
S1: 150.11.80.37
RA: 150.11.80.60

Departamento B
PC1: 150.11.80.65
PC2: 150.11.80.66
RB: 150.11.80.67

Departamento C
PC1: 150.11.81.20
PC2: 150.11.81.21
RB: 150.11.81.28

Departamento D
PC1: 150.11.81.33
PC2: 150.11.81.35
RB: 150.11.81.36

3.2 Qual a máscara de rede que usou (em formato decimal)? Quantos *hosts* IP pode interligar no máximo em cada departamento? Quantos prefixos de sub-rede ficam disponíveis para uso futuro? Justifique.

A máscara passou de 20 para 23 sendo o seu valor decimal: 255.255.254.0
Como a nossa máscara é de 23 bits ficamos com 32-23=9 bits alteráveis.
Por este motivo em cada departamento será possível atribuir 2^9-1-2(endereços reservados) = 509 endereços de IP para host.

3.3 Garanta e verifique que conectividade IP entre as várias redes locais da organização REDES é mantida. Explique como procedeu.

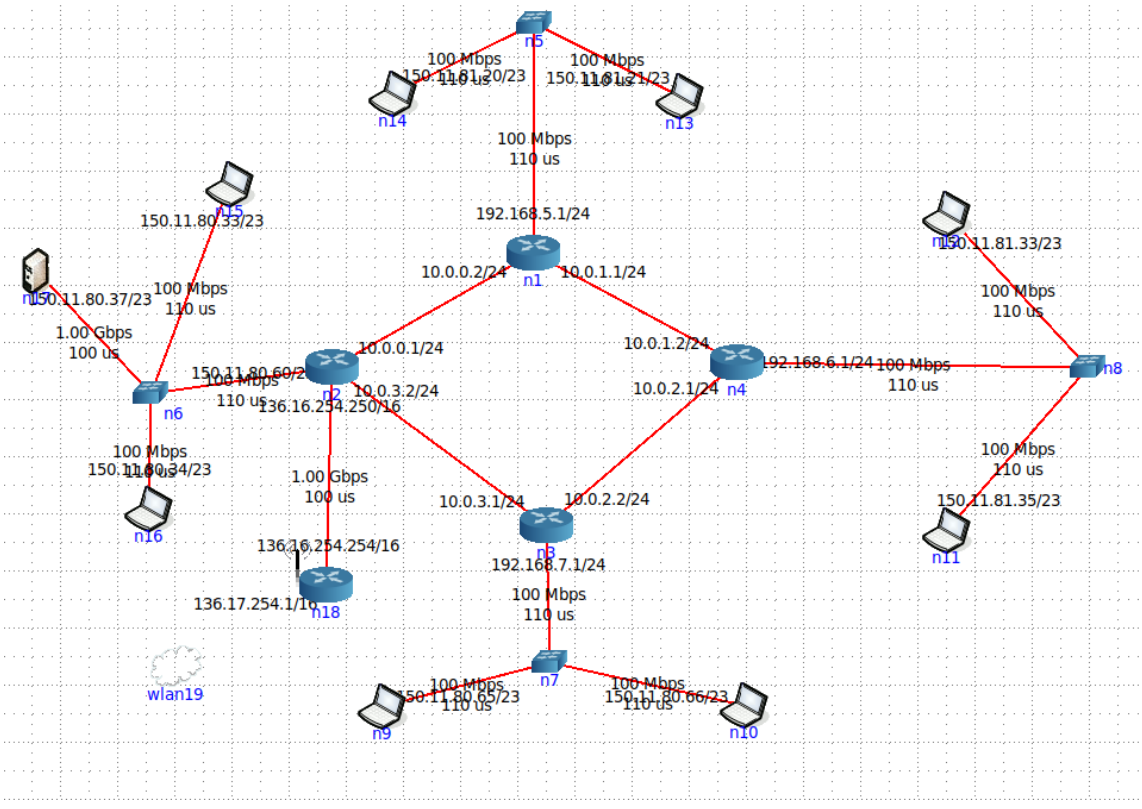


Figura 49

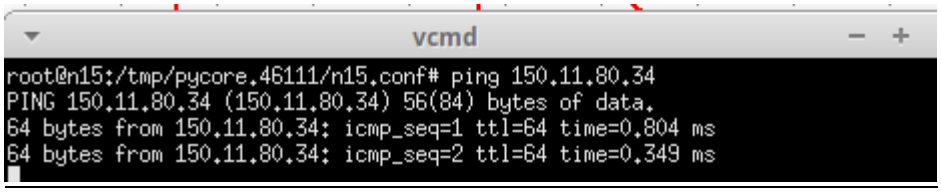


Figura 50

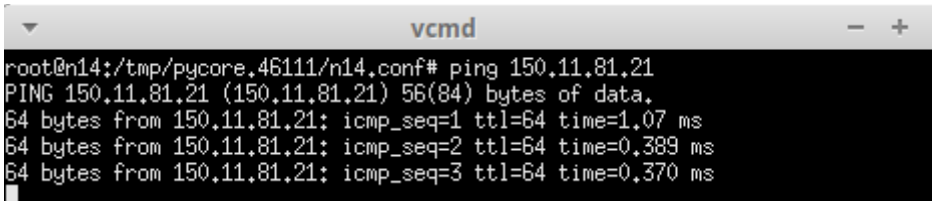


Figura 51

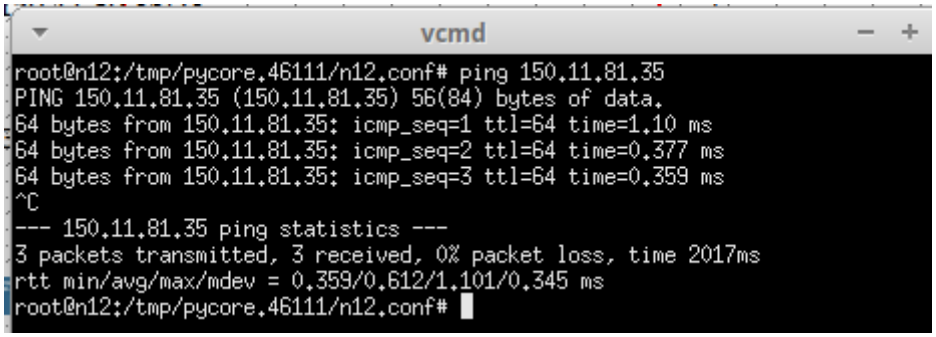


Figura 52

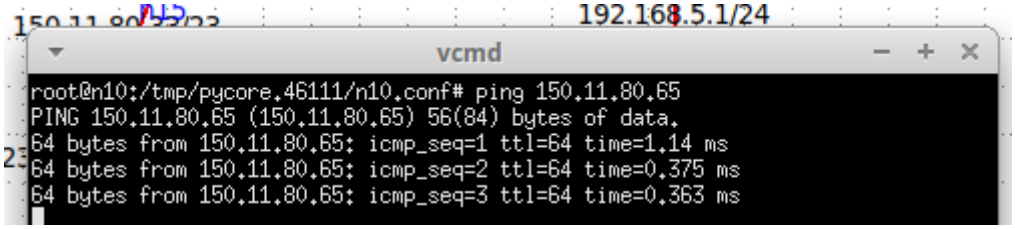


Figura 53

3. Conclusão

Na 1ª parte deste trabalho, foi feita a análise do tráfego ICMP e do protocolo IPv4 tal como alguns casos onde ocorria fragmentação, devido ao pacote IP ter tamanho superior ao MTU da rede.

Com isto, conseguimos enfatizar a nossa capacidade de análise de resultados e das flags, análise do tráfego ICMP no Wireshark, e também a análise de comandos como o tracert, bem como a importância e influencia do TTL na comunicação entre equipamentos.

Já na 2ª parte, procedemos à análise nomeadamente de tabelas de endereçamento e encaminhamento IP, sendo que com isto conseguimos ter um conhecimento maior sobre subnetting. Deste modo analisámos o uso, mais uma vez, de alguns comandos como netstat -rn.

Já com exercício final, conseguimos perceber as formas de definir subnets tendo por base um endereço de rede IP.

Este trabalho foi bastante propicio para a melhoria do nosso conhecimento de subnetting, dado o facto de termos de efetuar todas as ações e cálculos por nós mesmos.

Na realização do trabalho, tivemos alguma duvidas, sobretudo na criação de tabelas de endereçamento. Contudo, com esforço, ajuda do respetivo docente da UC e com alguma pesquisa, pensámos ter comprido os objetivos estipulados inicialmente para o mesmo.