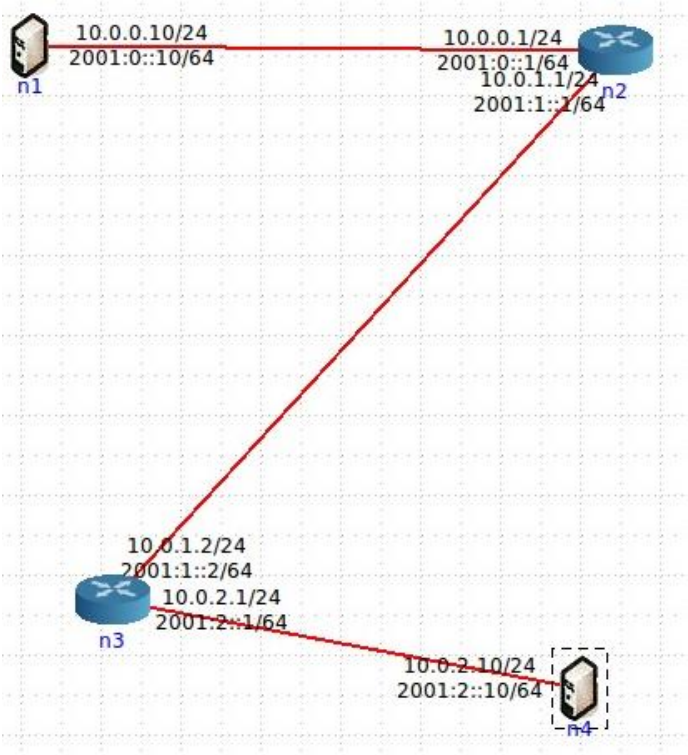
Hugo Sousa A91654, Isabel Inácio A91664, Pedro Martins A91681

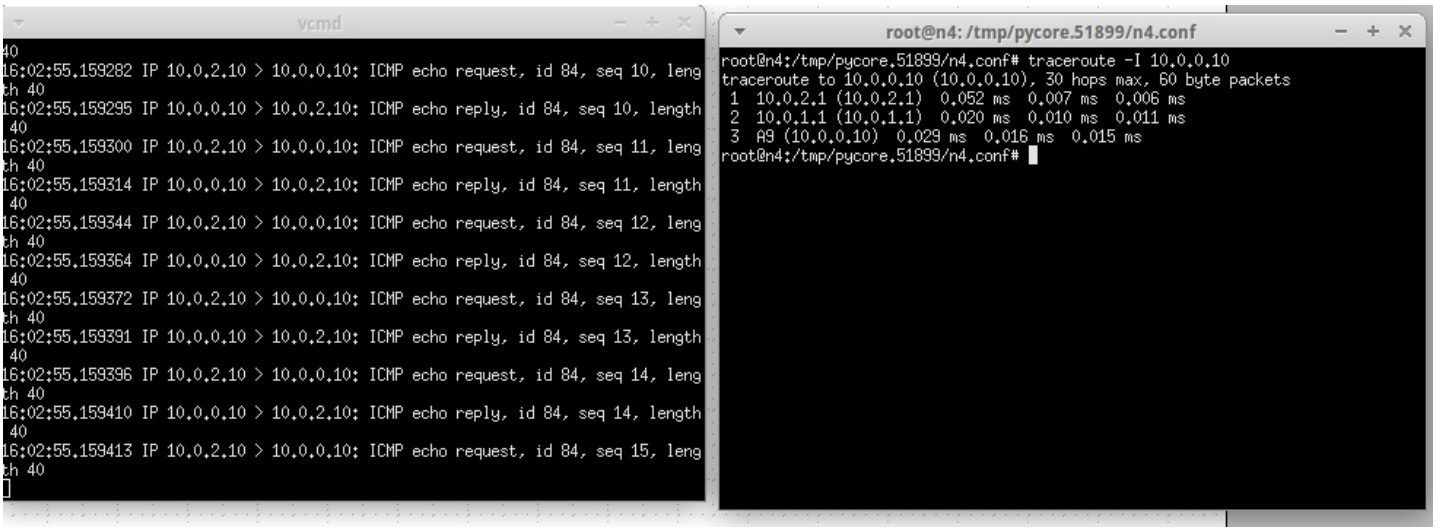
# TP4 - Relatório - Grupo 1

## Parte 1

1. **Prepare uma topologia CORE para verificar o comportamento do *traceroute*. Ligue um host n1 a um *router* n2, que se liga a um *router* n3 que, por sua vez, se liga a um host n4.**




a) **Active o *wireshark* ou o *tcpdump* no nó 4. Numa *shell* de n4, execute o comando *traceroute -I* para o endereço IP do n1.**



**b) Registe e analise o tráfego ICMP enviado por n4 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.**

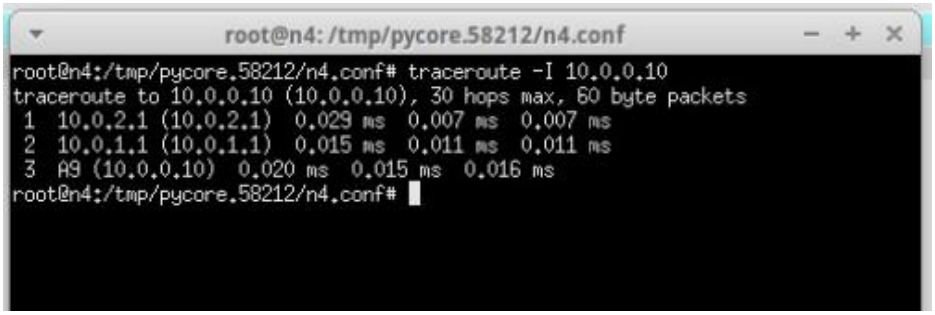
| | | | | | |
|---|---|---|---|---|---|
| 3 | 9.125299 | 10.0.2.10 | 10.0.0.10 | ICMP | 74 Echo (ping) request id=0x008f, seq=1/256, ttl=1 |
| 4 | 9.125318 | 10.0.2.10 | 10.0.2.10 | ICMP | 102 Time-to-live exceeded (Time to live exceeded in transit) |
| 5 | 9.125324 | 10.0.2.10 | 10.0.0.10 | ICMP | 74 Echo (ping) request id=0x008f, seq=2/512, ttl=1 |
| 6 | 9.125329 | 10.0.2.10 | 10.0.2.10 | ICMP | 102 Time-to-live exceeded (Time to live exceeded in transit) |
| 7 | 9.125333 | 10.0.2.10 | 10.0.0.10 | ICMP | 74 Echo (ping) request id=0x008f, seq=3/768, ttl=1 |
| 8 | 9.125337 | 10.0.2.10 | 10.0.2.10 | ICMP | 102 Time-to-live exceeded (Time to live exceeded in transit) |
| 9 | 9.125341 | 10.0.2.10 | 10.0.0.10 | ICMP | 74 Echo (ping) request id=0x008f, seq=4/1024, ttl=2 |
| 10 | 9.125354 | 10.0.1.10 | 10.0.2.10 | ICMP | 102 Time-to-live exceeded (Time to live exceeded in transit) |
| 11 | 9.125359 | 10.0.2.10 | 10.0.0.10 | ICMP | 74 Echo (ping) request id=0x008f, seq=5/1280, ttl=2 |
| 12 | 9.125368 | 10.0.1.10 | 10.0.2.10 | ICMP | 102 Time-to-live exceeded (Time to live exceeded in transit) |
| 13 | 9.125372 | 10.0.2.10 | 10.0.0.10 | ICMP | 74 Echo (ping) request id=0x008f, seq=6/1536, ttl=2 |
| 14 | 9.125381 | 10.0.1.10 | 10.0.2.10 | ICMP | 102 Time-to-live exceeded (Time to live exceeded in transit) |
| 15 | 9.125386 | 10.0.2.10 | 10.0.0.10 | ICMP | 74 Echo (ping) request id=0x008f, seq=7/1792, ttl=3 |
| 16 | 9.125403 | 10.0.0.10 | 10.0.2.10 | ICMP | 74 Echo (ping) reply id=0x008f, seq=7/1792, ttl=62 |

**R:** O host n4, enviou três pacotes com TTL = 1 para n1, mas como o TTL era insuficiente, ao chegar a n3 é enviada uma mensagem de erro para o host, que além de informar que o TTL é insuficiente, também fornece o endereço do router n3. Posteriormente n4 reenvia os três pacotes, mas desta vez com TTL = 2, mais uma vez, como o TTL é insuficiente, ao chegar a n2 é enviada uma mensagem de erro contendo também o endereço de n2. Por fim n4 reenvia mais uma vez os três pacotes, mas desta vez com TTL = 3, como foi suficiente para chegar a n1, não foi enviada uma mensagem de erro. Concluímos assim o percurso efetuado pelos pacotes de n4 até n1, bem como o TTL mínimo necessário para o envio dos mesmos.

**c) Qual deve ser o valor inicial mínimo do campo TTL para alcançar o destino n1?**

**R:** O valor inicial mínimo do campo TTL para alcançar o destino n1 deve ser 3.

**d) Qual o tempo médio de ida-e-volta (RTT - round-trip time) obtido?**




**R:** O tempo médio é de (0.020 + 0.015 + 0.016) /3 = 0.017 ms.

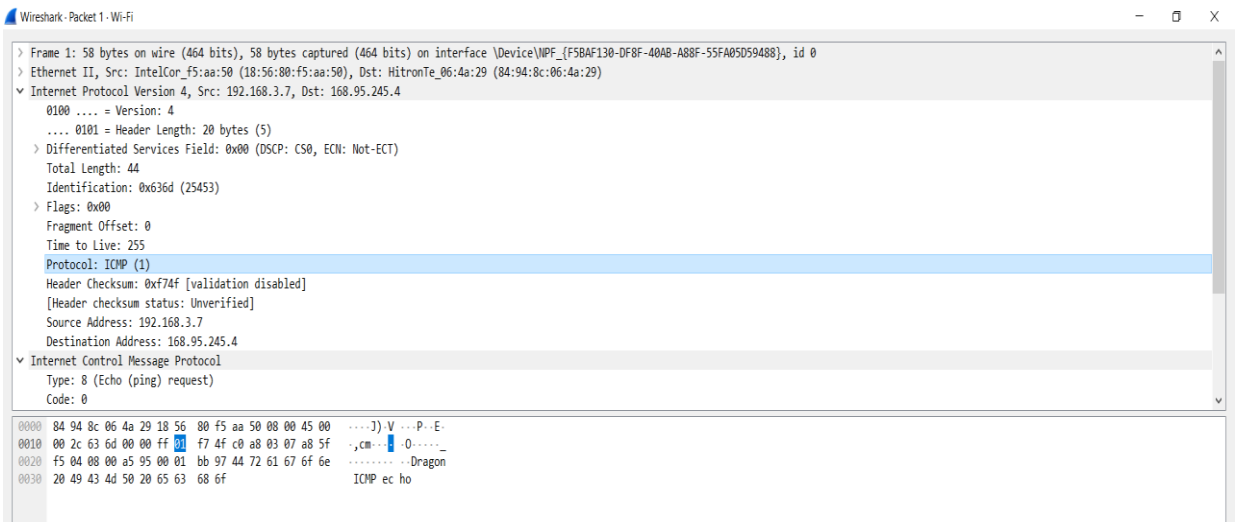
**2. <u>Procedimento:</u> Usando o *wireshark* capture o tráfego gerado pelo *traceroute* para os seguintes tamanhos de pacote: (i) <u>por defeito</u> e (ii) <u>30XX bytes</u> (XX é o seu número de grupo). <u>Utilize como máquina destino o host marco.uminho.pt.</u> Com base no tráfego capturado, identifique os pedidos ICMP *Echo Request* e o conjunto de mensagens devolvidas em resposta a esses pedidos.**

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

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 192.168.3.7 | 168.95.245.4 | ICMP | 58 | Echo (ping) request id=0x0001, seq=48023/38843, ttl=255 (reply in 2) |

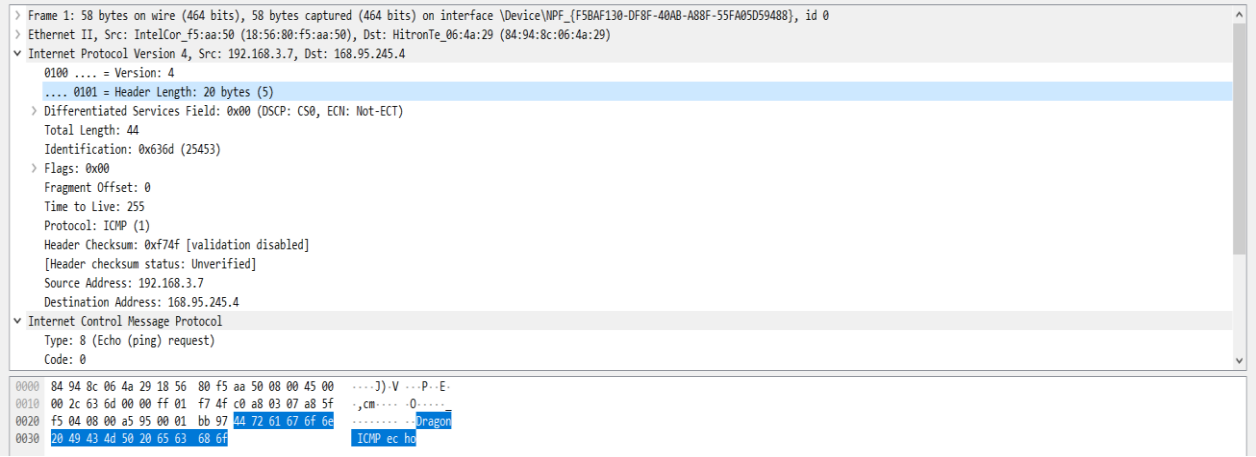
**R:** O endereço IP da interface ativa no meu computador é **192.168.3.7**.

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



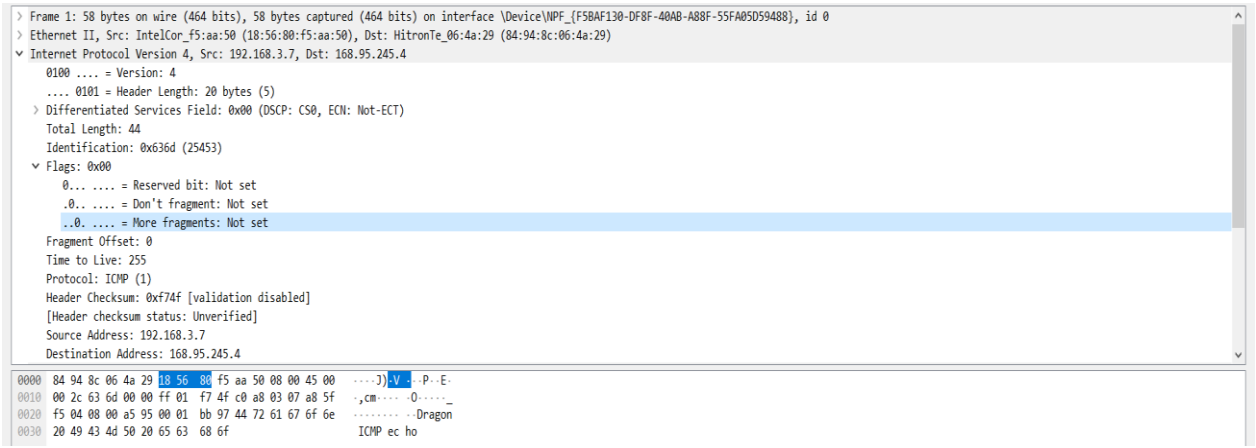

**R:** Como é possível visualizar na imagem o valor do campo protocolo é -> **ICMP (1)**. Este campo identifica o protocolo usado para transmitir o datagrama.

**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*?**




**R:** Como é apresentado na imagem o número de bytes existentes no cabeçalho IP(v4) é **20**. Os bytes do payload são calculados subtraindo o número de bytes total que neste caso são **44** com o número de bytes do cabeçalho IP(v4), ou seja, fica **44-20=22 bytes**.

**d) O datagrama IP foi fragmentado? Justifique.**




**R:** O datagrama IP não foi fragmentado, pois como é possível observar na imagem o segmento **more fragments** está apresentado como (**Not set**). E como não existe fragmentação também o **Fragment Offset** aparece a **0**.

**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 com base no IP gerado na sua máquina. Observa algum padrão nos valores do campo de Identificação do datagrama IP e TTL?**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 192.168.3.7 | 168.95.245.4 | ICMP | 58 | Echo (ping) request id=0x0001, seq=48023/38843, ttl=255 (reply in 2) |
| 3 | 0.448909 | 192.168.3.7 | 10.4.63.254 | NBNS | 92 | Name query NBSTAT *<00><00><00><00><00><00><00><00><00><00><00><00><00><00><00><00> |
| 4 | 0.490649 | 192.168.3.7 | 10.137.206.113 | NBNS | 92 | Name query NBSTAT *<00><00><00><00><00><00><00><00><00><00><00><00><00><00><00><00> |
| 5 | 0.536466 | 192.168.3.7 | 10.255.12.78 | NBNS | 92 | Name query NBSTAT *<00><00><00><00><00><00><00><00><00><00><00><00><00><00><00><00> |
| 6 | 0.578802 | 192.168.3.7 | 10.255.12.73 | NBNS | 92 | Name query NBSTAT *<00><00><00><00><00><00><00><00><00><00><00><00><00><00><00><00> |
| 8 | 1.278700 | 192.168.3.7 | 168.95.245.4 | ICMP | 58 | Echo (ping) request id=0x0001, seq=48024/39099, ttl=255 (reply in 16) |
| 9 | 1.317090 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48025/39355, ttl=255 (reply in 10) |
| 11 | 1.355744 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48026/39611, ttl=1 (no response found!) |
| 12 | 1.393936 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48027/39867, ttl=2 (no response found!) |
| 14 | 1.432311 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48028/40123, ttl=3 (no response found!) |
| 17 | 1.470363 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48029/40379, ttl=4 (no response found!) |
| 19 | 1.509144 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48030/40635, ttl=5 (no response found!) |
| 21 | 1.547378 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48031/40891, ttl=6 (no response found!) |
| 22 | 1.585863 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48032/41147, ttl=7 (no response found!) |
| 23 | 1.624075 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48033/41403, ttl=8 (no response found!) |
| 24 | 1.662504 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48034/41659, ttl=9 (no response found!) |
| 26 | 1.701183 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48035/41915, ttl=10 (no response found!) |
| 28 | 1.739329 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48036/42171, ttl=11 (no response found!) |
| 29 | 1.778015 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48037/42427, ttl=12 (no response found!) |
| 30 | 1.816844 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48038/42683, ttl=13 (no response found!) |
| 31 | 1.855695 | 192.168.3.7 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=48039/42939, ttl=14 (reply in 32) |

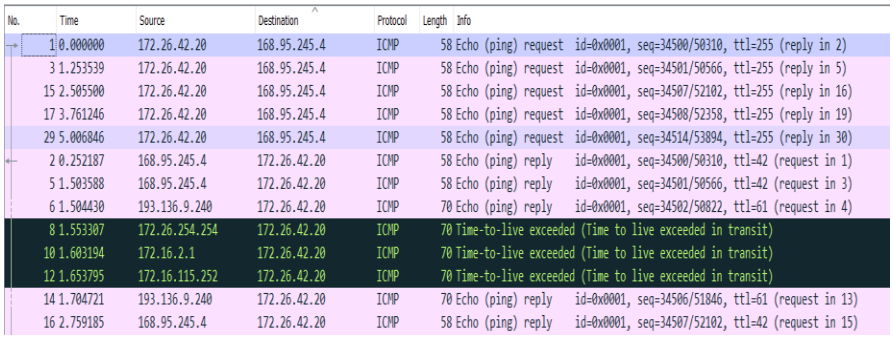
```
> Frame 11: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{F5BAF130-DF8F-40AB-A88F-55FA05D59488}, id 0
> Ethernet II, Src: IntelCor_f5:aa:50 (18:56:80:f5:aa:50), Dst: HitronTe_06:4a:29 (84:94:8c:06:4a:29)
v Internet Protocol Version 4, Src: 192.168.3.7, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0xa82b (43051)
  > Flags: 0x00
    Fragment Offset: 0
  > Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x8272 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.3.7
    Destination Address: 193.136.9.240
```

```
> Frame 12: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{F5BAF130-DF8F-40AB-A88F-55FA05D59488}, id 0
> Ethernet II, Src: IntelCor_f5:aa:50 (18:56:80:f5:aa:50), Dst: HitronTe_06:4a:29 (84:94:8c:06:4a:29)
v Internet Protocol Version 4, Src: 192.168.3.7, 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: 0xa82c (43052)
  > Flags: 0x00
    Fragment Offset: 0
  > Time to Live: 2
    Protocol: ICMP (1)
    Header Checksum: 0x8171 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.3.7
    Destination Address: 193.136.9.240
```

```
> Frame 14: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{F5BAF130-DF8F-40AB-A88F-55FA05D59488}, id 0
> Ethernet II, Src: IntelCor_f5:aa:50 (18:56:80:f5:aa:50), Dst: HitronTe_06:4a:29 (84:94:8c:06:4a:29)
v Internet Protocol Version 4, Src: 192.168.3.7, 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: 0xa82d (43053)
  > Flags: 0x00
    Fragment Offset: 0
  > Time to Live: 3
    Protocol: ICMP (1)
    Header Checksum: 0x8070 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.3.7
    Destination Address: 193.136.9.240
```

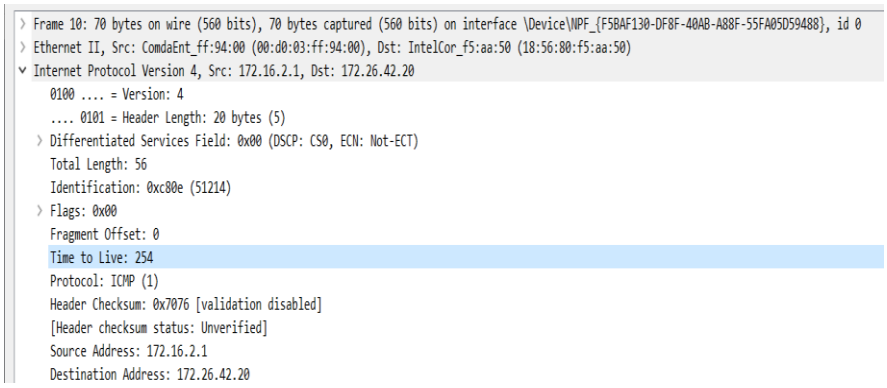
**R:** Se observarmos as imagens em cima apresentadas podemos ver que a parte referente aos valores do campo de identificação do datagrama IP nos frames 11, 12 e 14 vão aumentando 1 unidade ao valor de frame para frame. Relativamente ao campo Time to live, já que foi usado o Pingplotter observamos também que aumenta 1 unidade por frame.

**f) A seguir (com os pacotes ordenados por endereço destino) encontre a série de respostas ICMP TTL *exceeded* enviadas ao seu computador pelo primeiro router. Qual é o valor do campo TTL? Esse valor permanece constante para todas as mensagens de resposta ICMP TTL *exceeded* enviadas? Porquê?**

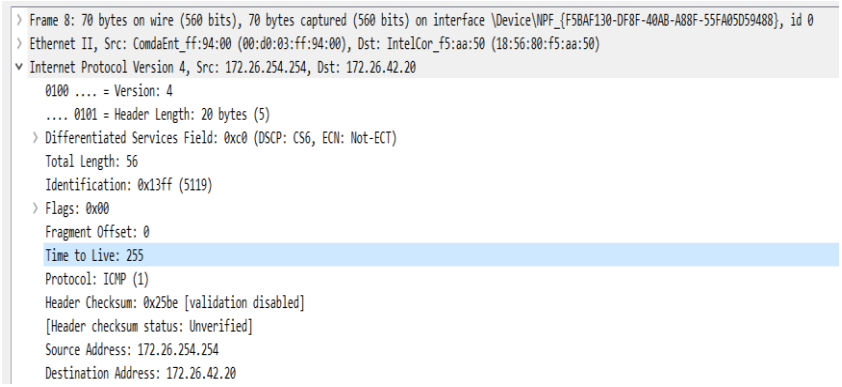


| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 172.26.42.20 | 168.95.245.4 | ICMP | 58 | Echo (ping) request id=0x0001, seq=34500/50310, ttl=255 (reply in 2) |
| 3 | 1.253539 | 172.26.42.20 | 168.95.245.4 | ICMP | 58 | Echo (ping) request id=0x0001, seq=34501/50566, ttl=255 (reply in 5) |
| 15 | 2.505500 | 172.26.42.20 | 168.95.245.4 | ICMP | 58 | Echo (ping) request id=0x0001, seq=34507/52102, ttl=255 (reply in 16) |
| 17 | 3.761246 | 172.26.42.20 | 168.95.245.4 | ICMP | 58 | Echo (ping) request id=0x0001, seq=34508/52358, ttl=255 (reply in 19) |
| 29 | 5.006846 | 172.26.42.20 | 168.95.245.4 | ICMP | 58 | Echo (ping) request id=0x0001, seq=34514/53894, ttl=255 (reply in 30) |
| 2 | 0.252187 | 168.95.245.4 | 172.26.42.20 | ICMP | 58 | Echo (ping) reply id=0x0001, seq=34500/50310, ttl=42 (request in 1) |
| 5 | 1.503588 | 168.95.245.4 | 172.26.42.20 | ICMP | 58 | Echo (ping) reply id=0x0001, seq=34501/50566, ttl=42 (request in 3) |
| 6 | 1.504430 | 193.136.9.240 | 172.26.42.20 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=34502/50822, ttl=61 (request in 4) |
| 8 | 1.553307 | 172.26.254.254 | 172.26.42.20 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 10 | 1.603194 | 172.16.2.1 | 172.26.42.20 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 12 | 1.653795 | 172.16.115.252 | 172.26.42.20 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 14 | 1.704721 | 193.136.9.240 | 172.26.42.20 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=34506/51846, ttl=61 (request in 13) |
| 16 | 2.759185 | 168.95.245.4 | 172.26.42.20 | ICMP | 58 | Echo (ping) reply id=0x0001, seq=34507/52102, ttl=42 (request in 15) |

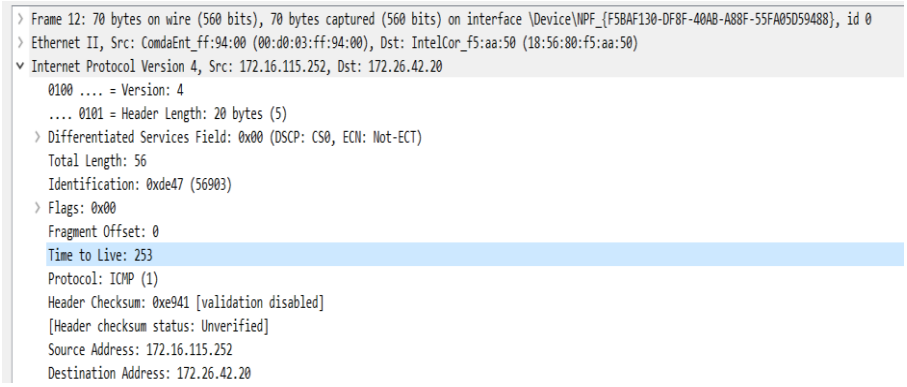
**(captura realizada na aula tp)**



```
> Frame 10: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{F5BAF130-DF8F-40AB-A88F-55FA05D59488}, id 0
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: IntelCor_f5:aa:50 (18:56:80:f5:aa:50)
v Internet Protocol Version 4, Src: 172.16.2.1, Dst: 172.26.42.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0xc80e (51214)
  > Flags: 0x00
    Fragment Offset: 0
    Time to Live: 254
    Protocol: ICMP (1)
    Header Checksum: 0x7076 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.16.2.1
    Destination Address: 172.26.42.20
```



```
> Frame 8: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{F5BAF130-DF8F-40AB-A88F-55FA05D59488}, id 0
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: IntelCor_f5:aa:50 (18:56:80:f5:aa:50)
v Internet Protocol Version 4, Src: 172.26.254.254, Dst: 172.26.42.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x13ff (5119)
  > Flags: 0x00
    Fragment Offset: 0
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x25be [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.254.254
    Destination Address: 172.26.42.20
```




**R:** O primeiro valor apresentado é 255. Este valor não vai permanecer constante como é possível ver nas imagens em cima.

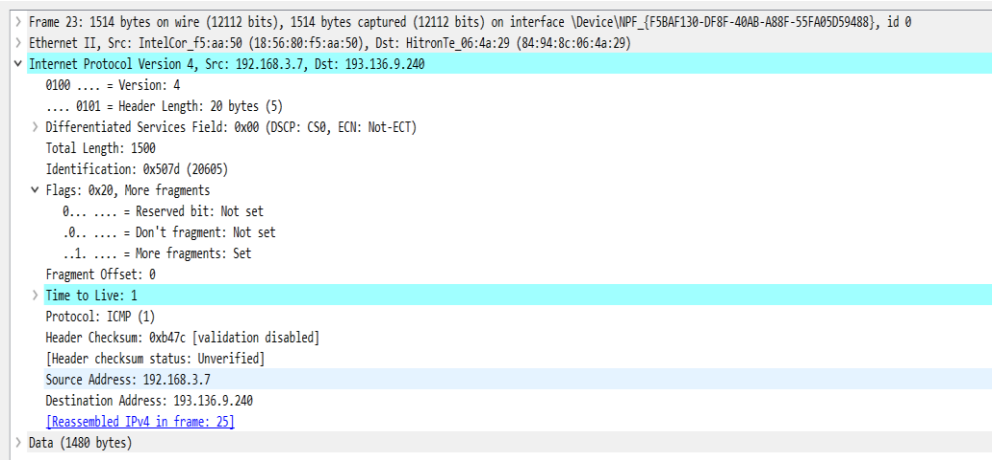
Esta alteração do valor deve-se ao facto de que cada router diferente vai fazer com que o TTL decremente 1unidade até chegar ao destino.

**3. Pretende-se agora analisar a fragmentação de pacotes IP. Reponha a ordem do tráfego capturado usando a coluna do tempo de captura. Observe o tráfego depois do tamanho de pacote ter sido definido em 30XX *bytes*.**

**a) Localize a primeira mensagem ICMP. A mensagem foi fragmentada? Porque é que houve (ou não) necessidade de o fazer?**

**R:** Como nós queríamos capturar pacotes com 3021bytes, e os frames de Ethernet só conseguem carregar até1500 bytes de dados, então conseguimos afirmar que a mensagem foi fragmentada em 3.

**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 se trata do primeiro fragmento? Qual é o tamanho deste datagrama IP?**

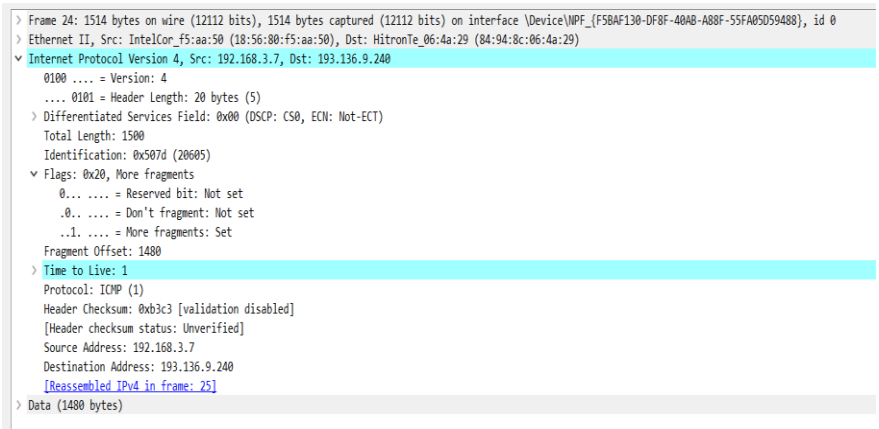



**R:** Como podemos observar na figura no campo referente às **flags**, é nos mencionada a existência de mais fragmentos.

Podemos afirmar que é o primeiro fragmento devido ao valor apresentado no **Fragment offset**, que é **0**.

O tamanho do datagrama IP é **1500bytes**.

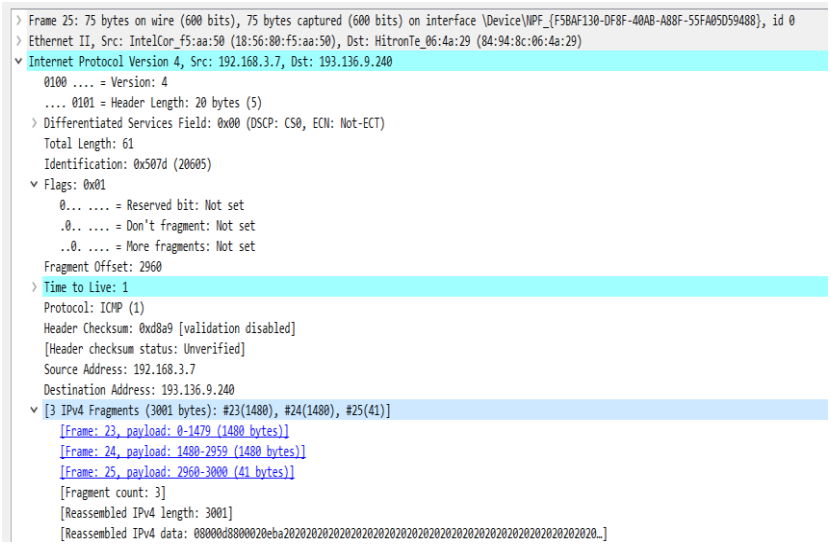
**c) Imprima o segundo fragmento do datagrama IP original. Que informação do cabeçalho IP indica que não se trata do 1º fragmento? Há mais fragmentos? O que nos permite afirmar isso?**




**R:** Uma vez que podemos observar na figura que o campo **Fragment Offset** se apresenta com **1480**, logo é diferente de **0** o que nos permite afirmar que não se trata do primeiro fragmento.

Podemos sim dizer que existem mais fragmentos dado que no campo **flags**, nos afirma isso.

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




**R:** Como é visível na figura foram criados 3 fragmentos a partir do datagrama original.

O último fragmento correspondente ao datagrama original pode ser detetado quando o campo **More fragments** apresenta-se como (**Not set**) e o campo **Fragment Offset** diferente de **0**. De acordo com a figura podemos afirmar que este é o último fragmento.

**e)** **Indique, resumindo, os campos que mudam no cabeçalho IP entre os diferentes fragmentos, e verifique a forma como essa informação permite reconstruir o datagrama original.**

**R:** Observando as 3 figuras das alíneas anteriores, podemos fazer comparações tanto entre o primeiro e segundo fragmento, como o segundo e o último fragmento e obter assim as principais diferenças. Dito isto na primeira comparação conseguimos analisar diferenças nos campos **Fragment offset** e no **Header checksum**. Na segunda observamos diferenças nos campos **Total Lenght**, **More fragments**, **Fragment offset** e no **Header checksum**.

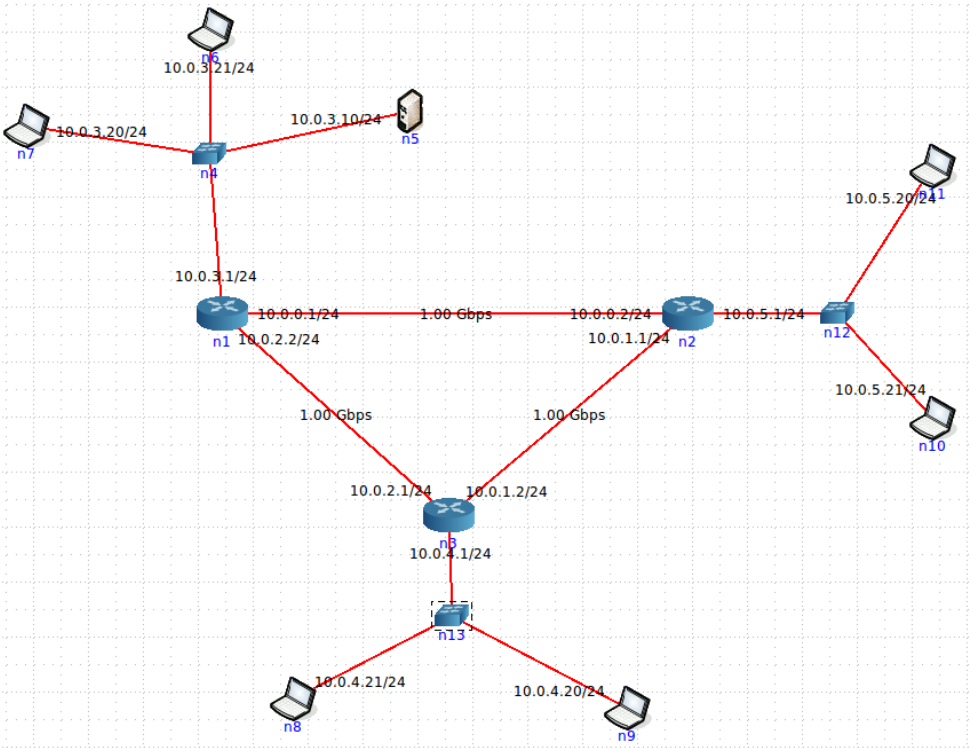
Como o campo **Identification** é igual em todos os fragmentos, no destino é possível saber quais dos fragmentos vão originar um datagrama.

Com o campo **More fragments** conseguimos saber se ainda existem mais fragmentos, sendo que quando este aparece mencionado como (Not set) se refere ao ultimo. O **Fragment count** que aparece na imagem do último fragmento permite-nos saber em quantos fragmentos foi dividido o datagrama original. E por último o campo **Fragment offset** diz-nos em que posição no datagrama original se encontra cada fragmento.

## Parte 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. Se preferir, pode incluir uma imagem que ilustre de forma clara a topologia e o endereçamento.**



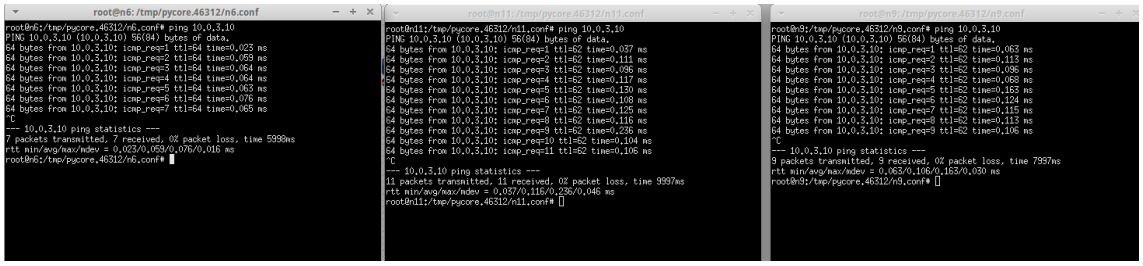

**b) Tratam-se de endereços públicos ou privados? Porquê?**

**R:** Tratam-se de endereços privados, porque são encontrados dentro do intervalo 10.0.0.0 até 10.255.255.255.

**c) Porque razão não é atribuído um endereço IP aos *switches*?**

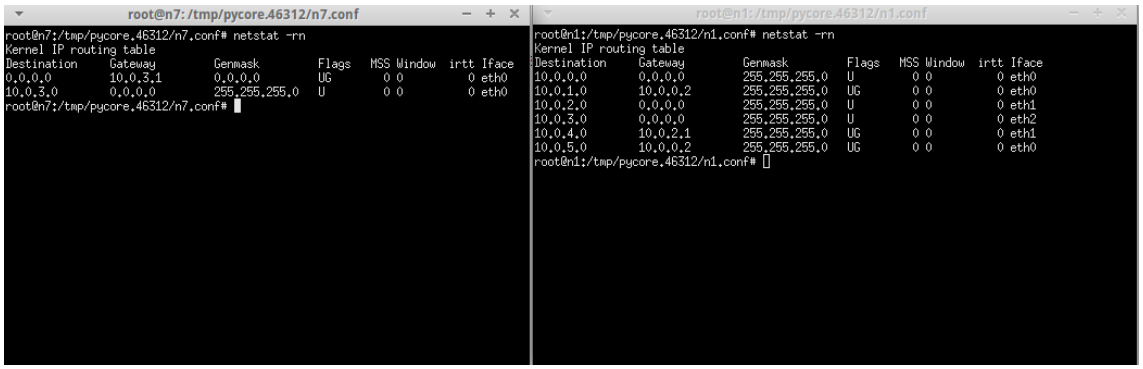
**R:** Devido ao facto do switch se encontrar no nível 2, este não possui endereços lógicos, como os endereços IP, apenas endereços físicos.

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



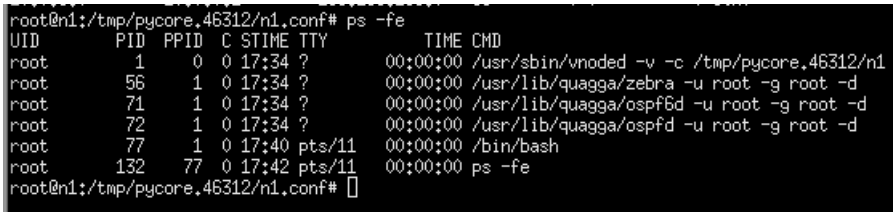

## 2. Para o *router* e um *laptop* do departamento A:

**a) Execute o comando <u>netstat –rn</u> 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 (<u>man netstat</u>).**



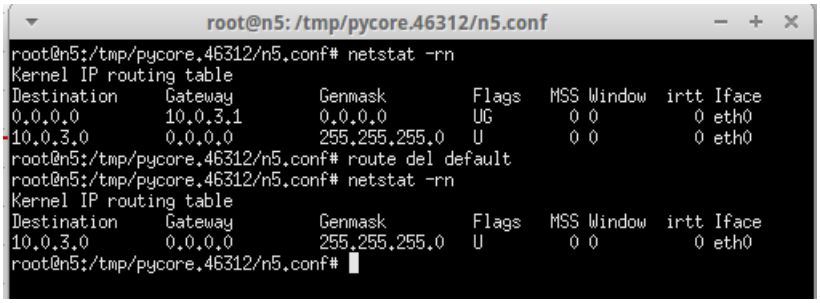

**R:** O **Destinacion** é a rede destino. O **Gateway** é o endereço IP que permite chegar à rede destino. A **Genmask** é a máscara de cada uma das redes. A **Flags** pode ser "U" e "G", "U" significa que podemos chegar ao destino partindo do endereço que foi utilizado e o "G" significa que é necessário utilizar o Gateway para o destino através do endereço que foi utilizado. A **MSS** é o que nos indica o tamanho máximo de segmento padrão para conexões TCP nessa rota. A **Window** é o que nos indica o tamanho da janela padrão também para conexões TCP nessa rota. A **Irtt** é o que indica o tempo inicial de ida e volta para esta rota. O **Iface** é o que nos mostra a interface de rede, se não tiver apenas uma, teria "lo", para loopback, "eth0", primeiro dispositivo Ethernet, e "eth1", para o segundo dispositivo Ethernet e assim continua para o número de interfaces instaladas.

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



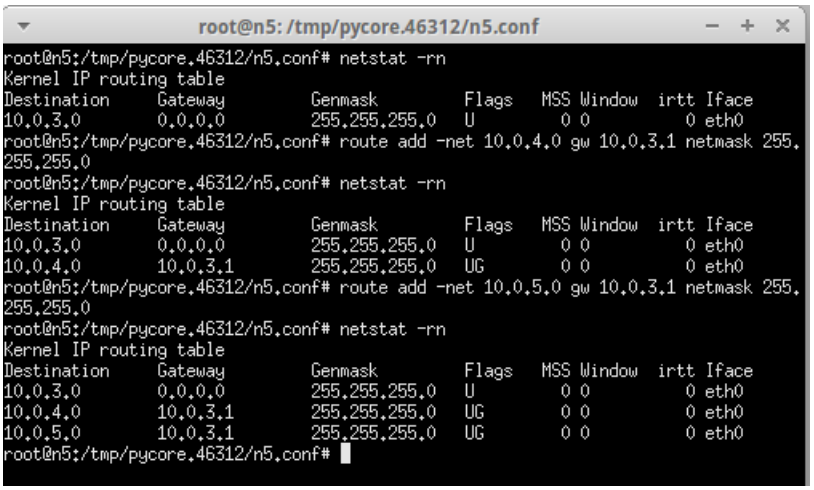

**R:** Como há ospf significa que está a ser utilizado o encaminhamento dinâmico, pois os routers trocam informação routing entre eles.

**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 localizado no departamento A. Use o comando route delete para o efeito. Que implicações tem esta medida para os utilizadores da empresa que acedem ao servidor. Justifique.**

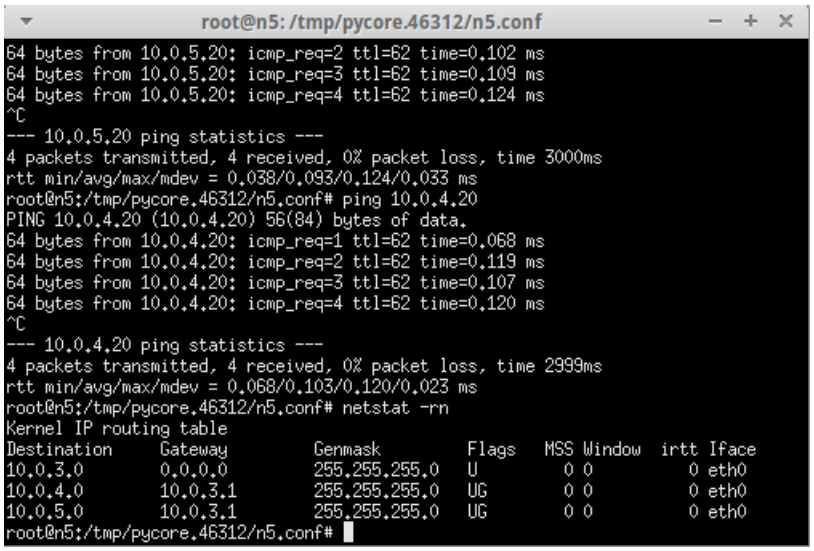



**R:** Ao tirar o default, os utilizadores apenas conseguem aceder a máquinas que estejam ligadas à mesma rede, pois a rota foi retirada para as máquinas das outras redes.

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



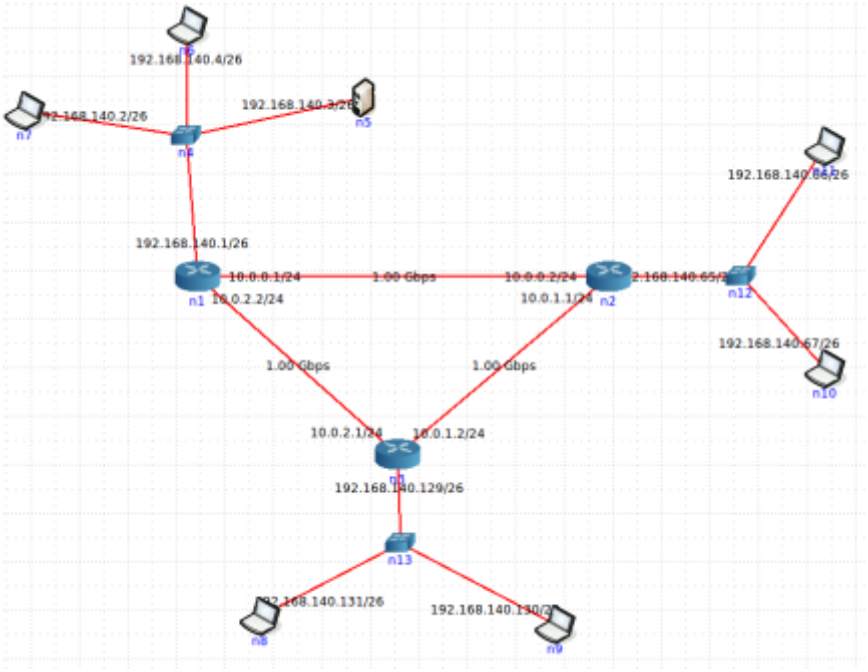

**e) Teste a nova política de encaminhamento garantindo que o servidor está novamente acessível, utilizando para o efeito o comando *ping*. Registe a nova tabela de encaminhamento do servidor.**




## 3. Definição de Sub-redes

**Considere a topologia definida anteriormente. Assuma que o endereçamento entre os routers se mantém inalterado, contudo, o endereçamento em cada departamento deve ser redefinido.**

**1) Assumindo que dispõe apenas de um único endereço de rede IP classe C 192.168.140.0/24, defina um novo esquema de endereçamento para as redes dos departamentos (mantendo a rede de core inalterada) e atribua endereços às interfaces dos vários sistemas envolvidos. Deve justificar as opções usadas.**




**R:** Para as sub-redes usamos dois bits, portanto temos 4 sub-redes: 0, 64, 128, 192. Os valores de host possíveis são: [1-62], [65-126], [129-190] e [193-254].

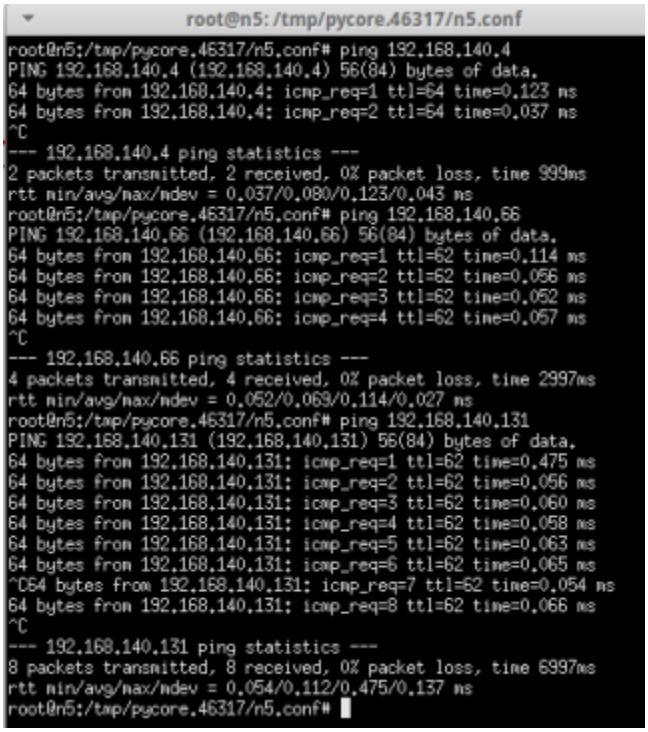
**2) Qual a máscara de rede que usou (em formato decimal)? Justifique.**

**R:** A máscara de rede usada foi: 255.255.255.192 em decimal /26.

**3) Quantos *hosts* IP pode interligar em cada departamento? Justifique.**

**R:** Em cada departamento podemos ter 62 hosts, porque os valores dos hosts possíveis para os departamentos A, B e C são [1-62], [65-126] e [129-190] respetivamente.

**4) Garanta que conectividade IP entre as várias redes locais da empresa LCCnet é mantida.**




## Conclusões:

Com estes exercícios fomos capazes de perceber mais sobre a matéria dada nas aulas teóricas como por exemplo, subnetting, máscaras, tabelas de encaminhamento e de endereçamento, VLSM, CICR, etc.