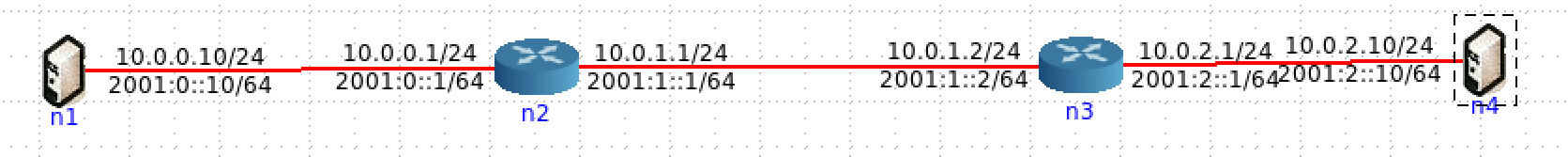
**TP4 – Protocolo IPv4**

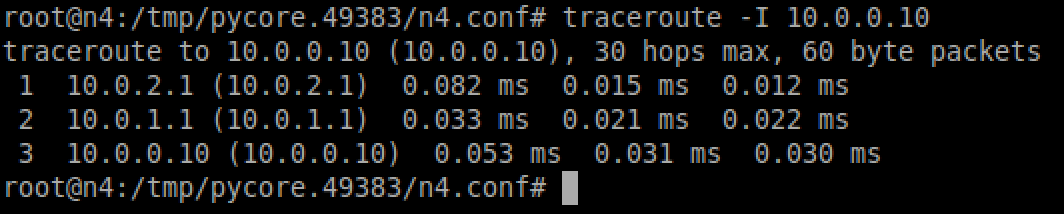
**Questões e Respostas**

**Parte I**

1. **Prepare uma topologia CORE para verificar o comportamento do *traceroute*. Ligue um *host* n1 a um router n2; o router n2 a um router n3 que, por sua vez, se liga a um *host* n4 (note que pode não existir conectividade IP imediata entre n1 e n4 até que o *routing* estabilize).**

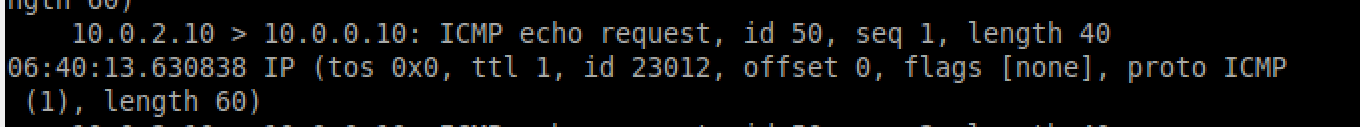


1. **Active o *wireshark* ou o *tcpdump* no *host* n4. Numa *shell* de n4, execute o comando *traceroute* -I para o endereço IP do *host* n1.**

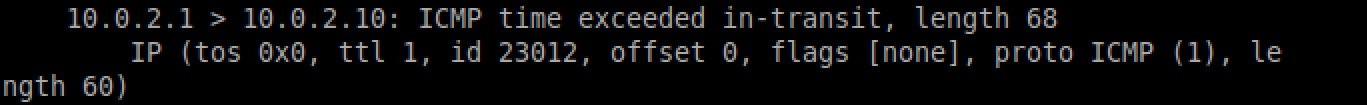


1. **Registe e analise o tráfego ICMP enviado por n4 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.**

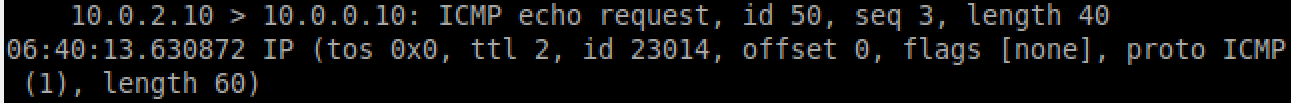
Primeiramente, n4 envia três datagramas (correspondentes a um *echo request*) com o campo TTL (*Time To Live*) igual a 1.

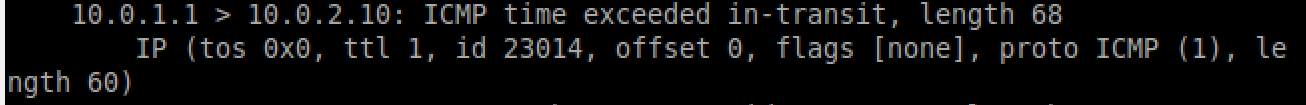


Como o número de saltos é inferior ao mínimo necessário para chegar de n4 a n1, é recebida uma mensagem de controlo ICMP (*Internet Control Message Protocol*) informando da falha no envio (i.e. *time exceeded in-transit*), procedente de n3. É de notar, no entanto, que apesar de a mensagem ICMP fazer referência a um tempo excedido, na verdade, o que aconteceu foi que o datagrama utilizou todos os saltos possíveis (definidos pelo TTL), sendo que a falha no envio não se deve a nenhuma razão temporal.

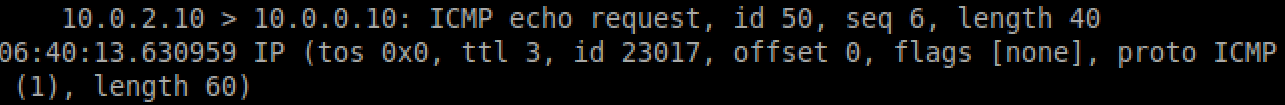


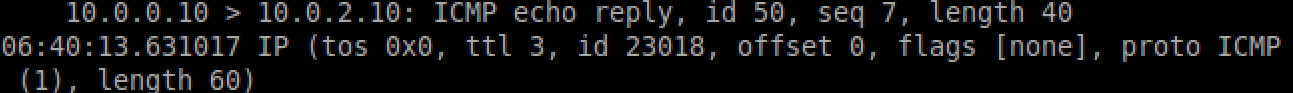
De seguida, são enviados mais três datagramas com o campo TTL igual a 2, ocorrendo o mesmo (sendo que a mensagem ICMP, *time exceeded in-transit*, é enviada por n2 e não por n3).



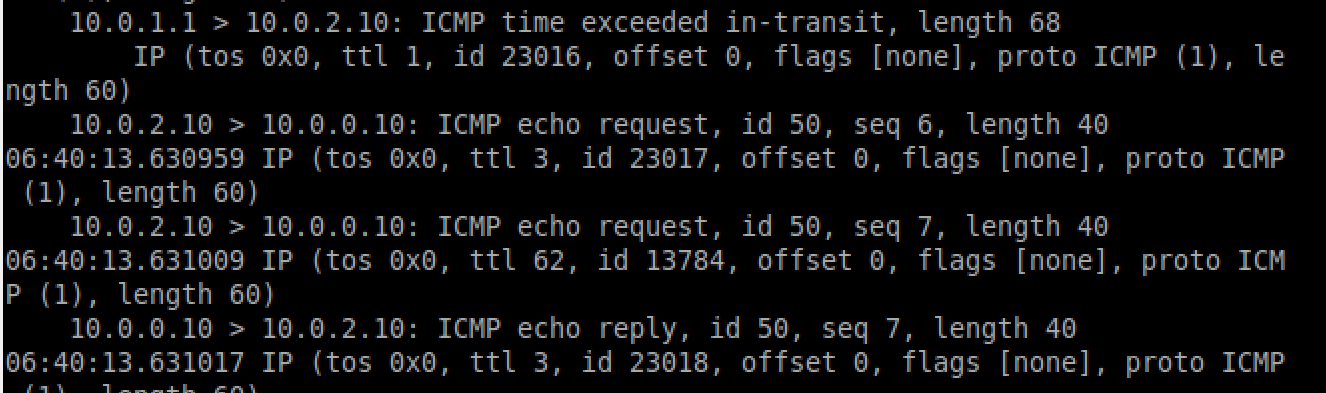


Por último, são enviados outros três datagramas mas agora com o TTL igual a 3 pelo que, como já se atingiu número mínimo de saltos para chegar de n4 a n1, é recebida a resposta ao *echo* *request* (i.e. um *echo* *reply*) procedente de n1.





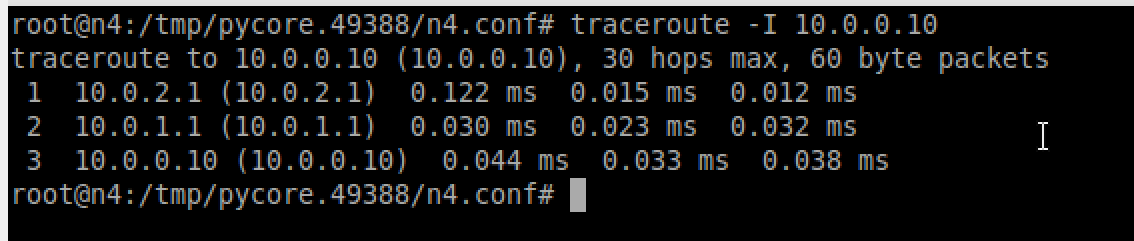
1. **Qual deve ser o valor inicial mínimo do campo TTL para alcançar o destino n1? Verifique na prática que a sua resposta está correta.**



O valor inicial mínimo do campo TTL para alcançar o destino n1 deve ser 3 (de n4 para n3 (1), de n3 para n2 (2) e de n2 para n1 (3)\).

Na prática, é isso que acontece: verifica-se que para os datagramas enviados por n4 com o TTL igual a 3, é enviado (por n1) a resposta ao *echo request* (i.e. um *echo reply*).

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



Para se calcular o *Round-Trip Time*, basta calcular a média dos valores obtidos para cada uma das três tramas enviadas:

* TTL 1 (n3) : = 0.050 ms
* TTL 2 (n2) : = 0.028 ms
* TLL 3 (n1) : = 0.038 ms

1. **Pretende‐se agora usar o *traceroute* na sua máquina nativa, e gerar de datagramas IP de diferentes tamanhos.**
2. **Qual é o endereço IP da interface ativa do seu computador?**

C:\Users\win8\AppData\Local\Microsoft\Windows\INetCacheContent.Word\2-a-IP.PNG

O endereço IP da interface ativa do nosso computador é 192.168.100.200.

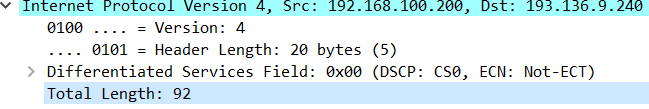
1. **Qual é o valor do campo protocolo? O que identifica?**

C:\Users\win8\AppData\Local\Microsoft\Windows\INetCacheContent.Word\2-b-PROTOCOL.PNG

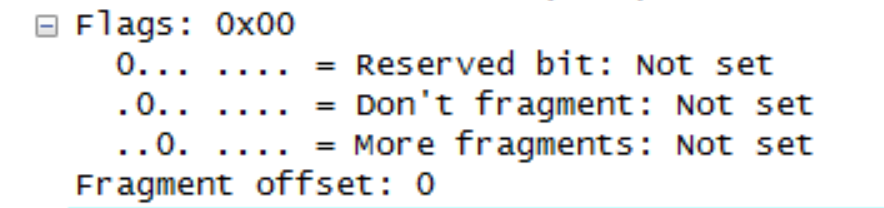
O valor do campo protocolo é ICMP (1). Identifica o *Internet Control Message Protocol*.

1. **Quantos *bytes* tem o cabeçalho IP(v4)? Quantos bytes tem o campo de dados (*payload*) do datagrama? Como se calcula o tamanho do *payload*?**

O cabeçalho IPv4 tem 20 bytes. O campo de dados (*payload*) do datagrama tem 72 bytes. Como o tamanho do datagrama a nível do IPv4 é igual a 92 bytes e sabemos que o tamanho do cabeçalho é 20 bytes, o *payload* é dado pela subtração destes dois valores.

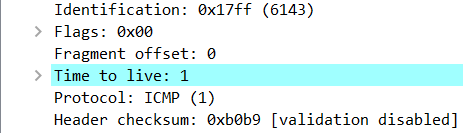


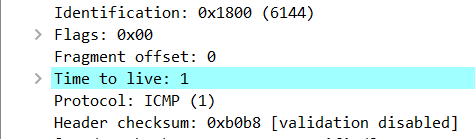
1. **O datagrama IP foi fragmentado? Justifique.**

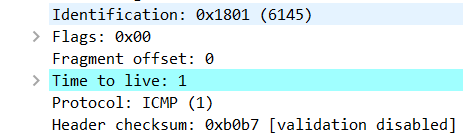


O datagrama não foi fragmentado visto que o campo *More fragments* está a 0 e o *Fragment offset* também está a 0. Ou seja, como o payload nesta trama começa na posição 0 (do datagrama original) – segundo o *Fragment* *offset* – e como não se esperam mais fragmentos desse datagrama original – segundo o *More fragments* – pode-se concluir que esta trama transporta o datagrama original na totalidade.

1. **Ordene os pacotes capturados de acordo com o endereço IP fonte (e.g., selecionando o cabeçalho da coluna *Source*), e analise a sequência de tráfego ICMP gerado a partir do endereço IP atribuído à sua máquina. Para a sequência de mensagens ICMP enviadas pelo seu computador, indique que campos do cabeçalho IP variam de pacote para pacote.**

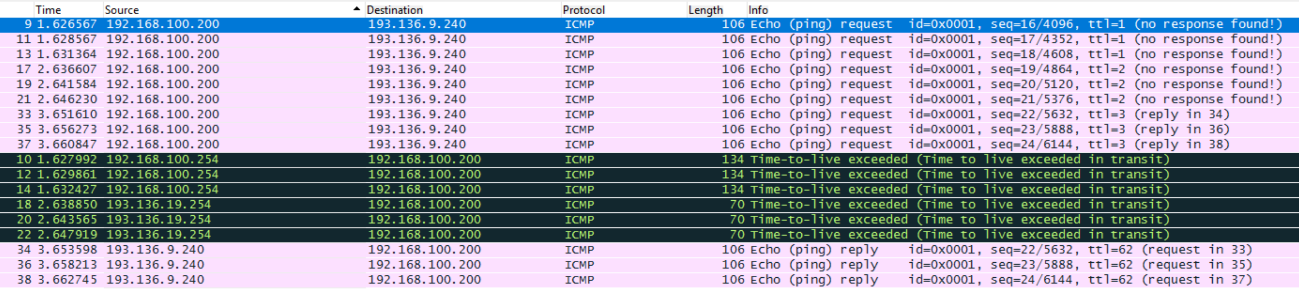






Segundo o *Wireshark*, os campos que variam são: a identificação (*Identification*), o TTL e o *header checksum*.

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



São enviadas três tramas com TTL igual a 1, três com TTL igual a 2 e outras três com TTL igual a 3.

Só as últimas três tramas chegam ao destino (33, 35 e 37). Todas as outras deram origem a respostas ICMP do tipo “*Time to live exceeded”* (saltos insuficientes para atingir o destino).

A campo da identificação (*Identification*) é sempre incrementado em uma unidade para as tramas enviadas para o mesmo destino.

1. **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?**

Para os *Echo Requests* relativos às tramas com TTL igual a 1, o valor do campo TTL das mensagens de resposta ICMP TTL *exceeded* é igual a 64.

Para os *Echo Requests* relativos às tramas com TTL igual a 2, o valor do campo TTL das mensagens de resposta ICMP TTL *exceeded* é igual a 254.

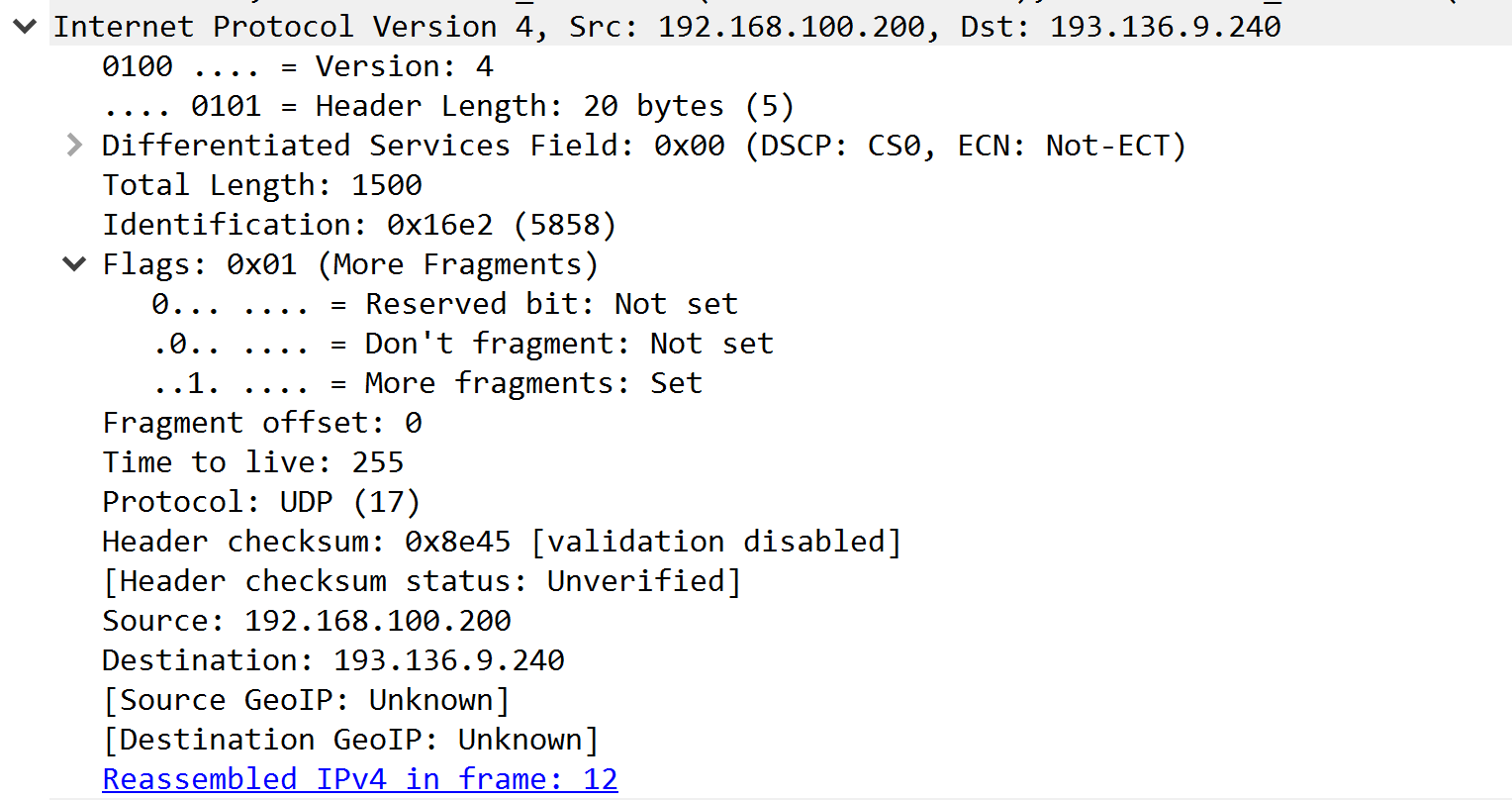
O valor do campo TTL não permanece constante para todas as mensagens devido ao facto de que nenhum *router*, ao receber uma trama com TTL igual a 0, sabe quantos saltos foram dados para trás e, por isso, envia um valor por defeito (que pode ser diferente para cada *router*) nas respostas ICMP TTL *exceeded*. Caso o valor do TTL não seja suficiente para alcançar o destino (i.e. a origem do datagrama), é enviada uma nova resposta ICMP com um TTL maior, repetindo-se este processo até ser obtida uma confirmação de receção.

1. **Pretende-se agora analisar a fragmentação de pacotes IP. Reponha a ordem do tráfego capturado usando a coluna do tempo de captura. Observe o tráfego depois do tamanho de pacote ter sido definido para 40XX bytes.**
2. **Localize a primeira mensagem ICMP. Porque é que houve necessidade de fragmentar o pacote inicial?**



Houve necessidade de fragmentar o pacote inicial porque o protocolo IP só suporta pacotes até 1500 bytes. Como o tamanho foi definido para 4021 bytes, houve necessidade de fragmentar o pacote inicial em três mais pequenos.

1. **Imprima o primeiro fragmento do datagrama IP segmentado. Que informação no cabeçalho indica que o datagrama foi fragmentado? Que informação no cabeçalho IP indica que se trata do primeiro fragmento? Qual é o tamanho deste datagrama IP?**

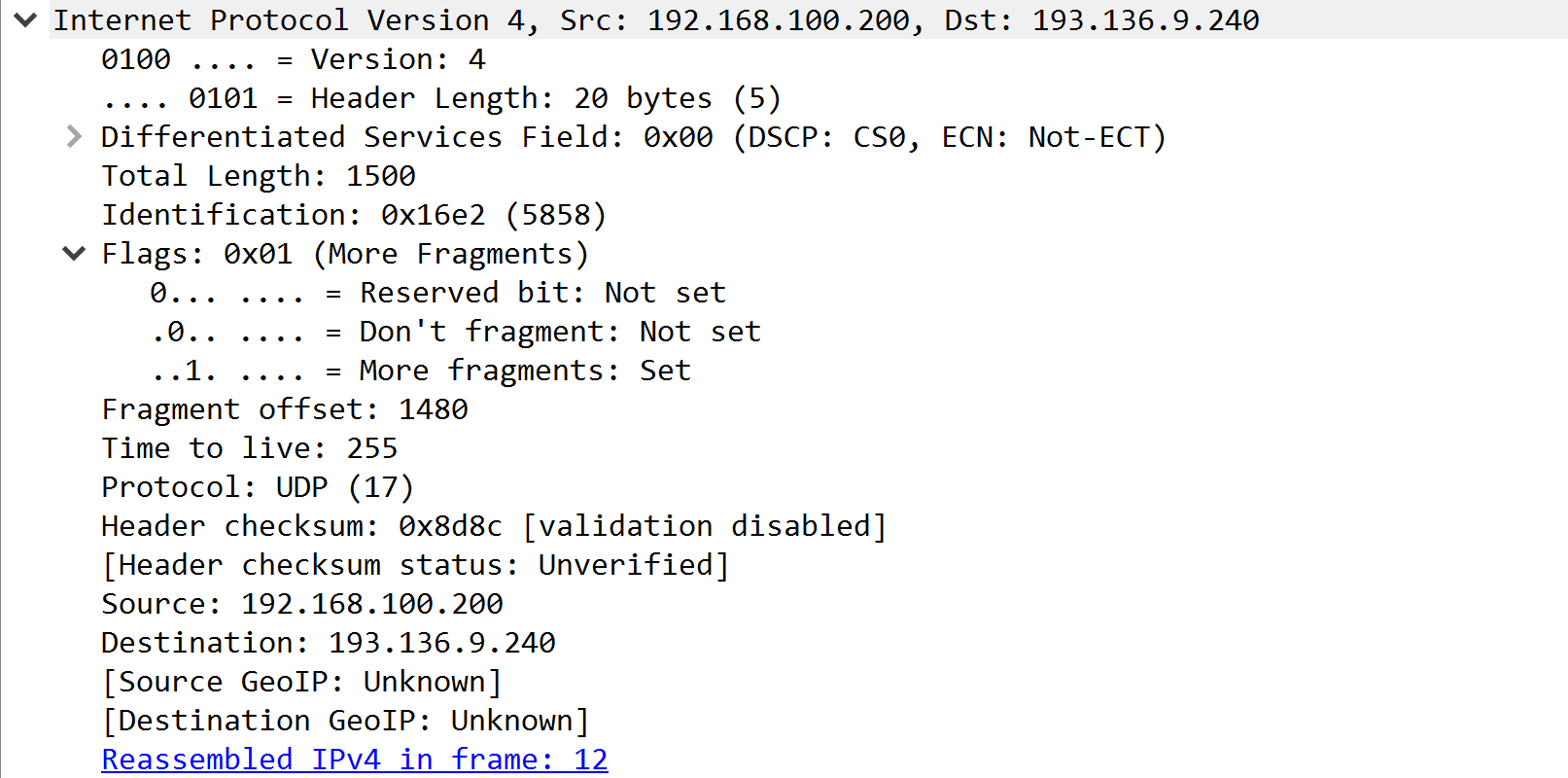


A informação do cabeçalho que indica que o datagrama foi fragmentada é a *flag* *More fragments*. Verifica-se que se trata do primeiro fragmento devido ao campo do *fragment offset* estar a zero.



O tamanho do datagrama IP é de 1514 bytes.

1. **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?**

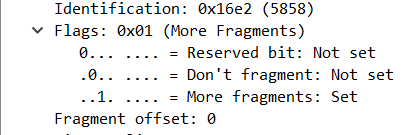


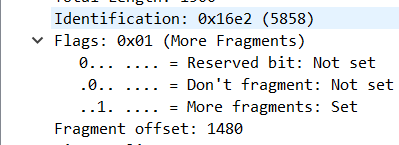
Podemos verificar que não se trata do primeiro fragmento devido ao campo do *fragment offset* que está a 1480. Sim, ainda existem mais fragmentos. Podemos confirmar tal informação a partir do valor da *flag* relativa à existência de mais fragmentos (“*More fragments”*) que está a 1.

É importante notar que podemos verificar que se tratam de fragmentos correspondentes ao mesmo datagrama original a partir do campo de identificação (*Identification*).

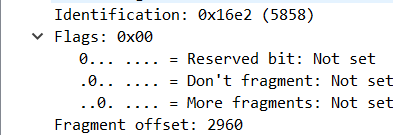
1. **Quantos fragmentos foram criados a partir do datagrama original? Como se deteta o último fragmento correspondente ao datagrama original?**

Foram criados três fragmentos a partir do datagrama original. Em primeiro lugar, deteta-se um fragmento correspondente ao datagrama original através do valor hexa do identificador (*Identification* – que se mantém constante para todos os fragmentos do datagrama original).





Assim, para detetar o último fragmento correspondente ao datagrama original, primeiro confirma-se o campo *Identification* e, de seguida, verifica-se que se trata mesmo do último fragmento através da *flag* relativa à existência de mais fragmentos (*More fragments*) que se deve encontrar a 0.



1. **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.**

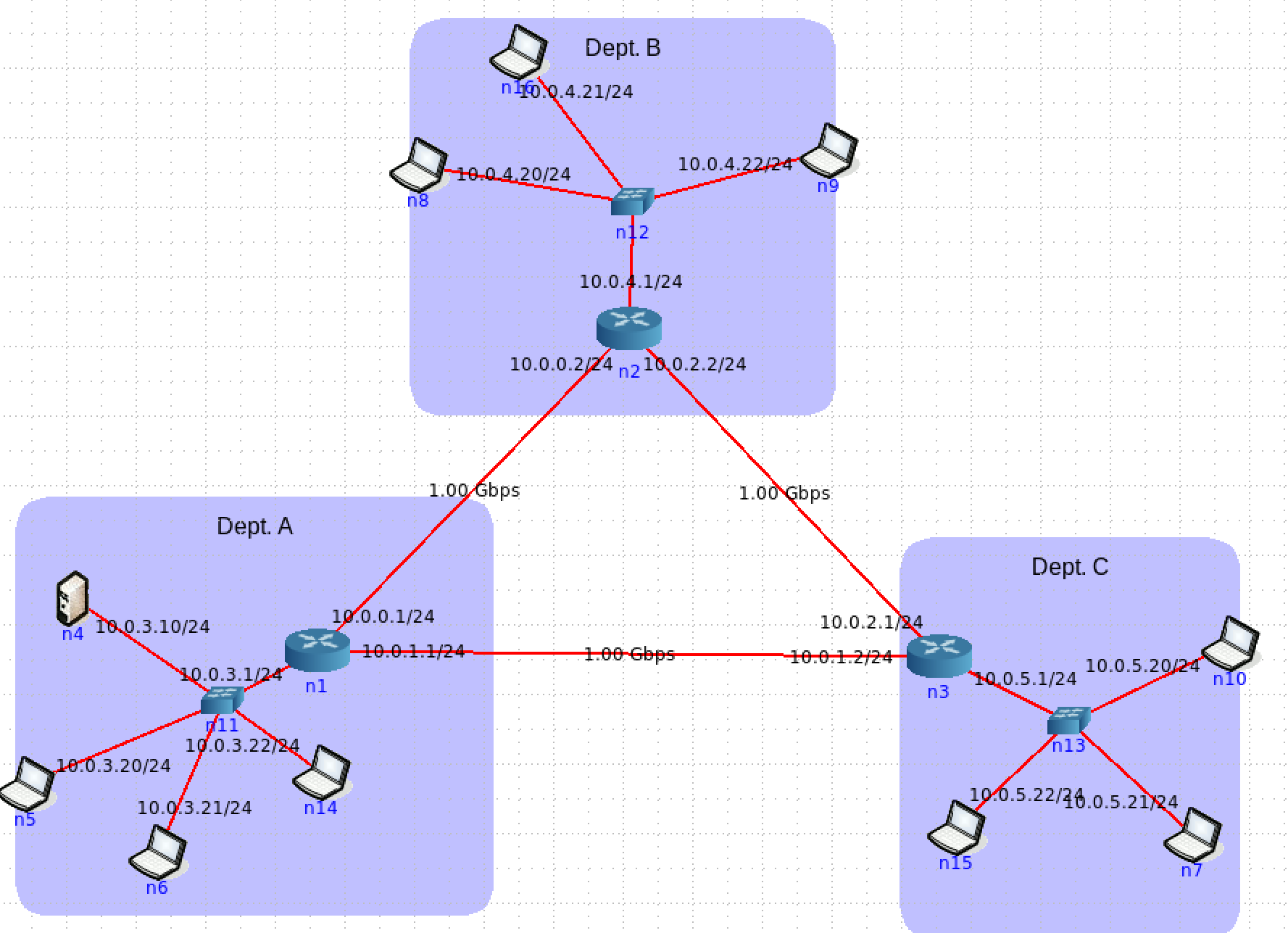
Como se pode verificar nas imagens anteriores (que são relativas ao mesmo datagrama original), os únicos campos que mudam são: *More fragments* e *Fragment offset*. Sabemos que fazem todos parte do mesmo datagrama original porque o campo *Identification* é igual para os três.

Através do *Fragment offset* sabemos em que posição do datagrama original a informação desta trama começa. Com a *flag* *More fragments* conseguimos saber se existem mais fragmentos desse datagrama para chegar. Saberemos que estamos no último fragmento quando a *Identification* for igual às restantes (imagem anterior) e *More fragments* for igual a 0. Com estas informações conseguimos reconstruir o datagrama original a partir dos vários fragmentos.

**Parte II**

**2 – Endereçamento e Encaminhamento IP**

1. **Atenda aos endereços IP atribuídos automaticamente pelo CORE aos diversos equipamentos da topologia.**
2. **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.**

****

1. **Tratam‐se de endereços públicos ou privados? Porquê?**

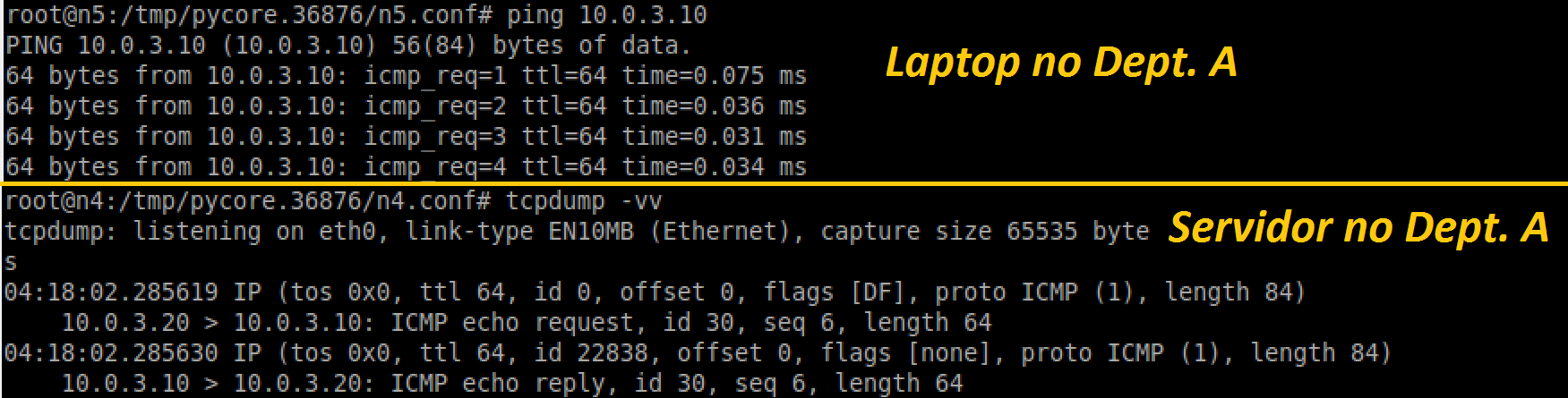
Tratam-se de endereços privados pois correspondem a endereços IP da classe A (entre 10.0.0.0 e 10.255.255.255). São também designados de *unique local access* (ULA), sendo endereços reservados à rede local dos departamentos do MIEInet, não sendo possível aceder-lhes diretamente através da rede internet global.

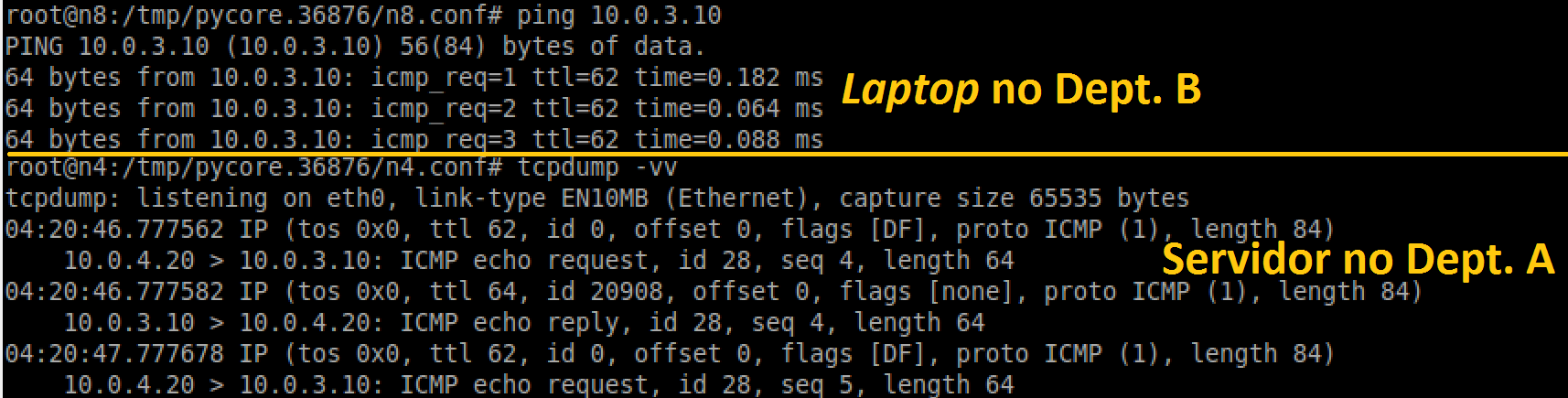
1. **Porque razão não é atribuído um endereço IP aos *switches*?**

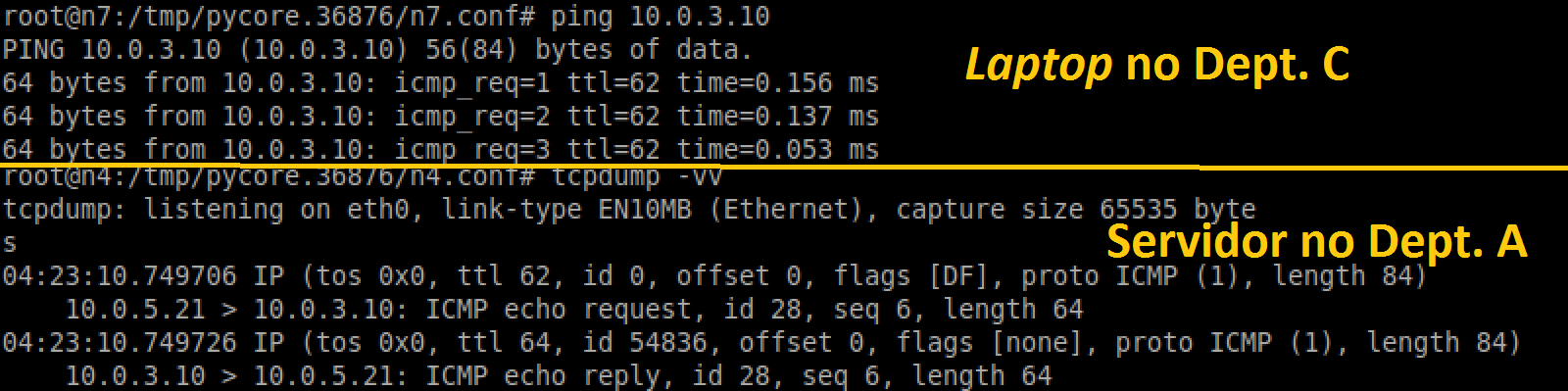
Não é atribuído um endereço IP aos *switches* pois estes são apenas dispositivos de ligação da rede (e não um *host*), que atua ao nível da Ethernet, enviando tramas deste tipo e encaminhando o tráfego da rede.

1. **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).**

Para testar a conectividade, enviámos múltiplos *pings* de um portátil de cada departamento (n5 em A, n8 em B e n7 em C) para o servidor presente no departamento A (n4).





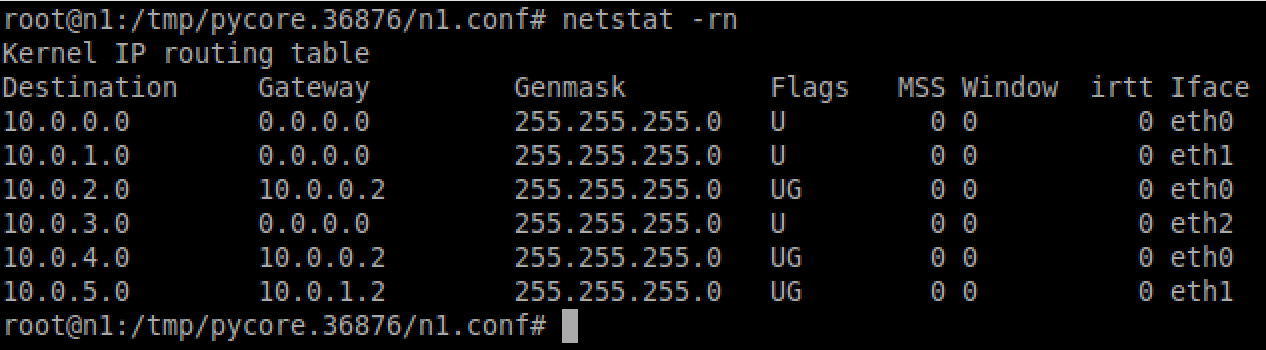


1. **Para o router e um laptop do departamento A:**
2. **Execute o comando *netstat -rn* por forma a poder consultar a tabela de encaminhamento *unicast* (IPv4). Inclua no seu relatório as tabelas de encaminhamento obtidas; interprete as várias entradas de cada tabela. Se necessário, consulte o manual respetivo (*man* *netstat*).**

Para o router n1**:**

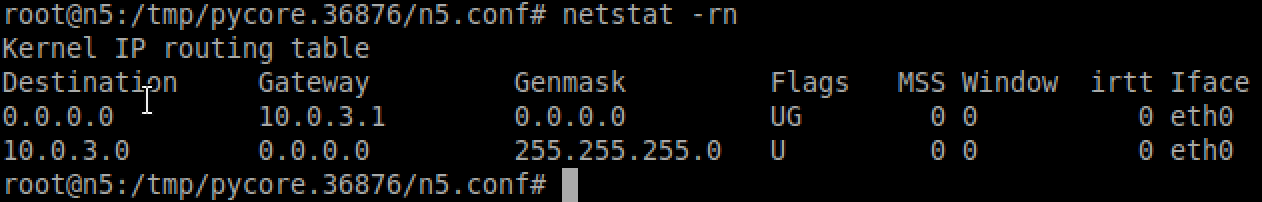
Cada uma das entradas da tabela diz-nos que para enviar um datagrama destinado a *Destination*, este tem de ser entregue a *Gateway* e sair pela interface *Iface*. Mais concretamente:

* A primeira linha diz-nos que para enviar um datagrama com destino 10.0.0.0, este tem de ser entregue a 0.0.0.0 e sair pela interface eth0.
* A segunda linha diz-nos que para enviar um datagrama com destino 10.0.1.0, este tem de ser entregue a 0.0.0.0 e sair pela interface eth1.
* A terceira linha diz-nos que para enviar um datagrama com destino 10.0.2.0, este tem de ser entregue a 10.0.0.2 e sair pela interface eth0.
* A quarta linha diz-nos que para enviar um datagrama com destino 10.0.3.0, este tem de ser entregue a 0.0.0.0 e sair pela interface eth2.
* A quinta linha diz-nos que para enviar um datagrama com destino 10.0.4.0, este tem de ser entregue a 0.0.0.0 e sair pela interface eth3.
* A sexta linha diz-nos que para enviar um datagrama com destino 10.0.5.0, este tem de ser entregue a 10.0.1.2 e sair pela interface eth1.
* A sétima linha diz-nos que para enviar um datagrama com destino 10.0.6.0, este tem de ser entregue a 10.0.0.2 e sair pela interface eth0.



Para o laptop n5:

* A primeira linha diz-nos que para enviar um datagrama com destino 0.0.0.0 (*default*), este tem de ser entregue a 10.0.4.1 e sair pela interface eth0.
* A segunda linha diz-nos que para enviar um datagrama com destino 10.0.4.0, este tem de ser entregue a 0.0.0.0 e sair pela interface eth0.



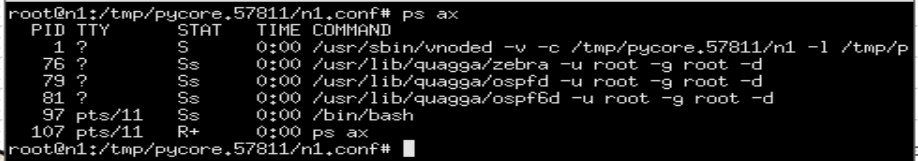
Quando o *gateway* é 0.0.0.0, isto indica-nos que o destino já está ligado, diretamente, ao própriodispositivo de origem (*router/laptop*). No contexto do problema, isto acontece (i.e. *gateway* a 0.0.0.0) quando o *router* n1 comunica com n2, n3, n4 e, através do *switch* n11, com n5, n6 e n14 ou quando o laptop n5 comunica com n1, n6 e n14.

Quando a *destination* é 0.0.0.0, estamos perante um caminho por defeito (ou *default*) – como, por exemplo, na segunda entrada da tabela de encaminhamento de n5 – que nos indica, no caso de n5, que quando for necessário enviar um datagrama cujo IP de destino não esteja na subrede 10.0.4.0 (a única outra entrada da tabela), deverá ser enviado, por defeito, através do dispositivo correspondente ao *gateway* 10.0.4.1 (*router* n1), que se encarregará de encaminhar o datagrama ao destino.

Na terceira coluna das tabelas de encaminhamento podemos ainda verificar a presença das *flags* U e G que fazem referência, respetivamente, à validade de um caminho (a *source* e a *destination* estão, de facto, conectadas – diretamente ou não) *e* ao facto do encaminhamento passar por um *gateway* intermédio. Quando a flag G não aparece nalguma entrada, podemos dizer que o dispositivo de destino se encontra ligado diretamente ao de origem.

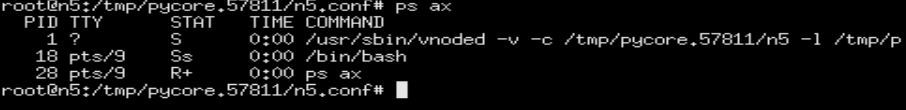
1. **Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico (sugestão: analise que processos estão a correr em cada sistema).**

Na lista de processos no *router* n1:



verificamos que o *daemon* *quagga* está a correr. Isto significa que temos encaminhamento dinâmico.

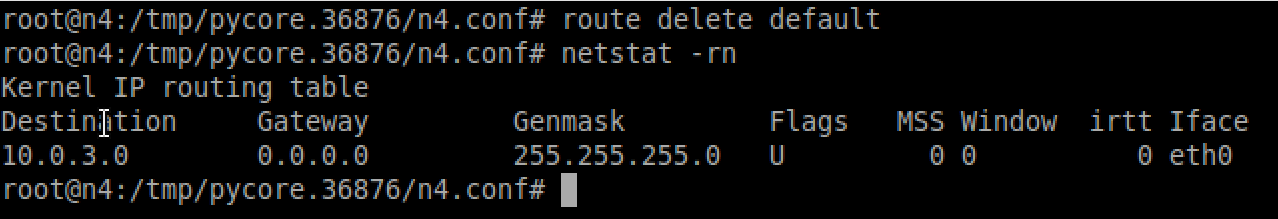
Contrariamente, no *laptop* n5, verifica-se que este *daemon* não está a correr:



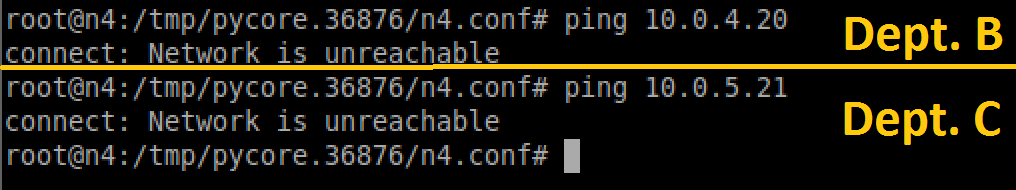
Ou seja, temos no *laptop* encaminhamento estático.

Podemos então concluir que ocorrem processos de *routing* dinâmico no *core* da rede (i.e. nos três *routers* do sistema) e não nas suas subredes dos departamentos.

1. **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.**



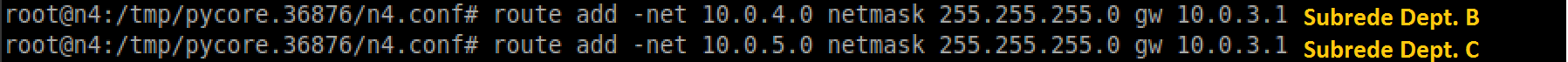
Ao utilizar o comando *route* *delete* *default* no servidor n4, a *route* por *default* (0.0.0.0) foi eliminada como podemos ver no *netstat* *-rn* anteriormente executado. As implicações que esta medida irá ter para os utilizadores é a impossibilidade de o servidor conseguir encaminhar datagramas cujo IP destino não seja pertencente à subrede 10.0.3.0. Como podemos ver nas imagens, é impossível fazer *ping* desde o servidor até máquinas nas subredes dos departamentos B e C.



1. **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.**

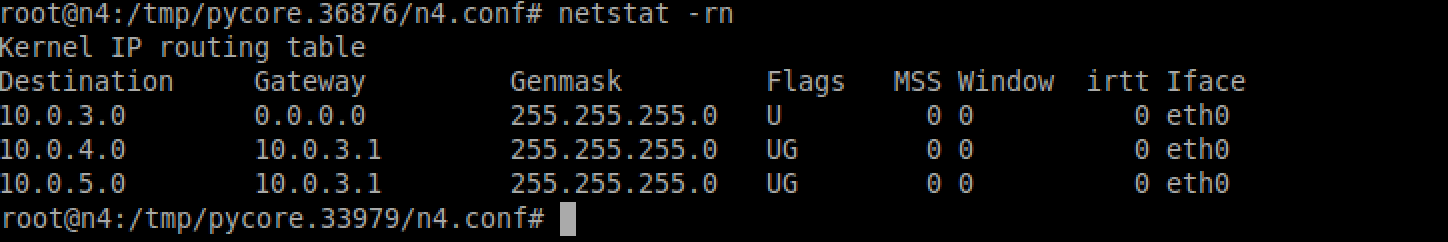
Tentámos executar os comandos:

* route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.3.1 (Subrede Departamento B)
* route add -net 10.0.6.0 netmask 255.255.255.0 gw 10.0.3.1 (Subrede Departamento C)

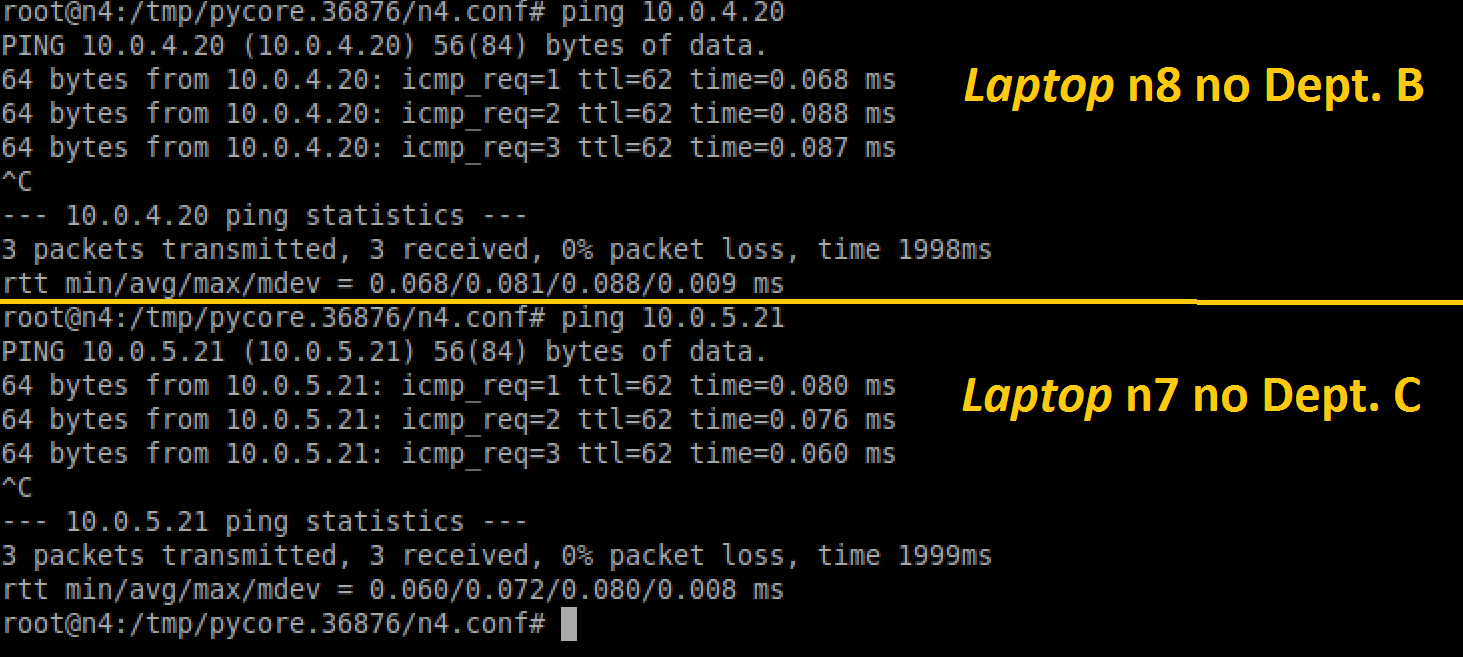


1. **Teste a nova política de encaminhamento garantindo que o servidor está novamente acessível, utilizando para o efeito o comando *ping*. Registe a nova tabela de encaminhamento do servidor.**

Como se pode verificar na imagem seguinte, as Subredes B e C já se encontra na tabela de encaminhamento do servidor.



Verifica-se que há acessibilidade do servidor a essas Subredes visto ser possível efetuar um *ping* a cada uma delas.



**3 – Definição de Subredes**

**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 192.168.128.0/24, 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.**

Temos 32 bits ao todo para o endereço IP. Destes temos:

Como temos 3 departamentos, serão no mínimo necessárias 3 Subredes. O número de bits mínimo que poderemos ter para identificar cada Subrede é 3, o que dá um máximo de 6 Subredes:

Se temos 3 bits para as Subredes, teremos então:

O que dá:

Sabendo que dos 8 bits que temos para gerir, teremos o endereço para cada interface que será igual aos 3 bits de identificação de Subrede mais os 5 bits disponíveis para atribuir por interface (exceto o 00000 e 11111, pois são reservados):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Bits p/ Subrede | Bits p/ Interface | | Endereço p/ Interface (dec.) | |
| Mín. | Máx. | Mín. | Máx. |
| **Reservado** | .000 | - | - | - | - |
| Subrede 1 | .001 | 00001 | 11110 | 33 | 62 |
| Subrede 2 | .010 | 00001 | 11110 | 65 | 94 |
| Subrede 3 | .011 | 00001 | 11110 | 97 | 126 |
| Subrede 4 | .100 | 00001 | 11110 | 129 | 158 |
| Subrede 5 | .101 | 00001 | 11110 | 161 | 190 |
| Subrede 6 | .110 | 00001 | 11110 | 193 | 222 |
| **Reservado** | .111 | - | - | - | - |

**Tabela 1 –** Distribuição de endereços

Na **Tabela 1** temos a atribuição de endereços para 6 Subredes. Vamos considerar que as três primeiras Subredes nessa tabela são, respetivamente, para os Departamentos A, B e C, o que resulta em:

Dept. A -> (bits p/ identificar a Subrede) 00100000 -> 192.168.128.32/27

Dept. B -> (bits p/ identificar a Subrede) 01000000 -> 192.168.128.64/27

Dept. C -> (bits p/ identificar a Subrede) 01100000 -> 192.168.128.96/27

O número 27 serve para indicar que dos 32 bits do IP, os primeiros 27 identificam a rede.

Como sabemos o endereço máximo e mínimo para cada Subrede (**Tabela 1**), podemos atribuir em cada departamento os endereços IP para as diversas interfaces (desde que cumpram os limites):

|  |  |  |
| --- | --- | --- |
| **Dispositivo** | **IP** | **Máscara de Rede** |
| n1 (router) | 192.168.128.33/27 | 255.255.255.224 |
| n4 (*host*) | 192.168.128.34/27 | 255.255.255.224 |
| n5 (*laptop*) | 192.168.128.35/27 | 255.255.255.224 |
| n6 (*laptop*) | 192.168.128.61/27 | 255.255.255.224 |
| n14 (*laptop*) | 192.168.128.62/27 | 255.255.255.224 |

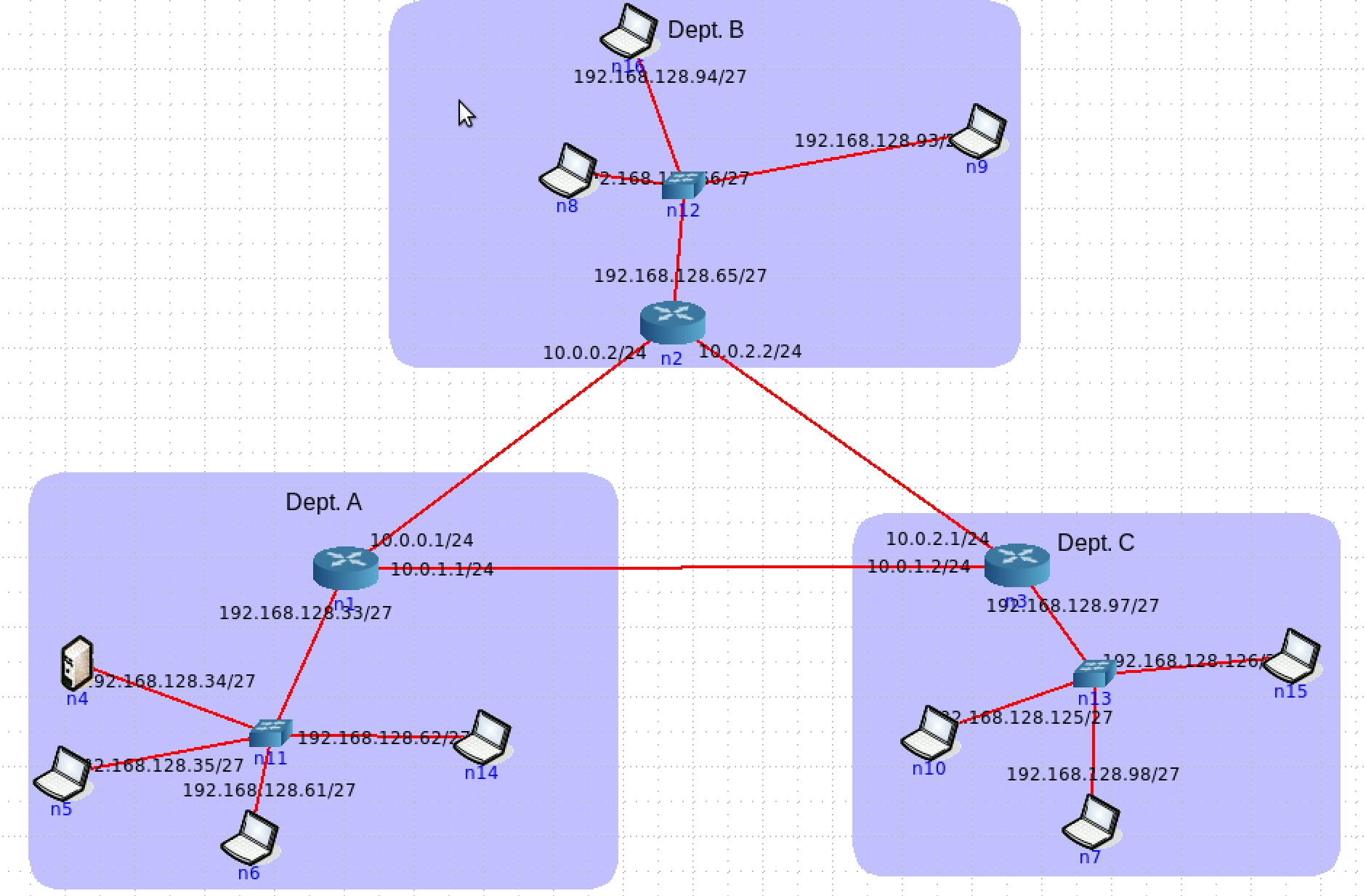
**Tabela 2** – Endereços IP da Subrede no Departamento A

|  |  |  |
| --- | --- | --- |
| **Dispositivo** | **IP** | **Máscara de Rede** |
| n2 (router) | 192.168.128.65/27 | 255.255.255.224 |
| n8 (*laptop*) | 192.168.128.66/27 | 255.255.255.224 |
| n9 (*laptop*) | 192.168.128.93/27 | 255.255.255.224 |
| n16 (*laptop*) | 192.168.128.94/27 | 255.255.255.224 |

**Tabela 3** – Endereços IP da Subrede no Departamento B

|  |  |  |
| --- | --- | --- |
| **Dispositivo** | **IP** | **Máscara de Rede** |
| n3 (router) | 192.168.128.97/27 | 255.255.255.224 |
| n7 (*laptop*) | 192.168.128.98/27 | 255.255.255.224 |
| n10 (*laptop*) | 192.168.128.125/27 | 255.255.255.224 |
| n15 (*laptop*) | 192.168.128.126/27 | 255.255.255.224 |

**Tabela 4** – Endereços IP da Subrede no Departamento C

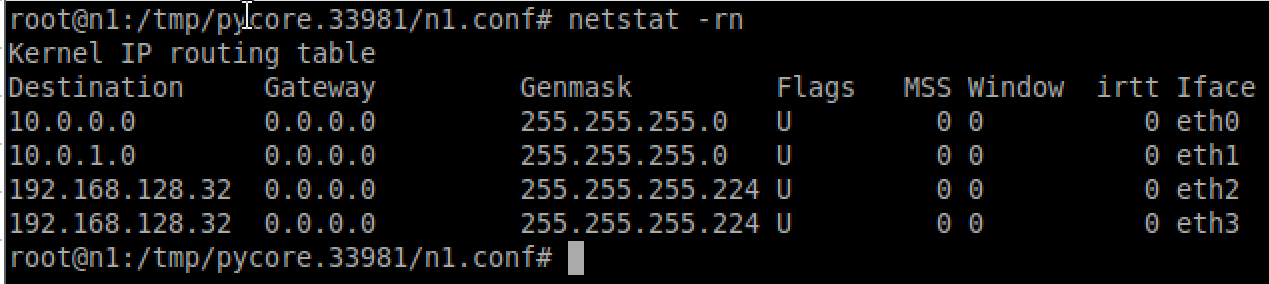


1. **Qual a máscara de rede que usou (em formato decimal)? Justifique.**

Como se trata de um endereço Classe C, sabemos que a máscara será 255.255.255.0. Isto indica que os 3 primeiros octetos são para identificar a rede. Como foi calculado antes, do último octeto apenas os 3 primeiros bits serão utilizados para identificar a Subrede. Ou seja, teremos em binário algo género:

Posto isto, sabemos que a máscara de rede utilizada é 255.255.255.224.

Podemos verificar isto numa tabela de endereçamento (nesta caso da do *router* no Departamento A):



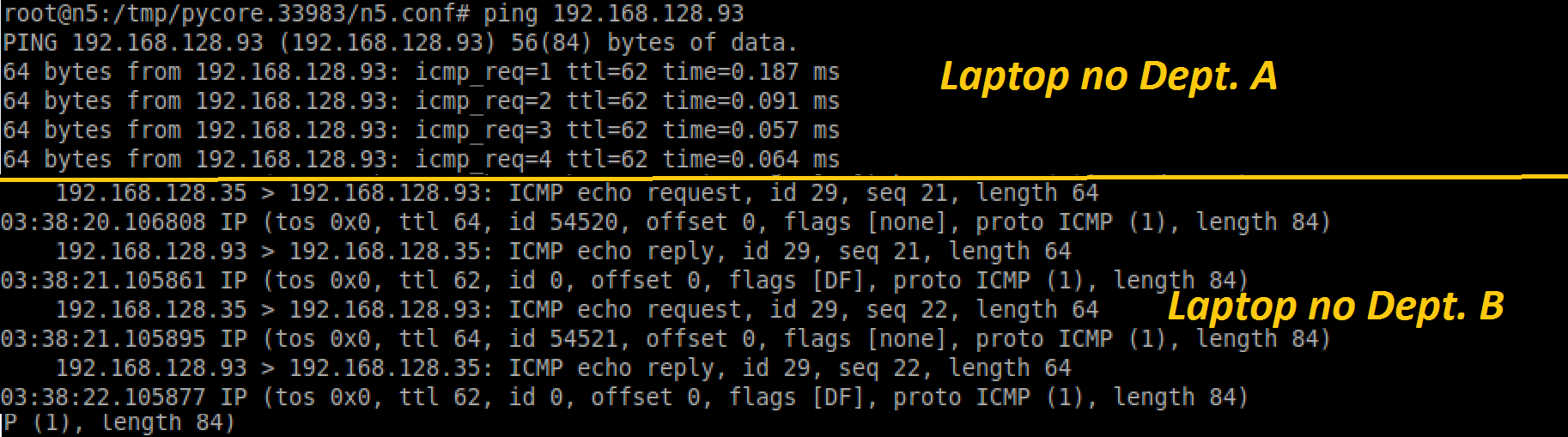
1. **Quantos *hosts* IP pode interligar em cada departamento? Justifique.**

Temos 8 bits para gerir endereços. Destes, os primeiros 3 bits identificam a Subrede, ficando 5 bits para atribuir às interfaces. Contudo, temos 2 endereços reservados – o 00000 e o 11111. Temos assim:

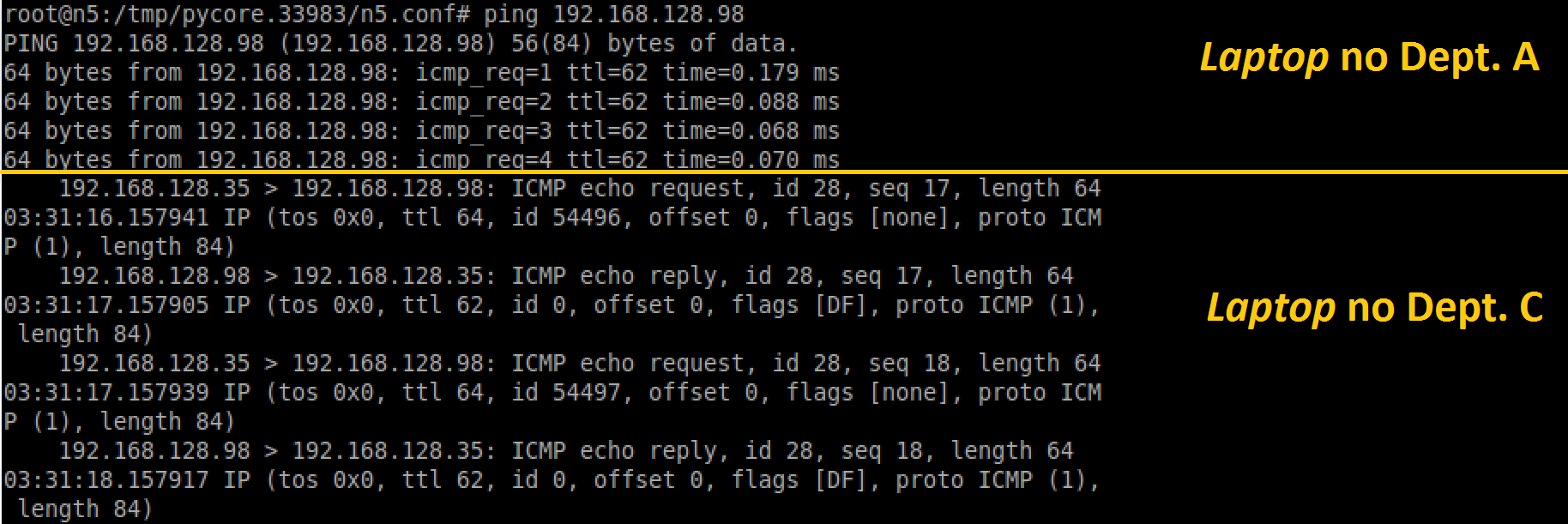
1. **Garanta que conectividade IP entre as várias redes locais da empresa MIEInet é mantida.**

Podemos verificar a conectividade entre as várias redes locais fazendo *ping* entre máquinas de diferentes departamentos:

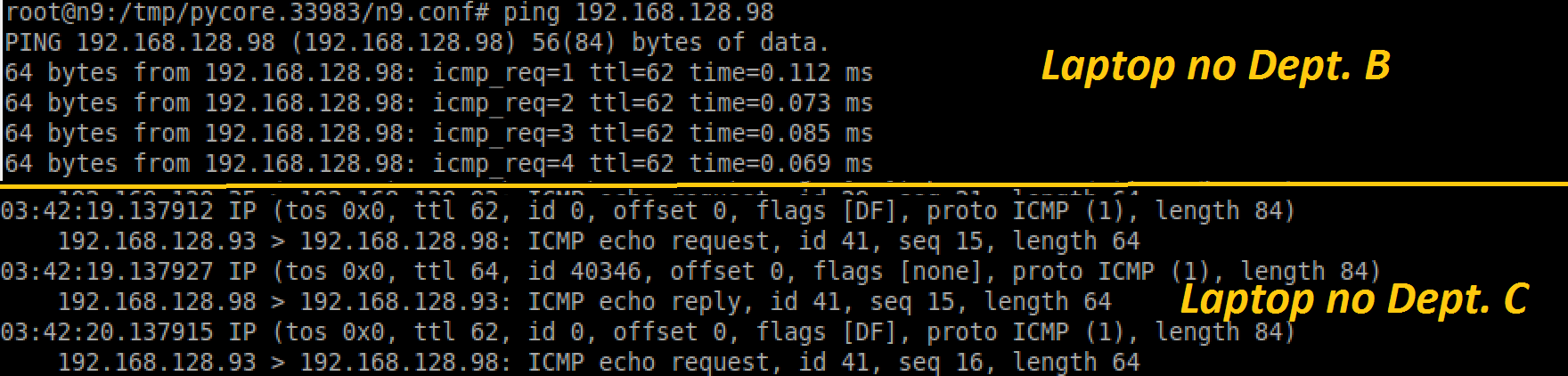
* Ping de uma máquina em A (n5) para B (n9)



* Ping de uma máquina em A (n5) para C (n7)



* Ping de uma máquina em B (n9) para C (n7)



Podemos assim efetivamente verificar que as máquinas nos diferentes departamentos conseguem comunicar entre si.

**Conclusões**

Este quarto trabalho prático permitiu-nos pôr em prática os conhecimentos teóricos adquiridos nas aulas de Redes de Computadores e, assim, compreender melhor os mesmos de um ponto de vista mais real. Desta vez, debruçámo-nos sobre o protocolo IPv4.

Na primeira parte, abordámos o papel do campo TTL (*Time To Live)* dos datagramas IP na comunicação entre dispositivos e as respetivas respostas ICMP, bem como a ocorrência de fragmentação dos datagramas que ocorre quando são enviados datagramas com tamanho superior ao suportado pelo protocolo IPv4.

Na segunda parte deste trabalho foi abordado, em primeiro lugar, o endereçamento e encaminhamento que ocorrem ainda ao nível do IP. Compreendemos a função das tabelas de encaminhamento, os encaminhamentos estático e dinâmico e a remoção e adição de rotas numa rede. Em segundo lugar, abordámos a definição de subredes (*subnetting*) e esquemas de endereçamento, bem como a utilização de máscaras de rede.