



NOVEMBRO 2020

**Universidade do Minho**  
Escola de Engenharia

# TP2

## REDES DE COMPUTADORES

### **Grupo 54**

Adriano Maior, a89483

Joel Martins, a89575

Manuel Moreira, a89471

## Índice

### TP2: Protocolo IP (Parte I)

1.	5
a.	Ative o wireshark ou o tcpdump no Cliente1. Numa shell do Cliente1, execute o comando traceroute -I para o endereço IP do Servidor1. ....5
b.	Registre e analise o tráfego ICMP enviado pelo Cliente1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.....5
c.	Qual deve ser o valor inicial mínimo do campo TTL para alcançar o Servidor1? Verifique na prática que a sua resposta está correta.....6
d.	Calcule o valor médio do tempo de ida-e-volta (Round-Trip Time) obtido?.....6
2.	6
a.	Qual é o endereço IP da interface ativa do seu computador? .....6
b.	Qual é o valor do campo protocolo? O que identifica? .....6
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? .....7
d.	O datagrama IP foi fragmentado? Justifique. ....7
e.	Ordene os pacotes capturados de acordo com o endereço IP fonte (e.g., selecionando o cabeçalho da coluna Source), e analise a sequência de tráfego ICMP gerado a partir do endereço IP atribuído à interface da sua máquina. Para a sequência de mensagens ICMP enviadas pelo seu computador, indique que campos do cabeçalho IP variam de pacote para pacote. ....8
f.	Observa algum padrão nos valores do campo de Identificação do datagrama IP e TTL? .8
g.	Ordene o tráfego capturado por endereço destino e encontre a série de respostas ICMP TTL exceeded enviadas ao seu computador. Qual é o valor do campo TTL? Esse valor permanece constante para todas as mensagens de resposta ICMP TTL exceeded enviados ao seu host? Porquê? .....9
3.	10
a.	Localize a primeira mensagem ICMP. Porque é que houve necessidade de fragmentar o pacote inicial? .....10
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? .....10
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? .....11
d.	Quantos fragmentos foram criados a partir do datagrama original? Como se detecta o último fragmento correspondente ao datagrama original? .....11
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. ....12

## TP2: Protocolo IP (Parte II)

1.	13
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. ....13
b.	Tratam-se de endereços públicos ou privados? Porquê? .....13
c.	Porque razão não é atribuído um endereço IP aos switches? .....13
d.	Usando o comando ping certifique-se que existe conectividade IP entre os laptops dos vários departamentos e o servidor do departamento A (basta certificar-se da conectividade de um laptop por departamento). ....14
e.	Verifique se existe conectividade IP do router de acesso RISP para o servidor S1. ....15
2.	15
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). ....15
b.	Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico (sugestão: analise que processos estão a correr em cada sistema, por exemplo, ps -ax). ....17
	É utilizado um encaminhamento dinâmico, pois as rotas são definidas de forma automática através da troca de informação entre os routers. ....17
c.	Admita que, por questões administrativas, a rota por defeito (0.0.0.0 ou default) deve ser retirada definitivamente da tabela de encaminhamento do servidor S1 localizado no departamento A. Use o comando route delete para o efeito. Que implicações tem esta medida para os utilizadores da organização MIEI-RC que acedem ao servidor. Justifique. ....17
d.	Adicione as rotas estáticas necessárias para restaurar a conectividade para o servidor S1, por forma a contornar a restrição imposta na alínea c). Utilize para o efeito o comando route add e registe os comandos que usou. ....18
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. ....18
3.	19
a.	Considere que dispõe apenas do endereço de rede IP 130.XX.96.0/19, em que XX é o decimal correspondendo ao seu número de grupo (PLXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo a rede de acesso e core inalteradas) e atribua endereços às interfaces dos vários sistemas envolvidos. Assuma que todos os endereços de sub-redes são usáveis. Deve justificar as opções usadas. ....19
b.	Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP pode interligar em cada departamento? Justifique. ....20
c.	Garanta e verifique que conectividade IP entre as várias redes locais da organização MIEI-RC é mantida. Explique como procedeu. ....21



# TP2: Protocolo IP (Parte I)

1. Prepare uma topologia CORE para verificar o comportamento do traceroute. Ligue um host (pc) Cliente1 a um router R2; o router R2 a um router R3, que por sua vez, se liga a um host (servidor) Servidor1. (Note que pode não existir conectividade IP imediata entre o Cliente1 e o Servidor1 até que o anúncio de rotas estabilize). Ajuste o nome dos equipamentos atribuídos por defeito para a topologia do enunciado.

- a. Ative o wireshark ou o tcpdump no Cliente1. Numa shell do Cliente1, execute o comando `traceroute -I` para o endereço IP do Servidor1.

```
vcmd
root@Cliente1:/tmp/pycore.38895/Cliente1.conf# traceroute -I 10.0.3.10
traceroute to 10.0.3.10 (10.0.3.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1)  0.032 ms  0.007 ms  0.006 ms
 2 10.0.2.2 (10.0.2.2)  0.049 ms  0.015 ms  0.009 ms
 3 10.0.3.10 (10.0.3.10) 0.018 ms  0.012 ms  0.012 ms
root@Cliente1:/tmp/pycore.38895/Cliente1.conf#
```

Execução do comando traceroute

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

2	0.108803999	Te80::200:TT:Teaa:5	TT02::5	OSPF	90 Hello Packet
3	6.516035299	10.0.0.20	10.0.3.10	ICMP	74 Echo (ping) request id=0x002d, seq=1/256, ttl=1 (no response found!)
4	6.516057339	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
5	6.516064974	10.0.0.20	10.0.3.10	ICMP	74 Echo (ping) request id=0x002d, seq=2/512, ttl=1 (no response found!)
6	6.516068670	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
7	6.516071877	10.0.0.20	10.0.3.10	ICMP	74 Echo (ping) request id=0x002d, seq=3/768, ttl=1 (no response found!)
8	6.516074672	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
9	6.516078325	10.0.0.20	10.0.3.10	ICMP	74 Echo (ping) request id=0x002d, seq=4/1024, ttl=2 (no response found!)
10	6.516089876	10.0.2.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
11	6.516093038	10.0.0.20	10.0.3.10	ICMP	74 Echo (ping) request id=0x002d, seq=5/1280, ttl=2 (no response found!)
12	6.516097944	10.0.2.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
13	6.516100958	10.0.0.20	10.0.3.10	ICMP	74 Echo (ping) request id=0x002d, seq=6/1536, ttl=2 (no response found!)
14	6.516105578	10.0.2.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
15	6.516109250	10.0.0.20	10.0.3.10	ICMP	74 Echo (ping) request id=0x002d, seq=7/1792, ttl=3 (reply in 16)
16	6.516121568	10.0.3.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x002d, seq=7/1792, ttl=62 (request in 15)
17	6.516125518	10.0.0.20	10.0.3.10	ICMP	74 Echo (ping) request id=0x002d, seq=8/2048, ttl=3 (reply in 18)
18	6.516132194	10.0.3.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x002d, seq=8/2048, ttl=62 (request in 17)
19	6.516135324	10.0.0.20	10.0.3.10	ICMP	74 Echo (ping) request id=0x002d, seq=9/2304, ttl=3 (reply in 20)
20	6.516141783	10.0.3.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x002d, seq=9/2304, ttl=62 (request in 19)
21	6.516145566	10.0.0.20	10.0.3.10	ICMP	74 Echo (ping) request id=0x002d, seq=10/2560, ttl=4 (reply in 22)
22	6.516152805	10.0.3.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x002d, seq=10/2560, ttl=62 (request in 21)

Tráfego ICMP

O comportamento que aconteceu foi o esperado, uma vez que foram enviados pacotes com TTL inicial igual a 1 e este foi incrementando unitariamente. Sendo que quando o TTL não é suficiente para chegar ao destino, o router que recebe o pacote com o TTL a 1 descarta-o e envia uma resposta ao host que enviou o Echo Request. Quando

o pacote consegue chegar ao destino, então é enviado um Echo Reply para o nodo de origem.

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

TTL = 3, o que pode ser verificado na imagem da alínea anterior.

- d. Calcule o valor médio do tempo de ida-e-volta (Round-Trip Time) obtido?

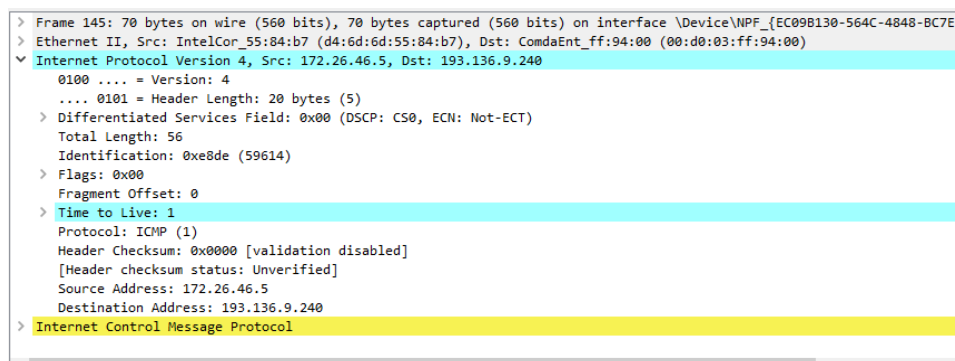
$$\frac{(0.018 + 0.012 + 0.012)}{3} = 0.014 \text{ ms}$$

Os valores foram retirados da figura x.

**2. Selecione a primeira mensagem ICMP capturada (referente a (i) tamanho por defeito) e centre a análise no nível protocolar IP (expandir o tab correspondente na janela de detalhe do Wireshark). Através da análise do cabeçalho IP diga:**

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

Após a abertura do primeiro pacote ICMP enviado podemos ver que o IP de origem é: 172.26.46.5



*Primeiro fragmento do datagrama IPv4*

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

Dentro do campo protocolo é possível ver a identificação do pacote ICMP, cujo respetivo código é 1.

```

> Frame 145: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{EC09B130-564C-4848-BC7E-
> Ethernet II, Src: IntelCor_55:84:b7 (d4:6d:6d:55:84:b7), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.46.5, 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: 0xe8de (59614)
  > Flags: 0x00
    Fragment Offset: 0
  > Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.46.5
    Destination Address: 193.136.9.240
> Internet Control Message Protocol

```

*Primeiro fragmento do datagrama IPv4*

**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?**

O cabeçalho IPv4 é tem um tamanho de 20 bytes, portanto para descobrir o tamanho do campo de dados apenas temos de subtrair ao tamanho total de um pacote IPv4, ou seja, o tamanho do cabeçalho. Ficamos com um total de 36 bytes para payload.

```

> Frame 145: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{EC09B130-564C-4848-BC7E-
> Ethernet II, Src: IntelCor_55:84:b7 (d4:6d:6d:55:84:b7), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.46.5, 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: 0xe8de (59614)
  > Flags: 0x00
    Fragment Offset: 0
  > Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.46.5
    Destination Address: 193.136.9.240
> Internet Control Message Protocol

```

*Primeiro fragmento do datagrama IPv4*

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

O pacote IPv4 não foi fragmentado pois a flag indicadora de mais fragmentos está a 0 tal como o offset, portanto não existe qualquer pacote anterior a este.

```

> Frame 145: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{EC09B130-564C-4848-BC7E-
> Ethernet II, Src: IntelCor_55:84:b7 (d4:6d:6d:55:84:b7), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.46.5, 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: 0xe8de (59614)
  > Flags: 0x00
    Fragment Offset: 0
  > Time to Live: 1
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.46.5
    Destination Address: 193.136.9.240
> Internet Control Message Protocol

```

*Primeiro fragmento do datagrama IPv4*

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

Podemos ver que as únicas variações de pacote para pacote dão-se ao nível do campo de identificação e no campo ttl.

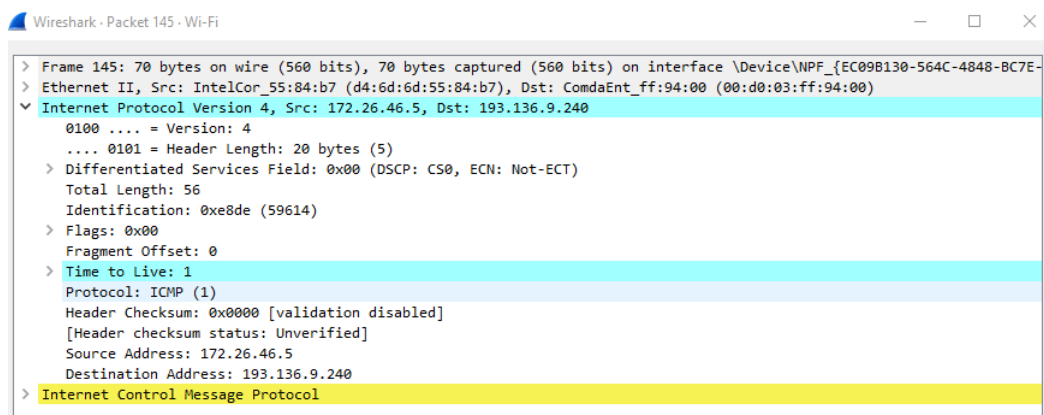
150	3.093979	172.16.115.252	172.26.46.5	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
160	5.631228	172.16.115.252	172.26.46.5	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
170	8.131026	172.16.115.252	172.26.46.5	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
148	3.055608	172.16.2.1	172.26.46.5	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
158	5.508705	172.16.2.1	172.26.46.5	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
168	8.081274	172.16.2.1	172.26.46.5	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
146	3.026807	172.26.254.254	172.26.46.5	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
156	5.530761	172.26.254.254	172.26.46.5	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
166	8.030551	172.26.254.254	172.26.46.5	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
144	2.976165	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=281/6401, ttl=255 (no response found)
145	3.014343	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=282/6657, ttl=1 (no response found)
147	3.054251	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=283/6913, ttl=2 (no response found)
149	3.092572	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=284/7169, ttl=3 (no response found)
151	3.132172	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=285/7425, ttl=4 (reply in 152)
153	5.477874	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=286/7681, ttl=255 (reply in 154)
155	5.528659	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=287/7937, ttl=1 (no response found)
157	5.578985	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=288/8193, ttl=2 (no response found)
159	5.628905	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=289/8449, ttl=3 (no response found)
161	5.679219	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=290/8705, ttl=4 (reply in 162)
163	7.978501	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=291/8961, ttl=255 (reply in 164)
165	8.028344	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=292/9217, ttl=1 (no response found)
167	8.078802	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=293/9473, ttl=2 (no response found)
169	8.129238	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=294/9729, ttl=3 (no response found)
171	8.179640	172.26.46.5	193.136.9.240	ICMP	70 Echo (ping) request id=0x0001, seq=295/9985, ttl=4 (reply in 172)
152	3.133933	193.136.9.240	172.26.46.5	ICMP	70 Echo (ping) reply id=0x0001, seq=285/7425, ttl=61 (request in 151)
154	5.482175	193.136.9.240	172.26.46.5	ICMP	70 Echo (ping) reply id=0x0001, seq=286/7681, ttl=61 (request in 153)

> Frame 148: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF\_{EC09B130-564C-4848-BC7E-3061BD2279A7}, id 0  
 > Ethernet II, Src: ComdaEnt\_ff:94:00 (00:d0:03:ff:94:00), Dst: IntelCor\_55:84:b7 (d4:6d:6d:55:84:b7)  
 > Internet Protocol Version 4, Src: 172.16.2.1, Dst: 172.26.46.5

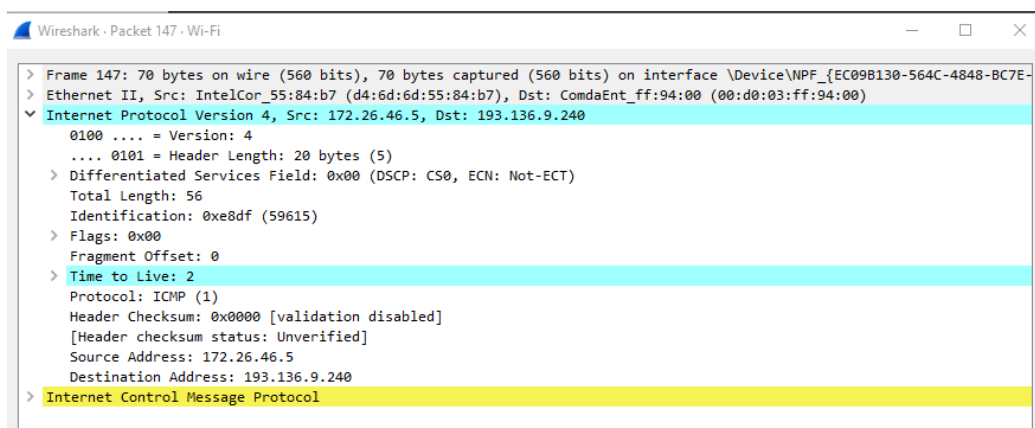
Tráfego ordenado pelo endereço IP fonte

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

A identificação de cada pacote incrementa 1 unidade assim como o ttl.







Dois pacotes IP consecutivos

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

O valor do campo ttl varia entre 253 e 255. Não, o valor varia ligeiramente. Depende do router que o enviou uma vez que o pacote pode já ter saltado.

No.	Time	Source	Destination	Protocol	Length	Info
146	3.026807	172.26.254.254	172.26.46.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
148	3.055608	172.16.2.1	172.26.46.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
150	3.093979	172.16.115.252	172.26.46.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
152	3.133933	193.136.9.240	172.26.46.5	ICMP	70	Echo (ping) reply id=0x0001, seq=285/7425, ttl=61 (request in 151)
154	5.482175	193.136.9.240	172.26.46.5	ICMP	70	Echo (ping) reply id=0x0001, seq=286/7681, ttl=61 (request in 153)
156	5.530761	172.26.254.254	172.26.46.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
158	5.580705	172.16.2.1	172.26.46.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
160	5.631228	172.16.115.252	172.26.46.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
162	5.681463	193.136.9.240	172.26.46.5	ICMP	70	Echo (ping) reply id=0x0001, seq=290/8705, ttl=61 (request in 161)
164	7.983385	193.136.9.240	172.26.46.5	ICMP	70	Echo (ping) reply id=0x0001, seq=291/8961, ttl=61 (request in 163)
166	8.030551	172.26.254.254	172.26.46.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
168	8.081274	172.16.2.1	172.26.46.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
170	8.131026	172.16.115.252	172.26.46.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
172	8.182051	193.136.9.240	172.26.46.5	ICMP	70	Echo (ping) reply id=0x0001, seq=295/9985, ttl=61 (request in 171)
144	2.976165	172.26.46.5	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=281/6401, ttl=255 (no response found)
145	3.014343	172.26.46.5	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=282/6657, ttl=1 (no response found)
147	3.054251	172.26.46.5	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=283/6913, ttl=2 (no response found)
149	3.092572	172.26.46.5	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=284/7169, ttl=3 (no response found)
151	3.132172	172.26.46.5	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=285/7425, ttl=4 (reply in 152)
153	5.477874	172.26.46.5	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=286/7681, ttl=255 (reply in 154)
155	5.528659	172.26.46.5	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=287/7937, ttl=1 (no response found)

Tráfego ordenado por endereço de destino

IP	Name
172.26.254.254	172.26.254.254
172.16.2.1	172.16.2.1
172.16.115.252	172.16.115.252
193.136.9.240	marco.uminho.pt

Endereços IP

**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 para 32XX bytes.**

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

Como somos o grupo 54, definimos o tamanho do pacote para 3254 bytes. Estando identificada a primeira mensagem ICMP, houve necessidade de fragmentar o pacote porque ele ultrapassava o tamanho limite do protocolo IPv4 que é 1500.

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?

[illegible]

Primeiro fragmento do datagrama IP

No cabeçalho, podemos ver que existe a flag “More fragments” e esta é igual a 1, logo o datagrama encontra-se fragmentado.

Podemos ver que é o primeiro fragmento do datagrama, pois o “Fragment Offset” é 0.

O tamanho deste datagrama IP é 1500, tal como indica em “Total length”.

- 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?

A informação do cabeçalho IP que nos indica que não se trata do 1º fragmento é o offset, que se encontra a 1480.

```
> Frame 2: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{EC09B130-564C-4848-BC77-00011D5B46F8}
> Ethernet II, Src: IntelCor_55:84:b7 (d4:6d:6d:55:84:b7), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.46.5, 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: 0x6e32 (28210)
    > Flags: 0x20, More fragments
    Fragment Offset: 1480
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.46.5
    Destination Address: 193.136.9.240
    [Reassembled IPv4 in frame: 3]
> Data (1480 bytes)
```

*Segundo fragmento do datagrama IP*

Sim, há mais fragmentos pois existe a flag “More fragments” que se encontra a 1.

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

```
> Frame 3: 308 bytes on wire (2464 bits), 308 bytes captured (2464 bits) on interface \Device\NPF_{EC09B130-564C-4848-BC77-00011D5B46F8}
> Ethernet II, Src: IntelCor_55:84:b7 (d4:6d:6d:55:84:b7), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.46.5, 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: 294
    Identification: 0x6e32 (28210)
    > Flags: 0x01
    Fragment Offset: 2960
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.46.5
    Destination Address: 193.136.9.240
    > [3 IPv4 Fragments (3234 bytes): #1(1480), #2(1480), #3(274)]
> Internet Control Message Protocol
```

*Terceiro fragmento do datagrama IP*

Foram criados 3 fragmentos a partir do datagrama original, como se pode ver na figura.

Este pacote já não tem a flag “More fragments”, que se encontra a 0, e o ID nos pacotes fragmentados mantém-se.

- 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.**

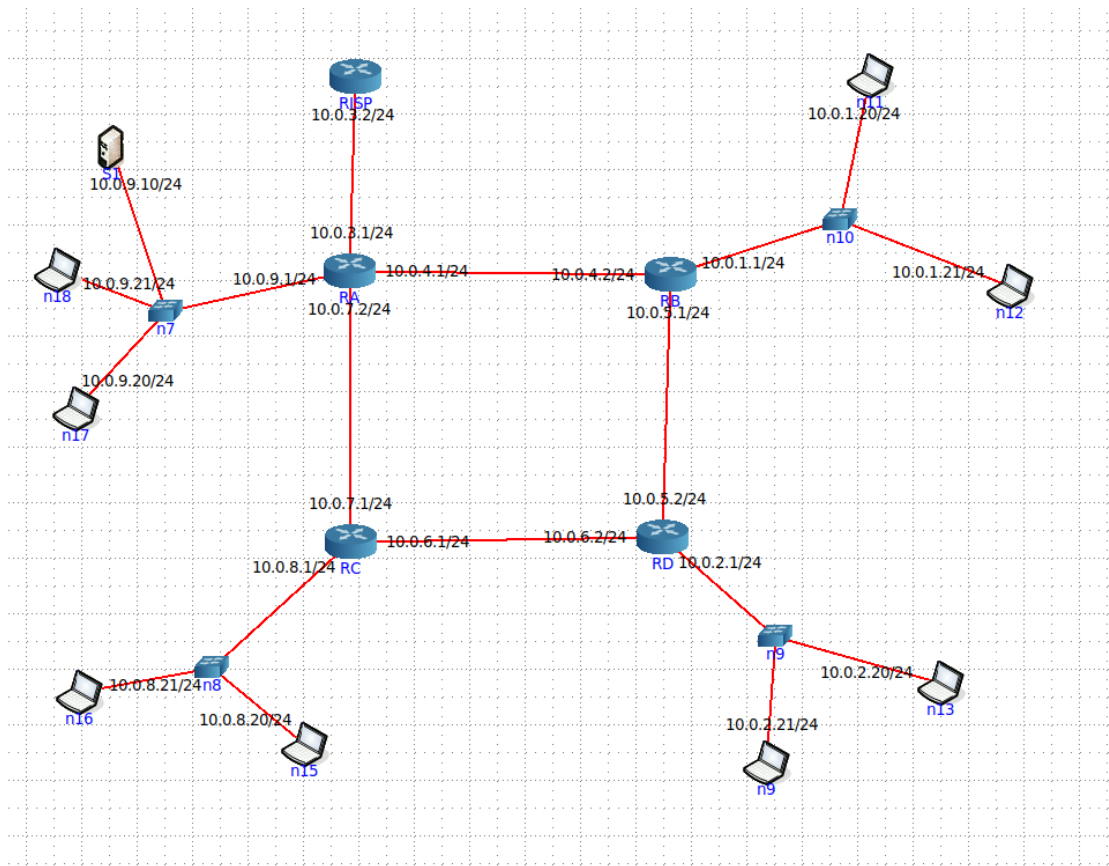
Entre os diferentes fragmentos, os campos que mudam no cabeçalho IP são a flag “More fragments”, o “Fragment Offset” e o “Total length”.

Para reconstruir o pacote basta utilizar o ID do pacote que se mantém igual e juntar os pacotes pela ordem crescente dos offsets.

## TP2: Protocolo IP (Parte II)

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.



Topologia de rede

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

São endereços privados uma vez que utilizam como prefixo 10/8 que é reservado para redes privadas.

- c. Porque razão não é atribuído um endereço IP aos switches?

Os switches não tem um IP atribuído porque operam através de MAC e não IPs.

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

```
vcmd
root@n16:/tmp/pycore.42193/n16.conf# ping 10.0.9.10
PING 10.0.9.10 (10.0.9.10) 56(84) bytes of data.
64 bytes from 10.0.9.10: icmp_seq=1 ttl=62 time=0.042 ms
64 bytes from 10.0.9.10: icmp_seq=2 ttl=62 time=0.064 ms
64 bytes from 10.0.9.10: icmp_seq=3 ttl=62 time=0.056 ms
^C
--- 10.0.9.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.042/0.054/0.064/0.009 ms
root@n16:/tmp/pycore.42193/n16.conf#

vcmd
root@n11:/tmp/pycore.42193/n11.conf# ping 10.0.9.10
PING 10.0.9.10 (10.0.9.10) 56(84) bytes of data.
64 bytes from 10.0.9.10: icmp_seq=1 ttl=62 time=0.083 ms
64 bytes from 10.0.9.10: icmp_seq=2 ttl=62 time=0.055 ms
64 bytes from 10.0.9.10: icmp_seq=3 ttl=62 time=0.060 ms
^C
--- 10.0.9.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2045ms
rtt min/avg/max/mdev = 0.055/0.066/0.083/0.012 ms
root@n11:/tmp/pycore.42193/n11.conf#

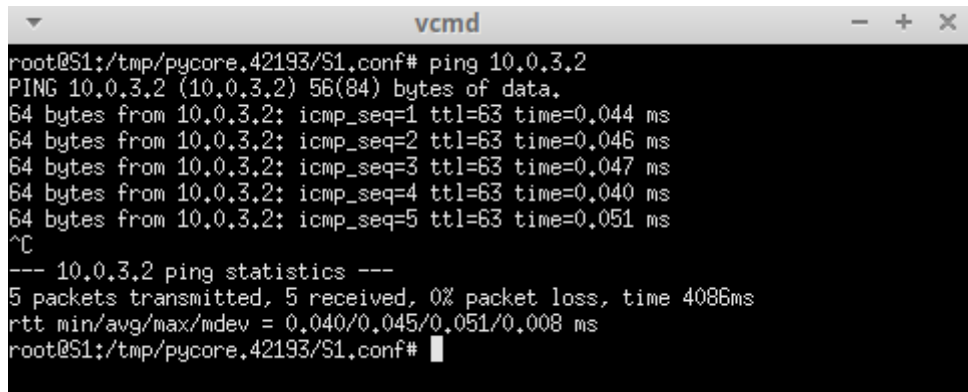
vcmd
root@n9:/tmp/pycore.42193/n9.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags        MSS Window  irtt Iface
0.0.0.0          10.0.2.1       0.0.0.0         UG           0 0        0 eth0
10.0.2.0         0.0.0.0        255.255.255.0   U            0 0        0 eth0
root@n9:/tmp/pycore.42193/n9.conf# ping 10.0.9.10
PING 10.0.9.10 (10.0.9.10) 56(84) bytes of data.
64 bytes from 10.0.9.10: icmp_seq=1 ttl=61 time=0.128 ms
64 bytes from 10.0.9.10: icmp_seq=2 ttl=61 time=0.061 ms
64 bytes from 10.0.9.10: icmp_seq=3 ttl=61 time=0.065 ms
^C
--- 10.0.9.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2054ms
rtt min/avg/max/mdev = 0.061/0.084/0.128/0.032 ms
root@n9:/tmp/pycore.42193/n9.conf#

vcmd
root@n18:/tmp/pycore.42193/n18.conf# ping 10.0.9.10
PING 10.0.9.10 (10.0.9.10) 56(84) bytes of data.
64 bytes from 10.0.9.10: icmp_seq=1 ttl=64 time=0.051 ms
64 bytes from 10.0.9.10: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 10.0.9.10: icmp_seq=3 ttl=64 time=0.036 ms
^C
--- 10.0.9.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2052ms
rtt min/avg/max/mdev = 0.032/0.039/0.051/0.010 ms
root@n18:/tmp/pycore.42193/n18.conf#
```

Testes de conectividade

Assim sendo, verifica-se a conectividade entre os vários departamentos e o servidor.

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



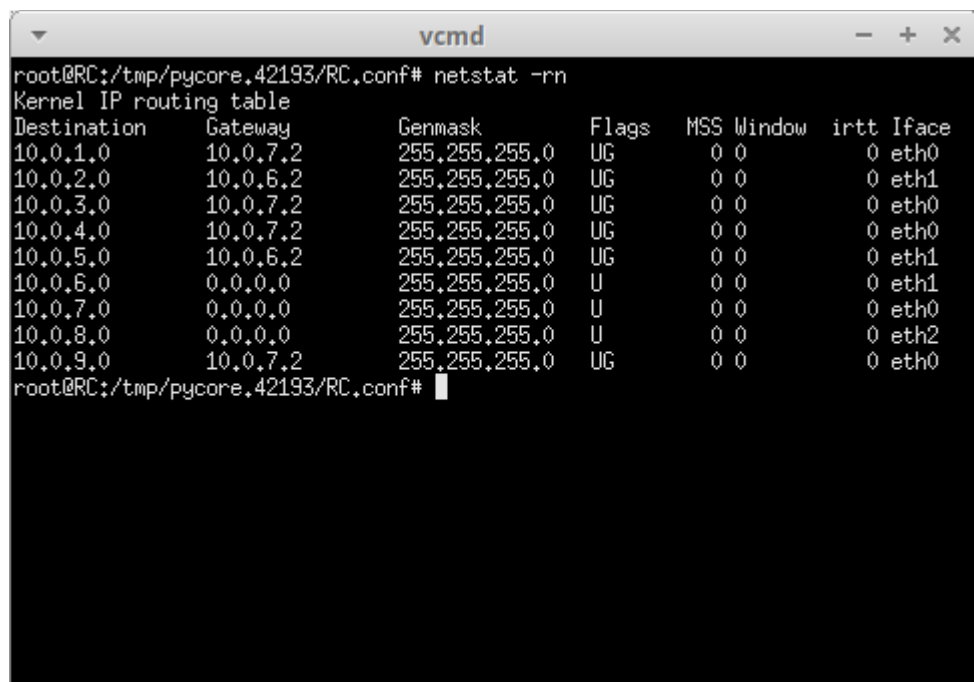
```
vcmd
root@S1:/tmp/pycore.42193/S1.conf# ping 10.0.3.2
PING 10.0.3.2 (10.0.3.2) 56(84) bytes of data:
64 bytes from 10.0.3.2: icmp_seq=1 ttl=63 time=0.044 ms
64 bytes from 10.0.3.2: icmp_seq=2 ttl=63 time=0.046 ms
64 bytes from 10.0.3.2: icmp_seq=3 ttl=63 time=0.047 ms
64 bytes from 10.0.3.2: icmp_seq=4 ttl=63 time=0.040 ms
64 bytes from 10.0.3.2: icmp_seq=5 ttl=63 time=0.051 ms
^C
--- 10.0.3.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4086ms
rtt min/avg/max/mdev = 0.040/0.045/0.051/0.008 ms
root@S1:/tmp/pycore.42193/S1.conf#
```

*Teste de conectividade com RISP*

Logo verifica-se que existe conectividade entre o servidor e router RISP.

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

- 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`).

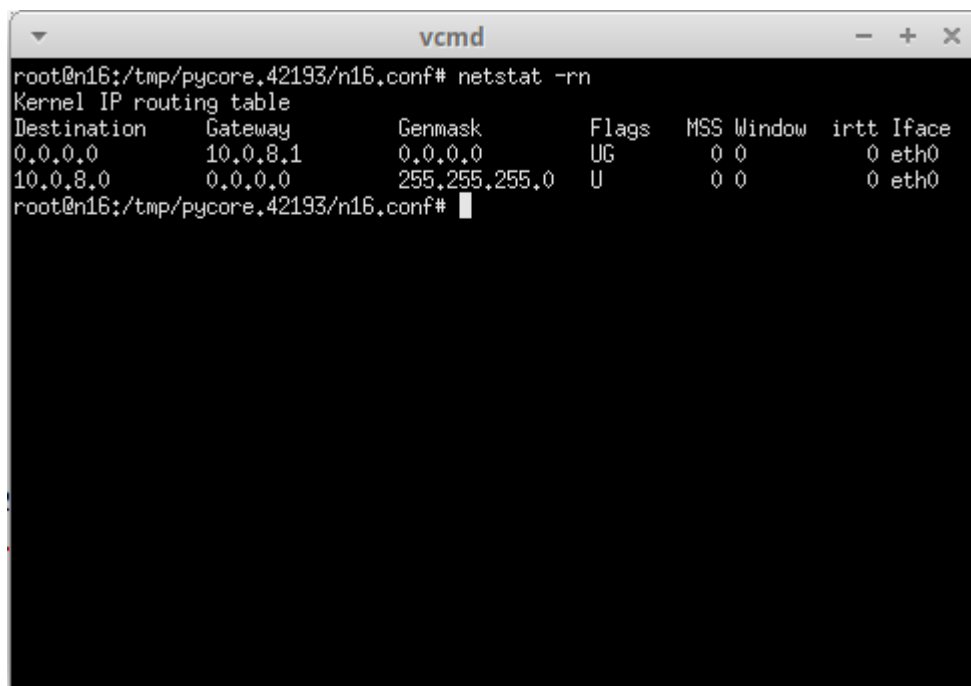


```
vcmd
root@RC:/tmp/pycore.42193/RC.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags         MSS Window  irtt  Iface
10.0.1.0          10.0.7.2        255.255.255.0   UG            0 0        0     eth0
10.0.2.0          10.0.6.2        255.255.255.0   UG            0 0        0     eth1
10.0.3.0          10.0.7.2        255.255.255.0   UG            0 0        0     eth0
10.0.4.0          10.0.7.2        255.255.255.0   UG            0 0        0     eth0
10.0.5.0          10.0.6.2        255.255.255.0   UG            0 0        0     eth1
10.0.6.0          0.0.0.0         255.255.255.0   U             0 0        0     eth1
10.0.7.0          0.0.0.0         255.255.255.0   U             0 0        0     eth0
10.0.8.0          0.0.0.0         255.255.255.0   U             0 0        0     eth2
10.0.9.0          10.0.7.2        255.255.255.0   UG            0 0        0     eth0
root@RC:/tmp/pycore.42193/RC.conf#
```

*Tabela de encaminhamento do router C*

Cada entrada da tabela refere-se a uma sub-rede diferente de destino. O gateway é o correspondente ao próximo salto no encaminhamento de pacotes para um determinado destino.

Tendo isto em consideração, quando o gateway é 0.0.0.0, como é o caso da 6ª, 7ª e 8ª entrada na tabela, sabemos que o router onde estamos vai ser responsável pelo encaminhamento dos pacotes diretamente, isto acontece quando o router em questão tem uma interface na rede de destino. Nos outros casos, gateway diferente de 0.0.0.0, o router limita-se a transmitir o pacote para o dispositivo identificado pelo IP indicado no gateway, que por sua vez fará este mesmo processo.



```
root@n16:/tmp/pycore.42193/n16.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.8.1 0.0.0.0 UG 0 0 0 eth0
10.0.8.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@n16:/tmp/pycore.42193/n16.conf#
```

*Tabela de encaminhamento de um portátil do departamento C*

A primeira entrada é a entrada default, em que a máscara é 0.0.0.0. Os pacotes dão match nesta entrada quando não existe nenhuma entrada em que consigam dar match. Ou seja, um pacote cujo destino não seja conhecido segue por esta entrada sendo enviado para o gateway que fará o seu encaminhamento.

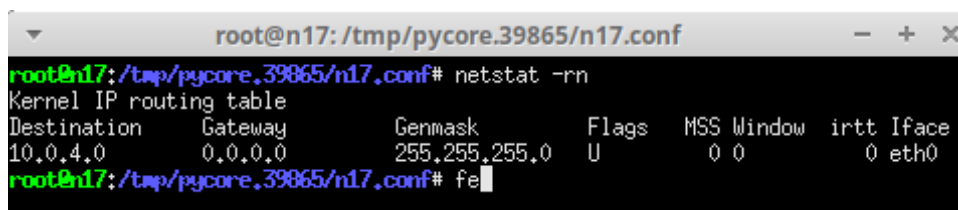
Na segunda entrada, o destino é a própria rede local, logo o gateway é 0.0.0.0, o que significa que o próprio dispositivo é capaz de transmitir a informação diretamente para o destino.



- b. Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico (sugestão: analise que processos estão a correr em cada sistema, por exemplo, `ps -ax`).

É utilizado um encaminhamento dinâmico, pois as rotas são definidas de forma automática através da troca de informação entre os routers.

- c. Admita que, por questões administrativas, a rota por defeito (0.0.0.0 ou default) deve ser retirada definitivamente da tabela de encaminhamento do servidor S1 localizado no departamento A. Use o comando `route delete` para o efeito. Que implicações tem esta medida para os utilizadores da organização MIEI-RC que acedem ao servidor. Justifique.



```
root@n17: /tmp/pycore.39865/n17.conf
root@n17: /tmp/pycore.39865/n17.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.4.0         0.0.0.0        255.255.255.0   U        0  0        0 eth0
root@n17: /tmp/pycore.39865/n17.conf# fe
```

*Tabela de encaminhamento após remoção da rota por defeito*



```
root@n11: /tmp/pycore.39865/n11.conf
root@n11: /tmp/pycore.39865/n11.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
^C
--- 10.0.4.10 ping statistics ---
16 packets transmitted, 0 received, 100% packet loss, time 15352ms
root@n11: /tmp/pycore.39865/n11.conf#
```

*Teste de conectividade com o servidor após a remoção da rota deste.*

O servidor perde toda a conectividade com hosts que não estejam na sua rede local, uma vez que, removendo a rota por defeito o servidor não tem definida a rota de envio de tráfego para redes não locais. Sendo assim é possível enviar dados para o servidor, mas não é possível obter qualquer resposta deste.

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

```
root@n17: /tmp/pycore.39865/n17.conf
root@n17: /tmp/pycore.39865/n17.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask          Flags   MSS Window  irtt Iface
10.0.4.0          0.0.0.0          255.255.255.0    U        0 0        0 eth0
10.0.5.0          10.0.4.1         255.255.255.0    UG        0 0        0 eth0
10.0.6.0          10.0.4.1         255.255.255.0    UG        0 0        0 eth0
10.0.7.0          10.0.4.1         255.255.255.0    UG        0 0        0 eth0
10.0.8.0          10.0.4.1         255.255.255.0    UG        0 0        0 eth0
root@n17: /tmp/pycore.39865/n17.conf#
```

*Tabela de encaminhamento após inserção das rotas estáticas*

Para inserção das rotas estáticas no servidor foram usados os comandos:

```
route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.4.1
```

```
route add -net 10.0.6.0 netmask 255.255.255.0 gw 10.0.4.1
```

```
route add -net 10.0.7.0 netmask 255.255.255.0 gw 10.0.4.1
```

```
route add -net 10.0.8.0 netmask 255.255.255.0 gw 10.0.4.1
```

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

```
root@n17: /tmp/pycore.39865/n17.conf
root@n17: /tmp/pycore.39865/n17.conf# ping 10.0.8.2
PING 10.0.8.2 (10.0.8.2) 56(84) bytes of data:
64 bytes from 10.0.8.2: icmp_seq=1 ttl=63 time=0.115 ms
64 bytes from 10.0.8.2: icmp_seq=2 ttl=63 time=0.104 ms
64 bytes from 10.0.8.2: icmp_seq=3 ttl=63 time=0.104 ms
64 bytes from 10.0.8.2: icmp_seq=4 ttl=63 time=0.108 ms
^C
--- 10.0.8.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3063ms
rtt min/avg/max/mdev = 0.104/0.107/0.115/0.013 ms
root@n17: /tmp/pycore.39865/n17.conf#
```

*Teste de conectividade com a internet*

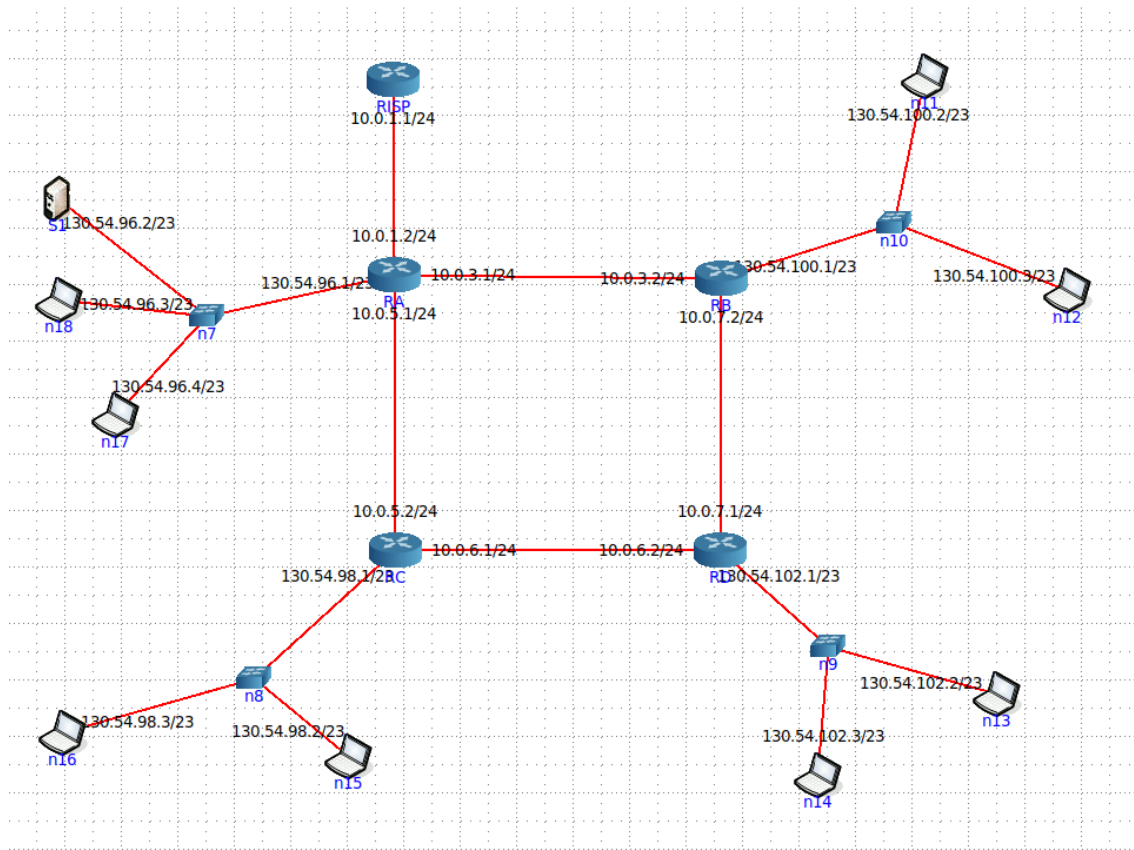
```
root@n11: /tmp/pycore.39865/n11.conf
root@n11: /tmp/pycore.39865/n11.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_seq=1 ttl=61 time=0.179 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=61 time=0.130 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=61 time=0.138 ms
^C
--- 10.0.4.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.130/0.149/0.179/0.021 ms
root@n11: /tmp/pycore.39865/n11.conf#
```

*Teste de conectividade com o servidor de uma rede não local*

É possível verificar pelas imagens que os testes de conectividade foram bem sucedidos.

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

- a. Considere que dispõe apenas do endereço de rede IP 130.XX.96.0/19, em que XX é o decimal correspondendo ao seu número de grupo (PLXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo a rede de acesso e core inalteradas) e atribua endereços às interfaces dos vários sistemas envolvidos. Assuma que todos os endereços de sub-redes são usáveis. Deve justificar as opções usadas.



Utilizando 4 bits para identificação de sub-redes temos como vantagem, a possibilidade de expansão, na medida em que ainda podemos ter mais 12 sub-redes. Por outro lado, esta escolha leva a que o número de endereços IP disponíveis para hosts seja mais reduzido. Outra opção que também poderia ser adequada seria utilizar 3 bits para representação de sub-redes o que também permitiria a expansão, dado que seria possível ter 8 sub-redes, mas

o número de endereços para hosts seria o dobro. Dependendo da situação uma opção poderá ser mais viável que a outra, no entanto, sem mais informações sobre o contexto da situação não conseguimos decidir qual será a escolha mais acertada.

#### **Sub-redes**

96 -> 011 0000 x – corresponde ao prefixo de sub-rede 130.54.96.0/23

98 -> 011 0001 x – corresponde ao prefixo de sub-rede 130.54.98.0/23

100 -> 011 0010 x – corresponde ao prefixo de sub-rede 130.54.100.0/23

102 -> 011 0011 x – corresponde ao prefixo de sub-rede 130.54.102.0/23

#### **Gama de endereços**

A -> 130.54.96.1/23 - 130.54.97.254/23

B -> 130.54.98.1/23 - 130.54.99.254/23

C -> 130.54.100.1/23 - 130.54.101.254/23

D -> 130.54.102.1/23 - 130.54.103.254/23

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

A máscara de rede utilizada foi 255.255.254.0/23, uma vez que se utilizou 4 bits para sub-redes e a máscara da rede original era 255.255.96.0/19.

Este modo de endereçamento permite a conexão de 510 hosts em cada departamento ( $2^9 - \text{IPRede} - \text{IPBroadcast}$ ).

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

```
vcmd
root@n13:/tmp/pycore.39181/n13.conf# ping 130.54.100.2
PING 130.54.100.2 (130.54.100.2) 56(84) bytes of data.
64 bytes from 130.54.100.2: icmp_seq=1 ttl=62 time=0.115 ms
64 bytes from 130.54.100.2: icmp_seq=2 ttl=62 time=0.121 ms
64 bytes from 130.54.100.2: icmp_seq=3 ttl=62 time=0.118 ms
64 bytes from 130.54.100.2: icmp_seq=4 ttl=62 time=0.153 ms
64 bytes from 130.54.100.2: icmp_seq=5 ttl=62 time=0.182 ms
^C
--- 130.54.100.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4095ms
rtt min/avg/max/mdev = 0.115/0.137/0.182/0.029 ms
root@n13:/tmp/pycore.39181/n13.conf# ping 130.54.96.2
PING 130.54.96.2 (130.54.96.2) 56(84) bytes of data.
64 bytes from 130.54.96.2: icmp_seq=1 ttl=61 time=0.089 ms
64 bytes from 130.54.96.2: icmp_seq=2 ttl=61 time=0.141 ms
64 bytes from 130.54.96.2: icmp_seq=3 ttl=61 time=0.369 ms
64 bytes from 130.54.96.2: icmp_seq=4 ttl=61 time=0.139 ms
^C
--- 130.54.96.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3123ms
rtt min/avg/max/mdev = 0.089/0.184/0.369/0.109 ms
root@n13:/tmp/pycore.39181/n13.conf# ping 130.54.98.3
PING 130.54.98.3 (130.54.98.3) 56(84) bytes of data.
64 bytes from 130.54.98.3: icmp_seq=1 ttl=62 time=0.123 ms
64 bytes from 130.54.98.3: icmp_seq=2 ttl=62 time=0.118 ms
64 bytes from 130.54.98.3: icmp_seq=3 ttl=62 time=0.122 ms
64 bytes from 130.54.98.3: icmp_seq=4 ttl=62 time=0.117 ms
64 bytes from 130.54.98.3: icmp_seq=5 ttl=62 time=0.126 ms
^C
--- 130.54.98.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4081ms
rtt min/avg/max/mdev = 0.117/0.121/0.126/0.007 ms
root@n13:/tmp/pycore.39181/n13.conf#

vcmd
root@n11:/tmp/pycore.39181/n11.conf# ping 130.54.98.2
PING 130.54.98.2 (130.54.98.2) 56(84) bytes of data.
64 bytes from 130.54.98.2: icmp_seq=1 ttl=61 time=0.090 ms
64 bytes from 130.54.98.2: icmp_seq=2 ttl=61 time=0.135 ms
64 bytes from 130.54.98.2: icmp_seq=3 ttl=61 time=0.137 ms
64 bytes from 130.54.98.2: icmp_seq=4 ttl=61 time=0.143 ms
^C
--- 130.54.98.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3095ms
rtt min/avg/max/mdev = 0.090/0.126/0.143/0.022 ms
root@n11:/tmp/pycore.39181/n11.conf# ping 130.54.96.2
PING 130.54.96.2 (130.54.96.2) 56(84) bytes of data.
64 bytes from 130.54.96.2: icmp_seq=1 ttl=62 time=0.053 ms
64 bytes from 130.54.96.2: icmp_seq=2 ttl=62 time=0.128 ms
64 bytes from 130.54.96.2: icmp_seq=3 ttl=62 time=0.185 ms
64 bytes from 130.54.96.2: icmp_seq=4 ttl=62 time=0.118 ms
^C
--- 130.54.96.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3057ms
rtt min/avg/max/mdev = 0.053/0.121/0.185/0.046 ms
root@n11:/tmp/pycore.39181/n11.conf#
```

```
vcmd
root@n16:/tmp/pycore,39181/n16.conf# ping 130.54.96.2
PING 130.54.96.2 (130.54.96.2) 56(84) bytes of data.
64 bytes from 130.54.96.2: icmp_seq=1 ttl=62 time=0.054 ms
64 bytes from 130.54.96.2: icmp_seq=2 ttl=62 time=0.212 ms
64 bytes from 130.54.96.2: icmp_seq=3 ttl=62 time=0.126 ms
64 bytes from 130.54.96.2: icmp_seq=4 ttl=62 time=0.125 ms
64 bytes from 130.54.96.2: icmp_seq=5 ttl=62 time=0.121 ms
^C
--- 130.54.96.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4074ms
rtt min/avg/max/mdev = 0.054/0.127/0.212/0.051 ms
root@n16:/tmp/pycore,39181/n16.conf#
```

*Testes de conectividade*

Foram feitos pings entre a rede D e as restantes, de seguida fez-se ping entre a rede B, C e A e por fim testou-se a conexão entre C e A.

# Conclusão

Neste trabalho, fomos capazes de consolidar os conhecimentos que nos foram transmitidos nas aulas teóricas. Com efeito, relativamente à 1ª fase, a possibilidade de construir redes manualmente através do CORE e de análise de tráfego ICMP, permitiu-nos visualizar todo o processo que é feito por um datagrama no seu processo de encaminhamento. De facto, esta experiência levou a que conseguíssemos compreender a variação do TTL dos pacotes quando passavam pelos diferentes routers e a maneira como a identificação dos pacotes incrementa. Além disso, foi possível analisar o cabeçalho de vários pacotes IPv4 o que por um lado permitiu-nos perceber melhor os conteúdos de cada pacote e por outro perceber as variações que se verificam na fragmentação de pacotes. No que toca, à 2ª fase, vimos que os hosts normalmente utilizam encaminhamento estático enquanto que os routers utilizam encaminhamento dinâmico. Em particular, é importante destacar o exercício de criação de rotas estáticas que se mostrou muito útil uma vez que para conseguir fazer as ligações entre o servidor S do departamento A e os restantes departamentos era necessário compreender bem o funcionamento das tabelas de encaminhamento. Por fim, a organização de sub-redes manualmente permitiu aprimorar os nossos conhecimentos nesta área e utilizá-los num exemplo concreto servindo assim de complemento às aulas teóricas. Em suma, este projeto permitiu-nos compreender melhor o protocolo IP e consolidar os conceitos que foram lecionados.