

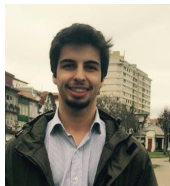
Universidade do Minho

Relatório - Redes de Computadores

MIEI - 3º ANO - 1º SEMESTRE
UNIVERSIDADE DO MINHO

PROTOCOLO IP

GRUPO 58



Dinis Peixoto
A75353



Ricardo Pereira
A74185



Marcelo Lima
A75210

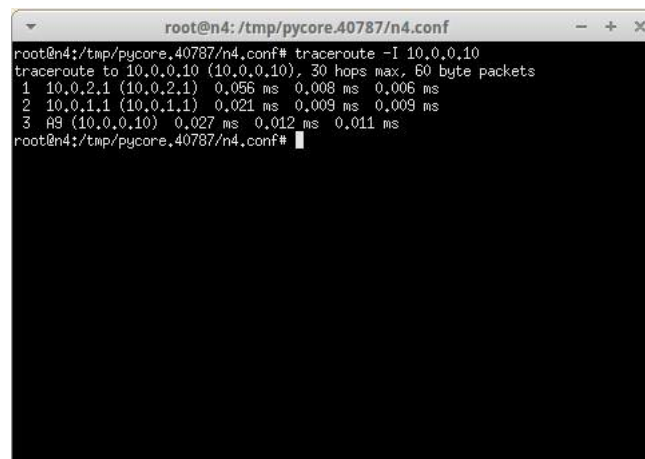
16 de Novembro de 2017

1. Questões e Respostas

1.1 Parte I

1.1.1

- a. 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.



```
root@n4: /tmp/pycore.40787/n4.conf
root@n4:/tmp/pycore.40787/n4.conf# traceroute -I 10.0.0.10
traceroute to 10.0.0.10 (10.0.0.10), 30 hops max, 60 byte packets
 1 10.0.2.1 (10.0.2.1)  0.056 ms  0.008 ms  0.006 ms
 2 10.0.1.1 (10.0.1.1)  0.021 ms  0.009 ms  0.009 ms
 3 R9 (10.0.0.10)  0.027 ms  0.012 ms  0.011 ms
root@n4:/tmp/pycore.40787/n4.conf#
```

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

No.	Time	Source	Destination	Protocol	Length	Info
19	56.276529	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=1/256, ttl=1
20	56.276543	10.0.2.1	10.0.2.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
21	56.276550	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=2/512, ttl=1
22	56.276555	10.0.2.1	10.0.2.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
23	56.276559	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=3/768, ttl=1
24	56.276563	10.0.2.1	10.0.2.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
25	56.276568	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=4/1024, ttl=2
26	56.276586	10.0.1.1	10.0.2.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
27	56.276590	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=5/1280, ttl=2
28	56.276597	10.0.1.1	10.0.2.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
29	56.276601	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=6/1536, ttl=2
30	56.276607	10.0.1.1	10.0.2.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
31	56.276611	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=7/1792, ttl=3
32	56.276636	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) reply id=0x0089, seq=7/1792, ttl=62
33	56.276642	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=8/2048, ttl=3
34	56.276651	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) reply id=0x0089, seq=8/2048, ttl=62
35	56.276655	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=9/2304, ttl=3
36	56.276663	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) reply id=0x0089, seq=9/2304, ttl=62
37	56.276667	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=10/2560, ttl=4
38	56.276675	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) reply id=0x0089, seq=10/2560, ttl=62
39	56.276679	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=11/2816, ttl=4
40	56.276687	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) reply id=0x0089, seq=11/2816, ttl=62
41	56.276690	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) request id=0x0089, seq=12/3072, ttl=4

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

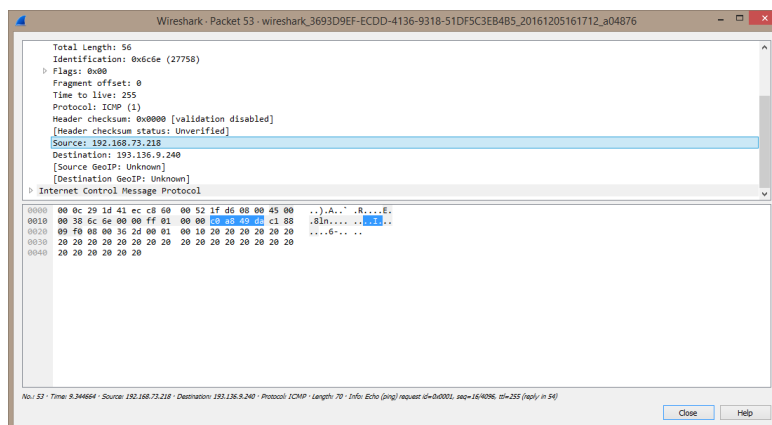
O valor inicial mínimo do campo TTL deve ser 3, para poder alcançar o destino n1, porque a partir do TTL 3 não se verifica a receção de mensagens de erro, mas verifica-se o envio de replys. Podemos constatar isto no *printscreen* anterior.

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

Tendo em conta os valores verificados no primeiro *printscreen* apresentado e fazendo a média destes podemos obter o valor do tempo médio do tempo de ida-e-volta, sendo este aproximadamente 0,035 ms.

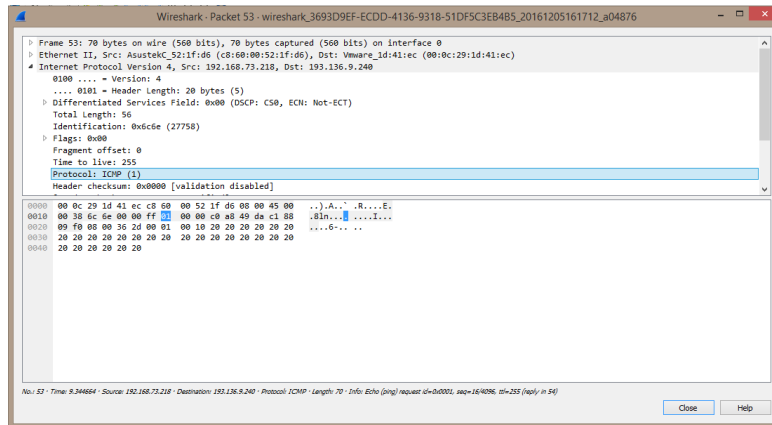
1.1.2

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



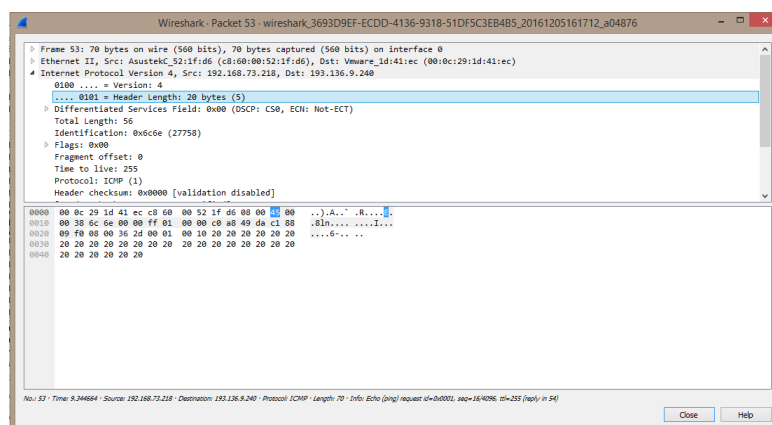
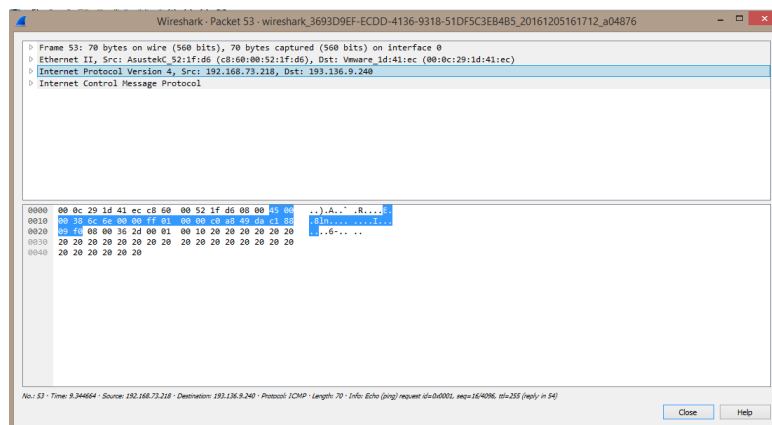
O endereço IP da interface ativa no computador utilizado é 192.168.73.218.

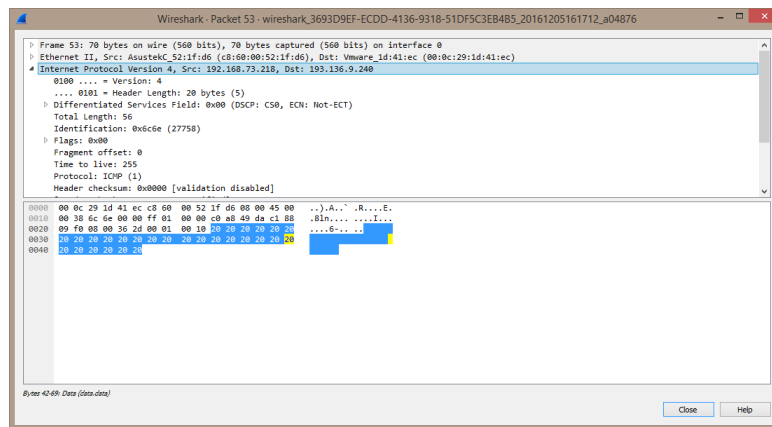
b. Qual é o valor do campo do protocolo? O que identifica?



O valor do campo do protocolo é 1 e identifica o protocolo ICMP.

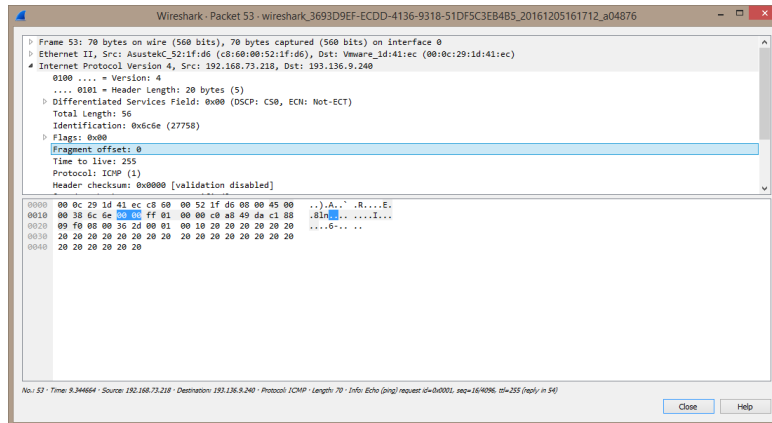
c. Quantos bytes tem o cabeçalho IP(v4)? Quantos bytes tem o campo de dados (payload) do datagrama? Como se calcula o tamanho do payload?





O cabeçalho IP(v4) tem 20 bytes. O tamanho do campo de dados (payload) é 36 bytes, este é calculado através da diferença do número total de bytes pelo número de bytes do cabeçalho, assim $56 - 20 = 36$ bytes.

d. O datagrama IP foi fragmentado? Justifique.



Não, porque o fragment offset e as flags, nomeadamente a flag *more fragments*, têm o valor 0. A flag *more fragments* tendo o valor nulo indica que não existem mais fragmentos, e o *fragment offset*, que indica a posição no datagrama que a porção de dados veiculada pelo fragmento, logo não ocorreu fragmentação uma vez que também tem valor nulo.

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

The image shows a Wireshark packet capture window. The packet list pane shows a sequence of packets, with the source and destination IP addresses, the protocol, and the length of the packet. The packets are ordered by time, and the source IP address is 192.168.73.218. The destination IP address is 193.136.9.240. The protocol is ICMP. The length of the packet is 70 bytes. The sequence of packets is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
53	9.344664	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=16/4096, ttl=255 (reply in 54)
55	9.408732	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=17/4352, ttl=1 (no response found)
56	9.408818	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=18/4608, ttl=2 (no response found)
59	9.433993	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=19/4864, ttl=3 (reply in 60)
61	9.474029	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=20/5120, ttl=4 (reply in 62)
63	9.514018	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=21/5376, ttl=5 (reply in 64)
65	9.569741	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=22/5632, ttl=6 (reply in 66)
67	9.618127	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=23/5888, ttl=7 (reply in 68)
69	9.658127	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=24/6144, ttl=8 (reply in 70)
71	9.698159	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=25/6400, ttl=9 (reply in 72)
73	9.731170	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=26/6656, ttl=10 (reply in 74)
75	9.771223	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=27/6912, ttl=11 (reply in 76)
77	9.811210	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=28/7168, ttl=12 (reply in 78)
79	9.818537	192.168.73.218	193.136.19.1	DNS	87	Standard query 0x38ed PTR 254.73.168.192.in-addr.arpa
81	9.820240	192.168.73.218	192.168.73.254	NBNS	92	Name query HOSTAT <00><00><00><00><00><00><00><00><00><00><00><00><00><00><00>
83	9.828510	192.168.73.218	224.0.0.252	LLMNR	87	Standard query 0x14a6 PTR 254.73.168.192.in-addr.arpa
84	9.851249	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=29/7424, ttl=13 (reply in 85)
86	9.891389	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=30/7680, ttl=14 (reply in 87)
88	9.931327	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=31/7936, ttl=15 (reply in 89)
90	9.971427	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=32/8192, ttl=16 (reply in 91)
92	10.011374	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=33/8448, ttl=17 (reply in 93)
96	10.051376	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=34/8704, ttl=18 (reply in 97)
98	10.091429	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=35/8960, ttl=19 (reply in 99)
100	10.131464	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=36/9216, ttl=20 (reply in 101)
102	10.171558	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=37/9472, ttl=21 (reply in 103)
104	10.201117	192.168.73.218	193.136.19.1	DNS	87	Standard query 0x72fe PTR 254.19.136.193.in-addr.arpa
106	10.211775	192.168.73.218	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=38/9728, ttl=22 (reply in 107)
109	10.238555	192.168.73.218	224.0.0.252	LLMNR	87	Standard query 0x14a6 PTR 254.73.168.192.in-addr.arpa

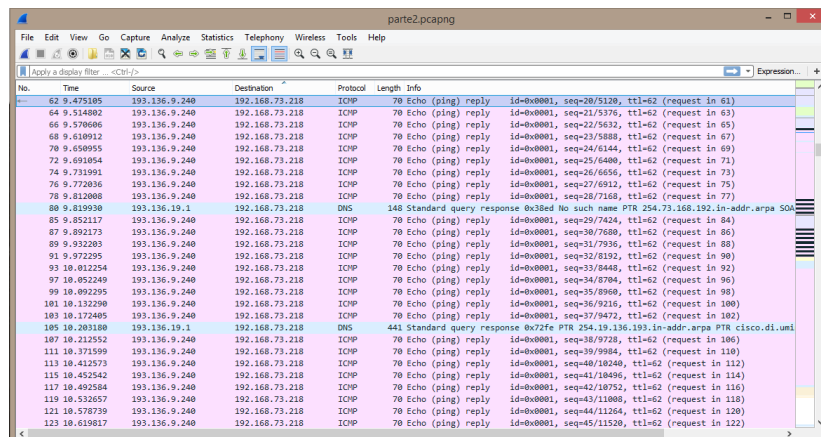
Os campos que variam são o identificador e o TTL.

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

O facto de incrementar 1 por cada pacote apresentado.

g. Ordene o tráfego capturado por endereço destino e encontre a série de resposta 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?

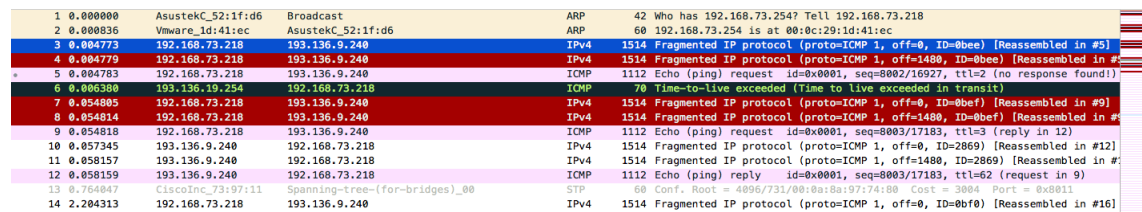


No.	Time	Source	Destination	Protocol	Length	Info
62	9.475105	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=20/5120, ttl=62 (request in 61)
64	9.514802	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=21/5376, ttl=62 (request in 63)
66	9.570606	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=22/5632, ttl=62 (request in 65)
68	9.610912	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=23/5888, ttl=62 (request in 67)
70	9.650955	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=24/6144, ttl=62 (request in 69)
72	9.691054	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=25/6400, ttl=62 (request in 71)
74	9.731991	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=26/6656, ttl=62 (request in 73)
76	9.772036	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=27/6912, ttl=62 (request in 75)
78	9.812008	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=28/7168, ttl=62 (request in 77)
80	9.819590	193.136.19.1	192.168.73.218	DNS	148	Standard query response 0x30ed No such name PTR 254.73.168.192.in-addr.arpa SOA
85	9.852117	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=29/7424, ttl=62 (request in 84)
87	9.892173	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=30/7680, ttl=62 (request in 86)
89	9.932203	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=31/7936, ttl=62 (request in 88)
91	9.972295	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=32/8192, ttl=62 (request in 90)
93	10.012254	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=33/8448, ttl=62 (request in 92)
97	10.052249	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=34/8704, ttl=62 (request in 96)
99	10.092295	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=35/8960, ttl=62 (request in 98)
101	10.132230	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=36/9216, ttl=62 (request in 100)
103	10.172405	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=37/9472, ttl=62 (request in 102)
105	10.203180	193.136.19.1	192.168.73.218	DNS	441	Standard query response 0x72fe PTR 254.19.136.193.in-addr.arpa PTR cisco.di.uni
107	10.212552	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=38/9728, ttl=62 (request in 106)
111	10.371599	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=39/9984, ttl=62 (request in 110)
113	10.412573	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=40/10240, ttl=62 (request in 112)
115	10.452542	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=41/10496, ttl=62 (request in 114)
117	10.492584	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=42/10752, ttl=62 (request in 116)
119	10.532657	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=43/11008, ttl=62 (request in 118)
121	10.578739	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=44/11264, ttl=62 (request in 120)
123	10.619817	193.136.9.240	192.168.73.218	ICMP	70	Echo (ping) reply id=0x0001, seq=45/11520, ttl=62 (request in 122)

O valor do campo TTL é 62. Este permanece constante para todas as mensagens de resposta *ICMP TTL exceeded* enviadas ao host, isto porque encontrou o destino quando enviou o TTL 62.

1.1.3

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



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	AsustekC_52:1f:d6	Broadcast	ARP	42	Who has 192.168.73.254? Tell 192.168.73.218
2	0.000036	Vmware_Id:41:ec	AsustekC_52:1f:d6	ARP	60	192.168.73.254 is at 00:0c:29:1d:41:ec
3	0.004773	192.168.73.218	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=0bee) [Reassembled in #5]
4	0.004779	192.168.73.218	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=0bee) [Reassembled in #5]
5	0.004783	192.168.73.218	193.136.9.240	ICMP	1112	Echo (ping) request id=0x0001, seq=8002/16927, ttl=2 (no response found!)
6	0.006380	193.136.19.254	192.168.73.218	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
7	0.054805	192.168.73.218	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=0bef) [Reassembled in #9]
8	0.054814	192.168.73.218	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=0bef) [Reassembled in #9]
9	0.054818	192.168.73.218	193.136.9.240	ICMP	1112	Echo (ping) request id=0x0001, seq=8003/17183, ttl=3 (reply in 12)
10	0.057345	193.136.9.240	192.168.73.218	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=2869) [Reassembled in #12]
11	0.058157	193.136.9.240	192.168.73.218	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2869) [Reassembled in #12]
12	0.058159	193.136.9.240	192.168.73.218	ICMP	1112	Echo (ping) reply id=0x0001, seq=8003/17183, ttl=62 (request in 9)
13	0.764847	CiscoInc_73:97:11	Spanning-tree-(for-bridges)_00	STP	60	Conf. Root = 4096/731/00:0a:8a:97:74:00 Cost = 3084 Port = 0x0011
14	2.204313	192.168.73.218	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=0bf0) [Reassembled in #16]

Ao aumentar a dimensão do pacote (para 4058 bytes), obriga a sua fragmentação, pois a transmissão na rede é efectuada apenas com pacotes de dimensão inferior (1500 bytes).

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?

```
Frame 3: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
Ethernet II, Src: AsustekC_52:1f:d6 (c8:60:00:52:1f:d6), Dst: Vmware_1d:41:ec (00:0c:29:1d:41:ec)
Internet Protocol Version 4, Src: 192.168.73.218, Dst: 193.136.9.240
0100 .... = Version: 4
... 0101 = Header Length: 20 bytes (5)
Differetiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1500
Identification: 0x0bee (3054)
Flags: 0x01 (More Fragments)
0... .... = Reserved bit: Not set
.0... .... = Don't fragment: Not set
..1. .... = More fragments: Set
Fragment offset: 0
Time to live: 2
Protocol: ICMP (1)
Header checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.73.218
Destination: 193.136.9.240
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Reassembled IPv4 in frame: 5
Data (1480 bytes)
```

O fragment offset e as flags, nomeadamente a flag *more fragments*, indica-nos se ocorreu fragmentação ou não. Como a flag *more fragments* tem o valor 1, o que indica que existem mais fragmentos. Para além disso, o *fragment offset*, estabelece a posição no datagrama que a porção de dados veiculada pelo fragmento, isto é, como o valor do *fragment offset* é 0, e a flag *more fragments* tem valor 1, logo corresponde ao 1º fragmento. O tamanho deste datagrama IP é 1480 bytes, sendo que os restantes 20 bytes do cabeçalho.

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

```
Frame 4: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
Ethernet II, Src: AsustekC_52:1f:d6 (c8:60:00:52:1f:d6), Dst: Vmware_1d:41:ec (00:0c:29:1d:41:ec)
Internet Protocol Version 4, Src: 192.168.73.218, Dst: 193.136.9.240
0100 .... = Version: 4
... 0101 = Header Length: 20 bytes (5)
Differetiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1500
Identification: 0x0bee (3054)
Flags: 0x01 (More Fragments)
0... .... = Reserved bit: Not set
.0... .... = Don't fragment: Not set
..1. .... = More fragments: Set
Fragment offset: 1480
Time to live: 2
Protocol: ICMP (1)
Header checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.73.218
Destination: 193.136.9.240
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Reassembled IPv4 in frame: 5
Data (1480 bytes)
```

Como o *fragment offset* tem o valor de 1480, logo não corresponde ao 1º fragmento, mas sim ao 2º fragmento. Para além disso, como a flag *more fragments* tem valor 1, logo existem mais fragmentos.

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

2	0.000836	Vmware_Id:41:ec	AsustekC_52:1f:d6	ARP	60	192.168.73.254 is at 00:0c:29:1d:41:ec
3	0.004773	192.168.73.218	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=0bee) [Reassembled in #5]
4	0.004779	192.168.73.218	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=0bee) [Reassembled in #5]
5	0.004783	192.168.73.218	193.136.9.240	ICMP	1112	Echo (ping) request id=0x0001, seq=8002/16927, ttl=2 (no response found!)
6	0.006380	193.136.19.254	192.168.73.218	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
7	0.054805	192.168.73.218	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=0bef) [Reassembled in #9]

```

Frame 5: 1112 bytes on wire (8896 bits), 1112 bytes captured (8896 bits) on interface 0
Ethernet II, Src: AsustekC_52:1f:d6 (c8:60:00:52:1f:d6), Dst: Vmware_Id:41:ec (00:0c:29:1d:41:ec)
Internet Protocol Version 4, Src: 192.168.73.218, 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: 1098
    Identification: 0x0bee (3054)
  Flags: 0x00
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
  Fragment offset: 2960
  Time to live: 2
  Protocol: ICMP (1)
  Header checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.73.218
  Destination: 193.136.9.240
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
  [3 IPv4 Fragments (4038 bytes): #3(1480), #4(1480), #5(1078)]
Internet Control Message Protocol

```

Foram criados 3 fragmentos, a partir do datagrama original. O que nos indica que é o último fragmento, é a flag *more fragments*, que no último fragmento tem o valor 0, como neste caso.

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.

```

Frame 3: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
Ethernet II, Src: AsustekC_52:1f:d6 (c8:60:00:52:1f:d6), Dst: Vmware_Id:41:ec (00:0c:29:1d:41:ec)
Internet Protocol Version 4, Src: 192.168.73.218, 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: 0x0bee (3054)
  Flags: 0x01 (More Fragments)
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
  Fragment offset: 0
  Time to live: 2
  Protocol: ICMP (1)
  Header checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.73.218
  Destination: 193.136.9.240
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
  Reassembled IPv4 in frame: 5
Data (1480 bytes)

```

```

Frame 4: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
Ethernet II, Src: AsustekC_52:1f:d6 (c8:60:00:52:1f:d6), Dst: Vmware_Id:41:ec (00:0c:29:1d:41:ec)
Internet Protocol Version 4, Src: 192.168.73.218, 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: 0x0bee (3054)
  Flags: 0x01 (More Fragments)
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
  Fragment offset: 1480
  Time to live: 2
  Protocol: ICMP (1)
  Header checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.73.218
  Destination: 193.136.9.240
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
  Reassembled IPv4 in frame: 5
Data (1480 bytes)

```

```

Frame 5: 1112 bytes on wire (8896 bits), 1112 bytes captured (8896 bits) on interface 0
Ethernet II, Src: AsustekC_52:1f:d6 (c8:60:00:52:1f:d6), Dst: Vmware_Id:41:ec (00:0c:29:1d:41:ec)
Internet Protocol Version 4, Src: 192.168.73.218, 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: 1098
    Identification: 0x0bee (3054)
  Flags: 0x00
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
  Fragment offset: 2960
  Time to live: 2
  Protocol: ICMP (1)
  Header checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.73.218
  Destination: 193.136.9.240
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
  [3 IPv4 Fragments (4038 bytes): #3(1480), #4(1480), #5(1078)]
Internet Control Message Protocol

```

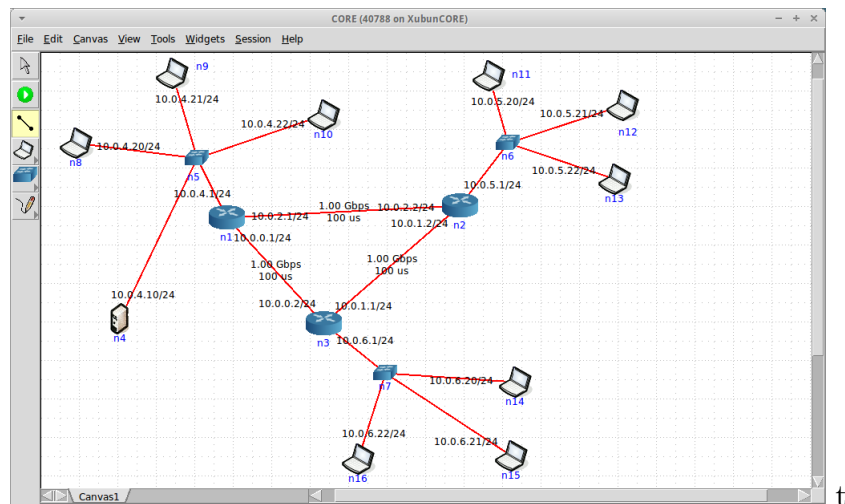
Os campos que mudam no cabeçalho IP entre os diferentes fragmentos, é a flag

more fragments e o *fragment offset*. O *more fragments* indica-nos se existe mais fragmentos ou não, e o *fragment offset* estabelece a posição no datagrama, a porção de dados veiculada pelo fragmento. Desta forma, conseguimos estabelecer a ligação entre os diferentes fragmentos, e reconstruir o datagrama original.

1.2 Parte II

1.2.1

a. Indique que endereços IP e máscaras de rede foram atribuídos pelo CORE a cada equipamento. Se preferir, pode incluir uma mensagem que ilustre de forma clara a topologia e o endereçamento.



Através do printscreen anterior podemos obter os endereços IP e máscaras de rede de cada equipamento.

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

Privados porque estão na gama de valores: 10.0.0.0 a 10.255.255.255.

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

O switch é um equipamento responsável pela interligação de equipamentos, este, não precisa de endereço IP, uma vez que estes registam o end. MAC de cada um dos dispositivos conetados a si, de maneira que, mais tarde, possa fazer um correto redirecionamento.

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).

The image displays three terminal windows stacked vertically, each showing the output of a ping command from a different host to the IP address 10.0.4.10. The windows are titled 'root@n11: /tmp/pycore.40797/n11.conf', 'root@n16: /tmp/pycore.40797/n16.conf', and 'root@n8: /tmp/pycore.40797/n8.conf'. Each window shows 11 successful ping requests with 64 bytes of data, a TTL of 62, and various response times in milliseconds.

```
root@n11: /tmp/pycore.40797/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_req=1 ttl=62 time=1.24 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=62 time=0.291 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=62 time=0.290 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=62 time=0.289 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=62 time=0.293 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=62 time=0.290 ms
64 bytes from 10.0.4.10: icmp_req=7 ttl=62 time=0.288 ms
64 bytes from 10.0.4.10: icmp_req=8 ttl=62 time=0.327 ms
64 bytes from 10.0.4.10: icmp_req=9 ttl=62 time=0.291 ms
64 bytes from 10.0.4.10: icmp_req=10 ttl=62 time=0.292 ms
64 bytes from 10.0.4.10: icmp_req=11 ttl=62 time=0.290 ms

root@n16: /tmp/pycore.40797/n16.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_req=1 ttl=62 time=0.796 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=62 time=0.289 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=62 time=0.299 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=62 time=0.298 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=62 time=0.287 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=62 time=0.292 ms
64 bytes from 10.0.4.10: icmp_req=7 ttl=62 time=0.294 ms
64 bytes from 10.0.4.10: icmp_req=8 ttl=62 time=0.314 ms

root@n8: /tmp/pycore.40797/n8.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_req=1 ttl=64 time=0.061 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=64 time=0.032 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=64 time=0.033 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=64 time=0.034 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=64 time=0.034 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=64 time=0.031 ms
64 bytes from 10.0.4.10: icmp_req=7 ttl=64 time=0.033 ms
64 bytes from 10.0.4.10: icmp_req=8 ttl=64 time=0.032 ms
```

Como podemos verificar pelos printscreens anteriores, facilmente verificamos que existe conectividade IP entre os laptops dos utilizadores e o servidor do departamento A.

1.2.2

a. Execute o comando *netstat -rn* por forma a poder consultar a tabela de encaminhamento unicast (IPv4). Inclua no seu relatório as tabelas de encaminhamento obtidas; interprete as várias entradas de cada tabela. Se necessário, consulte o manual respectivo (*man netstat*).

```
root@n1: /tmp/pycore.40802/n1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.0.1.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
10.0.4.0 0.0.0.0 255.255.255.0 U 0 0 0 eth3
10.0.5.0 10.0.2.2 255.255.255.0 UG 0 0 0 eth1
10.0.6.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
root@n1: /tmp/pycore.40802/n1.conf#
```

```
root@n8: /tmp/pycore.40802/n8.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.4.1 0.0.0.0 UG 0 0 0 eth0
10.0.4.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@n8: /tmp/pycore.40802/n8.conf#
```

b. Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico.

Está a ser usado o encaminhamento dinâmico, uma vez que os routers se adaptam a possíveis alterações na rede em que se encontram.

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.

```
root@n4: /tmp/pycore.44880/n4.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@n4: /tmp/pycore.44880/n4.conf#
```

Ao eliminarmos a rota por defeito, da tabela de encaminhamento do servidor localizado no departamento A, estamos a apagar a possibilidade de o host poder conectar-se com outros sistemas fora da rede do departamento A. Desta forma apenas consegue conectar-se aos laptops da rede do departamento A.

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.

```
root@n4: /tmp/pycore.44880/n4.conf# route add -net 10.0.6.0 netmask 255.255.255.0 gw 10.0.4.1
root@n4: /tmp/pycore.44880/n4.conf# route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.4.1
root@n4: /tmp/pycore.44880/n4.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
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
root@n4: /tmp/pycore.44880/n4.conf#
```

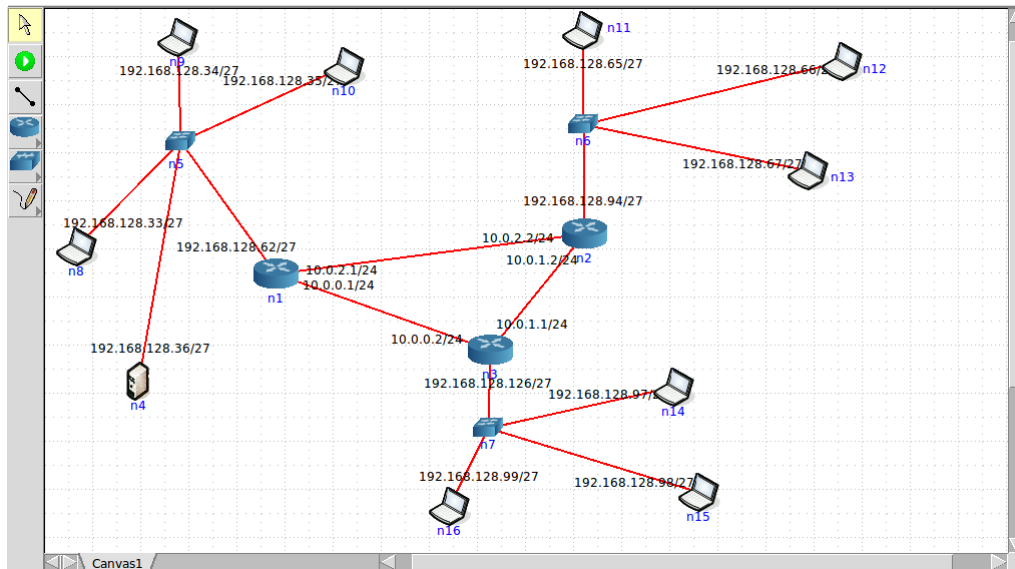
e. Teste a nova política de encaminhamento garantindo que o servidor está novamente acessível, utilizando para o efeito o comando *ping*. Registe a nova tabela de encaminhamento do servidor.

```
root@n4: /tmp/pycore.44880/n4.conf# ping 10.0.6.22
PING 10.0.6.22 (10.0.6.22) 56(84) bytes of data:
64 bytes from 10.0.6.22: icmp_req=1 ttl=62 time=0.115 ms
64 bytes from 10.0.6.22: icmp_req=2 ttl=62 time=0.047 ms
64 bytes from 10.0.6.22: icmp_req=3 ttl=62 time=0.047 ms
64 bytes from 10.0.6.22: icmp_req=4 ttl=62 time=0.055 ms
^C
--- 10.0.6.22 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.047/0.065/0.115/0.028 ms
root@n4: /tmp/pycore.44880/n4.conf# ping 10.0.5.22
PING 10.0.5.22 (10.0.5.22) 56(84) bytes of data:
64 bytes from 10.0.5.22: icmp_req=1 ttl=62 time=0.022 ms
64 bytes from 10.0.5.22: icmp_req=2 ttl=62 time=0.050 ms
64 bytes from 10.0.5.22: icmp_req=3 ttl=62 time=0.049 ms
64 bytes from 10.0.5.22: icmp_req=4 ttl=62 time=0.049 ms
^C
--- 10.0.5.22 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.022/0.042/0.050/0.013 ms
root@n4: /tmp/pycore.44880/n4.conf#
```

1.2.3

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.



Tendo uma rede IP classe C 192.168.128.0/24, podemos construir 3 sub-redes diferentes, com os endereços, 192.168.128.32/27, 192.168.128.64/27 e 192.168.128.96/27 (tendo outras opções de endereços para as sub-redes). Dos 8 bits para host, podemos manipula-los para formar as sub-redes. Desta forma, usamos 3 bits para identificar a sub-rede, e 5 bits, para os diferentes hosts. Desta forma, obtemos uma máscara de 255.255.255.224 (/27). Em cada sub-rede, atribuímos ao router o valor do limite superior da gama de valores válidos numa sub-rede, por exemplo, na sub-rede 192.168.128.32/27, o router tem o endereço, 192.168.128.62/27.

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

A máscara usada é 255.255.255.224 (/27). A máscara deve-se aos bits usados para indentificar a sub-rede, ou seja 3 bits reservados para a sub-rede, logo $24+3= 27$. Outra maneira de ver a máscara, é através dos 3 bits reservados. Os 3 bits (111), em decimal, significa, $128+64+32 = 224$, logo a máscara é 255.255.255.224.

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

Em cada departamento é possível interligar todos os hosts existentes na rede 192.168.128.0/24, ou seja, 10 hosts (1 host e 9 laptops). Isto deve-se porque mesmo alterando os IPs dos vários sistemas, a conectividade é mantida na mesma, porque as tabelas de encaminhamento garantem a relação entre os diferentes endereços IP.

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

```
root@n4: /tmp/pycore.44885/n4.conf
root@n4:/tmp/pycore.44885/n4.conf# ping 192.168.128.65
PING 192.168.128.65 (192.168.128.65) 56(84) bytes of data:
64 bytes from 192.168.128.65: icmp_req=1 ttl=62 time=0.146 ms
64 bytes from 192.168.128.65: icmp_req=2 ttl=62 time=0.062 ms
64 bytes from 192.168.128.65: icmp_req=3 ttl=62 time=0.060 ms
64 bytes from 192.168.128.65: icmp_req=4 ttl=62 time=0.061 ms
^C
--- 192.168.128.65 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.050/0.074/0.146/0.042 ms
root@n4:/tmp/pycore.44885/n4.conf#
```

2. *Conclusão*

Na primeira parte do trabalho, fizemos uma análise ao protocolo de Ipv4. Para isso, foi realizado uma topologia Core, para estudar o comportamento e analisarmos o tráfego ICMP enviado e tráfego ICMP recebido. Para além disso, tivemos em análise casos particulares, como a fragmentação de pacotes IP devido a sua dimensão. Também verificamos o seu comportamento e como identificar se ocorreu ou não uma fragmentação. Na realização da 2º parte deste trabalho prático podemos afirmar que ficamos bastante mais esclarecidos relativamente ao protocolo IPv4, nomeadamente, no endereçamento e encaminhamento IP, e também a sua análise. Com isso, numa topologia Core, vimos o funcionamento do encaminhamento entre 3 redes diferentes, e como manipular os endereços IP, para subnetting, e o seu encaminhamento respectivamente.