



**Universidade do Minho**  
Escola de Engenharia

2020

# REDES DE COMPUTADORES

TRABALHO PRÁTICO 2

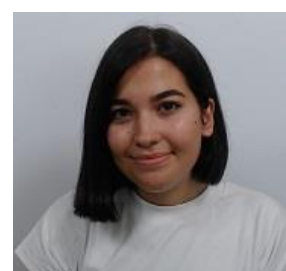
GRUPO 63



A89533 - Ana Carneiro



A89517 - Diogo Araújo



A89518 – Ema Dias

## Conteúdo

1. Protocolo IP (1ª Parte)	3
1.1 Exercício 1	3
1.1.1 Active o Wireshark ou o tcpdump no Cliente1. Numa shell do Cliente1, execute o comando <code>tracert -l</code> para o endereço IP do Servidor1.	3
1.1.2 Registre e analise o tráfego ICMP enviado pelo Cliente1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.	3
1.1.3 Qual deve ser o valor inicial mínimo do campo TTL para alcançar o Servidor1? Verifique na prática que a sua resposta está correta.	4
1.1.4 Calcule o valor médio do tempo de ida-e-volta (Round-Trip Time) obtido?	4
1.2 Exercício 2	4
1.2.1 Qual é o endereço IP da interface ativa do seu computador?	4
1.2.2 Qual é o valor do campo protocolo? O que identifica?	4
1.2.3 Quantos bytes tem o cabeçalho IP(v4)? Quantos bytes tem o campo de dados (payload) do datagrama? Como se calcula o tamanho do payload?	4
1.2.4 O datagrama IP foi fragmentado? Justifique.	5
1.2.5 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.	5
1.2.6 Observa algum padrão nos valores do campo de Identificação do datagrama IP e TTL?	5
1.2.7 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?	6
1.3 Exercício 3	6
1.3.1 Localize a primeira mensagem ICMP. Porque é que houve necessidade de fragmentar o pacote inicial?	6
1.3.2 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?	7
1.3.3 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?	7
1.3.4 Quantos fragmentos foram criados a partir do datagrama original? Como se detecta o último fragmento correspondente ao datagrama original?	8
1.3.5 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.	8
2. Protocolo IP (2ª Parte)	9
2.1 Exercício 1	9

2.1.1	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.	9
2.1.2	Trata-se de endereços públicos ou privados? Porquê?	9
2.1.3	Porque razão não é atribuído um endereço IP aos switches?	9
2.1.4	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).	10
2.1.5	Verifique se existe conectividade IP do router de acesso RISP para o servidor S1.	10
2.2	Exercício 2	11
2.2.1	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).	11
2.2.2	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).	12
2.2.3	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.	12
2.2.4	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.	12
2.2.5	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.	13
2.3	Exercício 3	13
2.3.1	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) atribua endereços às interfaces dos vários sistemas envolvidos. Assuma que todos os endereços de sub-redes são usáveis. Deve justificar as opções usadas.	13
2.3.2	Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP pode interligar em cada departamento? Justifique.	15
2.3.3	Garanta e verifique que conectividade IP entre as várias redes locais da organização MIEI-RC é mantida. Explique como procedeu.	15
	Conclusões	16

# 1. Protocolo IP (1ª Parte)

## 1.1 Exercício 1

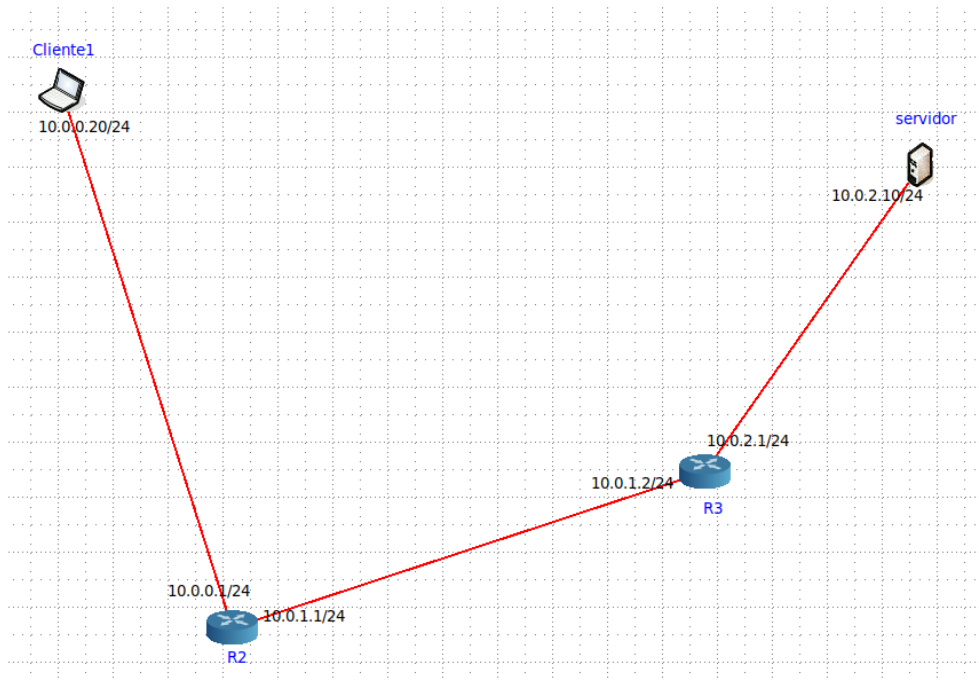


Figura 1: Rede do exercício 1

1.1.1 Active o wireshark ou o tcpdump no Cliente1. Numa shell do Cliente1, execute o comando `tracert -I` para o endereço IP do Servidor1.

```

root@Cliente1: /tmp/pycore.36371/Cliente1.conf
root@Cliente1: /tmp/pycore.36371/Cliente1.conf# traceroute -I 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1)  0.119 ms  0.042 ms  0.030 ms
 2 10.0.1.2 (10.0.1.2)  0.161 ms  0.081 ms  0.047 ms
 3 10.0.2.10 (10.0.2.10)  0.144 ms  0.037 ms  0.033 ms
root@Cliente1: /tmp/pycore.36371/Cliente1.conf#
  
```

Figura 2: Comando `tracert` na shell do Cliente1

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

9	40.005168409	10.0.0.1	224.0.0.5	OSPF	78 Hello Packet
10	40.320193823	fe80::200:ff:feaa:1	ff02::5	OSPF	90 Hello Packet
11	41.041760926	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=1/256, ttl=1 (no response found!)
12	41.041788474	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (time to live exceeded in transit)
13	41.041799371	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=2/512, ttl=1 (no response found!)
14	41.041804608	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (time to live exceeded in transit)
15	41.041808616	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=3/768, ttl=1 (no response found!)
16	41.041812319	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (time to live exceeded in transit)
17	41.041817731	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=4/1024, ttl=2 (no response found!)
18	41.041835433	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (time to live exceeded in transit)
19	41.041840016	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=5/1280, ttl=2 (no response found!)
20	41.041846078	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (time to live exceeded in transit)
21	41.041850374	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=6/1536, ttl=2 (no response found!)
22	41.041856685	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (time to live exceeded in transit)
23	41.041861542	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=7/1792, ttl=3 (reply in 24)
24	41.041903408	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x0028, seq=7/1792, ttl=62 (request in 23)
25	41.041910620	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=8/2048, ttl=3 (reply in 26)
26	41.041922047	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x0028, seq=8/2048, ttl=62 (request in 25)
27	41.041927784	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=9/2304, ttl=3 (reply in 28)
28	41.041939073	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x0028, seq=9/2304, ttl=62 (request in 27)
29	41.041945321	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=10/2560, ttl=4 (reply in 30)
30	41.041956174	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x0028, seq=10/2560, ttl=62 (request in 29)
31	41.041960987	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=11/2816, ttl=4 (reply in 32)
32	41.041971986	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x0028, seq=11/2816, ttl=62 (request in 31)
33	41.041976813	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=12/3072, ttl=4 (reply in 34)
34	41.041988185	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x0028, seq=12/3072, ttl=62 (request in 33)
35	41.041995759	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=13/3328, ttl=5 (reply in 36)
36	41.042008064	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x0028, seq=13/3328, ttl=62 (request in 35)
37	41.042013339	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0028, seq=14/3584, ttl=5 (reply in 38)

Figura 3: Captura de datagramas no Wireshark

Conseguimos ver através da análise de tráfego que os valores TTL vão aumentando ao longo da captura.

Para TTL=1 e TTL=2 conseguimos ver que o ICMP devolve uma mensagem “TTL exceed”, isto é, o datagrama enviado pelo cliente 1 nunca chegou ao destino (servidor 1) e como existem 3 routers nesta rede, quando um pacote com TTL=1 chega ao router 1 este decremente o TTL, ficando TTL=0, o que impede que o datagrama continue o percurso. O mesmo acontece com datagramas com TTL=2 no router 2.

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

O valor mínimo para TTL será 3, como se pode verificar na figura 3. Só a partir da linha 23 é que conseguimos com que o datagrama chegue ao destino, pois o TTL = 3.

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

$$\frac{4,1866 \times 10^{-5} + 1,1427 \times 10^{-5} + 1,1289 \times 10^{-5}}{3} = 2,15 \times 10^{-3} ms$$

## 1.2 Exercício 2

```
> Frame 15: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{5A512AA6-9B6A-4775-A17E-787BCDE7CEA4}, id 0
> Ethernet II, Src: IntelCor_b3:36:02 (7c:2a:31:b3:36:02), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
✓ Internet Protocol Version 4, Src: 172.26.55.252, 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: 0x188f (6287)
  ✓ Flags: 0x00
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    Fragment Offset: 0
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0xf3a6 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.55.252
    Destination Address: 193.136.9.240
  > Internet Control Message Protocol
```

Figura 4: Informação sobre a 1ª mensagem ICMP

### 1.2.1 Qual é o endereço IP da interface ativa do seu computador?

O endereço da source é 172.26.55.252.

### 1.2.2 Qual é o valor do campo protocolo? O que identifica?

O valor do campo protocolo é ICMP (Internet Control Message Protocol), que reporta erros com a transmissão dos datagramas à fonte original.

### 1.2.3 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 tem 20 bytes. O payload tem 36 bytes. O cálculo do payload é feito através da subtração entre a “header length” e o “total length”.

length – header = payload. Neste caso, o length é de 56 bytes.

### 1.2.4 O datagrama IP foi fragmentado? Justifique.

O datagrama não foi fragmentado, já que o “fragment offset” é zero e “more fragments” encontra-se a not set. Isto é, o datagrama encontra-se no início e não existe mais fragmentos à sua frente.

### 1.2.5 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.

15	4.025688	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=18/4608, ttl=255 (reply in 16)
17	4.064695	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=19/4864, ttl=1 (no response found!)
19	4.102771	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=20/5120, ttl=2 (no response found!)
21	4.142737	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=21/5376, ttl=3 (no response found!)
23	4.180846	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=22/5632, ttl=4 (reply in 24)
26	6.526130	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=23/5888, ttl=255 (reply in 27)
29	6.576020	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=24/6144, ttl=1 (no response found!)
31	6.595380	172.26.55.252	52.113.205.223	TLSv1.2	239 Application Data	
32	6.627319	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=25/6400, ttl=2 (no response found!)
34	6.677505	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=26/6656, ttl=3 (no response found!)
37	6.727954	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=27/6912, ttl=4 (reply in 38)
39	7.109184	172.26.55.252	162.159.137.232	TCP	55 49745 → 443 [ACK] Seq=1 Ack=1 Win=512 Len=1 [TCP segment of a reassembled PDU]	
42	9.026652	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=28/7168, ttl=255 (reply in 43)
44	9.076970	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=29/7424, ttl=1 (no response found!)
46	9.127282	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=30/7680, ttl=2 (no response found!)
48	9.178146	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=31/7936, ttl=3 (no response found!)
50	9.229150	172.26.55.252	193.136.9.240	ICMP	70 Echo (ping) request	id=0x0001, seq=32/8192, ttl=4 (reply in 51)
16	4.042546	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply	id=0x0001, seq=18/4608, ttl=61 (request in 15)
24	4.192292	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply	id=0x0001, seq=22/5632, ttl=61 (request in 23)
27	6.530315	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply	id=0x0001, seq=23/5888, ttl=61 (request in 26)
38	6.736630	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply	id=0x0001, seq=27/6912, ttl=61 (request in 37)
43	9.034145	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply	id=0x0001, seq=28/7168, ttl=61 (request in 42)
51	9.231209	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply	id=0x0001, seq=32/8192, ttl=61 (request in 50)
2	0.051498	40.67.251.132	172.26.55.252	TLSv1.2	227 Application Data	
28	6.575769	52.113.205.223	172.26.55.252	TLSv1.2	393 Application Data	

Figura 5: Mensagens enviadas na rede por ordem da Source

Os campos que variam são o Identification, TTL e o header checksum.

### 1.2.6 Observa algum padrão nos valores do campo de Identificação do datagrama IP e TTL?

Em relação ao campo identification, este é sempre inicializado com 0x18(...). Já em relação ao campo TTL, este vai aumentando ao longo da captura, contudo como inicialmente não se sabe qual é o TTL que possibilita que o datagrama chegue ao destino, o traceroute cria um datagrama com TTL = 255, para ter a certeza que atinge o destino. De seguida, é iniciado o TTL=1 e o traceroute vai aumentando o TTL até os datagramas conseguirem chegar ao destino.

### 1.2.7 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?

49	9.180336	172.16.115.252	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
47	9.129018	172.16.2.1	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
45	9.081098	172.26.254.254	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
43	9.034145	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply id=0x0001, seq=28/7168, ttl=61 (request in 42)
41	8.779361	13.107.136.254	172.26.55.252	TCP	60 443 → 49761 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40	7.140954	162.159.137.232	172.26.55.252	TCP	66 443 → 49745 [ACK] Seq=1 Ack=2 Win=67 Len=0 SLE=1 SRE=2
38	6.736630	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply id=0x0001, seq=27/6912, ttl=61 (request in 37)
36	6.687310	52.113.205.223	172.26.55.252	TCP	60 443 → 49762 [ACK] Seq=340 Ack=186 Win=2048 Len=0
35	6.680271	172.16.115.252	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
33	6.629424	172.16.2.1	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
30	6.578066	172.26.254.254	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
28	6.575769	52.113.205.223	172.26.55.252	TLSv1.2	393 Application Data
27	6.530315	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply id=0x0001, seq=23/5888, ttl=61 (request in 26)
25	6.234533	13.107.246.254	172.26.55.252	TCP	60 443 → 49752 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
24	4.192292	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply id=0x0001, seq=22/5632, ttl=61 (request in 23)
22	4.162055	172.16.115.252	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
20	4.114885	172.16.2.1	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
18	4.067901	172.26.254.254	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
16	4.042546	193.136.9.240	172.26.55.252	ICMP	70 Echo (ping) reply id=0x0001, seq=18/4608, ttl=61 (request in 15)
13	2.839637	93.184.220.29	172.26.55.252	TCP	60 80 → 49753 [ACK] Seq=1 Ack=1 Win=131 Len=0

Figura 6: Respostas ICMP TTL exceeded

O TTL é igual a 1 para todas as mensagens ICMP TTL exceeded mantendo se constante nesse valor, já que com esse valor o datagrama fica impedido de chegar ao destino e por isso retorna uma mensagem de erro.

### 1.3 Exercício 3

6	1.530943	52.113.199.187	172.26.55.252	TCP	66 443 → 49838 [ACK] Seq=1 Ack=2 Win=2050 Len=0 SLE=1 SRE=2
7	6.837923	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=1c3b) [Reassembled in #9]
8	6.837923	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1c3b) [Reassembled in #9]
9	6.837923	172.26.55.252	193.136.9.240	ICMP	317 Echo (ping) request id=0x0001, seq=958/48643, ttl=255 (reply in 12)
10	6.847832	193.136.9.240	172.26.55.252	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ab15) [Reassembled in #12]
11	6.855989	193.136.9.240	172.26.55.252	IPv4	317 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ab15) [Reassembled in #12]
12	6.855989	193.136.9.240	172.26.55.252	ICMP	1514 Echo (ping) reply id=0x0001, seq=958/48643, ttl=61 (request in 9)
13	6.876900	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=1c3c) [Reassembled in #15]
14	6.876900	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1c3c) [Reassembled in #15]
15	6.876900	172.26.55.252	193.136.9.240	ICMP	317 Echo (ping) request id=0x0001, seq=959/48899, ttl=1 (no response found!)
16	6.880000	172.26.254.254	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
17	6.915326	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=1c3d) [Reassembled in #19]
18	6.915326	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1c3d) [Reassembled in #19]
19	6.915326	172.26.55.252	193.136.9.240	ICMP	317 Echo (ping) request id=0x0001, seq=960/49155, ttl=2 (no response found!)
20	6.922358	172.16.2.1	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
21	6.955217	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=1c3e) [Reassembled in #23]
22	6.955217	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1c3e) [Reassembled in #23]
23	6.955217	172.26.55.252	193.136.9.240	ICMP	317 Echo (ping) request id=0x0001, seq=961/49411, ttl=3 (no response found!)
24	6.964438	172.16.115.252	172.26.55.252	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
25	6.993528	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=1c3f) [Reassembled in #27]
26	6.993528	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1c3f) [Reassembled in #27]
27	6.993528	172.26.55.252	193.136.9.240	ICMP	317 Echo (ping) request id=0x0001, seq=962/49667, ttl=4 (reply in 30)
28	7.002097	193.136.9.240	172.26.55.252	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ab44) [Reassembled in #30]
29	7.011174	193.136.9.240	172.26.55.252	IPv4	317 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=ab44) [Reassembled in #30]
30	7.011174	193.136.9.240	172.26.55.252	ICMP	1514 Echo (ping) reply id=0x0001, seq=962/49667, ttl=61 (request in 27)
31	9.338129	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=1c40) [Reassembled in #33]
32	9.338129	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1c40) [Reassembled in #33]
33	9.338129	172.26.55.252	193.136.9.240	ICMP	317 Echo (ping) request id=0x0001, seq=963/49923, ttl=255 (reply in 36)
34	9.360091	193.136.9.240	172.26.55.252	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=abef) [Reassembled in #36]
35	9.360091	193.136.9.240	172.26.55.252	IPv4	317 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=abef) [Reassembled in #36]
36	9.360091	193.136.9.240	172.26.55.252	ICMP	1514 Echo (ping) reply id=0x0001, seq=963/49923, ttl=61 (request in 33)
37	9.388665	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=1c41) [Reassembled in #39]
38	9.388665	172.26.55.252	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1c41) [Reassembled in #39]
39	9.388665	172.26.55.252	193.136.9.240	ICMP	317 Echo (ping) request id=0x0001, seq=964/50179, ttl=1 (no response found!)

Figura 7: Datagramas enviados na rede com tamanho 3263

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

A primeira mensagem ICMP é a 9. Como o tamanho dos datagramas (3263 bytes) são maiores que o MTU do Wi-Fi (Maximum Transmission Unit) é necessário haver fragmentação em 3 partes (tamanhos = 1500 + 1500 + 303 – está dentro do nº bytes necessário para transmitir o datagrama).



### 1.3.2 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 7: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{5A512AA6-9B6A-4775-A17E-787BCDE7CEA4}, id 0
> Ethernet II, Src: IntelCor_b3:36:02 (7c:2a:31:b3:36:02), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.55.252, 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: 0x1c3b (7227)
  > Flags: 0x20, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    Fragment Offset: 0
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0xca56 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.55.252
    Destination Address: 193.136.9.240
    [Reassembled IPv4 in frame: 9]
> Data (1480 bytes)
```

Figura 8: Informação sobre o 1º fragmento do datagrama

A informação que indica se o datagrama foi fragmentado é o “fragment offset” e o “more fragments”. Se o offset for diferente de 0 e o “more fragments” estiver em Set, então o datagrama está fragmentado, o que se verifica. Trata-se do 1º fragmento, pois o “fragment offset” encontra-se a 0. O tamanho deste datagrama é 1500 bytes, dado pelo “total length”.

### 1.3.3 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?

```
> Frame 8: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{5A512AA6-9B6A-4775-A17E-787BCDE7CEA4}, id 0
> Ethernet II, Src: IntelCor_b3:36:02 (7c:2a:31:b3:36:02), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.55.252, 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: 0x1c3b (7227)
  > Flags: 0x20, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    Fragment Offset: 1480
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0xc99d [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.55.252
    Destination Address: 193.136.9.240
    [Reassembled IPv4 in frame: 9]
> Data (1480 bytes)
```

Figura 9: Informação sobre o 2º fragmento do datagrama

O “fragment offset” mostra que não se trata do 1º fragmento, pois o offset é diferente de 0 (se fosse zero estaríamos no 1º fragmento). Há mais fragmentos, pois o campo “more fragments” está com bit a 1, ou seja, está a set.



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

```

Frame 9: 317 bytes on wire (2536 bits), 317 bytes captured (2536 bits) on interface \Device\NPF_{5A512AA6-9B6A-4775-A17E-787BCDE7CEA4}, id 0
Ethernet II, Src: IntelCor_b3:36:02 (7c:2a:31:b3:36:02), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.55.252, 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: 303
    Identification: 0x1c3b (7227)
  < Flags: 0x01
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    Fragment Offset: 2960
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0xed91 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.55.252
    Destination Address: 193.136.9.240
  > [ 3 IPv4 Fragments (3243 bytes): #7(1480), #8(1480), #9(283)]
Internet Control Message Protocol

```

**Figura 10: Informação sobre o 3º (último) fragmento do datagrama**

Foram criados 3 fragmentos. O último datagrama consegue-se detetar através do “fragment offset” e do “more fragments”. No 1º vai ter um nº diferente de 0, no 2º como é o último fragmento então não haverá mais fragmentos por isso o bit estará a 0 (not set).

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

Os campos que mudaram no cabeçalho entre fragmentos são o “fragment offset”, “header checksum”, “more fragments”. Iniciando no 1º datagrama (offset = 0, more=not set), este vai transportar 1480 bytes de informação assim, no próximo fragmento é necessário transportar os restantes dados a partir do bit 1480 (logo, offset=1480, more=not set). No último fragmento, como também foram transportados mais 1480 bytes de informação então o offset será 2960 (1480+1480) e como já não há mais fragmentos, então o “more fragments” estará a not set.

## 2. Protocolo IP (2ª Parte)

### 2.1 Exercício 1

2.1.1 Indique que endereços IP e máscaras de rede foram atribuídos pelo CORE a cada equipamento. Para simplificar, pode incluir uma imagem que ilustre de forma clara a topologia definida e o endereçamento usado.

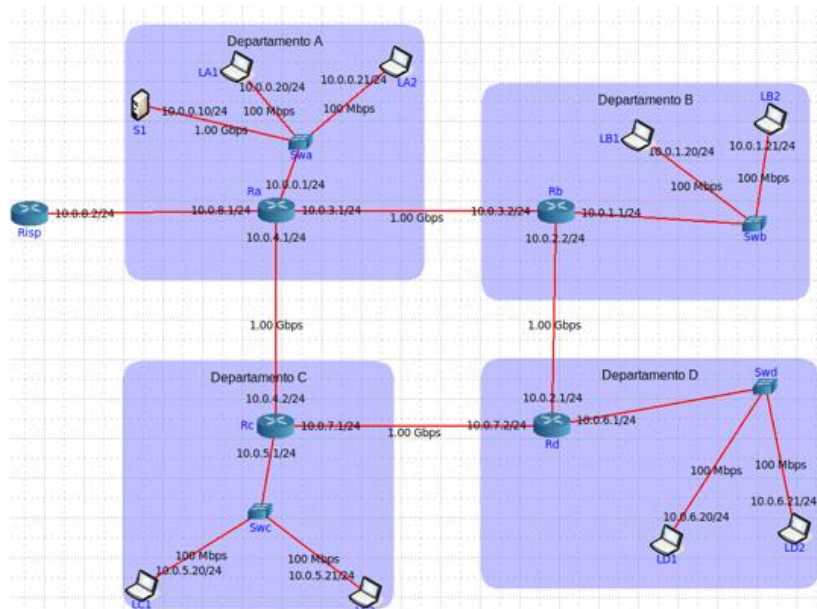


Figura 11: Topologia de rede

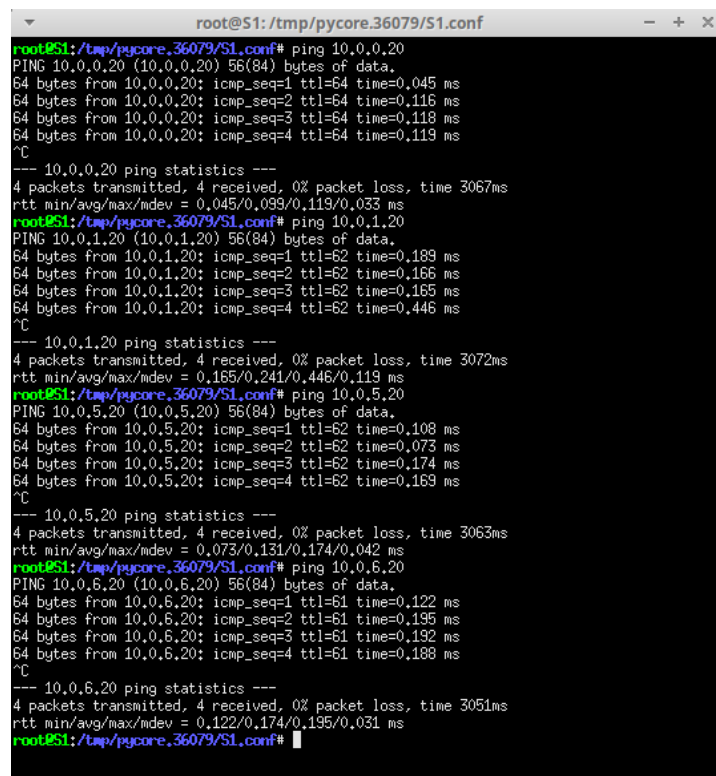
2.1.2 Trata-se de endereços públicos ou privados? Porquê?

Os endereços descritos na topologia de rede tratam-se de endereços privados pois utilizam como prefixo um dos blocos reservados a endereços privados (estabelecidos pelo IANA na norma RFC 1918). Os endereços pertencem ao intervalo de 10.0.0.0 a 10.255.255.255 (10.0.0.0/8).

2.1.3 Porque razão não é atribuído um endereço IP aos switches?

Os switches não têm um endereço IP atribuído pois só implementam a camada física e de ligação lógica da pilha protocolar (endereços IP são implementados na camada de rede, que é um nível acima).

- 2.1.4 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).

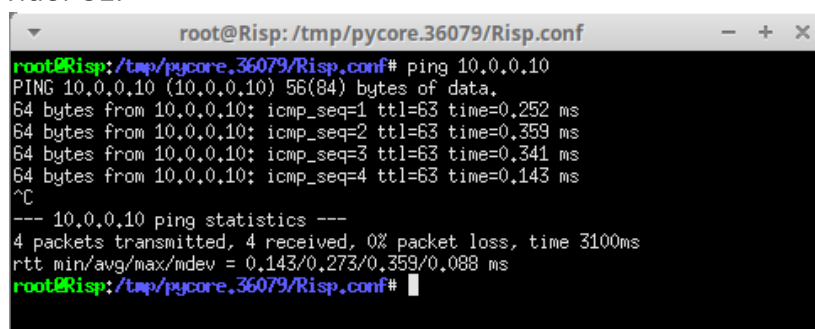


```
root@S1: /tmp/pycore.36079/S1.conf# ping 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_seq=1 ttl=64 time=0.045 ms
64 bytes from 10.0.0.20: icmp_seq=2 ttl=64 time=0.116 ms
64 bytes from 10.0.0.20: icmp_seq=3 ttl=64 time=0.118 ms
64 bytes from 10.0.0.20: icmp_seq=4 ttl=64 time=0.113 ms
^C
--- 10.0.0.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3067ms
rtt min/avg/max/mdev = 0.045/0.099/0.119/0.033 ms
root@S1: /tmp/pycore.36079/S1.conf# ping 10.0.1.20
PING 10.0.1.20 (10.0.1.20) 56(84) bytes of data.
64 bytes from 10.0.1.20: icmp_seq=1 ttl=62 time=0.189 ms
64 bytes from 10.0.1.20: icmp_seq=2 ttl=62 time=0.166 ms
64 bytes from 10.0.1.20: icmp_seq=3 ttl=62 time=0.165 ms
64 bytes from 10.0.1.20: icmp_seq=4 ttl=62 time=0.446 ms
^C
--- 10.0.1.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3072ms
rtt min/avg/max/mdev = 0.165/0.241/0.446/0.119 ms
root@S1: /tmp/pycore.36079/S1.conf# ping 10.0.5.20
PING 10.0.5.20 (10.0.5.20) 56(84) bytes of data.
64 bytes from 10.0.5.20: icmp_seq=1 ttl=62 time=0.108 ms
64 bytes from 10.0.5.20: icmp_seq=2 ttl=62 time=0.073 ms
64 bytes from 10.0.5.20: icmp_seq=3 ttl=62 time=0.174 ms
64 bytes from 10.0.5.20: icmp_seq=4 ttl=62 time=0.169 ms
^C
--- 10.0.5.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3063ms
rtt min/avg/max/mdev = 0.073/0.131/0.174/0.042 ms
root@S1: /tmp/pycore.36079/S1.conf# ping 10.0.6.20
PING 10.0.6.20 (10.0.6.20) 56(84) bytes of data.
64 bytes from 10.0.6.20: icmp_seq=1 ttl=61 time=0.122 ms
64 bytes from 10.0.6.20: icmp_seq=2 ttl=61 time=0.195 ms
64 bytes from 10.0.6.20: icmp_seq=3 ttl=61 time=0.192 ms
64 bytes from 10.0.6.20: icmp_seq=4 ttl=61 time=0.188 ms
^C
--- 10.0.6.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.122/0.174/0.195/0.031 ms
root@S1: /tmp/pycore.36079/S1.conf#
```

Figura 12: Conectividade IP entre o servidor e os departamentos

Pela interpretação da figura acima, concluímos que há conectividade IP do servidor S1 (do departamento A) para laptops de todos os departamentos, visível na transferência e recepção de pacotes entre S1 e os respectivos laptops.

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



```
root@Risp: /tmp/pycore.36079/Risp.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=63 time=0.252 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=63 time=0.359 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=63 time=0.341 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=63 time=0.143 ms
^C
--- 10.0.0.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3100ms
rtt min/avg/max/mdev = 0.143/0.273/0.359/0.088 ms
root@Risp: /tmp/pycore.36079/Risp.conf#
```

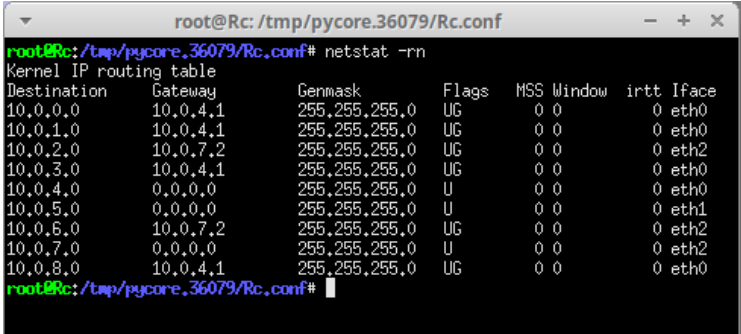
Figura 13: Conectividade IP entre o servidor e o RISP

Pela interpretação da figura acima, concluímos que há conectividade IP do router de acesso (Risp) para o servidor S1 (do departamento A).

## 2.2 Exercício 2

2.2.1 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`).

Podemos ler uma linha da tabela de encaminhamento abstratamente como: Um pacote/datagrama destinado á rede “Destination” será entregue na interface com endereço “Gateway” saindo pela interface local “Iface”. Cada linha possui ainda a máscara utilizada que, neste caso, será sempre 255.255.255.0 (máscara /24).



```

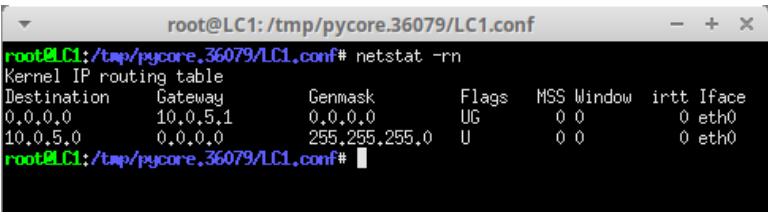
root@RC: /tmp/pycore.36079/Rc.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt  Iface
10.0.0.0         10.0.4.1        255.255.255.0   UG      0 0        0     eth0
10.0.1.0         10.0.4.1        255.255.255.0   UG      0 0        0     eth0
10.0.2.0         10.0.7.2        255.255.255.0   UG      0 0        0     eth2
10.0.3.0         10.0.4.1        255.255.255.0   UG      0 0        0     eth0
10.0.4.0         0.0.0.0         255.255.255.0   U       0 0        0     eth0
10.0.5.0         0.0.0.0         255.255.255.0   U       0 0        0     eth1
10.0.6.0         10.0.7.2        255.255.255.0   UG      0 0        0     eth2
10.0.7.0         0.0.0.0         255.255.255.0   U       0 0        0     eth2
10.0.8.0         10.0.4.1        255.255.255.0   UG      0 0        0     eth0

```

Figura 14: Tabela de encaminhamento do router RC

Analisando as entradas da tabela de encaminhamento do router do departamento C:

- Datagramas destinados às redes 10.0.0.0, 10.0.1.0, 10.0.3.0 e 10.0.8.0 serão entregues na interface com endereço IP 10.0.4.1 saindo pela interface eth0 do router do departamento C;
- Datagramas destinados às redes 10.0.2.0 e 10.0.6.0 serão entregues na interface com endereço IP 10.0.7.2 saindo pela interface eth2 do router do departamento C;
- Datagramas destinados às redes 10.0.4.0, 10.0.5.0 e 10.0.7.0 (redes diretamente ligadas ao router do departamento C) saem pelas interfaces eth0, eth1 e eth2 do router do departamento C, respetivamente.



```

root@LC1: /tmp/pycore.36079/LC1.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt  Iface
0.0.0.0          10.0.5.1        0.0.0.0         UG      0 0        0     eth0
10.0.5.0         0.0.0.0         255.255.255.0   U       0 0        0     eth0

```

Figura 15: Tabela de encaminhamento do LC1

Analisando as entradas da tabela de encaminhamento de um laptop do departamento C:

Caso o tráfego não seja para um host da rede 10.0.5.0, os pacotes são redirecionados, por defeito, para a interface com endereço IP 10.0.5.1, ou seja, para o

router de acesso do departamento C. Tráfego local (para a rede 10.0.5.0) não necessita de ir ao router de acesso.

- 2.2.2 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`).

```

root@Ra: /tmp/pycore.36079/Ra.conf# ps -ax
PID TTY STAT TIME COMMAND
  1 ?    S      0:00 /usr/local/bin/vncnode -v -c /tmp/pycore.36079/Ra -1 /
 67 ?    Ss     0:00 /usr/sbin/zebra -d
 73 ?    Ss     0:00 /usr/sbin/ospf6d -d
 77 ?    Ss     0:00 /usr/sbin/ospf6d -d
 85 pts/2 Ss     0:00 /bin/bash
 93 pts/2 R+     0:00 ps -ax
root@Ra: /tmp/pycore.36079/Ra.conf#

```

Figura 16: Processos a decorrer no sistema RA (router)

É usado encaminhamento dinâmico pois é utilizado um protocolo responsável pelo encaminhamento dinâmico (protocolo OSPFD).

- 2.2.3 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@S1: /tmp/pycore.36079/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.0.1 0.0.0.0 UG 0 0 0 eth0
10.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1: /tmp/pycore.36079/S1.conf# route delete default
root@S1: /tmp/pycore.36079/S1.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
root@S1: /tmp/pycore.36079/S1.conf#

```

Figura 17: Tabela de encaminhamento do S1 antes e após a remoção da rota por defeito

Após a remoção da rota por defeito (default ou 0.0.0.0), o servidor S1 do departamento A perde conectividade com todos os outros hosts que não pertencem à rede local, ou seja, perde conectividade com os departamentos B, C e D. Isto porque na nova tabela de encaminhamento apenas está definida a rota para a rede local, ou seja, para o departamento A. Assim, nenhum host de outros departamento(B, C e D) consegue receber dados de S1.

- 2.2.4 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@S1: /tmp/pycore.36105/S1.conf# route add default gw 10.0.0.1
root@S1: /tmp/pycore.36105/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.0.1 0.0.0.0 UG 0 0 0 eth0
10.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1: /tmp/pycore.36105/S1.conf#

```

Figura 18: Tabela de encaminhamento do S1 após adicionar a rota estática

2.2.5 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@S1:/tmp/pycore.36105/S1.conf# route add default gw 10.0.0.1
root@S1:/tmp/pycore.36105/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.0.1 0.0.0.0 UG 0 0 0 eth0
10.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
```

Figura 19: Tabela de encaminhamento do S1 após adicionar a rota estática

```
root@S1:/tmp/pycore.39805/S1.conf# ping 10.0.1.21
PING 10.0.1.21 (10.0.1.21) 56(84) bytes of data:
64 bytes from 10.0.1.21: icmp_seq=1 ttl=62 time=0.874 ms
64 bytes from 10.0.1.21: icmp_seq=2 ttl=62 time=0.413 ms
64 bytes from 10.0.1.21: icmp_seq=3 ttl=62 time=0.267 ms
64 bytes from 10.0.1.21: icmp_seq=4 ttl=62 time=0.429 ms
^C
--- 10.0.1.21 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3039ms
rtt min/avg/max/mdev = 0.267/0.495/0.874/0.228 ms
root@S1:/tmp/pycore.39805/S1.conf# ping 10.0.6.21
PING 10.0.6.21 (10.0.6.21) 56(84) bytes of data:
64 bytes from 10.0.6.21: icmp_seq=1 ttl=61 time=0.427 ms
64 bytes from 10.0.6.21: icmp_seq=2 ttl=61 time=0.416 ms
64 bytes from 10.0.6.21: icmp_seq=3 ttl=61 time=0.275 ms
64 bytes from 10.0.6.21: icmp_seq=4 ttl=61 time=0.432 ms
^C
--- 10.0.6.21 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3065ms
rtt min/avg/max/mdev = 0.275/0.387/0.432/0.068 ms
root@S1:/tmp/pycore.39805/S1.conf# ping 10.0.5.21
PING 10.0.5.21 (10.0.5.21) 56(84) bytes of data:
64 bytes from 10.0.5.21: icmp_seq=1 ttl=62 time=0.361 ms
64 bytes from 10.0.5.21: icmp_seq=2 ttl=62 time=0.245 ms
64 bytes from 10.0.5.21: icmp_seq=3 ttl=62 time=0.253 ms
64 bytes from 10.0.5.21: icmp_seq=4 ttl=62 time=0.246 ms
^C
--- 10.0.5.21 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3084ms
rtt min/avg/max/mdev = 0.245/0.276/0.361/0.050 ms
```

Figura 20: Conectividade IP entre o S1 e os vários departamentos

### 2.3 Exercício 3

2.3.1 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.

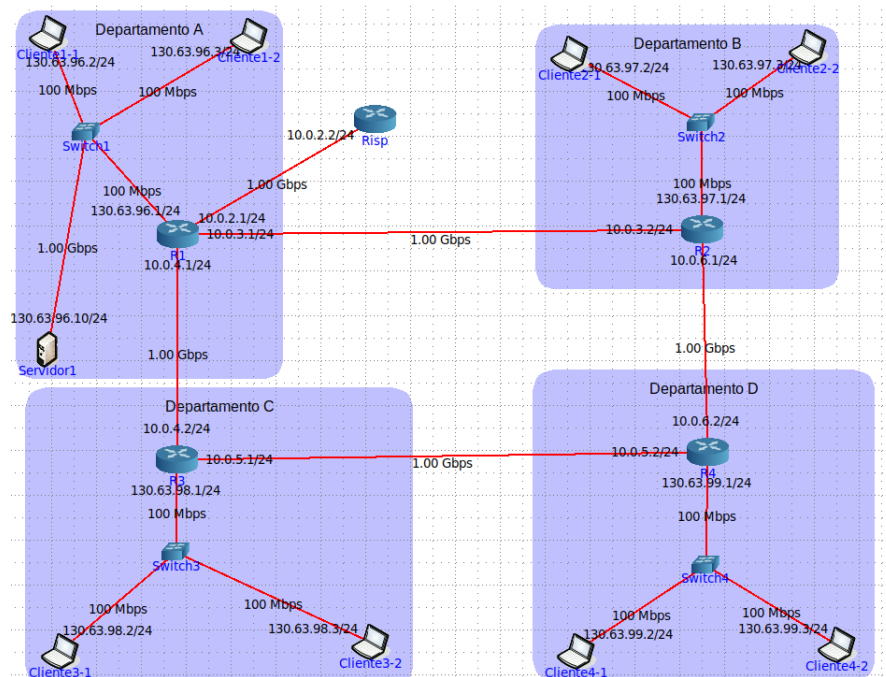


Figura 21: Topologia de rede

No nosso caso,  $XX=63$ , pelo que o IP da rede é 130.63.96.0/19.

Iremos usar 5 bits para definir as subredes (podemos definir  $2^5=32$  subredes, sendo possível aumentar o número de departamentos da nossa topologia posteriormente, se assim o desejarmos).

Iremos usar 8 bits para definir os hosts de cada subrede (podemos definir  $2^8=254$  hosts, sendo possível aumentar o número de hosts de cada departamento posteriormente, se assim o desejarmos).

Dos 4 octetos do endereço IP iremos utilizar assim os 3 primeiros octetos (24 bits) para definir a rede e o último octeto (8 bits) para definir os hosts. Utilizaremos a máscara 255.255.255.0 (/24).

Assim,

Para o departamento A:

-Cliente1-1: 130.63.96.2/24

-Cliente1-2: 130.63.96.3/24

-Servidor1: 130.63.96.10/24

-Interface do router de acesso do departamento A ligado á sub-rede A: 130.63.96.1/24

Para o departamento B:

-Cliente2-1: 130.63.97.2/24

-Cliente2-2: 130.63.97.3/24

-Interface do router de acesso do departamento B ligado á sub-rede B: 130.63.97.1/24

Para o departamento C:

-Cliente3-1: 130.63.98.2/24

-Cliente3-2: 130.63.98.3/24

-Interface do router de acesso do departamento C ligado á sub-rede C: 130.63.98.1/24

Para o departamento D:

-Cliente4-1: 130.63.99.2/24

-Cliente4-2: 130.63.99.3/24

-Interface do router de acesso do departamento D ligado á sub-rede D: 130.63.99.1/24



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

A máscara de rede que utilizamos foi um /24 (255.255.255.0). É possível interligar  $2^8 - 2 = 254$  hosts (utilizamos 8 bits para representar os hosts) em cada departamento. Os 2 endereços que subtraímos são o endereço de broadcast e endereço universal de rede (endereços reservados).

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

```

root@R1:/tmp/pycore.41921/R1.conf# ping -c 5 130.63.96,2
PING 130.63.96,2 (130.63.96,2) 56(84) bytes of data.
64 bytes from 130.63.96,2: icmp_seq=1 ttl=64 time=0.171 ms
64 bytes from 130.63.96,2: icmp_seq=2 ttl=64 time=0.027 ms
64 bytes from 130.63.96,2: icmp_seq=3 ttl=64 time=0.108 ms
64 bytes from 130.63.96,2: icmp_seq=4 ttl=64 time=0.105 ms
64 bytes from 130.63.96,2: icmp_seq=5 ttl=64 time=0.107 ms

--- 130.63.96,2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4094ms
rtt min/avg/max/ndev = 0.027/0.103/0.171/0.047 ms
root@R1:/tmp/pycore.41921/R1.conf# ping -c 5 130.63.98,2
PING 130.63.98,2 (130.63.98,2) 56(84) bytes of data.
64 bytes from 130.63.98,2: icmp_seq=1 ttl=63 time=0.284 ms
64 bytes from 130.63.98,2: icmp_seq=2 ttl=63 time=0.131 ms
64 bytes from 130.63.98,2: icmp_seq=3 ttl=63 time=0.137 ms
64 bytes from 130.63.98,2: icmp_seq=4 ttl=63 time=0.153 ms
64 bytes from 130.63.98,2: icmp_seq=5 ttl=63 time=0.155 ms

--- 130.63.98,2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4100ms
rtt min/avg/max/ndev = 0.131/0.172/0.284/0.056 ms
root@R1:/tmp/pycore.41921/R1.conf# ping -c 5 130.63.99,2
PING 130.63.99,2 (130.63.99,2) 56(84) bytes of data.
64 bytes from 130.63.99,2: icmp_seq=1 ttl=62 time=0.126 ms
64 bytes from 130.63.99,2: icmp_seq=2 ttl=62 time=0.187 ms
64 bytes from 130.63.99,2: icmp_seq=3 ttl=62 time=0.160 ms
64 bytes from 130.63.99,2: icmp_seq=4 ttl=62 time=0.185 ms
64 bytes from 130.63.99,2: icmp_seq=5 ttl=62 time=0.336 ms

--- 130.63.99,2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4103ms
rtt min/avg/max/ndev = 0.126/0.198/0.336/0.074 ms
root@R1:/tmp/pycore.41921/R1.conf# ping -c 5 130.63.97,2
PING 130.63.97,2 (130.63.97,2) 56(84) bytes of data.
64 bytes from 130.63.97,2: icmp_seq=1 ttl=63 time=0.184 ms
64 bytes from 130.63.97,2: icmp_seq=2 ttl=63 time=0.160 ms
64 bytes from 130.63.97,2: icmp_seq=3 ttl=63 time=0.153 ms
64 bytes from 130.63.97,2: icmp_seq=4 ttl=63 time=0.089 ms
64 bytes from 130.63.97,2: icmp_seq=5 ttl=63 time=0.137 ms

--- 130.63.97,2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4093ms
rtt min/avg/max/ndev = 0.089/0.144/0.184/0.034 ms
root@R1:/tmp/pycore.41921/R1.conf#

```

Figura 22: Conectividade IP entre R1 e os departamentos

```

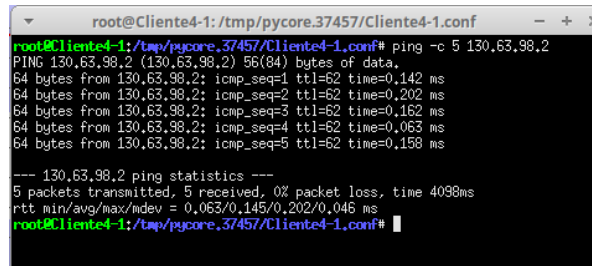
root@Cliente2-1:/tmp/pycore.37457/Cliente2-1.conf# ping -c 5 130.63.98,2
PING 130.63.98,2 (130.63.98,2) 56(84) bytes of data.
64 bytes from 130.63.98,2: icmp_seq=1 ttl=61 time=0.306 ms
64 bytes from 130.63.98,2: icmp_seq=2 ttl=61 time=0.179 ms
64 bytes from 130.63.98,2: icmp_seq=3 ttl=61 time=0.214 ms
64 bytes from 130.63.98,2: icmp_seq=4 ttl=61 time=0.188 ms
64 bytes from 130.63.98,2: icmp_seq=5 ttl=61 time=0.210 ms

--- 130.63.98,2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4139ms
rtt min/avg/max/ndev = 0.179/0.219/0.306/0.047 ms
root@Cliente2-1:/tmp/pycore.37457/Cliente2-1.conf# ping -c 5 130.63.99,2
PING 130.63.99,2 (130.63.99,2) 56(84) bytes of data.
64 bytes from 130.63.99,2: icmp_seq=1 ttl=62 time=0.087 ms
64 bytes from 130.63.99,2: icmp_seq=2 ttl=62 time=0.087 ms
64 bytes from 130.63.99,2: icmp_seq=3 ttl=62 time=0.157 ms
64 bytes from 130.63.99,2: icmp_seq=4 ttl=62 time=0.148 ms
64 bytes from 130.63.99,2: icmp_seq=5 ttl=62 time=0.346 ms

--- 130.63.99,2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4116ms
rtt min/avg/max/ndev = 0.087/0.165/0.346/0.095 ms
root@Cliente2-1:/tmp/pycore.37457/Cliente2-1.conf#

```

Figura 23: Conectividade IP entre os departamentos B, C e D



```
root@Cliente4-1: /tmp/pycore.37457/Cliente4-1.conf
root@Cliente4-1: /tmp/pycore.37457/Cliente4-1.conf# ping -c 5 130.63.98.2
PING 130.63.98.2 (130.63.98.2) 56(84) bytes of data:
64 bytes from 130.63.98.2: icmp_seq=1 ttl=62 time=0.142 ms
64 bytes from 130.63.98.2: icmp_seq=2 ttl=62 time=0.202 ms
64 bytes from 130.63.98.2: icmp_seq=3 ttl=62 time=0.162 ms
64 bytes from 130.63.98.2: icmp_seq=4 ttl=62 time=0.063 ms
64 bytes from 130.63.98.2: icmp_seq=5 ttl=62 time=0.158 ms

--- 130.63.98.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4098ms
rtt min/avg/max/mdev = 0.063/0.145/0.202/0.046 ms
root@Cliente4-1: /tmp/pycore.37457/Cliente4-1.conf#
```

Figura 24: Conectividade IP entre os departamentos C e D

## Conclusões

A realização deste trabalho permitiu-nos uma aplicação prática dos conceitos que aprendemos ao longo das aulas teóricas, seja da temática de Protocolo IPv4 como do Encaminhamento IP e Endereçamento. Desta maneira, na 1ª parte tivemos oportunidade de consolidar os conteúdos relativos a fragmentação e datagramas, onde para tal desenvolvemos uma topologia CORE. Assim, conseguimos aprofundar os conceitos de TTL, de Round-Trip Time, e até mesmo de payload e cabeçalho IP, utilizando como ferramenta o wireshark.

Já na 2ª parte, podemos nos deparar com análise de tabelas de encaminhamento, ou seja, com o endereçamento, de uma organização denominada de MIEI-RC, constituída com 4 departamentos. Desta forma, utilizamos com frequência comandos como o ping, e comandos de inserção e remoção de rotas. Para além disso, definimos tipos de encaminhamento, estático ou dinâmico e endereçamentos públicos ou privados. Nesta segunda parte, também se analisou o conceito de sub-redes através da criação de um novo esquema de endereçamento numa topologia de rede.

Resumindo, através deste trabalho pratico conseguimos consolidar todos os conceitos e nomenclaturas importantes ao capítulo do IPV4 e network layer.