

# eTP2: Protocolo IPv4 (Parte I)

Filipa Correia Parente, José André Martins Pereira, Ricardo André Gomes  
Petronilho

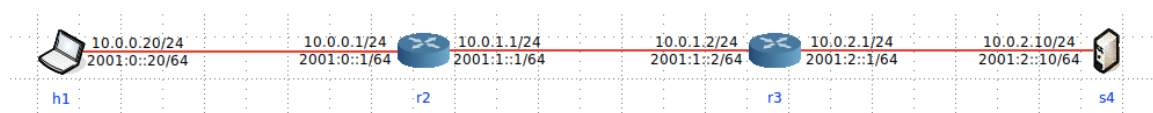
University of Minho, Department of Informatics, 4710-057 Braga, Portugal e-mail:  
{a82145,a82880,a81744}@alunos.uminho.pt

## Exercício 1:

a) Active o wireshark ou o tcpdump no pc h1. Numa shell de h1, execute o comando `traceroute -I` para o endereço IP do host s4.

### Resposta:

Através do emulador de redes CORE foi montada uma topologia envolvendo 1 pc, 2 routers, e 1 host tal como a seguinte figura ilustra.



De seguida é executado um traceroute no dispositivo h1 (pc) para o s4 (host).

```
root@h1:/tmp/pycore.45863/h1.conf# traceroute -I 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1 gateway (10.0.0.1)  0.253 ms  0.217 ms  0.212 ms
 2 10.0.1.2 (10.0.1.2)  0.206 ms  0.190 ms  0.183 ms
 3 10.0.2.10 (10.0.2.10)  0.176 ms  0.159 ms  0.151 ms
```

Como se pode observar no output obtido, é possível verificar a rota dos pacotes envidados pelo h1 para o s4 através do comando traceroute.

b) Registe e analise o tráfego ICMP enviado por h1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.

### Resposta:

Assim é feito o registo do tráfego gerado pelo comando traceroute, e analisado através do software wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	224.0.0.5	OSPF	78	Hello Packet
2	1.352665946	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0027, seq=1/256, <b>ttl=1</b> (no res)
3	1.352706774	10.0.0.1	10.0.0.20	ICMP	102	<b>Time-to-live exceeded</b> (Time to live exceeded in transit)
4	1.352722913	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0027, seq=2/512, ttl=1 (no res)
5	1.352731503	10.0.0.1	10.0.0.20	ICMP	102	<b>Time-to-live exceeded</b> (Time to live exceeded in transit)
6	1.352738779	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0027, seq=3/768, ttl=1 (no res)
7	1.352745354	10.0.0.1	10.0.0.20	ICMP	102	<b>Time-to-live exceeded</b> (Time to live exceeded in transit)
8	1.352753164	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0027, seq=4/1024, <b>ttl=2</b> (no res)
9	1.352777112	10.0.1.2	10.0.0.20	ICMP	102	<b>Time-to-live exceeded</b> (Time to live exceeded in transit)
10	1.352784232	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0027, seq=5/1280, ttl=2 (no res)
11	1.352795721	10.0.1.2	10.0.0.20	ICMP	102	<b>Time-to-live exceeded</b> (Time to live exceeded in transit)
12	1.352801948	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0027, seq=6/1536, ttl=2 (no res)
13	1.352812709	10.0.1.2	10.0.0.20	ICMP	102	<b>Time-to-live exceeded</b> (Time to live exceeded in transit)
14	1.352820137	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0027, seq=7/1792, <b>ttl=3</b> (reply)
15	1.352846740	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0027, seq=7/1792, <b>ttl=62</b> (request)
16	1.352855710	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0027, seq=8/2048, ttl=3 (reply)
17	1.352870684	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0027, seq=8/2048, ttl=62 (request)
18	1.352877261	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0027, seq=9/2304, ttl=3 (reply)
19	1.352891607	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0027, seq=9/2304, ttl=62 (request)

Os resultados obtidos na prática são coerentes com os resultados teóricos esperados.

### Explicação teórica:

Inicialmente, o traceroute envia 3 datagramas com o campo TTL (time-To-Live) igual a 1, o TTL é decrementado em cada gateway intermédio, caso o TTL atinga o valor 0 é enviado uma mensagem ICMP - Time-To-Live exceeded - no sentido contrário, permitindo assim, registar o endereço IP do gateway intermédio. De seguida são enviados novamente 3 pacotes com o TTL a 2 e assim sucessivamente até que o TTL inicial seja suficiente para o pacote atingir o destino.

No final obtêm-se a rota completa que os pacotes seguiram desde a origem até ao destino.

### Análise prática:

Como é possível observar na imagem acima, inicialmente, é enviado um pacote com TTL = 1, desde a origem h1 (10.0.0.20) com destino a s4 (10.0.2.10).

Ao passar pelo primeiro gateway r2 (10.0.0.1) o TTL é decrementado atingindo assim TTL = 0, desta forma, o dispositivo r2 envia uma mensagem ICMP á origem (h1) a informar que o TTL foi excedido.

Na figura a baixo são filtrados os dois pacotes referidos anteriormente.

10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0027, seq=1/256, <b>ttl=1</b> (no res)
10.0.0.1	10.0.0.20	ICMP	102 <b>Time-to-live exceeded</b> (Time to live exceeded in transit)

De seguida, o traceroute envia mais 2 pacotes com o TTL = 1, para assegurar a fiabilidade da informação recolhida.

O mesmo procedimento é feito para pacotes com TTL inicialmente a 2, no entanto quando chegam a r3 (10.0.1.2) o TTL é igual a 0, por isso são enviadas novamente 3 mensagens ICMP a informar que o TTL foi excedido.

10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0027, seq=4/1024, ttl=2 (no res
10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)

Finalmente, são enviados 3 pacotes com TTL = 3, desta forma os mesmo conseguem alcançar o destino (s4) antes do TTL tomar o valor de 0.

10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0027, seq=7/1792, ttl=3 (
10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x0027, seq=7/1792, ttl=62

Note-se que a mensagem ICMP reply chega ao destino com TTL = 62 pois inicialmente o pacote é enviado com TTL = 64 passando em 2 nodos intermédidos, decrementando o TTL duas vezes, ficando no fim com TTL = 62.

c) Qual deve ser o valor inicial mínimo do campo TTL para alcançar o destino s4? Verifique na prática que a sua resposta está correta.

**Resposta:**

Conlui-se que o **TTL = 3** inicial, é suficiente para os pacotes alcançarem o destino, este resultado confirma-se com a análise prática efetuada na alínea b).

O traceroute também informa o valor de ida-e-volta (Round-Trip Time) para cada um dos 3 pacotes em cada percurso.

```
root@h1:/tmp/pycore.45863/h1.conf# traceroute -I 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1 gateway (10.0.0.1) 0.253 ms 0.217 ms 0.212 ms
 2 10.0.1.2 (10.0.1.2) 0.206 ms 0.190 ms 0.183 ms
 3 10.0.2.10 (10.0.2.10) 0.176 ms 0.159 ms 0.151 ms
```

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

**Resposta:**

Verifica-se que o tempo médio de ida-e-volta no percurso total (3) é **0.162 ms**.

## Exercício 2:

No exercício 2 é feita a análise de tráfego num ambiente de redes real, sendo analisado o tráfego ICMP gerado pelo comando tracert com tamanho dos pacotes padrão.

```
▼ Internet Protocol Version 4, Src: 192.168.100.219, 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: 72
    Identification: 0x9740 (38720)
  ▼ Flags: 0x0000
    0... .. = Reserved bit: Not set
    .0.. .. = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment offset: 0
  ► Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.100.219
  Destination: 193.136.9.240
  ▼ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x60bf [correct]
    [Checksum Status: Good]
    Identifier (BE): 38719 (0x973f)
    Identifier (LE): 16279 (0x3f97)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
  ► [No response seen]
  ▼ Data (44 bytes)
    Data: 0000000000000000000000000000000000000000000000000000000000000000...
    [Length: 44]
```

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

**Resposta:**

O endereço IP da interface ativa do computador é 192.168.100.219 tal como se pode observar na região a vermelho.

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

**Resposta:**

O valor do campo protocolo é **1**. Identifica o protocolo **ICMP**. Tal é possível ser observado na região a azul.

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?

**Resposta:**

O cabeçalho tem 20 bytes como é verificado na região a verde. O campo payload tem 44 bytes como é possível verificar na região a preto no campo denominado **"Data"**. Este valor é obtido excluindo o tamanho do header do IPv4 e do ICMP ao tamanho total: **44 = 72 (tamanho total) – 20 (header do IPv4) – 8 (header do ICMP)**

d) O datagrama IP foi fragmentado? Justifique.

**Resposta:**

Como o campo **“More fragments”** não está inicializado (not set) e o campo **“Fragment offset”** tem o valor 0 significa que corresponde ao primeiro pacote e não está fragmentado. Tal é possível observar na região a rosa.

```
▼ Internet Protocol Version 4, Src: 192.168.100.219, 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: 72
  Identification: 0x9740 (38720)
  ► Flags: 0x0000
  ► Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.100.219
  Destination: 193.136.9.240

▼ Internet Protocol Version 4, Src: 192.168.100.219, 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: 72
  Identification: 0x9741 (38721)
  ► Flags: 0x0000
  ► Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.100.219
  Destination: 193.136.9.240
```

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.

**Resposta:**

É possível verificar que o campo **“Identification”** do IPv4 altera **sequencialmente** consoante o pacote, no entanto apesar de não ser possível verificar no tráfego analisado, o **“Header checksum”** também é alterado.

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

**Resposta:**

Podemos observar que o TTL muda sequencialmente a cada grupo de 3 pacotes (3 a TTL = 1, 3 a TTL = 2, etc) e os identificadores são sequenciais, ou seja, aumentam 1 a cada pacote enviado/recebido.

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?

**Resposta:**

Cada grupo de 3 pacotes enviados para o nosso computador têm todos o mesmo TTL visto que são enviados do mesmo router.

O primeiro grupo foi enviado com valor inicial TTL = 64 e alcançou o nosso computador com TTL = 64 pois não passou por nenhum nó intermédio.

O segundo grupo foi enviado com valor inicial TTL = 255 e alcançou o nosso computador com TTL = 254 pois passou por 1 nó intermédio.

O terceiro grupo foi enviado com valor inicial TTL = 64 e alcançou o nosso computador com TTL = 62 pois passou por 2 nós intermédios.

### Exercício 3:

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

Resposta:

Como o tamanho da fonte (3521 bytes) excede o MTU, 1500 bytes (capacidade máxima de um datagrama) é necessário recorrer ao processo de fragmentação dos dados.

```
▼ Internet Protocol Version 4, Src: 192.168.100.219, 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: 0x9e73 (40563)
  ▼ Flags: 0x2000, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    ...0 0000 0000 0000 = Fragment offset: 0
  ► Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.100.219
    Destination: 193.136.9.240
```

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?

Resposta:

Como se pode observar na região a azul, este datagrama é fragmentado (“**More fragments: Set**”) e corresponde ao primeiro fragmento (“**Fragment offset: 0**”). O tamanho total do pacote é 1500 como se verifica na região a vermelho.

```
▼ Internet Protocol Version 4, Src: 192.168.100.219, 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: 0x9e73 (40563)
  ▼ Flags: 0x20b9, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    ...0 0000 1011 1001 = Fragment offset: 185
  ► Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.100.219
    Destination: 193.136.9.240
    Reassembled IPv4 in frame: 3
```

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?

**Resposta:**

O datagrama não se trata do 1º fragmento como se pode comprovar com o valor do **“Fragment offset”** na região a vermelho. Existem mais fragmentos como se pode constatar com o valor de **“More fragments”** na região a azul.

```
▼ Internet Protocol Version 4, Src: 192.168.100.219, 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: 561
    Identification: 0x9e73 (40563)
  ▼ Flags: 0x0172
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0001 0111 0010 = Fragment offset: 370
  ► Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.100.219
    Destination: 193.136.9.240
  ► [3 IPv4 Fragments (3501 bytes): #1(1480), #2(1480), #3(541)]
```

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

**Resposta:**

Foram criados 3 fragmentos a partir do datagrama original. O último fragmento tem o **“Fragment offset”** diferente de 0 e **“More fragments”** sem atribuição. Tal é possível constatar na região a vermelho.

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.

**Resposta:**

Entre os diferentes fragmentos mudam 2 campos, o campo **“More fragments”** e o **“Fragment offset”**. O último pacote geralmente tem **dimensão** menor que os anteriores.

A reconstrução cronológica dos fragmentos é possível uma vez que o primeiro fragmento tem o **“More fragments: set”** e o **“Fragment offset: 0”** os sucessores têm o **“More**



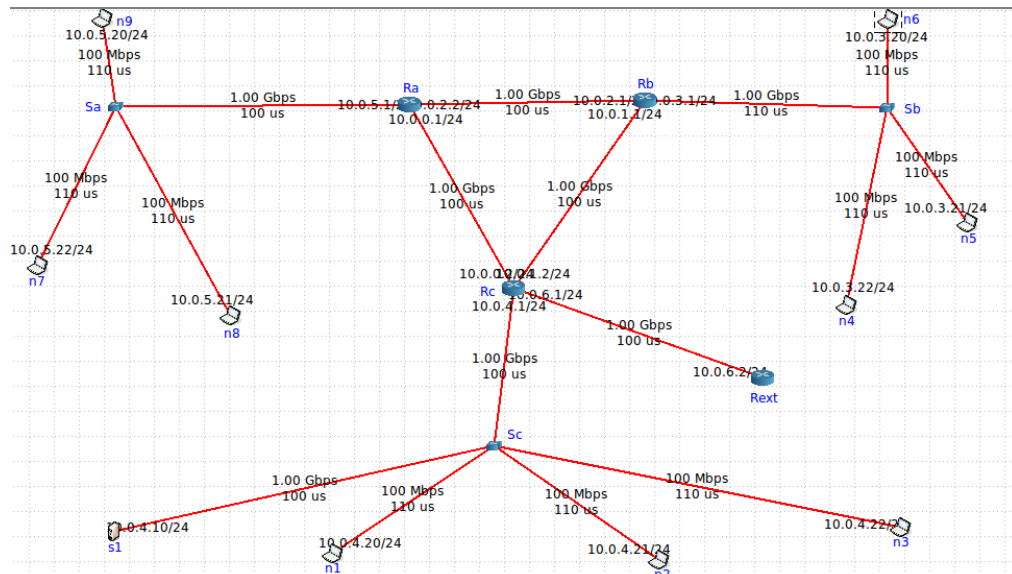
**fragments: set**” e o **“Fragment offset”** com valores incrementais, o último fragmento tem o **“More fragments: not set”** e o **“Fragment offset”** com o valor máximo.

# eTP2: Protocolo IP (Parte II)

## Exercício 1:

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.

**Resposta:**



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

**Resposta:**

Tratam-se de endereços privados uma vez que não são visíveis para o exterior da rede, o próprio padrão do endereço de IP é indicativo que é privado isto é, começa com 10.0.0.0 até 10.255.255.255.

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

**Resposta:**

Os switches não atuam em nível 3 (IP), apenas em nível 2 (ethernet).

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

**Resposta:**

Laptop no departamento A para o servidor S1.

```
root@n9: /tmp/pycore.38626/n9.conf
root@n9:/tmp/pycore.38626/n9.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=25.5 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=62 time=23.1 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=62 time=45.3 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=62 time=34.7 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=62 time=23.5 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=62 time=23.6 ms
^C
--- 10.0.4.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5028ms
rtt min/avg/max/mdev = 23.156/29.306/45.321/8.208 ms
root@n9:/tmp/pycore.38626/n9.conf#
```

Laptop no departamento B para o servidor S1.

```
root@n6: /tmp/pycore.38626/n6.conf
root@n6:/tmp/pycore.38626/n6.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=39.8 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=62 time=35.5 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=62 time=33.9 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=62 time=23.7 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=62 time=23.2 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=62 time=23.9 ms
64 bytes from 10.0.4.10: icmp_req=7 ttl=62 time=44.0 ms
^C
--- 10.0.4.10 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6030ms
rtt min/avg/max/mdev = 23.230/32.057/44.009/7.866 ms
root@n6:/tmp/pycore.38626/n6.conf#
```

Laptop no departamento C para o servidor S1.

```
root@n3: /tmp/pycore.38626/n3.conf
root@n3:/tmp/pycore.38626/n3.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=23.0 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=64 time=7.00 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=64 time=8.60 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=64 time=6.26 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=64 time=14.1 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=64 time=11.5 ms
64 bytes from 10.0.4.10: icmp_req=7 ttl=64 time=11.9 ms
^C
--- 10.0.4.10 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6039ms
rtt min/avg/max/mdev = 6.260/11.789/23.000/5.280 ms
root@n3:/tmp/pycore.38626/n3.conf#
```

Conclui-se que existe conectividade entre todos os laptops dos vários departamentos e o servidor S1.

e) Verifique se existe conectividade IP do router de acesso Rext para o servidor S1.

Resposta:

```
root@Rext: /tmp/pycore.42423/Rext.conf
root@Rext:/tmp/pycore.42423/Rext.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=63 time=1.00 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=63 time=2.37 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=63 time=0.839 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=63 time=1.17 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=63 time=0.872 ms
64 bytes from 10.0.4.10: icmp_req=6 ttl=63 time=1.15 ms
^C
--- 10.0.4.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5004ms
rtt min/avg/max/mdev = 0.839/1.237/2.378/0.525 ms
root@Rext:/tmp/pycore.42423/Rext.conf#
```

Como se verifica na imagem existe conectividade entre o router externo e o servidor S1.

## Exercício 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 respetivo (`man netstat`).

Resposta:

```
root@n9: /tmp/pycore.42423/n9.conf
root@n9:/tmp/pycore.42423/n9.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
root@n9:/tmp/pycore.42423/n9.conf#
```

O comando `netstat` (network status) apresenta informações sobre o estado atual da rede. A opção `-rn` apresenta as tabelas de encaminhamento, `-r` (**route**), sobre a forma de IP's numéricos, `-n` (**numeric host**). Como podemos ver na tabela existe encaminhamento do laptop (dispositivo atual - 0.0.0.0) para o router (10.0.5.0).

```
root@Ra: /tmp/pycore.42423/Ra.conf
root@Ra:/tmp/pycore.42423/Ra.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
10.0.3.0 10.0.2.1 255.255.255.0 UG 0 0 0 eth1
10.0.4.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
10.0.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
10.0.6.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
root@Ra:/tmp/pycore.42423/Ra.conf#
```

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

Resposta:

```
root@Ra:/tmp/pycore.42423/Ra.conf# ps ax
PID TTY STAT TIME COMMAND
1 ? S 0:00 /usr/sbin/vnoded -v -c /tmp/pycore.42423/Ra -l /tmp/p
58 ? Ss 0:00 /usr/lib/quagga/zebra -u root -g root -d
66 ? Ss 0:03 /usr/lib/quagga/ospfd -u root -g root -d
68 ? Ss 0:11 /usr/lib/quagga/ospfd -u root -g root -d
76 pts/7 Ss+ 0:00 /bin/bash
88 pts/5 Ss+ 0:00 /bin/bash
99 pts/9 Ss 0:00 /bin/bash
114 pts/9 R+ 0:00 ps ax
root@Ra:/tmp/pycore.42423/Ra.conf#
```

Como se pode verificar na imagem acima, estão a correr 3 processos do **Quagga**, este software é responsável por fazer encaminhamento dinâmico das rotas. Ainda é possível observar que está a ser usado encaminhamento dinâmico tanto para IPv4 (ospfd que implementa o algoritmo OSPF - Open Shorte**s**t Path First) como para IPv6 (ospfd6d que implementa o mesmo algoritmo numa versão diferente).

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 C. Use o comando `route delete` para o efeito. Que implicações tem esta medida para os utilizadores da empresa que acedem ao servidor. Justifique.

Resposta:

Após ser retirada da tabela de encaminhamento do servidor S1 a rota por defeito (0.0.0.0 ou *default*) a mesma encontra-se da seguinte forma.

```
root@s1:/tmp/pycore.42425/s1.conf# route delete default
root@s1:/tmp/pycore.42425/s1.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@s1:/tmp/pycore.42425/s1.conf# ping 0.0.0.0
```

A tentativa de conexão entre um laptop (n7) do departamento A para o servidor S1 foi falhada como a seguinte imagem evidencia.

```
root@n7:/tmp/pycore.42425/n7.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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2002ms
root@n7:/tmp/pycore.42425/n7.conf#
```

Ao eliminar a rota por defeito no servidor S1, o mesmo deixa de poder enviar pacotes para outras redes ou seja, existe caminho de ida entre os outros dispositivos fora da rede e o servidor mas não existe caminho de volta.

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.

**Resposta:**

A tabela de encaminhamento restaurada é a seguinte.

```
root@s1: /tmp/pycore.42425/s1.conf
root@s1:/tmp/pycore.42425/s1.conf# route add default gw 10.0.4.1
root@s1:/tmp/pycore.42425/s1.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@s1:/tmp/pycore.42425/s1.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.

**Resposta:**

Como a seguinte imagem demonstra o servidor S1 está de novo acessível, visto que temos conectividade, conexão (“caminho”) de ida e volta.

```
root@n7: /tmp/pycore.42425/n7.conf
root@n7:/tmp/pycore.42425/n7.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=1.43 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=62 time=1.63 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=62 time=1.92 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=62 time=4.46 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=62 time=2.32 ms
^C
--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 1.436/2.357/4.464/1.095 ms
root@n7:/tmp/pycore.42425/n7.conf#
```

### Exercício 3:

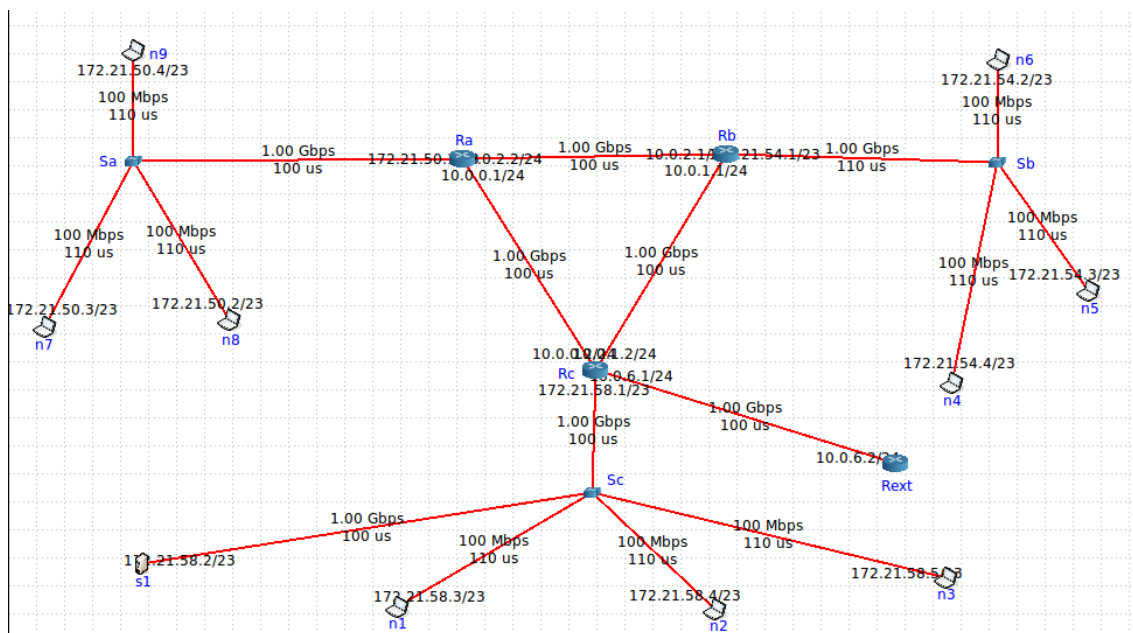
1) Considere que dispõe apenas do endereço de rede IP 172.XX.48.0/20, em que XX é o decimal correspondendo ao seu numero de grupo (PLXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo a rede de acesso e core inalteradas) e atribua endereços às interfaces dos vários sistemas envolvidos. **Deve justificar as opções usadas.**

#### Resposta:

O endereço IP inicial da rede proposto é o 172.21.48.0/20. Com base no número de departamentos existentes, neste caso são 3, é necessário definir pelo menos 3 sub-redes distintas. Assumimos que são reservados os dois endereços extremos (x.x.x.0) e o (x.x.x.255). Deste modo necessitamos de pelo menos 3 bits para identificar a sub-rede. Usando 3 bits conseguimos identificar 8 sub-redes distintas, no entanto excluindo os 2 endereços reservados, resta-nos 6 sub-redes. Assim ficamos com um novo endereço /23 variando entre **172.21.48.0**, **172.21.50.0**, 172.21.52.0, **172.21.54.0**, 172.21.56.0, **172.21.58.0**, 172.21.60.0, **172.21.62.0**, sendo que os endereços IP a vermelho são os reservados.

Os endereços IP a azul são os escolhidos para cada departamento, uma vez que pensamos numa possível expansão.

Desta forma propomos a seguinte topologia.



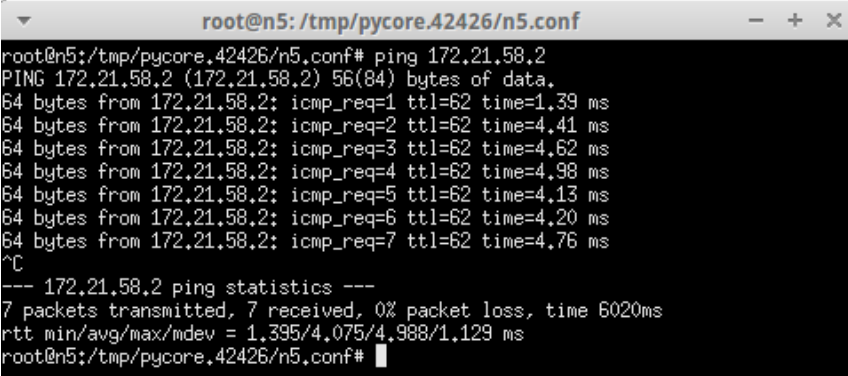
2) Qual a máscara de rede que usou (em formato decimal)? Quantos *hosts* IP pode interligar em cada departamento? Justifique.

**Resposta:**

Visto que o endereço de IP é **/23** temos uma máscara de rede **1111 1111 . 1111 1111 . 1111 1110 . 0000 0000** que em decimal é **255.255.254.0**. Em cada departamento é possível interligar **510** dispositivos, por exemplo a sub-rede C (172.21.58.0) varia entre 172.21.58.0 e 172.21.59.255, assumindo que se exclui o IP de identificação de rede e o de broadcast, ficamos com  $256 * 2 - 2 = 510$  endereços de IP possíveis de associar aos dispositivos na rede.

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

**Resposta:**



```
root@n5: /tmp/pycore.42426/n5.conf
root@n5:/tmp/pycore.42426/n5.conf# ping 172.21.58.2
PING 172.21.58.2 (172.21.58.2) 56(84) bytes of data:
64 bytes from 172.21.58.2: icmp_req=1 ttl=62 time=1.39 ms
64 bytes from 172.21.58.2: icmp_req=2 ttl=62 time=4.41 ms
64 bytes from 172.21.58.2: icmp_req=3 ttl=62 time=4.62 ms
64 bytes from 172.21.58.2: icmp_req=4 ttl=62 time=4.98 ms
64 bytes from 172.21.58.2: icmp_req=5 ttl=62 time=4.13 ms
64 bytes from 172.21.58.2: icmp_req=6 ttl=62 time=4.20 ms
64 bytes from 172.21.58.2: icmp_req=7 ttl=62 time=4.76 ms
^C
--- 172.21.58.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6020ms
rtt min/avg/max/mdev = 1.395/4.075/4.988/1.129 ms
root@n5:/tmp/pycore.42426/n5.conf#
```

Como se pode observar na imagem a conectividade verifica-se após a implementação do sub-neting.



## Conclusão:

Relativamente à realização da primeira parte, surgiram alguns desafios, nomeadamente no estabelecimento do “routing” no emulador de rede utilizado (CORE), e na interpretação dos resultados retirados do Wireshark, uma vez que existiam valores de TTL (time to live) que não correspondiam ao habitual. Por exemplo, numa rede com 2 routers e um host o time to live deveria ter decrementado no máximo 3, contudo decrementou a mais. Posteriormente, verificou-se que era um erro do emulador.

No que diz respeito à realização da segunda parte, o principal desafio foi o estabelecimento do routing dinâmico no CORE, uma vez que aconteciam momentos em que o “ping” funcionava, e outras em que não funcionava. Mais uma vez se verificou que o problema estava no delay no estabelecimento do “routing” no CORE.

Contudo, é de destacar a importância que este trabalho prático teve no aprofundamento dos conhecimentos acerca da forma como o protocolo IPv4 atua ao nível do “Network layer”, bem como na influência que o sub-netting tem na organização de uma rede, apesar da diminuição de endereços válidos disponíveis.