



UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

Redes de Computadores
Trabalho Prático 2
Protocolo IPv4 e
Endereçamento e Encaminhamento
IP
Grupo Nº 18

Ariana Lousada (A87998)
Carlos Gomes (A77185)
Pedro Pereira (A80627)

22 de maio de 2021

Conteúdo

1	Questões e Respostas	3
1.1	1ª Parte	3
1.1.1	Exercício 1	3
1.1.2	Exercício 2	5
1.1.3	Exercício 3	7
1.2	2ª Parte	8
1.2.1	Exercício 1	8
1.2.2	Exercício 2	11
1.2.3	Exercício 3	15
2	Conclusão e Análise de Resultados	17

Capítulo 1

Questões e Respostas

Para a resolução deste trabalho, foram-nos propostas várias questões, as quais vamos passar a responder neste capítulo:

1.1 1ª Parte

1.1.1 Exercício 1

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

```
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) 0.078 ms 0.023 ms 0.019 ms
 2 10.0.1.2 (10.0.1.2) 0.045 ms 0.031 ms 0.026 ms
 3 10.0.2.10 (10.0.2.10) 0.096 ms 0.036 ms 0.033 ms
root@Cliente:/tmp/pycore.40263/Cliente.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) 0.071 ms 0.022 ms 0.016 ms
 2 10.0.1.2 (10.0.1.2) 0.035 ms 0.024 ms 0.022 ms
 3 10.0.2.10 (10.0.2.10) 0.040 ms 0.032 ms 0.028 ms
root@Cliente:/tmp/pycore.40263/Cliente.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) 0.072 ms 0.022 ms 0.017 ms
 2 10.0.1.2 (10.0.1.2) 0.050 ms 0.025 ms 0.023 ms
 3 10.0.2.10 (10.0.2.10) 0.040 ms 0.030 ms 0.029 ms
root@Cliente:/tmp/pycore.40263/Cliente.conf#
```

Figura 1.1: Comando traceroute -I

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=1/256, ttl=1 (no response found!)
2	0.000000000	10.0.0.1	10.0.2.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
3	0.000042766	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=2/512, ttl=1 (no response found!)
4	0.000046524	10.0.0.1	10.0.2.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
5	0.000050061	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=3/768, ttl=1 (no response found!)
6	0.000053113	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
7	0.000056890	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=4/1024, ttl=2 (no response found!)
8	0.000072270	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
9	0.000075732	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=5/1280, ttl=2 (no response found!)
10	0.000081742	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
11	0.000084797	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=6/1536, ttl=2 (no response found!)
12	0.000089705	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
13	0.000093567	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=7/1792, ttl=3 (reply in 14)
14	0.000103478	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x002a, seq=7/1792, ttl=62 (request in 13)
15	0.000113913	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=8/2048, ttl=3 (reply in 16)
16	0.000120663	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x002a, seq=8/2048, ttl=62 (request in 15)

Figura 1.2: TTL

Ao analisarmos os resultados obtidos, podemos constatar que o envio de pacotes teve duas fases: i) pacotes com TTL abaixo de 3 e ii) pacotes com TTL acima de 3.

Na primeira fase, podemos observar que os pacotes com TTL=1 e TTL=2 foram descartados pelos routers e para cada um destes foi recebido um pacote Time-to-live exceed. Já na segunda fase, nenhum deles foi descartado tendo como resposta pacotes Echo(ping) reply.

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

Obtivemos um valor mínimo do campo TTL equivalente a 3, tal como seria esperado teoricamente.

- 1.d) Calcule o valor médio do tempo de ida-e-volta (Round-TripTime) obtido.

Uma vez que o 1^a pacote teve um RTT bastante superior aos restantes, provavelmente devido a atrasos causados pelo sistema operativo em si e não inerentes à rede, decidimos ignorar esse valor. Sendo assim o valor médio foi calculado da seguinte forma:

$$\frac{36 + 33 + 40 + 32 + 28 + 40 + 30 + 29}{8} = 0.0335ms \quad (1.1)$$

1.1.2 Exercício 2

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

```
▼ Internet Protocol Version 4, Src: 172.26.36.54, 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: 60
    Identification: 0x2149 (8521)
  ▶ Flags: 0x0000
    Fragment offset: 0
  ▶ Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0xfcaf [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.36.54
    Destination: 193.136.9.240
```

Figura 1.3: Cabeçalho IP

Como se pode observar na figura, a máquina utilizada para teste tem um endereço IP de 172.26.36.54.

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

Verificámos que o valor equivale a 1, que corresponde ao ICMP.

2.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?

Ao analisarmos o campo Header length na figura 2.3, podemos concluir que o cabeçalho IP(v4) tem 20 bytes (5). O payload do datagrama corresponde a 40. O tamanho do payload pode ser obtido através da diferença entre o número de bytes e o tamanho do cabeçalho do datagrama (*Total length - header length*).

2.d) O datagrama IP foi fragmentado? Justifique.

Não, uma vez que ambos o *fragment offset* e o *flag more fragments* são 0 (na primeira mensagem capturada).

2.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.

Os únicos campos do cabeçalho que vão variando são a *Identificação*, *time-to-live* e a *header checksum*.

5	0.016977368	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=1/256, ttl=1 (no response found!)
6	0.017000712	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=2/512, ttl=1 (no response found!)
7	0.017011963	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=3/768, ttl=1 (no response found!)
8	0.017024492	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=4/1024, ttl=2 (no response found!)
9	0.017072692	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=5/1280, ttl=2 (no response found!)
10	0.017116761	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=6/1536, ttl=2 (no response found!)
11	0.017159265	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=7/1792, ttl=3 (no response found!)
12	0.017200262	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=8/2048, ttl=3 (no response found!)
13	0.017208326	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=9/2304, ttl=3 (no response found!)
14	0.017251794	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=10/2560, ttl=4 (reply in 29)
15	0.017259945	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=11/2816, ttl=4 (reply in 30)
16	0.017297269	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=12/3072, ttl=4 (reply in 31)
17	0.017306270	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=13/3328, ttl=5 (reply in 32)
18	0.017340757	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=14/3584, ttl=5 (reply in 34)
19	0.017348816	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=15/3840, ttl=5 (reply in 35)
20	0.017386285	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=16/4096, ttl=6 (reply in 36)
22	0.019796688	172.26.36.54	193.136.9.240	ICMP	74	Echo (ping) request	id=0x78bc, seq=17/4352, ttl=6 (reply in 39)

Figura 1.4: Pacotes capturados

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

O ID incrementa por 1 de pacote em pacote, enquanto que o TTL incrementa por 1 de 3 em 3 pacotes.

<p>Frame 5: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp3s0, id 0</p> <p>Ethernet II, Src: AzureWav_9c:ab:05 (88:a5:89:9c:ab:05), Dst: ComdaEnt_ff:94:00 (00:00:03:ff:94:00)</p> <p>Destination: ComdaEnt_ff:94:00 (00:00:03:ff:94:00)</p> <p>Address: ComdaEnt_ff:94:00 (00:00:03:ff:94:00)</p> <p>.....0..... = 10 bit: Globally unique address (factory default)</p> <p>.....0..... = 10 bit: Individual address (unicast)</p> <p>Source: AzureWav_9c:ab:05 (88:a5:89:9c:ab:05)</p> <p>Address: AzureWav_9c:ab:05 (88:a5:89:9c:ab:05)</p> <p>.....0..... = 10 bit: Globally unique address (factory default)</p> <p>.....0..... = 10 bit: Individual address (unicast)</p> <p>Type: IPv4 (0x0800)</p> <p>Internet Protocol Version 4, Src: 172.26.36.54, Dst: 193.136.9.240</p> <p>0100 = Version: 4</p> <p>.....0101 = Header Length: 20 bytes (5)</p> <p>Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>0000 00.. = Differentiated Services Codepoint: Default (0)</p> <p>.....00 = Explicit Congestion Notification: Not ECT-Capable Transport (0)</p> <p>Total Length: 60</p> <p>Identification: 0x2149 (8521)</p> <p>Flags: 0x0000</p> <p>Fragment offset: 0</p> <p>Time to live: 1</p> <p>Protocol: ICMP (1)</p> <p>Header checksum: 0xfcaf [validation disabled]</p> <p>0000 00 00 03 ff 94 00 00 00 88 a5 89 9c ab 05 00 00 45 00 0010 00 3c 21 49 00 00 01 01 fc aa ac 1a 24 36 c1 88E\$ 0020 09 f0 08 00 09 0d 78 bc 00 01 48 49 4a 4b 4c 4dx..KLM 0030 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d WOPQSTU VWXYZ[] 0040 5e 5f 60 61 62 63 64 65 66 67 ^_`abcde fg</p>	<p>Frame 6: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp3s0, id 0</p> <p>Ethernet II, Src: AzureWav_9c:ab:05 (88:a5:89:9c:ab:05), Dst: ComdaEnt_ff:94:00 (00:00:03:ff:94:00)</p> <p>Destination: ComdaEnt_ff:94:00 (00:00:03:ff:94:00)</p> <p>Address: ComdaEnt_ff:94:00 (00:00:03:ff:94:00)</p> <p>.....0..... = 10 bit: Globally unique address (factory default)</p> <p>.....0..... = 10 bit: Individual address (unicast)</p> <p>Source: AzureWav_9c:ab:05 (88:a5:89:9c:ab:05)</p> <p>Address: AzureWav_9c:ab:05 (88:a5:89:9c:ab:05)</p> <p>.....0..... = 10 bit: Globally unique address (factory default)</p> <p>.....0..... = 10 bit: Individual address (unicast)</p> <p>Type: IPv4 (0x0800)</p> <p>Internet Protocol Version 4, Src: 172.26.36.54, Dst: 193.136.9.240</p> <p>0100 = Version: 4</p> <p>.....0101 = Header Length: 20 bytes (5)</p> <p>Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>0000 00.. = Differentiated Services Codepoint: Default (0)</p> <p>.....00 = Explicit Congestion Notification: Not ECT-Capable Transport (0)</p> <p>Total Length: 60</p> <p>Identification: 0x214a (8522)</p> <p>Flags: 0x0000</p> <p>Fragment offset: 0</p> <p>Time to live: 1</p> <p>Protocol: ICMP (1)</p> <p>Header checksum: 0xfcae [validation disabled]</p> <p>0000 00 00 03 ff 94 00 00 00 88 a5 89 9c ab 05 00 00 45 00 0010 00 3c 21 4a 00 00 01 01 fc aa ac 1a 24 36 c1 88E\$ 0020 09 f0 08 00 09 bc 78 bc 00 02 10 49 4a 4b 4c 4dx..LM 0030 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d WOPQSTU VWXYZ[] 0040 5e 5f 60 61 62 63 64 65 66 67 ^_`abcde fg</p>
---	--

Figura 1.5

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

O ttl é de 1 e permanece constante. O TTL é decrementado em cada salto e quando é 1 e não está no destino (sendo decrementado passaria a 0) é enviado para trás.

1.1.3 Exercício 3

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

Foi necessário fragmentar o pacote inicial, uma vez que o limite da camada IP é de 1500 enquanto que o tamanho do pacote enviado é de 3218.

- 3.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?**

A *flag more fragments* a 1 indica que foi fragmentado. O *fragment offset* a 0 indica que se trata do primeiro fragmento. Com isto, podemos portanto dizer que tamanho do datagrama IP é de 1500 bytes.

- 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?**

O *fragment offset* a 1480 indica que não se trata do primeiro fragmento. Há mais fragmentos pois a *flag more fragments* está a 1.

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

Foram criados 3 fragmentos a partir do datagrama original. O último fragmento é detetável pela *flag more fragments*, uma vez que está a 0.

- 3.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.**

Os campos que mudam são a *total length*, o *fragment offset* e as *flags* (e consequentemente a *header checksum*). Uma vez que os fragmentos têm a mesma identificação e sabendo o offset de cada um, é possível reconstruir o datagrama original.

1.2 2ª Parte

1.2.1 Exercício 1

- 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.

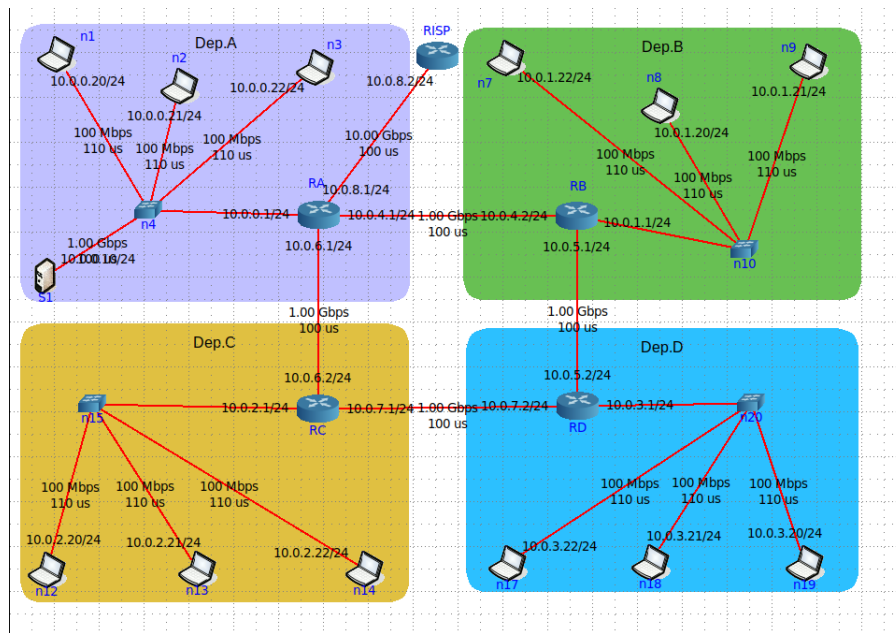


Figura 1.6: Topologia criada

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

Os endereços presentes são privados, uma vez que os seus IP's localizam-se entre 10.0.0.0 e 10.255.255.255.

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

Os *switches* não possuem um valor de IP porque a sua função é simplesmente transmitir/reencaminhar as ligações.

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

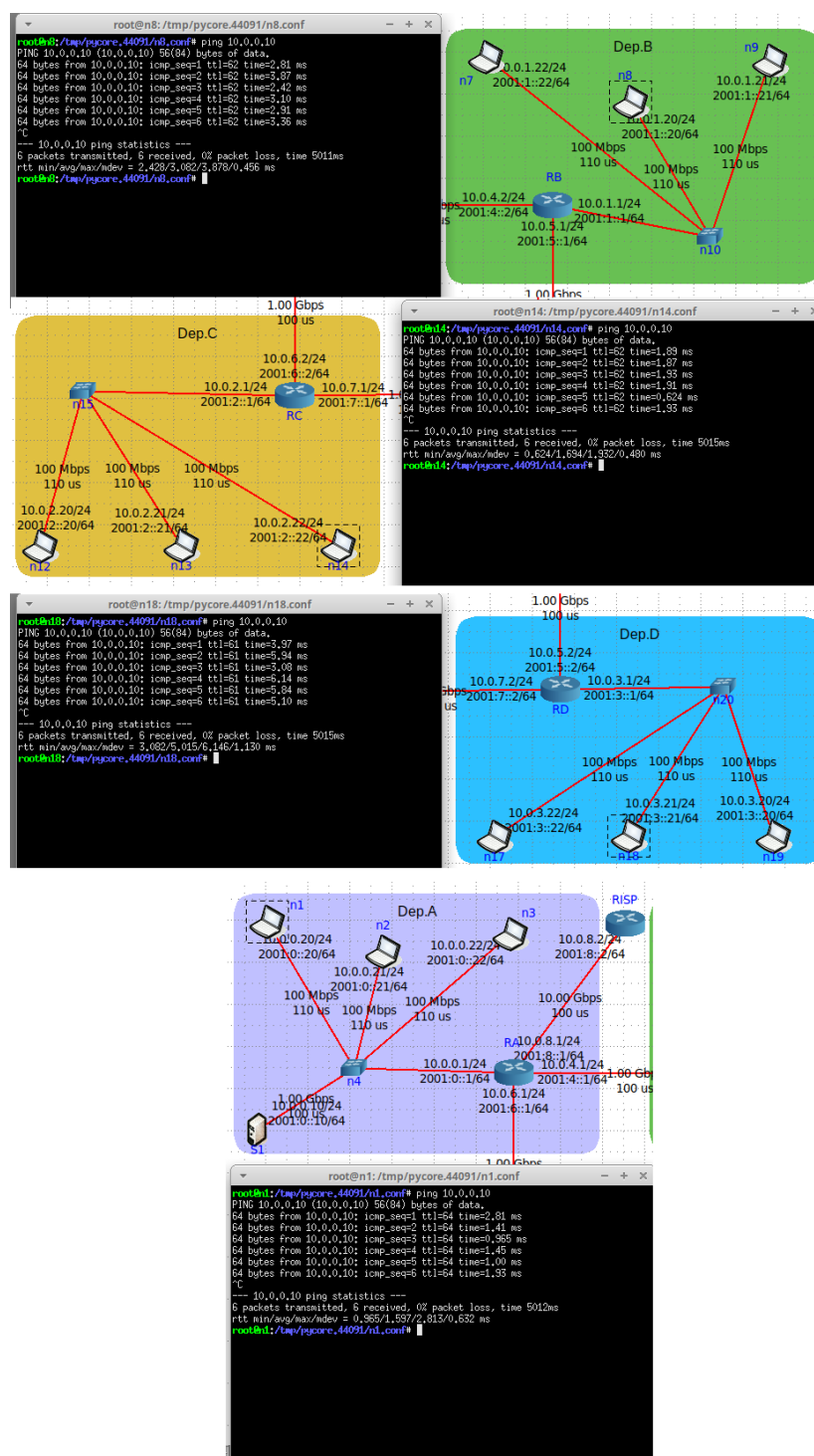


Figura 1.7: Execução do comando *ping* por Departamento

Pela figura apresentada, podemos observar que o comando *ping* foi executado em todos os departamentos, o que mostra que existe de facto conectividade IP em cada departamento.

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

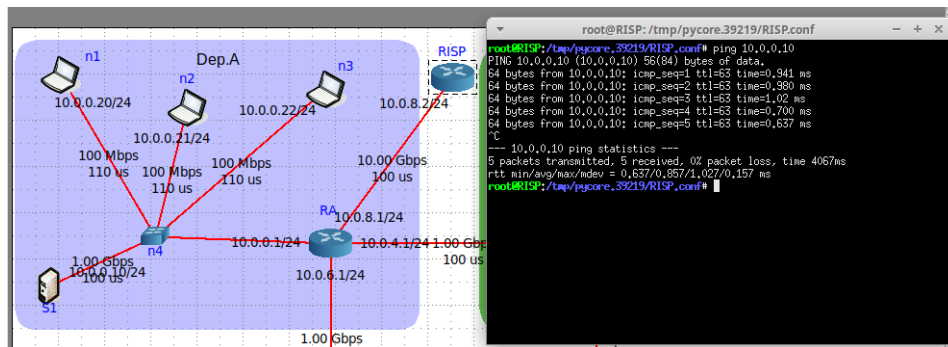
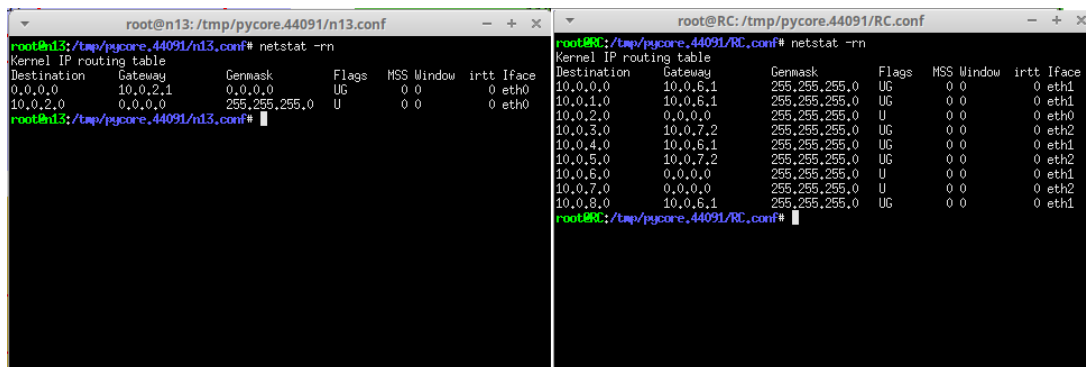


Figura 1.8: Ligação entre RISP e S1

De modo a verificar a conexão entre as duas entidades referidas, executou-se mais uma vez o comando *ping*, que nos mostrou que existe de facto conectividade entre o servidor S1 e o router RISP.

1.2.2 Exercício 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 respetivo (`man netstat`).



```
root@n13:/tmp/pycore.44091/n13.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.2.1 0.0.0.0 UG 0 0 0 eth0
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0

root@RC:/tmp/pycore.44091/RC.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 10.0.6.1 255.255.255.0 UG 0 0 0 eth1
10.0.1.0 10.0.6.1 255.255.255.0 UG 0 0 0 eth1
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.0.3.0 10.0.7.2 255.255.255.0 UG 0 0 0 eth2
10.0.4.0 10.0.6.1 255.255.255.0 UG 0 0 0 eth1
10.0.5.0 10.0.7.2 255.255.255.0 UG 0 0 0 eth2
10.0.6.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
10.0.7.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
10.0.8.0 10.0.6.1 255.255.255.0 UG 0 0 0 eth1
```

Figura 1.9: Execução do comando `netstat`

Para a tabela de endereçamento do laptop do departamento C:

A 1ª entrada indica o encaminhamento para os endereços *default*, devendo ser encaminhados para a interface do router do departamento C. A 2ª entrada indica o encaminhamento de pacotes para uma interface da rede do departamento C, logo o encaminhamento é direto.

Para a tabela de endereçamento do router do departamento C:

A 1ª e 2ª entradas indicam o encaminhamento para os endereços do departamento A e B, respetivamente, devendo ser encaminhados para a interface do router do departamento A. A 3ª entrada indica o encaminhamento de pacotes para uma interface da rede do departamento C, logo o encaminhamento é direto. A 4ª e 6ª indicam o encaminhamento para os endereços do departamento D e para uma das interfaces do router do departamento D, respetivamente, devendo ser encaminhadas para a interface deste. A 5ª e 9ª indicam o encaminhamento para uma das interfaces do router do departamento B e para o router do ISP, respetivamente, devendo ser encaminhadas para a interface do router A. As 7ª e 8ª entradas indicam o encaminhamento para as interfaces do próprio router C, não sendo necessário encaminhamento.

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

O encaminhamento é estático nos *hosts* e dinâmico nos *routers*. Nos routers, está a ser utilizado o encaminhamento dinâmico, uma vez que é utilizado o OSPF, que é uma técnica de *dynamic routing*. O encaminhamento dinâmico altera automaticamente com alterações na rede. Uma vez que os

routers podem sofrer muitas alterações nas ligações porque qualquer coisa se poder ligar a ele, por isso é conveniente que o seu encaminhamento seja dinâmico.

Contudo, nos hosts temos encaminhamento estático, que pode ser alterado manualmente (uma vez que um host está ligado ao router e por isso não é necessário). A rede pode mudar mas o routing só muda se este for propositadamente alterado.

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

```

root@S1:/tmp/pycore.46243/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.46243/S1.conf# route delete default
root@S1:/tmp/pycore.46243/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.46243/S1.conf#

```

Figura 1.10

Utilizadores da organização MIEI-RC que não pertençam ao departamento A não vão conseguir aceder ao servidor S1, uma vez que o S1 não tem rota de encaminhamento definida para os IP's não pertencentes à rede do departamento A.

- 2.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 registre os comandos que usou

```

root@S1:/tmp/pycore.46243/S1.conf# route add -net 10.0.1.0 netmask 255.255.255.0 gw 10.0.0.1
root@S1:/tmp/pycore.46243/S1.conf# route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.0.1
root@S1:/tmp/pycore.46243/S1.conf# route add -net 10.0.3.0 netmask 255.255.255.0 gw 10.0.0.1
root@S1:/tmp/pycore.46243/S1.conf# route add -net 10.0.8.0 netmask 255.255.255.0 gw 10.0.0.1

```

Figura 1.11

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

```
root@S1:/tmp/pycore.46243/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
10.0.1.0          10.0.0.1        255.255.255.0   UG       0 0        0 eth0
10.0.2.0          10.0.0.1        255.255.255.0   UG       0 0        0 eth0
10.0.3.0          10.0.0.1        255.255.255.0   UG       0 0        0 eth0
10.0.8.0          10.0.0.1        255.255.255.0   UG       0 0        0 eth0
root@S1:/tmp/pycore.46243/S1.conf#
```

Figura 1.12

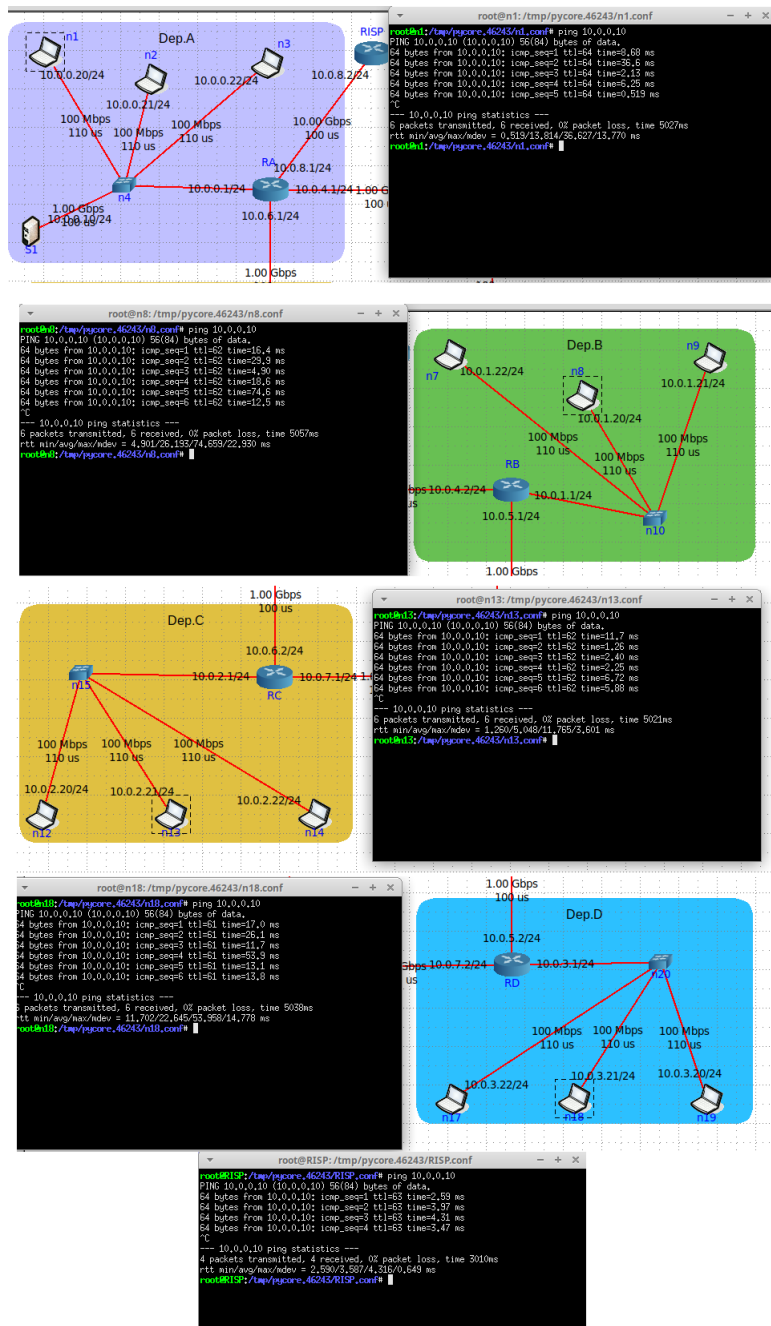


Figura 1.13: Execução do comando *ping* por Departamento

1.2.3 Exercício 3

- 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.

Dos 13 bits disponíveis para *subnetting*, 3 foram usados para identificação de sub-redes. Isto permite a existencia de 8 sub-redes, permitindo assim uma expansão futura à rede atual com 4 sub-redes. Os restantes 10 bits são usados para identificação de host interfaces.

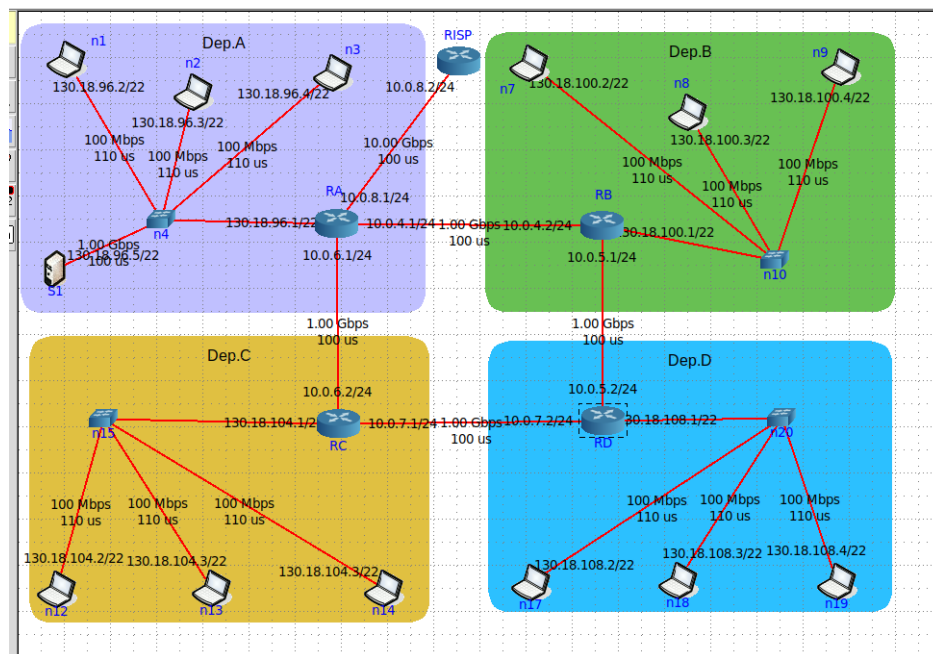


Figura 1.14

- 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 utilizada é 255.255.252.0., isto porque tem 22 bits de máscara. É possível a ligação de 1022 hosts a cada departamento ($2^{10} - 2$).
- 3.3) Garanta e verifique que conectividade IP entre as várias redes locais da organização MIEI-RC é mantida. Explique como procedeu.

Para verificarmos a conectividade IP entre os quatro departamentos, simplesmente executámos o comando ping num dos *hosts* em cada rede.

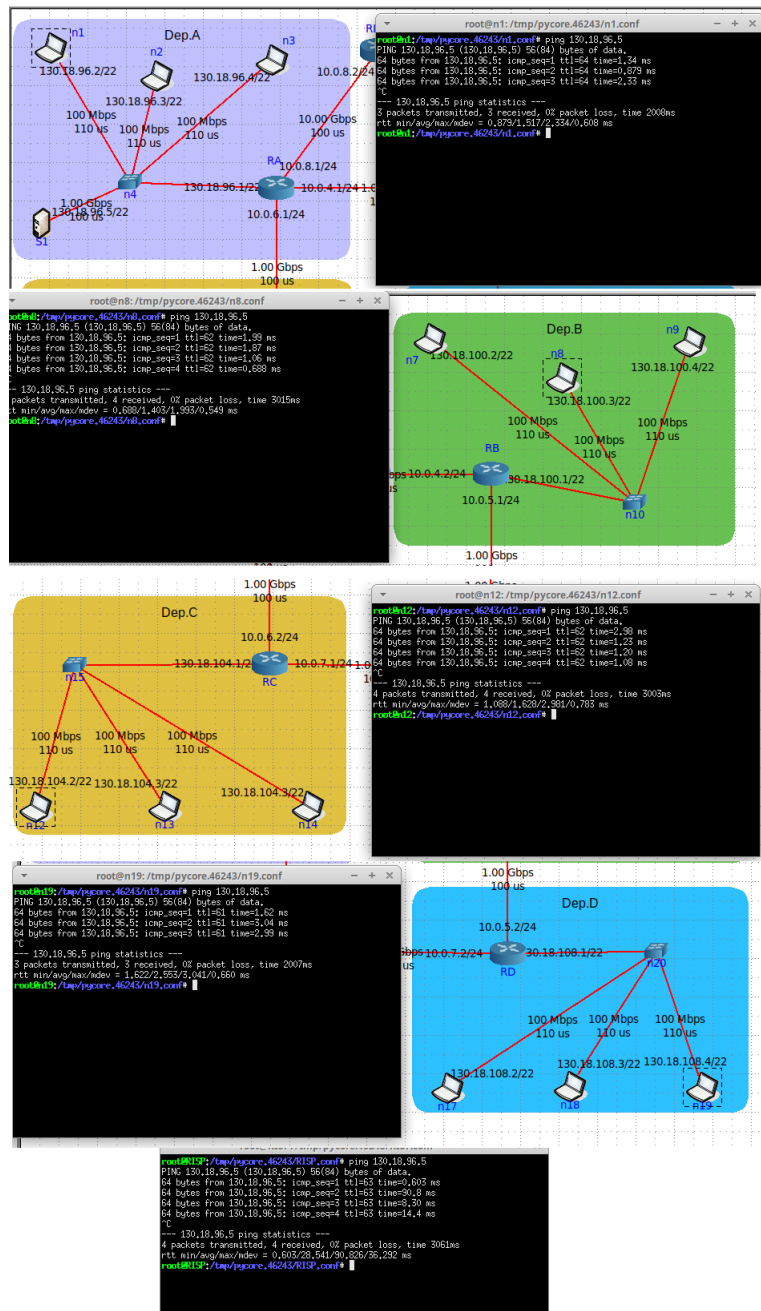


Figura 1.15: Execução do comando `ping` em cada departamento

Capítulo 2

Conclusão e Análise de Resultados

Com a elaboração deste trabalho conseguimos ter uma experiência mais prática com os conceitos lecionados nas aulas teóricas.

Na primeira parte, o objetivo principal foi analisar o IP e para isso tivemos que analisar o formato dos pacotes/datagramas IP e a fragmentação dos mesmos, usando uma topologia desenvolvida no Core, que nos permitiu acompanhar passo a passo todo o processo, também com o auxílio do WireShark (que nos possibilitou analisar mais detalhadamente os pacotes enviados e recebidos pela rede).

Na 2ª parte focámos no estudo e perceção do endereçamento e encaminhamento do que foi estudado na 1ª parte. Construímos novamente uma topologia na tentativa de simularmos um cenário próximo do real, removendo, alterando e até adicionando endereços para uma melhor compreensão de como são feitos os transportes de pacotes numa rede.

Em suma, com este projeto conseguimos perceber as dificuldades e a organização necessária dos vários componentes constituintes de uma rede, assim como a importância de cada um destes para o seu correto funcionamento.