

# **TP2: Protocolo IP**

Diogo Afonso Costa, Daniel Maia, and Vitor Castro

University of Minho, Department of Informatics, 4710-057 Braga, Portugal  
e-mail: {a78034,a77531,a77870}@alunos.uminho.pt

**Abstract.** Resumo...

## **1 Introdução**

## **2 Parte I - Datagramas e Fragmentação**

### **2.1 Exercício 1.b.**

**Questão**

**Resposta**

**Realização**

### **2.2 Exercício 1.c.**

**Questão**

**Resposta**

**Realização**

### **2.3 Exercício 1.d.**

**Questão**

**Resposta**

**Realização**

### **2.4 Exercício 2.a.**

**Questão**

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

## Resposta

O endereço IP é 192.168.100.216.

## Realização

43	3.903920123	192.168.100.216	192.168.100.216	UNDEF	519	Standard query response 0x7472 A marco.uminho.pt A 192.168.9.248 NS unsa
44	3.903910826	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=1/256, ttl=1 (no response found!)
45	3.903938117	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=2/512, ttl=1 (no response found!)
46	3.903947143	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=3/768, ttl=1 (no response found!)
47	3.903956246	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=4/1024, ttl=2 (no response found!)
48	3.903964672	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=5/1280, ttl=2 (no response found!)
49	3.903970621	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=6/1536, ttl=2 (no response found!)
50	3.903975159	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=7/1792, ttl=3 (reply in 63)
51	3.903982289	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=8/2048, ttl=3 (reply in 64)
52	3.903988913	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=9/2304, ttl=3 (reply in 66)
53	3.903994876	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=10/2560, ttl=4 (reply in 67)
54	3.904000785	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=11/2816, ttl=4 (reply in 68)
55	3.904009311	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=12/3072, ttl=4 (reply in 69)
56	3.904015173	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=13/3328, ttl=5 (reply in 70)
57	3.904020348	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=14/3584, ttl=5 (reply in 71)
58	3.904023767	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=15/3840, ttl=5 (reply in 72)
59	3.904032851	192.168.100.216	193.136.9.240	ICMP	74	Echo (ping) request id=0x3eal, seq=16/4096, ttl=6 (reply in 73)
60	3.904260758	192.168.100.216	192.168.100.216	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)

Fig. 1: Identificação do endereço IP.

## 2.5 Exercício 2.b.

### Questão

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

### Resposta

O campo protocolo tem o valor "ICMP (1)". ICMP significa *Internet Control Message Protocol*. Este é utilizado para reportar erros no processamento de datagramas. Efetivamente, dentro dos possíveis erros temos, *destination unreachable* (quando o datagrama não consegue alcançar o destino), *time exceeded message* (quando um *gateway* processa um datagrama e descobre que o *TTL* é zero e tem que descartar o datagrama e consequentemente notificar o *host*), *echo request/reply* (quando são enviadas mensagens para funções de teste e controle da rede (*request*), caso a máquina esteja ligada responde com um *reply*) [1] [2]. Como as mensagens ICMP encontram-se ao nível de rede, estas são também elas encapsuladas em datagramas IP que, consequentemente, usam o protocolo IP.

Assim sendo, analisando a primeira mensagem ICMP, nomeadamente no separador do *Internet Protocol Version 4*, percebemos que se trata de uma mensagem que vem num protocolo *ICMP*. Além disso, é possível concluir que se trata de uma mensagem de *echo request*, se observarmos o campo *Type* no separador *Internet Control Message Protocol*. Desta forma, pode-se concluir que o computador usado para a resolução deste trabalho está a tentar perceber se consegue estabelecer uma ligação com o *host* marco.uminho.pt e para isso usa mensagens *ICMP* do tipo *echo request*.

### Realização

```

Internet Protocol Version 4, Src: 192.168.100.216, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    ► Differentiated Services Field: 0x00 (DSCH: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0xb224 (45604)
    ► Flags: 0x00
    Fragment offset: 0
    Time to live: 1
    Protocol: ICMP (1)
        Header checksum: 0x16a4 [validation disabled]
        [Header checksum status: Unverified]
        Source: 192.168.100.216
        Destination: 193.136.9.240
        [Source GeoIP: Unknown]
        ► [Destination GeoIP: Portugal]
    Internet Control Message Protocol
        Type: 8 (Echo (ping) request)
            Code: 0
            Checksum: 0x43d8 [correct]
            [Checksum Status: Good]
            Identifier (BE): 16033 (0x3e1)
            Identifier (LE): 41278 (0xa13e)
            Sequence number (BE): 1 (0x0001)
            Sequence number (LE): 256 (0x0100)
        ► [No response seen]
    Data (32 bytes)

```

Fig. 2: Identificação do campo *Protocol*.

## 2.6 Exercício 2.c.

### Questão

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 IPv4 tem 20 bytes.  
O campo de dados (payload) do datagrama tem 40 bytes.  
O cálculo do payload é feito retirando o tamanho do cabeçalho ao tamanho total do datagrama (60 bytes). Desta forma, basta fazer  $60 - 20 = 40\text{bytes}$ .

### Realização

```

Internet Protocol Version 4, Src: 192.168.100.216, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    ► Differentiated Services Field: 0x00 (DSCH: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0xb224 (45604)
    ► Flags: 0x00
    Fragment offset: 0
    Time to live: 1
    Protocol: ICMP (1)
        Header checksum: 0x16a4 [validation disabled]
        [Header checksum status: Unverified]
        Source: 192.168.100.216
        Destination: 193.136.9.240
        [Source GeoIP: Unknown]
        ► [Destination GeoIP: Portugal]
    Internet Control Message Protocol

```

Fig. 3: Identificação do tamanho do *header* e do *payload*.

## 2.7 Exercício 2.d.

### Questão

O datagrama IP foi fragmentado? Justifique.

### Resposta

A fragmentação acontece quando o tamanho total do datagrama excede o *MTU* disponível. Tendo em conta que por defeito o *traceroute* usa 60 bytes por datagrama e tem-se um *MTU* disponível de 1500 bytes, podemos conjeturar que não haverá fragmentação.

A verificação se um datagrama foi ou não fragmentado é feita com base em dois valores, o *fragment offset* (indica o *offset* em que o datagrama atual encaixa no datagrama original) e a *flag more fragments* (indica se existe mais fragmentos). Neste datagrama em específico o *fragment offset* = 0 e a *flag more fragments* = 0. Desta forma, tendo em conta o *fragment offset*, sabe-se que se o datagrama foi fragmentado então ele é necessariamente o primeiro. Além disso, se analisarmos a *flag more fragments* concluímos que para além do datagrama atual não existe mais nenhum associado a este.

Assim sendo, conjugando a informação dos dois parâmetros percebe-se que se o datagrama é o primeiro e não existe mais nenhum associado, então este é único e não foi fragmentado.

### Realização

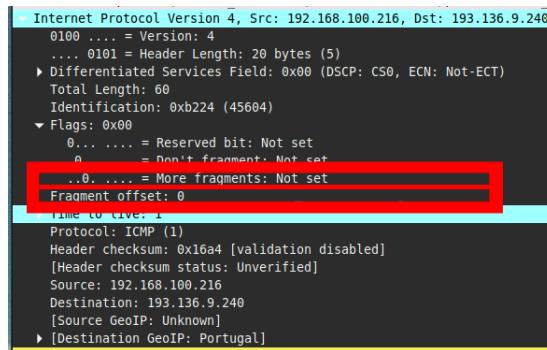


Fig. 4: Fragmentação.

## 2.8 Exercício 2.e.

### Questão

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 sua máquina. Para a sequência de mensagens ICMP enviadas pelo seu computador, indique que campos do cabeçalho IP variam de pacote para pacote.

## Resposta

Os campos que vêm os seus valores alterados correspondem à *identification*, *header checksum* e *time to live (TTL)*.

A *identificação* muda pois este campo identifica unicamente cada datagrama e visto que estes são sempre diferentes então o campo também o será.

O *header checksum* permite verificar que determinado header foi ou não corrompido. Desta forma o *checksum* identifica um determinado *header* num determinado estado. Assim sendo, o *checksum* muda pois este campo utiliza no seu algoritmo todas as palavras de 16 bits do *header* [1]. Efetivamente, sabendo que o *header*, propriamente dito, muda de datagrama para datagrama então o seu *checksum* também vai mudar.

O *TTL* muda pois a máquina que está a ser usada está a tentar contactar o *host* marco.uminho.pt. Começa por tentar com *TTL* = 1 e o pacote é descartado. Desta forma, vai aumentando o *TTL* até conseguir chegar ao *host* pretendido.

## Realização

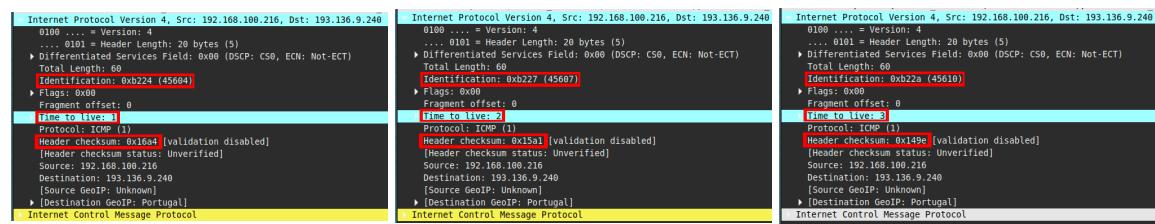


Fig. 5: Campos que mudam.

## 2.9 Exercício 2.f.

### Questão

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

## Resposta

O campo da indentificação corresponde a um valor que é incrementado e que identifica unicamente o datagrama em questão. Por exemplo, se o primeiro datagrama tiver *Identification* 0xb224 (45604), então o datagrama seguinte terá o valor 0xb225 (45605).

O TTL corresponde a uma variável que vai sendo decrementada sempre que é interse-tada por um *router*. Visto que na primeira mensagem o TTL é 1, o datagrama é descartado imediatamente no primeiro *router*. Deste modo, é enviado, de seguida, um novo datagrama com TTL a 2 com a esperança de que este chegue desta vez ao destino. Caso não chegue, então no próximo datagrama o TTL será aumentado, e assim sucessivamente, até chegar ao destino.

## Realização

A relação entre *TTL* pode ser observada na figura 8.

<pre> Internet Protocol Version 4, Src: 192.168.100.216, Dst: 193.136.9.248   0100 .... = Version: 4   .... 0101 = Header Length: 20 bytes (5)   ► Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)   Total Length: 60   Identification: 0xb224 (45604)   ► Flags: 0x00   Fragment offset: 0   Time to live: 1   Protocol: ICMP (1)   Header checksum: 0x16a4 [validation disabled]   [Header checksum status: Unverified]   Source: 192.168.100.216   Destination: 193.136.9.248   [Source GeoIP: Unknown]   ► [Destination GeoIP: Portugal]   Internet Control Message Protocol </pre>	<pre> Internet Protocol Version 4, Src: 192.168.100.216, Dst: 193.136.9.248   0100 .... = Version: 4   .... 0101 = Header Length: 20 bytes (5)   ► Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)   Total Length: 60   Identification: 0xb225 (45605)   ► Flags: 0x00   Fragment offset: 0   Time to live: 1   Protocol: ICMP (1)   Header checksum: 0x16a3 [validation disabled]   [Header checksum status: Unverified]   Source: 192.168.100.216   Destination: 193.136.9.248   [Source GeoIP: Unknown]   ► [Destination GeoIP: Portugal]   Internet Control Message Protocol </pre>
---	---

Fig. 6: Identificação.

## 2.10 Exercício 2.g.

### Questão

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

O IP 192.168.100.254 referencia a interface do primeiro router de acesso (visto ter o três primeiros campos iguais ao da máquina em que os testes estão a ser feitos e o último utiliza um número convencionado para ser utilizado para identificar a interface IP do router dentro da rede 192.168.100, na qual a máquina de testes se encontra).

Desta forma, quando analisamos os datagramas do 192.168.100.254 percebemos que estes têm todos o *TTL* = 64. O *TTL* toma um valor exageradamente elevado, devido ao desconhecimento que este tem da distância a que o *host* de destino se encontra. Por defeito, este router quando não tem informação da distância a que o *host* de destino se encontra envia datagramas com *TTL* = 64 e por isso todas os seus datagramas tem *TTL* igual.

Além disso, é possível concluir que as três mensagens recebidas deste router com *ICMP TTL exceeded* são a resposta ao envio feito pela máquina de testes de três datagramas de *echo (ping) request* com *TTL* apenas de 1. Estes datagramas chegaram ao *router* de acesso e ficaram com o *TTL* a 0 e a exceção veio enviado de no datagrama *ICMP TTL exceeded*.

Após estes datagramas é possível identificar um segundo conjunto de datagramas desta vez da interface IP 193.136.19.254. Pelo mesmo raciocínio podemos assumir que o IP desta interface identifica um *router*. Este novo *router* envia datagramas com *TTL* = 254 (constante). Este valor é considerável pela mesma razão explicitada anteriormente. Estes datagramas são a resposta a 3 datagramas enviados pela máquina de testes com *TTL* = 2, o que nos leva a concluir que é o segundo router por qual o nosso datagrama teve de passar para chegar ao marco.uminho.pt.

### Realização

## 2.11 Exercício 3.a.

### Questão

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

<pre> Internet Protocol Version 4, Src: 192.168.100.254, Dst: 192.168.100.216 0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) ► Differentiated Services Field: 0xc0 (DSGP: CS6, ECN: Not-ECT) Total Length: 88 Identification: 0xfef4c (61266) ► Flags: 0x00 Fragment offset: 0 Time to live: 64 Protocol: ICMP [1] Header checksum: 0x3f71 [validation disabled] [Header checksum status: Unverified] Source: 192.168.100.254 Destination: 192.168.100.216 [Source GeoIP: Unknown] [Destination GeoIP: Unknown] [Destination GeoIP: Unknown] Internet Control Message Protocol </pre>	<pre> Internet Protocol Version 4, Src: 192.168.100.254, Dst: 192.168.100.216 0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) ► Differentiated Services Field: 0xc0 (DSGP: CS6, ECN: Not-ECT) Total Length: 88 Identification: 0xef4d4 (61261) ► Flags: 0x00 Fragment offset: 0 Time to live: 64 Protocol: ICMP [1] Header checksum: 0x3f70 [validation disabled] [Header checksum status: Unverified] Source: 192.168.100.254 Destination: 192.168.100.216 [Source GeoIP: Unknown] [Destination GeoIP: Unknown] [Destination GeoIP: Unknown] Internet Control Message Protocol </pre>	<pre> Internet Protocol Version 4, Src: 192.168.100.254, Dst: 192.168.100.216 0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) ► Differentiated Services Field: 0xc0 (DSGP: CS6, ECN: Not-ECT) Total Length: 88 Identification: 0xef4e4 (61262) ► Flags: 0x00 Fragment offset: 0 Time to live: 64 Protocol: ICMP [1] Header checksum: 0x3f6f6 [validation disabled] [Header checksum status: Unverified] Source: 192.168.100.254 Destination: 192.168.100.216 [Source GeoIP: Unknown] [Destination GeoIP: Unknown] [Destination GeoIP: Unknown] Internet Control Message Protocol </pre>
---	---	--

Fig. 7: Router de acesso com  $TTL = 64$ .

<pre> Internet Protocol Version 4, Src: 193.136.19.254, Dst: 192.168.100.216 0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) ► Differentiated Services Field: 0xc0 (DSGP: CS6, ECN: Not-ECT) Total Length: 56 Identification: 0x1147 (4423) ► Flags: 0x00 Fragment offset: 0 Time to live: 254 Protocol: ICMP [1] Header checksum: 0xafb6 [validation disabled] [Header checksum status: Unverified] Source: 193.136.19.254 Destination: 192.168.100.216 [Source GeoIP: Portugal] [Destination GeoIP: Unknown] [Destination GeoIP: Unknown] Internet Control Message Protocol </pre>	<pre> Internet Protocol Version 4, Src: 193.136.19.254, Dst: 192.168.100.216 0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) ► Differentiated Services Field: 0xc0 (DSGP: CS6, ECN: Not-ECT) Total Length: 56 Identification: 0x1148 (4424) ► Flags: 0x00 Fragment offset: 0 Time to live: 254 Protocol: ICMP [1] Header checksum: 0xbfb5 [validation disabled] [Header checksum status: Unverified] Source: 193.136.19.254 Destination: 192.168.100.216 [Source GeoIP: Portugal] [Destination GeoIP: Unknown] [Destination GeoIP: Unknown] Internet Control Message Protocol </pre>	<pre> Internet Protocol Version 4, Src: 193.136.19.254, Dst: 192.168.100.216 0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) ► Differentiated Services Field: 0xc0 (DSGP: CS6, ECN: Not-ECT) Total Length: 56 Identification: 0x1149 (4425) ► Flags: 0x00 Fragment offset: 0 Time to live: 254 Protocol: ICMP [1] Header checksum: 0xbfb6 [validation disabled] [Header checksum status: Unverified] Source: 193.136.19.254 Destination: 192.168.100.216 [Source GeoIP: Portugal] [Destination GeoIP: Unknown] [Destination GeoIP: Unknown] Internet Control Message Protocol </pre>
---	---	---

Fig. 8: Segundo router com  $TTL = 254$ .

## Resposta

O pacote inicial tem um *total length* de 3033 bytes. O MTU (*Maximum Transmission Unit*) disponível é de apenas 1500 bytes e, como tal, o pacote inicial tem de ser fragmentado para que este possa ser transmitido na rede.

## Realização

No.	Time	Source	Destination	Protocol	Length	Info	
11	9.318600576	192.168.100.216	192.168.100.254	DNS	80	Standard query 0xa8ca A dl-debug.dropbox.com	
12	9.318606341	192.168.100.216	192.168.100.254	DNS	80	Standard query 0xbcdb AAAA dl-debug.dropbox.com CNAME block-debug.xdro	
13	9.656884877	192.168.100.216	192.168.100.254	DNS	80	Standard query 0x3d4f A 0x1147 0x1148 0x1149 debug.dropbox.com CNAME block-debug.xdro	
14	9.656884877	192.168.100.216	192.168.100.254	DNS	79	Standard query 0x3d4f A 0x1147 0x1148 0x1149 marco.uninho.pt	
15	9.649417934	192.168.100.216	192.168.100.254	DNS	75	Standard query 0x7e08 AAAA marco.uninho.pt	
16	9.650136243	192.168.100.254	192.168.100.216	DNS	347	Standard query response 0x3d4f A marco.uninho.pt A 193.136.9.248 NS dns3.uninho.pt	
17	9.656723657	192.168.100.254	192.168.100.216	DNS	129	Standard query response 0x7e08 AAAA marco.uninho.pt SOA dns.uninho.pt	
18	9.656885080	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, 10=813) [Reassembled in #20]	
19	9.656889397	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4, 10=813) [Reassembled in #20]	
*	20	9.656889397	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=8, 10=813) [Reassembled in #20]
21	9.656894641	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=12, 10=814) [Reassembled in #23]	
22	9.656897423	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=16, 10=814) [Reassembled in #23]	
23	9.656898295	192.168.100.216	193.136.9.248	ICMP	87	Echo (ping) request id=0xa4f3, seq=2/512, ttl=1 (no response found!)	
24	9.656904685	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, 10=815) [Reassembled in #26]	
25	9.656904685	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4, 10=815) [Reassembled in #26]	
26	9.656905537	192.168.100.216	193.136.9.248	ICMP	87	Echo (ping) request id=0xa4f3, seq=3/512, ttl=1 (no response found!)	
27	9.656910217	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, 10=816) [Reassembled in #29]	
28	9.656910217	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4, 10=816) [Reassembled in #29]	
29	9.656911393	192.168.100.216	193.136.9.248	ICMP	87	Echo (ping) request id=0xa4f3, seq=1/1024, ttl=2 (no response found!)	
> Frame 18: 1514 bytes on wire (12112 bits), 1514 bytes captured on interface 0							
> Ethernet II, Src: QnctecQo 0:0:85:44 (2c:60:0c:0:85:44), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)							
> Internet Protocol Version 4, Src: 192.168.100.216, Dst: 193.136.9.248							
0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) Differentiated Services Field: 0x00 (DSGP: CS0, ECN: Not-ECT) Total length: 1500 Identification: 0x4037 Flags: 0x01 (More Fragments) 0... .... = Reserved bit: Not set ...0.... = Don't Fragment: Not set ...1.... = More fragments: Set Fragment offset: 0 Time to live: 1 Protocol: ICMP [1] Header checksum: 0xfbfa [validation disabled] [Header checksum status: Unverified] Source: 192.168.100.216 Destination: 193.136.9.248 [Source GeoIP: Unknown]							

Fig. 9: A MTU é menor do que o tamanho do pacote.

## 2.12 Exercício 3.b.

### Questão

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

A informação de que o datagrama foi fragmentado é dada através da *flag more fragments* que se encontra a 1 em conjugação com o campo *fragment offset*, que neste caso se encontra a 0. Desta forma, podemos concluir que este datagrama é efetivamente o primeiro (*fragment offset = 0*) de um PDU fragmentado (*flag more fragments = 1*). Este é um datagrama IP de 1480 bytes (MTU de 1500 bytes, dos quais 20 bytes servem de *header*).

### Realização

No.	Time	Source	Destination	Protocol	Length	Info
11	9.318600576	192.168.100.216	192.168.100.254	DNS	80	Standard query 0xa8ca A d1-debug.dropbox.com
12	9.3186006341	192.168.100.216	192.168.100.254	DNS	80	Standard query 0xbcd4 AAAA d1-debug.dropbox.com
13	9.319254879	192.168.100.254	192.168.100.216	DNS	198	Standard query response 0xbcd4 AAAA d1-debug.dropbox.com CNAME block-debug.x.drc
14	9.649429727	192.168.100.216	192.168.100.254	DNS	75	Standard query 0x3dcf A marco.uminho.pt
15	9.649437058	192.168.100.216	192.168.100.254	DNS	75	Standard query 0x706f AAAA marco.uminho.pt
16	9.655723658	192.168.100.216	192.168.100.254	DNS	342	Standard query response 0x706f AAAA marco.uminho.pt A 193.136.0.240 NC dns5.uminho.pt
17	9.655723657	192.168.100.216	192.168.100.254	DNS	129	Standard query response 0x706f AAAA marco.uminho.pt SOA dns5.uminho.pt
18	9.656885085	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a813] (Reassembled in #20)
19	9.656885097	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=+1480, ID=a813] (Reassembled in #20)
20	9.656890359	192.168.100.216	193.136.9.248	ICMP	87	Echo (ping) request id=0xa4f3, seq=1/256, ttl=1 (no response found!)
21	9.656894646	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a814] (Reassembled in #23)
22	9.656894645	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=+1480, ID=a814] (Reassembled in #23)
23	9.656893255	192.168.100.216	193.136.9.248	ICMP	87	Echo (ping) request id=0xa4f3, seq=2/256, ttl=1 (no response found!)
24	9.656903505	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a815] (Reassembled in #26)
25	9.656904606	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=+1480, ID=a815] (Reassembled in #26)
26	9.656905558	192.168.100.216	193.136.9.248	ICMP	87	Echo (ping) request id=0xa4f3, seq=3/256, ttl=1 (no response found!)
27	9.656909521	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a816] (Reassembled in #29)
28	9.656910424	192.168.100.216	193.136.9.248	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=+1480, ID=a816] (Reassembled in #29)
29	9.656911306	192.168.100.216	193.136.9.248	ICMP	67	Echo (ping) request id=0xa4af3, seq=4/1024, ttl=2 (no response found!)
0100 0000 0000 0000 - Header length: 20 bytes (5) > Version: 4 > Header Length: 20 bytes (5) > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 1590 Identification: 0xa813 (43027) Flags: 0x01 (More Fragments) 0... .... - Reserved bit: Not set ...0... - More Fragment: Not set 1.... - More fragments: Set Fragment offset: 0 > Time to live: 1 Protocol: ICMP (1) Header checksum: 0xb14 [validated disabled] [Header checksum status: Unverified] [Header checksum value: 0xb14] Destination: 193.136.0.248 [Source GeoIP: Unknown] [Destination GeoIP: Unknown] Reassembled IPv4 in frame: 20 > Data (1480 bytes)						

Fig. 10: As flags e o fragment offset confirmam que é o primeiro fragmento.

### 2.13 Exercício 3.c.

#### Questão

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

Sabe-se que este não é o primeiro segmento pois o datagrama em questão tem *fragment offset* = 1480. Isto significa que, no datagrama original, este segmento começará a partir do byte 1480 do datagrama. Analisando a *flag more fragments*, percebe-se existem mais fragmentos, visto que esta é igual a 1.

No.	Time	Source	Destination	Protocol	Length	Info
11	9.3.18690576	192.168.100.216	192.168.100.254	DNS	88	Standard query 0x80ca A d1-debug.dropbox.com
12	9.3.18690591	192.168.100.216	192.168.100.254	DNS	88	Standard query 0x80ca AAAA d1-debug.dropbox.com
13	9.3.19254079	192.168.100.254	192.168.100.216	DNS	100	Standard query response 0x80d4 AAAA d1-debug.dropbox.com CNAME block-debug.x.dropbox.com
14	9.6.64942977	192.168.100.254	192.168.100.216	DNS	75	Standard query 0x3dcf A marco.uminho.pt
15	9.6.649437034	192.168.100.216	192.168.100.254	DNS	75	Standard query 0x79d8 AAAA marco.uminho.pt
16	9.6.650136248	192.168.100.254	192.168.100.216	DNS	347	Standard query response 0x3dcf A marco.uminho.pt A 193.136.9.24 NS dns3.uminho.pt
17	9.6.656723657	192.168.100.254	192.168.100.216	DNS	129	Standard query response 0x79d8 AAAA marco.uminho.pt SOA dns3.uminho.pt
18	9.6.656895865	192.168.100.216	193.136.9.240	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a813] [Reassembled in #20]
19	9.6.656895865	192.168.100.216	193.136.9.240	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a813] [Reassembled in #20]
20	9.6.656895865	192.168.100.216	193.136.9.240	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a814] [Reassembled in #21]
21	9.6.656896461	192.168.100.216	193.136.9.240	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a814] [Reassembled in #21]
22	9.6.656897423	192.168.100.216	193.136.9.240	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a814] [Reassembled in #23]
23	9.6.656898295	192.168.100.216	193.136.9.240	ICMP	87	Echo (ping) request id=0x4af3, seq=2/512, ttl=1 (no response found!)
24	9.6.656903585	192.168.100.216	193.136.9.240	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a815] [Reassembled in #26]
25	9.6.656904665	192.168.100.216	193.136.9.240	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a815] [Reassembled in #26]
26	9.6.656905155	192.168.100.216	193.136.9.240	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a815] [Reassembled in #26]
27	9.6.656909571	192.168.100.216	193.136.9.240	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=0, ID=a816] [Reassembled in #29]
28	9.6.656910428	192.168.100.216	193.136.9.240	IPv4	1514	Fragmented IP protocol [proto=ICMP 1, off=1480, ID=a816] [Reassembled in #29]
29	9.6.656911383	192.168.100.216	193.136.9.240	ICMP	87	Echo (ping) request id=0x4af3, seq=4/1024, ttl=2 (no response found!)

Frame 19: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0  
 Ethernet II, Src: Quantaco\_0:b1:85:44 (2c:60:8c:b1:85:44), Dst: Vmware\_42:19:f0 (00:0c:29:d2:19:f0)  
 ▾ Internet Protocol Version 4, Src: 192.168.100.216, Dst: 193.136.9.240  
 0100 ... = Version 4  
 .0... = Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECN)  
 Total Length: 1500  
 Identification: 0xa813 (43627)  
 ✓ Flags: 0x01 (More Fragments)  
 .0... .... = Reserve bit: Not set  
 .0... .... = Don't fragment: Not set  
 .1... = More fragments: Set  
 Fragment offset: 1480  
 ▾ Internet Layer:  
 Protocol: ICMP (1)  
 Header checksum: 0xfa5b [validation disabled]  
 [Header checksum status: Unverified]  
 Source: 192.168.100.216  
 Destination: 193.136.9.240  
 [Source GeoIP: Unknown]

Fig. 11: As flags e o fragment offset confirmam que é o segundo fragmento.

## Realização

### 2.14 Exercício 3.d.

#### Questão

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

#### Resposta

Analisando o parâmetro *identification* dos datagramas capturados pelo Wireshark, percebe-se que existem três datagramas com o mesmo valor o que revela que pertencem ao mesmo PDU (*identification* = 0xa813). Desta forma podemos concluir que o datagrama foi fragmentado duas vezes resultando em três fragmentos.

A deteção do último fragmento é feita através da análise da *flag more fragments*. Se esta estiver a 1 então existem mais segmentos. Caso contrário, o segmento em questão é o último.

#### Realização

### 2.15 Exercício 3.e.

#### Questão

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.

Fig. 12: Encontrando vários fragmentos com a mesma *identification*, é possível saber a sua ordem através do *fragment offset* e flags.

## Resposta

Os campos que mudam são *Total Length*, *more fragments* e *Fragment offset*.

No primeiro fragmento, o *Total Length* é igual ao MTU menos o *Header Length*, a flag *more fragments* é igual a 1 e o *Fragment offset* é igual a 0.

Entre o primeiro fragmento e o último fragmento não inclusives, o *Total Length* é igual ao MTU menos o *Header Length*, a flag *more fragments* é igual a 1 e o *Fragment offset* é igual a  $(MTU - Header Length) * (n - 1)$ , n sendo o n-ésimo fragmento do datagrama (n = 1, 2, ...).

No último fragmento, o *Total Length* é menor ou igual ao MTU, a flag *more fragments* é igual a 0 e o *Fragment offset* é igual a  $(MTU - Header Length) * (n - 1)$ , n sendo o número de fragmentos do datagrama.

## **Realização**

### 3 Parte II - Endereçamento e Encaminhamento IP

#### 3.1 Exercício 2.1.a

##### Questão

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

A máscara de rede utilizada é 255.255.255.0. Isto deve-se a facto de todos os IPv4 atribuídos aos diferentes componentes da rede terem /24 como quantidade de bits que identificam a rede.

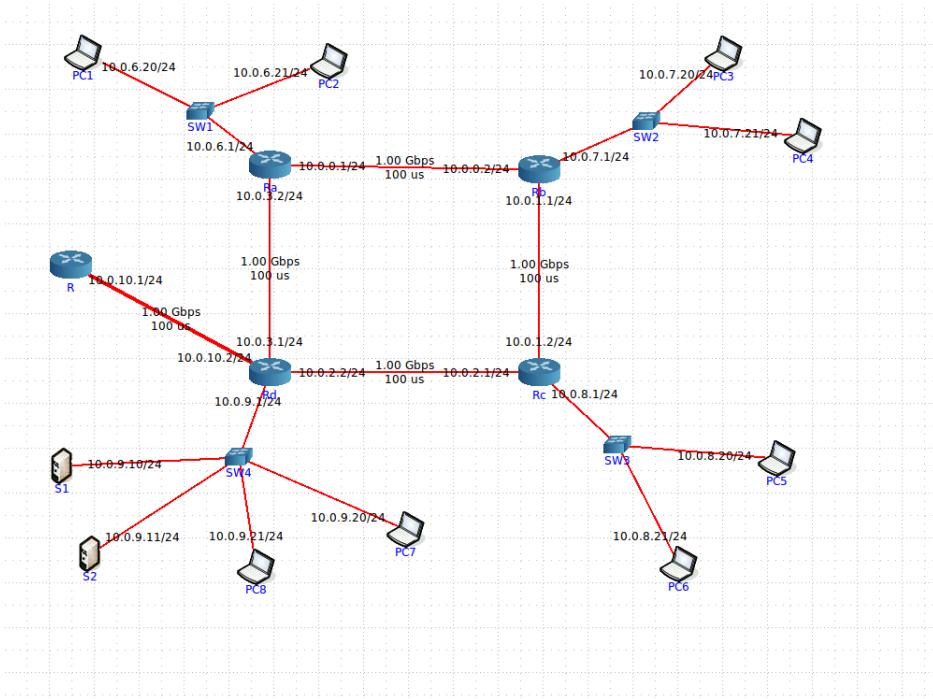


Fig. 13: Topologia da rede.

#### 3.2 Exercício 2.1.b

##### Questão

Tratam-se de endereços públicos ou privados? Porquê?

##### Resposta

Tratam-se de endereços privados pois o primeiro octeto é 00001010, que em decimal representa 10. Sendo este o primeiro octeto podemos afirmar que é um endereço IP privado visto que pertence ao conjunto de prefixos que identificam um endereçamento privado. Os conjuntos de endereços privados são os seguintes:

- Start address: 10.0.0.0 → End Address: 10.255.255.255
- Start address: 172.16.0.0 → End Address: 172.31.255.255
- Start address: 192.168.0.0 → End Address: 192.168.255.255
- Start address: 169.254.0.0 → End Address: 169.254.255.255

### **Realização**

Pela observação da topologia da rede 13.

### **3.3 Exercício 2.1.c**

#### **Questão**

Porque razão não é atribuído um endereço IP aos switches?

#### **Resposta**

Na topologia em questão se vier um pacote com destino ao PC7, este vem com endereço de destino de 10.0.9.20/24. Para tal, o pacote vem do *router* R e chega ao *router* Rd. O *router* Rd aplica a NetMask 255.255.255.0 ao endereço de destino e obtém 10.0.9.0. Consegue fazer match no endereço 10.0.9.1 e envia o pacote para o SW4. Ao chegar ao SW4, este avalia o endereço MAC de destino e duas situações podem acontecer. Se este tiver nos seus registos o endereço, então envia diretamente ao host especificado. Caso não tenha ainda essa informação envia para todos os hosts a ele ligados.

Efetivamente, os *switches* não operam ao nível de rede e como tal a camada de rede, nomeadamente o IP, é completamente transparente para estes dispositivos.

Através do exemplo a cima podemos concluir que não é necessário que este tenha IP pois os pacotes são enviados para o *switch* com base no IP da interface de saída do *router*. Após o datagrama chegar ao *switch* este quanto muito usa endereços MAC por forma a fazer *forward* do datagrama para o *host* respetivo.

### **3.4 Exercício 2.1.d**

#### **Questão**

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

## Resposta

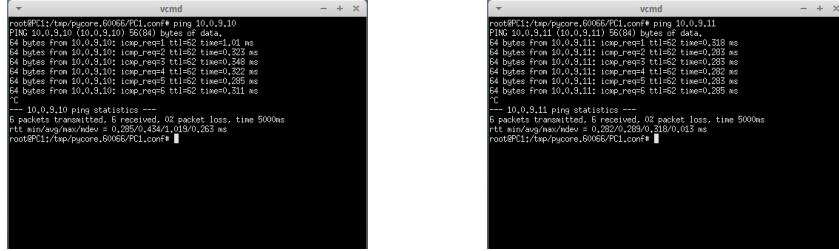


Fig. 14: Teste de conectividade entre PC1 e os servidores S1 (à esquerda) e S2 (à direita).

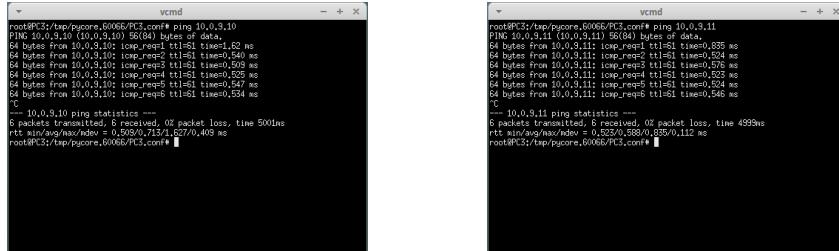


Fig. 15: Teste de conectividade entre PC3 e os servidores S1 (à esquerda) e S2 (à direita).

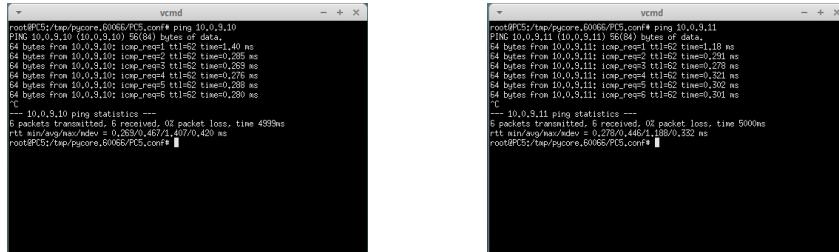


Fig. 16: Teste de conectividade entre PC5 e os servidores S1 (à esquerda) e S2 (à direita).

Tendo em conta os resultados do comando *ping* pode-se afirmar que existe conectividade entre os *hosts* dos diferentes departamentos e os servidores do departamento D.

### 3.5 Exercício 2.1.e

## Questão

Verifique se existe conectividade IP do router de acesso para os servidores S1 e S2.

## Resposta

```

root@PC7:/tmp/pycore_60065/PC7.conf# ping 10.0.9.10
PING 10.0.9.10 (10.0.9.10) 64(48) bytes of data.
64 bytes from 10.0.9.10: icmp_seq=1 ttl=64 time=0.028 ms
64 bytes from 10.0.9.10: icmp_seq=2 ttl=64 time=0.028 ms
64 bytes from 10.0.9.10: icmp_seq=3 ttl=64 time=0.028 ms
64 bytes from 10.0.9.10: icmp_seq=4 ttl=64 time=0.028 ms
64 bytes from 10.0.9.10: icmp_seq=5 ttl=64 time=0.028 ms
64 bytes from 10.0.9.10: icmp_seq=6 ttl=64 time=0.028 ms
64 bytes from 10.0.9.10: icmp_seq=7 ttl=64 time=0.028 ms
...
-- 10.0.9.10 ping statistics --
6 packets transmitted, 6 received, 0% packet loss, time 499ms
rtt min/avg/max/mdev = 0.028/0.028/0.028 ms
root@PC7:/tmp/pycore_60065/PC7.conf#
root@PC7:/tmp/pycore_60065/PC7.conf# ping 10.0.9.11
PING 10.0.9.11 (10.0.9.11) 64(48) bytes of data.
64 bytes from 10.0.9.11: icmp_seq=1 ttl=64 time=0.024 ms
64 bytes from 10.0.9.11: icmp_seq=2 ttl=64 time=0.024 ms
64 bytes from 10.0.9.11: icmp_seq=3 ttl=64 time=0.023 ms
64 bytes from 10.0.9.11: icmp_seq=4 ttl=64 time=0.023 ms
64 bytes from 10.0.9.11: icmp_seq=5 ttl=64 time=0.027 ms
64 bytes from 10.0.9.11: icmp_seq=6 ttl=64 time=0.028 ms
64 bytes from 10.0.9.11: icmp_seq=7 ttl=64 time=0.028 ms
...
-- 10.0.9.11 ping statistics --
7 packets transmitted, 7 received, 0% packet loss, time 599ms
rtt min/avg/max/mdev = 0.023/0.029/0.029 ms
root@PC7:/tmp/pycore_60065/PC7.conf#

```

Fig. 17: Teste de conectividade entre PC7 e os servidores S1 (à esquerda) e S2 (à direita).

```

root@R:/tmp/pycore_60065/R.conf# ping 10.0.9.10
PING 10.0.9.10 (10.0.9.10) 64(48) bytes of data.
64 bytes from 10.0.9.10: icmp_seq=1 ttl=63 time=0.261 ms
64 bytes from 10.0.9.10: icmp_seq=2 ttl=63 time=0.261 ms
64 bytes from 10.0.9.10: icmp_seq=3 ttl=63 time=0.299 ms
64 bytes from 10.0.9.10: icmp_seq=4 ttl=63 time=0.299 ms
64 bytes from 10.0.9.10: icmp_seq=5 ttl=63 time=0.299 ms
64 bytes from 10.0.9.10: icmp_seq=6 ttl=63 time=0.271 ms
64 bytes from 10.0.9.10: icmp_seq=7 ttl=63 time=0.271 ms
...
-- 10.0.9.10 ping statistics --
6 packets transmitted, 6 received, 0% packet loss, time 499ms
rtt min/avg/max/mdev = 0.261/0.324/0.574/0.113 ms
root@R:/tmp/pycore_60065/R.conf#
root@R:/tmp/pycore_60065/R.conf# ping 10.0.9.11
PING 10.0.9.11 (10.0.9.11) 64(48) bytes of data.
64 bytes from 10.0.9.11: icmp_seq=1 ttl=63 time=0.265 ms
64 bytes from 10.0.9.11: icmp_seq=2 ttl=63 time=0.265 ms
64 bytes from 10.0.9.11: icmp_seq=3 ttl=63 time=0.265 ms
64 bytes from 10.0.9.11: icmp_seq=4 ttl=63 time=0.289 ms
64 bytes from 10.0.9.11: icmp_seq=5 ttl=63 time=0.289 ms
64 bytes from 10.0.9.11: icmp_seq=6 ttl=63 time=0.274 ms
64 bytes from 10.0.9.11: icmp_seq=7 ttl=63 time=0.274 ms
...
-- 10.0.9.11 ping statistics --
6 packets transmitted, 6 received, 0% packet loss, time 500ms
rtt min/avg/max/mdev = 0.265/0.292/0.348/0.056 ms
root@R:/tmp/pycore_60065/R.conf#

```

Fig. 18: Teste de conectividade entre R e os servidores S1 (à esquerda) e S2 (à direita).

Tendo em conta os resultados obtidos em 19, podemos afirmar que existe conectividade entre o *router* R e os servidores S1 e S2.

### 3.6 Exercício 2.2.a

#### Questão

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

A primeira linha na tabela de endereçamento do PC3 representa a rota por defeito, isto é, quando entre no PC3 um datagrama com um IP que não faz *match* com nenhum IP da coluna *Destination* então este é reencaminhado para o *router* de acesso, ou seja, 10.0.7.1 ficando este responsável pelo seu encaminhamento. Denotando que a primeira linha, por ser um endereço estático, tem a menor prioridade. Esta linha faz *match* com qualquer endereço pois tem uma *Genmask* igual a 0.0.0.0. Efetivamente, quando chega à máquina PC3, por exemplo, um datagrama com o IP 10.0.8.20, este é comparado com todas as linhas da tabela. Posteriormente é aplicada a máscara obtendo-se 0.0.0.0. O resultado é depois comparado com o campo *Destination* e faz *match*. De seguida é enviado para o *router* de acesso explicitado pelo campo *Gateway*.

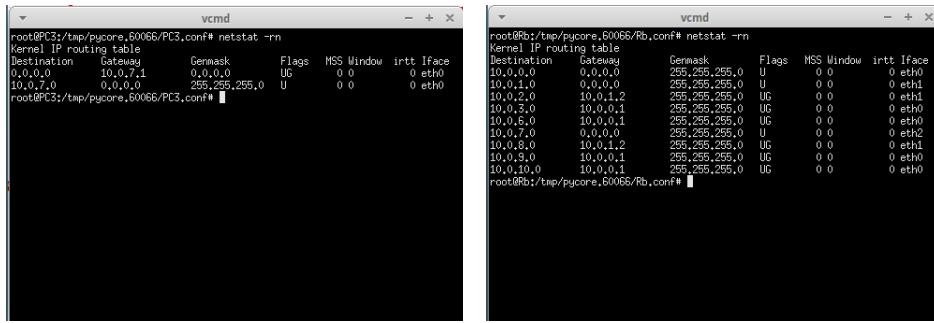
A segunda linha é responsável por encaminhar, através da própria máquina, o trânsito local, ou seja, o próprio PC3 encaminha os datagramas para *hosts* vizinhos. Deste modo, todos os datagramas que tiverem como prefixo de rede 10.0.7 vão fazer *match* na segunda linha, isto porque, chegando por exemplo o datagrama com o IP 10.0.7.21, é aplicada a máscara 255.255.255.0 e resulta a *destination* 10.0.7.0. Posteriormente é enviado para o endereço dito no *Gateway*, 0.0.0.0, ou seja, fica responsável por encaminhar este datagrama a própria máquina. Este processo de encaminhamento resulta pois os *hosts* tem conhecimento da topologia da rede local.

A tabela de encaminhamento do *router* Rb não incluí rotas por defeito pois todas as rotas existentes na rede encontram-se identificadas na tabela à partida.

Através da análise da topologia na rede criada com o CORE percebemos que o *router* Rb deverá ser responsável por encaminhar para a rede 10.0.0.0, 10.0.1.0, e 10.0.7.0. A tabela de encaminhamento confirma as nossas suspeitas, pois todas as linhas que têm os endereços IP atrás indicados têm como *Gateway* 0.0.0.0, que significa que o próprio *router* fica responsável por encaminhar esses datagramas.

Todas as outras redes existentes encontram-se na tabela com o endereço de encaminhamento apropriado. Nomeadamente, caso os datagramas tenham como destino a rede 10.0.x.0, em que  $x=2,3,6,8,9,10$ , estes são reencaminhados para uma das duas redes a que o *router* Rb tem acesso, isto é, 10.0.0.1 ou 10.0.1.2. Estas entradas em específico na tabela têm a flag G no campo *Flags*, visto não ser um encaminhamento direto.

### Realização



The figure consists of two side-by-side terminal windows. Both windows have a title bar 'vcmd' and a command prompt at the bottom. The left window shows the routing table for PC3, with the command 'root@PC3:/tmp/pycore\_60066/PC3.conf# netstat -rn'. The output is:

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
0.0.0.0	10.0.7.1	0.0.0.0	UG	0 0	0	0	eth0
10.0.7.0	0.0.0.0	255.255.255.0	U	0 0	0	0	eth0

The right window shows the routing table for Rb, with the command 'root@Rb:/tmp/pycore\_60066/Rb.conf# netstat -rn'. The output is:

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.0.0	0.0.0.0	255.255.255.0	U	0 0	0	0	eth0
10.0.0.0	0.0.0.0	255.255.255.0	U	0 0	0	0	eth1
10.0.3.0	10.0.1.2	255.255.255.0	UC	0 0	0	0	eth1
10.0.3.0	10.0.1.1	255.255.255.0	UC	0 0	0	0	eth0
10.0.6.0	10.0.0.1	255.255.255.0	UG	0 0	0	0	eth0
10.0.7.0	0.0.0.0	255.255.255.0	U	0 0	0	0	eth2
10.0.8.0	10.0.1.2	255.255.255.0	UG	0 0	0	0	eth1
10.0.9.0	10.0.0.1	255.255.255.0	UG	0 0	0	0	eth0
10.0.10.0	10.0.0.1	255.255.255.0	UG	0 0	0	0	eth0

Fig. 19: Tabelas de encaminhamento do PC3 (à esquerda) e do Rb (à direita).

### 3.7 Exercício 2.2.b

#### Questão

#### Resposta

#### Realização

### 3.8 Exercício 2.2.c

#### Questão

#### Resposta

#### Realização

**3.9 Exercício 2.2.d**

**Questão**

**Resposta**

**Realização**

**3.10 Exercício 2.2.e**

**Questão**

**Resposta**

**Realização**

**3.11 Exercício 3.1**

**Questão**

**Resposta**

**Realização**

**3.12 Exercício 3.2**

**Questão**

**Resposta**

**Realização**

**3.13 Exercício 3.3**

**Questão**

(a) Delay and jitter	(b) Delay and loss
(c) Delay and throughput	(d) Jitter and loss
(e) Jitter and throughput	(f) Loss and throughput

Fig. 20: Tabela exemplo.

## Resposta

### Realização

According to Table 20...

## 4 Conclusions

Neste trabalho...

## References

1. : Internet Control Message Protocol. RFC 792 (1981)
2. Wikipedia: Internet control message protocol. [https://pt.wikipedia.org/wiki/Internet\\_Control\\_Message\\_Protocol](https://pt.wikipedia.org/wiki/Internet_Control_Message_Protocol) (2017) [Online; accessed a 4-Novembro-2017].