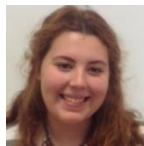




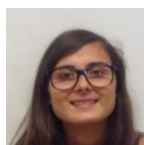
**Universidade do Minho**  
Escola de Engenharia

**Redes de Computadores:**  
Protocolo IPv4

**Grupo de Trabalho 2**



Ana Esmeralda Fernandes A74321



Bárbara Nadine Freitas Oliveira A75614



Miguel Dias Miranda A74726

**Novembro de 2016**



## **Conteúdo**

Datagramas IP e fragmentação .....	3
Captura de tráfego IP – Questão I .....	3
Captura de tráfego IP - Questão II .....	5
Captura de tráfego IP – Questão III .....	9
Endereçamento e Encaminhamento IP .....	12
Endereçamento e Encaminhamento IP – Questão I .....	12
Endereçamento e Encaminhamento IP – Questão II .....	14
Definição de Sub-Redes .....	19
Conclusão .....	22

## Datagramas IP e fragmentação

### Captura de tráfego IP – Questão I

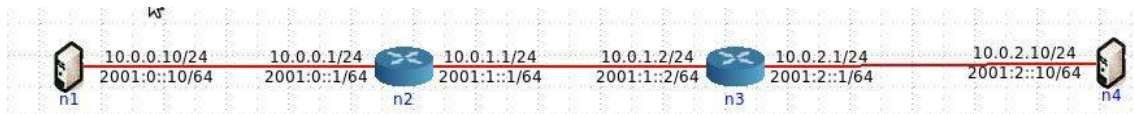


Figura 1- Topologia inicial do sistema

a) Active o wireshark ou o tcpdump no host n4. Numa shell de n4, execute o comando `traceroute -I` para o endereço IP do host n1.

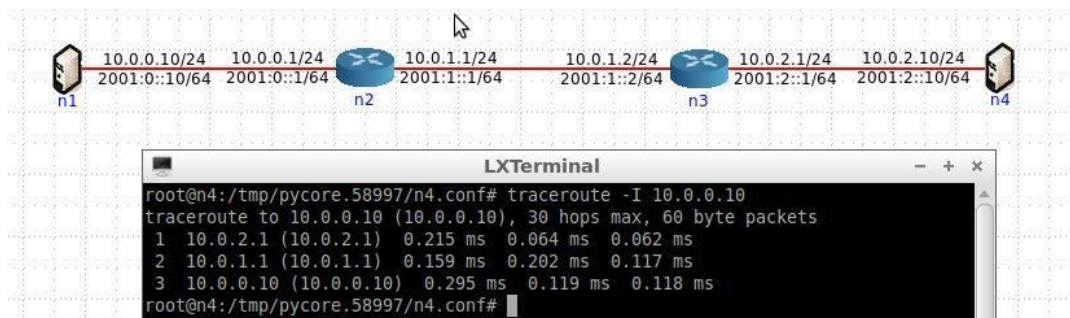


Figura 2 - Execução, numa shell de n4, do comando `'traceroute -I 10.0.0.10'`, onde o endereço IP usado no comando é referente ao host n1

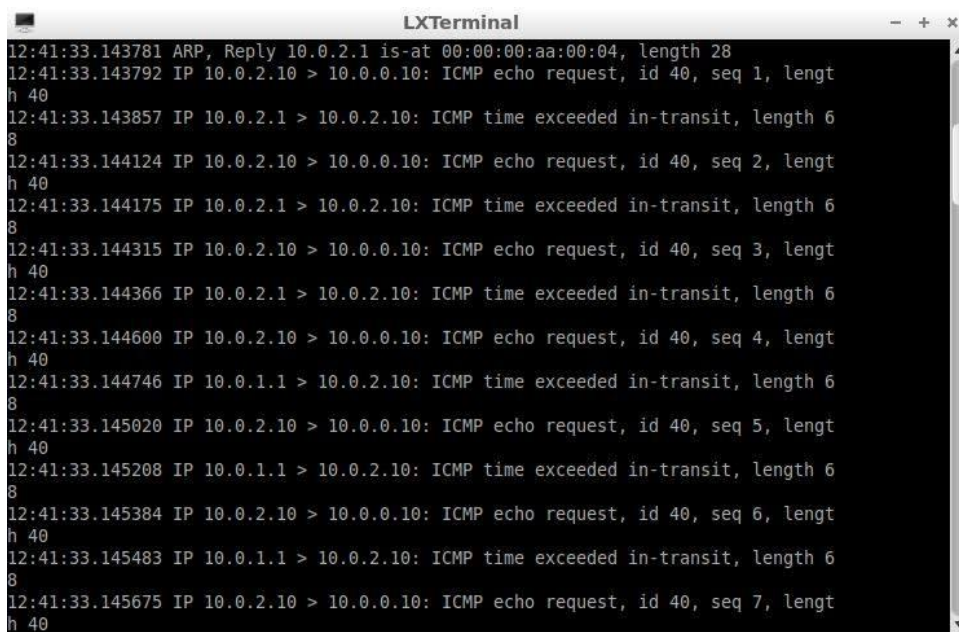


Figura 3 - Análise, numa shell de n4, do comando `tcpdump`

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

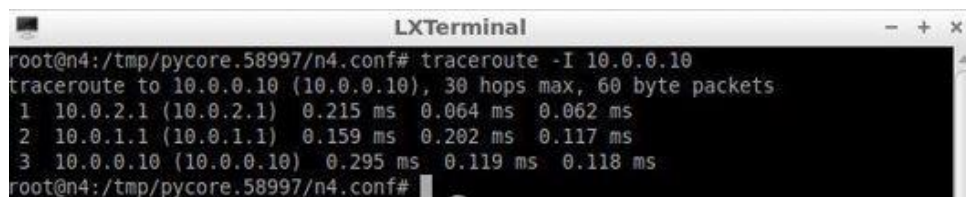
Por parte do host n4 o tráfego ICMP gerado caracteriza-se sempre por ser pedidos 'ICMP request' e o tráfego ICMP recebido varia entre pacotes do tipo 'ICMP time-exceeded', quando o valor do campo TTL não é suficiente para que o pacote dê os "saltos" necessários até ao host destino, ou pacotes ICMP do tipo 'ICMP Reply' quando o pacote chega ao seu destino e este confirma a n4 a sua receção.

Quando o host n4 recebe pacotes do tipo 'ICMP time exceeded' o seu procedimento é enviar ao mesmo destino um novo 'ICMP request' mas com o valor do campo TTL incrementado em 1 unidade.

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

O valor mínimo do campo TTL devia de ser 3 de forma a conseguir chegar até ao seu destino. Podemos verificar que este valor faz sentido, pois pelo sistema da rede que esboçamos no CORE vemos que existem 3 ligações/saltos que são necessários fazer para que o pacote enviado por n4 chegue a n1.

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



```
LXTerminal
root@n4:/tmp/pycore.58997/n4.conf# traceroute -I 10.0.0.10
traceroute to 10.0.0.10 (10.0.0.10), 30 hops max, 60 byte packets
 1 10.0.2.1 (10.0.2.1) 0.215 ms 0.064 ms 0.062 ms
 2 10.0.1.1 (10.0.1.1) 0.159 ms 0.202 ms 0.117 ms
 3 10.0.0.10 (10.0.0.10) 0.295 ms 0.119 ms 0.118 ms
root@n4:/tmp/pycore.58997/n4.conf#
```

Figura 4 - Resultado Traceroute em n4

Como queremos analisar o tempo de ida-e-volta até ao host n1, cujo endereço IP é 10.0.0.10, usamos os valores registados na linha 3 do comando traceroute da imagem anterior.

Assim, o tempo médio será  $\frac{0.295+0.119+0.118}{3} = 0.1773$ .

## Captura de tráfego IP - Questão II

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

O endereço IP da interface ativa do computador usado é 192.168.100.158.

```
> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
> Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.19.20
> Internet Control Message Protocol
```

Figura 5 - O endereço IP da interface ativa do computador usado é 192.168.100.158.

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

O campo protocolo tem o valor 1 que identifica o ICMP foi usado via IP.

```
> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
▼ Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.19.20
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x6398 (25496)
  > Flags: 0x00
    Fragment offset: 0
    Time to live: 255
    Protocol: ICMP (1)
    Header checksum: 0x5e49 [validation disabled]
```

Figura 6 - O campo protocolo tem o valor 1 que identifica o ICMP foi usado via IP.

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

O cabeçalho do IPv4 tem 20 bytes.

Como nesta captura foi usado o tamanho do pacote definido por omissão pelo PingPlotter de 56 bytes, o campo de dados do datagrama tem 36 bytes.

Este payload calcula-se subtraindo-se ao tamanho do pacote os 20 bytes ocupados pelo header do IPv4 ( $56 - 20 = 36$ ).

```
▼ Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.19.20
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
```

Figura 7 - Tamanho cabeçalho IPv4

```
> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
▼ Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.19.20
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x6398 (25496)
```

Figura 8 – Tamanho total do datagrama

**d. O datagrama foi fragmentado? Justifique**

O datagrama IP da frame 1 estudada não foi fragmentado. Pela análise do campo Flags da opção IPv4 vemos que o valor que contem é 0x00. Isto indica que a opção Don't fragment está marcada como Not Set.

Alem disto, por analise da questão anterior, vemos que o datagrama IPv4 mais o cabeçalho deste protocolo tem exatamente o tamanho máximo de 56 bytes configurado por omissão no PingPlotter, logo, não existe razões para fragmentar este pacote.

```
> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on i
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_d2
▼ Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.19.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x6398 (25496)
    ▼ Flags: 0x00
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
```

Figura 9 - Valor do campo Flags na opção IPv4

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1348/17413, ttl=255 (reply in 2)
3	0.050604	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1349/17669, ttl=1 (no response found!)
5	0.101007	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1350/17925, ttl=2 (reply in 6)
15	2.501019	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1351/18181, ttl=255 (reply in 16)
17	2.551545	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1352/18437, ttl=1 (no response found!)
19	2.601489	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1353/18693, ttl=2 (reply in 20)
23	5.002154	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1354/18949, ttl=255 (reply in 24)
25	5.052550	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1355/19205, ttl=1 (no response found!)
27	5.103050	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1356/19461, ttl=2 (reply in 28)
34	7.502355	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1357/19717, ttl=255 (reply in 35)
36	7.553055	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1358/19973, ttl=1 (no response found!)
38	7.603229	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1359/20229, ttl=2 (reply in 39)
42	10.002672	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1360/20485, ttl=255 (reply in 43)
44	10.053195	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1361/20741, ttl=1 (no response found!)
46	10.103341	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1362/20997, ttl=2 (reply in 47)
60	12.503629	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1363/21253, ttl=255 (reply in 61)
62	12.554009	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1364/21509, ttl=1 (no response found!)
64	12.604841	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1365/21765, ttl=2 (reply in 65)
70	15.004034	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1366/22021, ttl=255 (reply in 71)
72	15.054127	192.168.100.158	193.136.19.20	ICMP	70	Echo (ping) request id=0x0001, seq=1367/22277, ttl=1 (no response found!)

Figura 10 - Vista geral dos pacotes capturados com origem no IP do portátil usado no estudo.

```
> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_00:0c:29:1c:72:b3 (08:00:27:1c:72:b3)
> Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.145.101
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x6398 (25496)

> Frame 3: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_00:0c:29:1c:72:b3 (08:00:27:1c:72:b3)
> Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.145.101
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x6399 (25497)

> Frame 5: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_00:0c:29:1c:72:b3 (08:00:27:1c:72:b3)
> Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.145.101
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x639a (25498)

> Frame 15: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_00:0c:29:1c:72:b3 (08:00:27:1c:72:b3)
> Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.145.101
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x639b (25499)
```

Figura 11 - Valores do campo Identification da opção IPv4 de diferentes pacotes ICMP

De algumas frames analisadas é possível ver que o campo identification varia de pacote para pacote, algo que faz sentido porque este ID deve ser único e ter o maior período de 'vida' possível. Quando este ID é especificado, juntamente com os valores do campo Flags acerca da possível fragmentação do pacote, é possível no destino ser feita a montagem dos pacotes fragmentados a partir do ID e do número do fragmento.

```
> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_00:0c:29:1c:72:b3 (08:00:27:1c:72:b3)
> Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.145.101
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x6398 (25496)
  > Flags: 0x00
  Fragment offset: 0
  Time to live: 255
  Protocol: ICMP (1)
  Header checksum: 0x5e49 [validation disabled]

> Frame 3: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_00:0c:29:1c:72:b3 (08:00:27:1c:72:b3)
> Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.145.101
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x6399 (25497)
  > Flags: 0x00
  Fragment offset: 0
  Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x5c49 [validation disabled]

> Frame 5: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_00:0c:29:1c:72:b3 (08:00:27:1c:72:b3)
> Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.145.101
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x639a (25498)
  > Flags: 0x00
  Fragment offset: 0
  Time to live: 2
  Protocol: ICMP (1)
  Header checksum: 0x5b48 [validation disabled]

> Frame 15: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_00:0c:29:1c:72:b3 (08:00:27:1c:72:b3)
> Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.145.101
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x639b (25499)
  > Flags: 0x00
  Fragment offset: 0
  Time to live: 255
  Protocol: ICMP (1)
  Header checksum: 0x5e46 [validation disabled]
```

Figura 12 Valor do campo Header checkSum da opção IPv4 para diferentes pacotes ICMP

Ainda dentro da opção IPv4 é possível ver para diferentes frames que o campo Header Checksum é diferente. Este campo é usado para verificar a integridade ou existência de erros nos dados do pacote.



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

<p>&gt; Frame 1: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x6398 (25496)</p>	<p>&gt; Frame 3: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x6399 (25497)</p>	<p>&gt; Frame 5: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x639a (25498)</p>	<p>&gt; Frame 15: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x639b (25499)</p>
<p>&gt; Frame 17: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x639c (25500)</p>	<p>&gt; Frame 19: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x639d (25501)</p>	<p>&gt; Frame 23: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x639e (25502)</p>	<p>&gt; Frame 25: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x639f (25503)</p>
<p>&gt; Frame 27: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x63a0 (25504)</p>	<p>&gt; Frame 34: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x63a1 (25505)</p>	<p>&gt; Frame 36: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x63a2 (25506)</p>	<p>&gt; Frame 38: 70 bytes on wire (560 bit)</p> <p>&gt; Ethernet II, Src: AsustekC_21:3e:f1, Dst: AsustekC_21:3e:f1</p> <p>&gt; Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 b</p> <p>&gt; Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 56</p> <p>Identification: 0x63a3 (25507)</p>

Figura 13 - Valor do campo ID de algumas tramas ICMP com origem no computador usado para o estudo.

Pela observação de alguns frames do tipo ICMP e do valor do seu campo ID é possível concluir que estes são incrementados sequenciais de uma unidade face ao valor do mesmo campo no pacote anterior.

**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 enviadas ao seu host? Porquê?**

```
Total Length: 84
Identification: 0x8c8d (35981)
> Flags: 0x00
Fragment offset: 0
Time to live: 64
```

Figura 14 - Valor do campo TTL

O campo TTL apresenta o valor 64.

Sim, este valor permanece constante quando as frames têm como destino o endereço o computador usado. Isto acontece porque o router a que este está ligado é sempre o mesmo, logo, o número de passos até as mensagens chegarem até ele é o mesmo.



### Captura de tráfego IP – Questão III

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

Com o uso do PingPlotter definiu-se o tamanho do pacote ICMP para 4022Bytes. Com esta alteração, como o tamanho máximo do pacote por rede cablada Ethernet (usada para realizar a captura) é de 1500Bytes existe então a necessidade de fragmentar o pacote para que este possa ser enviado pelos protocolos usados na rede Ethernet.

```
> Frame 9: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_d2:19:f0 (00:0c:29:19:f0:00)
v Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.19.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x617c (24956)
    v 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)
```

Figura 15 - análise do primeiro pacote fragmentado com origem no IP da máquina usada para a captura

**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?**

Pela opção 'More fragments' do campo 'Flags' é possível ver que este campo tem o valor 1, que indica que a opção de fragmentação está definida como 'Set'. Pelo valor do campo 'Fragment offset' é possível ver que se trata do primeiro fragmento, pois é colocado no offset 0, ou seja, na posição inicial quando se for reconstruir o pacote inicial.

Sabemos que o datagrama tem 1500Bytes de tamanho, visto no campo 'total length' sendo que destes 20 Bytes são para o cabeçalho IP.

```
> Frame 9: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_d2:19:f0 (00:0c:29:19:f0:00)
v Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.19.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x617c (24956)
    v 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)
```

Figura 16 - Informações sobre o primeiro pacote fragmentado da nossa captura

- 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?**

Como o campo 'Fragment offset' tem o valor 1480, quando o pacote inicial for reconstruído, este datagrama em análise deve ser colocado na posição/offset 1480. Assim este não é o datagrama inicial porque esse vai ocupar desde a posição 0 até à posição 1480.

Embora os pacotes tenham um tamanho máximo de 1500Bytes, como 20bytes são usados para o cabeçalho do protocolo, só 1480 é que são efetivamente dados para reconstrução do pacote original.

Existem mais fragmentos porque a opção 'More Fragments' volta a estar definida com o valor 1 e, portanto, existem mais fragmentos.

```
> Frame 10: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 b
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_d2
v Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.19.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x617c (24956)
v Flags: 0x01 (More Fragments)
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    Fragment offset: 1480
> Time to live: 1
    Protocol: ICMP (1)
```

*Figura 17 - informações sobre o segundo fragmento*

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

Foram criados 3 fragmentos a partir do datagrama inicial. Além da análise dos Datagramas que de facto irão compor o fragmento inicial, facilmente podíamos concluir isto porque sabendo que o tamanho máximo dos pacotes é de 1500 bytes e o nosso pacote tem 4022bytes, serão então precisos 3 fragmentos para o dividir.

O primeiro e o segundo fragmentos transportarão ambos 1480 bytes de dados e o último fragmento, transportará apenas 1062.

```
> Frame 11: 1076 bytes on wire (8608 bits), 1076 bytes captured (8608 bit)
> Ethernet II, Src: AsustekC_21:3e:62 (30:5a:3a:21:3e:62), Dst: Vmware_d2
> Internet Protocol Version 4, Src: 192.168.100.158, Dst: 193.136.19.20
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 1062
  Identification: 0x617c (24956)
> Flags: 0x00
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..0. .... = More fragments: Not set
  Fragment offset: 2960
> Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x5906 [validation disabled]
```

Figura 18 - informações sobre o ultimo datagrama do pacote em análise

Pelo valor do campo 'More fragments', que tem o valor 0, sabemos que já não vão haver mais fragmentos referentes a este Id.

**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.**

O valor do campo 'Total Length' é diferente nos datagramas fragmentados de um mesmo pacote. Este valor não é sempre 1500Bytes (limite máximo da rede Ethernet) pois o ultimo fragmento, regra geral, tem sempre um tamanho inferior a 1500Bytes, com os últimos dados do pacote a reconstruir.

Exatamente por causa deste processo de reconstrução dos pacotes originais fragmentados para envio pela rede cablada, mudam também os campos 'Fragment offset' e 'Flags'. O campo 'Fragment offset' muda porque cada datagrama corresponde a um fragmento que deve ser colocado numa posição específica para reconstruir o pacote original e, portanto, aqui vai sempre surgir um valor diferente. O campo 'Flags' varia pois este indica se vão ou não existir mais fragmentos para aquele id.

Dentro do protocolo IP varia também o valor do campo Header Checksum pois este tem é usado para verificar a integridade dos dados e existência de erros.

## Endereçamento e Encaminhamento IP

### Nota

Aquando da realização da seguinte questão I e II sobre endereçamento e encaminhamento IP, por má interpretação do enunciado, ligamos o servidor do departamento A e uma interface do router do referido departamento e não ao switch onde estão ligados os laptops da rede do departamento. Só reparamos que o servidor do departamento A devia também ele estar ligado ao switch deste departamento quando chegamos á questão III e à implementação de sub-redes no sistema.

Por falta de tempo, e por concluirmos que o teor das respostas seria o mesmo, alteramos a disposição da arquitetura CORE apenas na questão III, sobre sub-redes, e mantivemos a arquitetura com origem na interpretação errada do enunciado para a realização da questão I e II.

### Endereçamento e Encaminhamento IP – Questão I

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

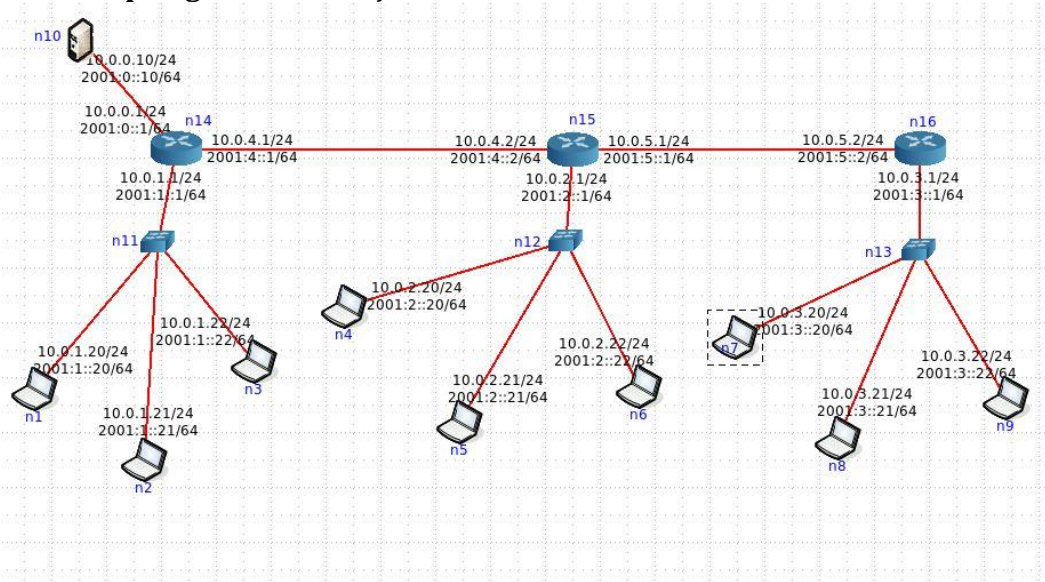


Figura 19 - Esquema da arquitetura CORE (Ler NOTA inicial)

Os routers n14, n15 e n16 representam, respetivamente, os routers dos departamentos A, B e C.

De igual modo, os switches n11, n12 e n13 representam os switches que se ligam aos routers do seu departamento. Cada switch tem a si ligado três computadores.

De referir ainda o host (servidor) n10 que pertence ao departamento A e se liga ao router do referido departamento.



Os endereços de cada interface podem ser vistos no primeiro valor e o endereço da máscara no segundo endereço (em baixo do primeiro). Por exemplo, para o computador n1, o seu endereço é 10.0.1.20 e o seu endereço máscara é 2001.1::20

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

Tratam-se de endereços privados porque se encontra na gama de endereços de 10.0.0.0 até 10.255.255.25 (redes com endereços 10.0.0.0/8).

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

Como os switches usados estão ligados a uma rede Ethernet estes apenas trocam e distribuem pacotes Ethernet entre os destinos finais e o router a que estão ligados. Neste nível de ligação não existem endereços IP e, portanto, em geral os switches não apresentam endereços IP.

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

Por análise do resultado do comando ping, feito a partir de um computador de cada sistema (departamento) até ao Host (servidor) n10 vemos que todos os pacotes enviados são recebidos e, portanto, existem conectividade entre os computadores dos diferentes departamentos e o servidor do departamento A.

```
64 bytes from 10.0.0.10: icmp_req=22 ttl=62 time=0.230 ms
64 bytes from 10.0.0.10: icmp_req=23 ttl=62 time=0.167 ms
64 bytes from 10.0.0.10: icmp_req=24 ttl=62 time=0.186 ms
64 bytes from 10.0.0.10: icmp_req=25 ttl=62 time=0.150 ms
^C
--- 10.0.0.10 ping statistics ---
25 packets transmitted, 25 received, 0% packet loss, time 23998ms
rtt min/avg/max/mdev = 0.148/0.193/0.480/0.071 ms
root@n4:/tmp/pycore.59029/n4.conf#
```

Figura 20 - comando ping do computador n4 (departamento B) para o host n10 do departamento A

```
64 bytes from 10.0.0.10: icmp_req=21 ttl=61 time=0.213 ms
64 bytes from 10.0.0.10: icmp_req=22 ttl=61 time=0.182 ms
64 bytes from 10.0.0.10: icmp_req=23 ttl=61 time=0.267 ms
64 bytes from 10.0.0.10: icmp_req=24 ttl=61 time=0.184 ms
64 bytes from 10.0.0.10: icmp_req=25 ttl=61 time=0.184 ms
^C
--- 10.0.0.10 ping statistics ---
25 packets transmitted, 25 received, 0% packet loss, time 24000ms
rtt min/avg/max/mdev = 0.182/0.217/0.517/0.072 ms
root@n7:/tmp/pycore.59029/n7.conf#
```

Figura 21- comando ping do computador n7 (departamento C) para o host n10 do departamento A

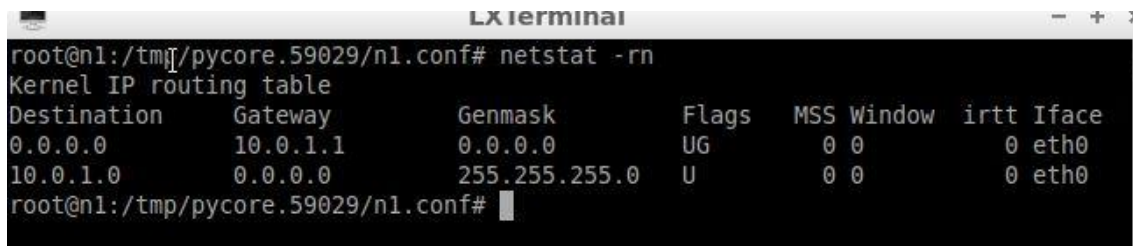
```
64 bytes from 10.0.0.10: icmp_req=17 ttl=63 time=0.121 ms
64 bytes from 10.0.0.10: icmp_req=18 ttl=63 time=0.115 ms
64 bytes from 10.0.0.10: icmp_req=19 ttl=63 time=0.159 ms
^C
--- 10.0.0.10 ping statistics ---
19 packets transmitted, 19 received, 0% packet loss, time 18005ms
rtt min/avg/max/mdev = 0.113/0.152/0.380/0.065 ms
root@n1:/tmp/pycore.59029/n1.conf#
```

Figura 22- comando ping do computador n1 (departamento A) para o host n10 do departamento A

## Endereçamento e Encaminhamento IP – Questão II

- 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`).**

Para a atual questão usamos como objeto de estudo o computador representado por n1 e o router do departamento A representado pelo router n14.



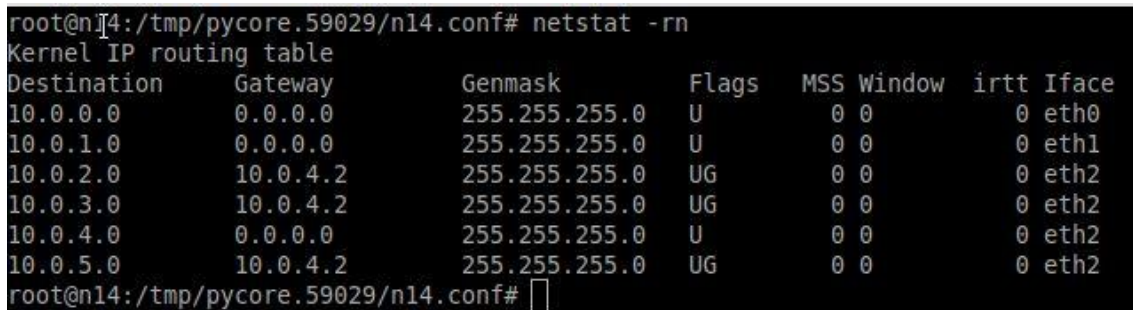
```

root@n1:/tmp/pycore.59029/n1.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask        Flags   MSS Window  irtt Iface
0.0.0.0          10.0.1.1        0.0.0.0        UG          0  0        0 eth0
10.0.1.0         0.0.0.0         255.255.255.0  U          0  0        0 eth0
root@n1:/tmp/pycore.59029/n1.conf#

```

Figura 23 - Tabela de encaminhamento do computador n1 da rede do Departamento A

Na primeira linha representa-se o procedimento a tomar quando o portátil recebe um pacote com IP default (0.0.0.0). Quando não existe encaminhamento definido na tabela, o caminho por defeito do pacote é ir para o router da rede, com o endereço 10.0.1.1 (endereço da interface do router a que o portátil está ligado) e este trata de encaminhar o pacote conforme o encaminhamento da sua tabela. Quando o destino é o endereço 10.0.1.0 não é especificado nenhum encaminhamento porque o destino é o próprio computador.



```

root@n14:/tmp/pycore.59029/n14.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          0.0.0.0         255.255.255.0  U          0  0        0 eth1
10.0.2.0          10.0.4.2        255.255.255.0  UG          0  0        0 eth2
10.0.3.0          10.0.4.2        255.255.255.0  UG          0  0        0 eth2
10.0.4.0          0.0.0.0         255.255.255.0  U          0  0        0 eth2
10.0.5.0          10.0.4.2        255.255.255.0  UG          0  0        0 eth2
root@n14:/tmp/pycore.59029/n14.conf#

```

Figura 24 - Tabela de encaminhamento do router do Departamento A

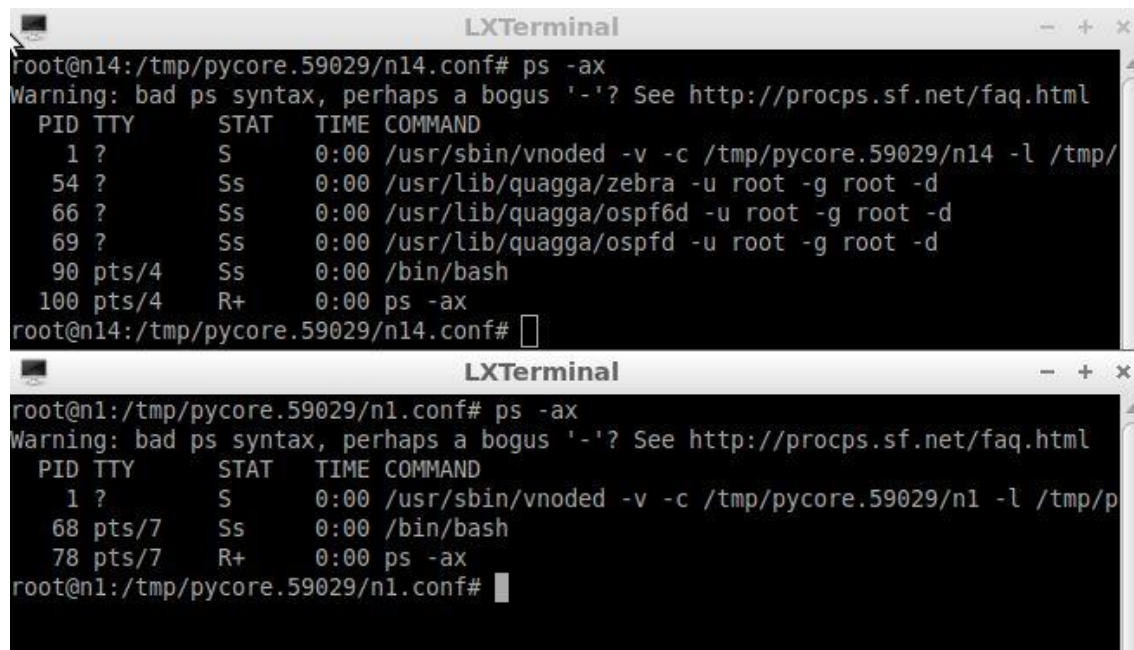
Quando o router do departamento A recebe pacotes cujo destino têm como endereço os endereços das suas interfaces ( IPs 10.0.0.0, 10.0.1.0, 10.0.4.0 ) não é feito nenhum encaminhamento porque o destino é o próprio router.

Quando o destino é o IP 10.0.2.0 (endereços dos computadores da rede do departamento B) os pacotes são enviados para a interface que liga o router B ao router A, neste caso com o endereço 10.0.4.2. Será depois o router B, com a sua tabela de distribuição que encaminhará os pacotes até ao destino final.

Quando o destino é o endereço 10.0.5.0 ( endereço da interface que liga o router B ao router C) ou os endereços 10.0.3.0 (endereços dos computadores que compõe a rede do router C) o encaminhamento é feito para a interface do router B com o endereço 10.0.4.2 (interface que liga o router A ao router B). Será depois, de forma semelhante, o router B a encaminhar os pacotes até ao router C e este, por sua vez, a encaminhar os pacotes para o seu destino final na sua rede.

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

Pela análise dos processos que o router e o computador executam concluímos que o router apresenta encaminhamento dinâmico, quer pela sua tabela de encaminhamento quer pelo número de processo que executa e o computador, pelo tráfego mais específico que recebe e por não ter processos a correr, apresenta encaminhamento estático.



```

root@n14:/tmp/pycore.59029/n14.conf# ps -ax
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
  PID TTY          STAT       TIME COMMAND
    1 ?            S          0:00 /usr/sbin/vnoded -v -c /tmp/pycore.59029/n14 -l /tmp/
   54 ?            Ss         0:00 /usr/lib/quagga/zebra -u root -g root -d
   66 ?            Ss         0:00 /usr/lib/quagga/ospfd -u root -g root -d
   69 ?            Ss         0:00 /usr/lib/quagga/ospfd -u root -g root -d
   90 pts/4        Ss         0:00 /bin/bash
  100 pts/4        R+         0:00 ps -ax
root@n14:/tmp/pycore.59029/n14.conf#

root@n1:/tmp/pycore.59029/n1.conf# ps -ax
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
  PID TTY          STAT       TIME COMMAND
    1 ?            S          0:00 /usr/sbin/vnoded -v -c /tmp/pycore.59029/n1 -l /tmp/p
   68 pts/7        Ss         0:00 /bin/bash
   78 pts/7        R+         0:00 ps -ax
root@n1:/tmp/pycore.59029/n1.conf#

```

Figura 25- Execução do comando 'ps -ax' no router n14 do departamento A e no computador n1 da rede do departamento A, respetivamente

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

Ao apagar a linha que indicava o encaminhamento a tomar para todos os endereços IP marcados como default (ou não conhecidos pelo host), o host perde a ligação com a interface do router do departamento A que estava ligado (e a quem encaminhava estes pacotes), e portanto, não consegue comunicar com nenhum router ou computador de qualquer um dos departamentos.

Ao realizar um ping do host n10 para um computador escolhido aleatoriamente de cada uma das redes dos diferentes departamentos, vemos que a mensagem não chega ao destino e é marcada com a mensagem "Network is unreachable". De igual modo, fazer ping a partir de um dos computadores das várias redes para o host n10, apesar de serem enviados nenhum pacote é recebido de volta na origem do ping.



```

root@n10:/tmp/pycore.59033/n10.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@n10:/tmp/pycore.59033/n10.conf# route delete default
root@n10:/tmp/pycore.59033/n10.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@n10:/tmp/pycore.59033/n10.conf#

```

Figura 26 - Remoção da linha que continha o procedimento de encaminhamento para a rota por defeito

```

root@n10:/tmp/pycore.59033/n10.conf# ping 10.0.2.20
connect: Network is unreachable
root@n10:/tmp/pycore.59033/n10.conf# ping 10.0.3.22
connect: Network is unreachable
root@n10:/tmp/pycore.59033/n10.conf# ping 10.0.1.21
connect: Network is unreachable
root@n10:/tmp/pycore.59033/n10.conf#

```

Figura 27 - Realização do comando ping no servidor n10 do departamento A para os computadores n4 (na rede do departamento B), n9 (na rede do departamento C) e n2 (na rede do departamento A). Todos os pedidos recebem a mensagem 'Network is unreachable' porque o router não sabe para onde encaminhar estes endereços que vão para um destino que não o seu endereço.

**d. Adicione as rotas estáticas necessárias para restaurar a conectividade para o servidor, por forma a contornar a restrição imposta em c). Utilize para o efeito o comando `route add` e registe os comandos que usou.**

Para reestabelecer o encaminhamento do servidor n10 foi usado o comando 'route add -net [endereço\_dest] netmask 255.255.255.0 gw [endereço\_gw]'.

Como o host n10 só está ligado a uma interface, em específico a uma do router do departamento A, o endereço do campo gateway será sempre o endereço 10.0.0.1 dessa mesma interface do router, que posteriormente encaminhará o tráfego até ao destino.

Como estamos a trabalhar com endereços do tipo x/24, a máscara de rede é o endereço 255.255.255.0 pois só o último octeto é que caracteriza o host.

Criaram-se assim três rotas, que definem o encaminhamento a realizar quando o endereço destino é, respetivamente, as redes 10.0.1.0 (do departamento A), 10.0.2.0 (do departamento B) e 10.0.3.0 (do departamento C).

Comandos usados:

'route add 10.0.1.0 netmask 255.255.255.0 gw 10.0.0.1' - encaminha o tráfego destinado para a rede do departamento A.

'route add 10.0.2.0 netmask 255.255.255.0 gw 10.0.0.1' - encaminha o tráfego destinado para a rede do departamento B.

'route add 10.0.3.0 netmask 255.255.255.0 gw 10.0.0.1' - encaminha o tráfego destinado para a rede do departamento C.

```

root@n10:/tmp/pycore.59038/n10.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@n10:/tmp/pycore.59038/n10.conf# route add -net 10.0.1.0 netmask 255.255.255
.0 gw 10.0.0.1
root@n10:/tmp/pycore.59038/n10.conf# route add -net 10.0.2.0 netmask 255.255.255
.0 gw 10.0.0.1
root@n10:/tmp/pycore.59038/n10.conf# route add -net 10.0.3.0 netmask 255.255.255
.0 gw 10.0.0.1
root@n10:/tmp/pycore.59038/n10.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

```

Figura 28 - comandos usados para restaurar a conectividade do servidor com o router e a rede

- 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.**

Para testar se o servidor estava novamente acessível, realizamos neste o comando ping para um computador na rede de cada um dos departamentos. Pelos resultados vimos que o ping foi bem-sucedido e portanto o servidor está novamente acessível.

```

root@n10:/tmp/pycore.59038/n10.conf# ping 10.0.1.20
PING 10.0.1.20 (10.0.1.20) 56(84) bytes of data.
64 bytes from 10.0.1.20: icmp_req=1 ttl=63 time=0.341 ms
64 bytes from 10.0.1.20: icmp_req=2 ttl=63 time=0.374 ms
64 bytes from 10.0.1.20: icmp_req=3 ttl=63 time=0.374 ms
^C
--- 10.0.1.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.341/0.363/0.374/0.015 ms

```

Figura 29 - Ping do servidor para o computador n1 (na rede do departamento A)

```

root@n10:/tmp/pycore.59038/n10.conf# ping 10.0.2.20
PING 10.0.2.20 (10.0.2.20) 56(84) bytes of data.
64 bytes from 10.0.2.20: icmp_req=1 ttl=62 time=0.381 ms
64 bytes from 10.0.2.20: icmp_req=2 ttl=62 time=0.158 ms
64 bytes from 10.0.2.20: icmp_req=3 ttl=62 time=0.300 ms
^C
--- 10.0.2.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.158/0.279/0.381/0.094 ms

```

Figura 30 - Ping do servidor para o computador n4 (na rede do departamento B)



```
root@n10:/tmp/pycore.59038/n10.conf# ping 10.0.3.20
PING 10.0.3.20 (10.0.3.20) 56(84) bytes of data.
64 bytes from 10.0.3.20: icmp_req=1 ttl=61 time=0.473 ms
64 bytes from 10.0.3.20: icmp_req=2 ttl=61 time=0.831 ms
64 bytes from 10.0.3.20: icmp_req=3 ttl=61 time=0.649 ms
^C
--- 10.0.3.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2124ms
rtt min/avg/max/mdev = 0.473/0.651/0.831/0.146 ms
```

*Figura 31 - Ping do servidor para o computador n7 (na rede do departamento C)*

## Definição de Sub-Redes

- 1) **Assumindo que dispõe apenas de um único endereço de rede IP classe C 192.168.128.0/24, defina um novo esquema de endereçamento para as redes dos departamentos (mantendo a rede de core inalterada) e atribua endereços às interfaces dos vários sistemas envolvidos. Deve justificar as opções usadas.** Com o endereço 192.168.128.0/24 que nos foi atribuído temos à nossa responsabilidade a gestão de um octeto, por forma a ter endereços suficientes para as interfaces dos router e para os utilizadores ou hosts do sistema. No nosso esquema CORE, identificamos 3 redes e, portanto, precisamos de uma divisão do octeto que permita identificar pelo menos 3 endereços de sub-redes.

Dividir o octeto em 3 bits para a máscara de subrede e 5 bits para a identificação do host, permite ter exatamente 6 sub-redes e cerca de 30 dispositivos ligados a cada subrede. Como na eventualidade de no futuro serem necessárias mais de 6 sub-redes, esta divisão já estaria no seu limite e seria necessária uma nova gestão e reimplementação de todos os endereços da rede. Assim, optamos por uma divisão do octeto em 4 bits para a subrede e 4 bits para identificar as interfaces. Apesar de isto só permitir cerca de 14 ligações em cada subrede, permite que novas sub-redes sejam criadas e estruturadas no futuro, sem alterar a implementação atual.

192.168.128.0/24 -> 8bits livres para gestão

192.168.128.      |       
 Identificar Identificar  
 subrede interfaces host  
 $2^4 - 2 = 14$   $2^4 - 2 = 14$

0000	0001	0010	0011	0100	0101	0110	0111
Reservado	Subrede 1	Subrede 2	Subrede 3	Subrede 4	Subrede 5	Subrede 6	Subrede 7
1000	1001	1010	1011	1100	1101	1110	1111
Subrede 8	Subrede 9	Subrede 10	Subrede 11	Subrede 12	Subrede 13	Subrede 14	Reservado

Às sub-redes do nosso sistema atribuímos os 3 primeiros valores da tabela apresentada anteriormente. Os endereços são do tipo x/28 porque dos 24 que anteriormente identificavam a rede, existem mais 4 reservados para identificar as sub-redes.

Subrede 1 – 192.168.128.16/28

Subrede 3 – 192.168.128.48/28

Subrede 2 – 192.168.128.32/28

A subrede 1 terá a gama de endereços uteis para interfaces desde 192.168.128.17 até 192.168.128.31;

A subrede 2 terá a gama de endereços uteis para interfaces desde 192.168.128.33 até 192.168.128.47;

A subrede 3 terá a gama de endereços uteis para interfaces desde 192.168.128.49 até 192.168.128.63;

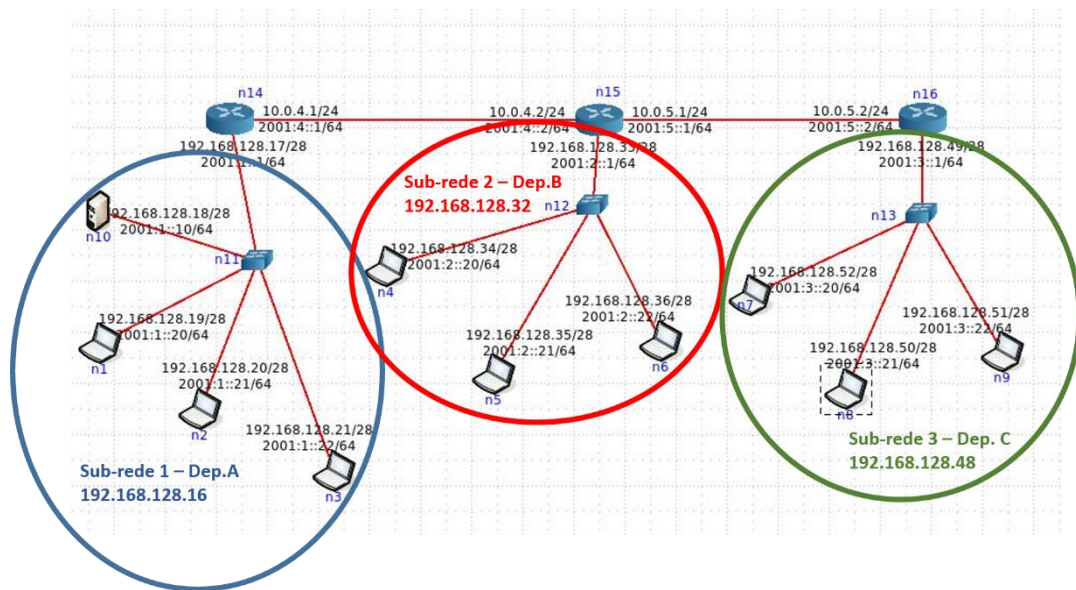
A distribuição das redes é ilustrada pela seguinte figura com o modelo CORE

Departamento A com a sub-rede 192.168.128.16

Departamento B com sub-rede 192.168.128.32

Departamento C com sub-rede 192.168.128.48





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

Como os endereços são do tipo x/28 (explicado na questão 1 deste grupo) os endereços máscara da rede são 255.255.255.240.

Como existem 4 bits do ultimo octeto a definir as sub-redes  $2^7 + 2^6 + 2^5 + 2^4 = 240$

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

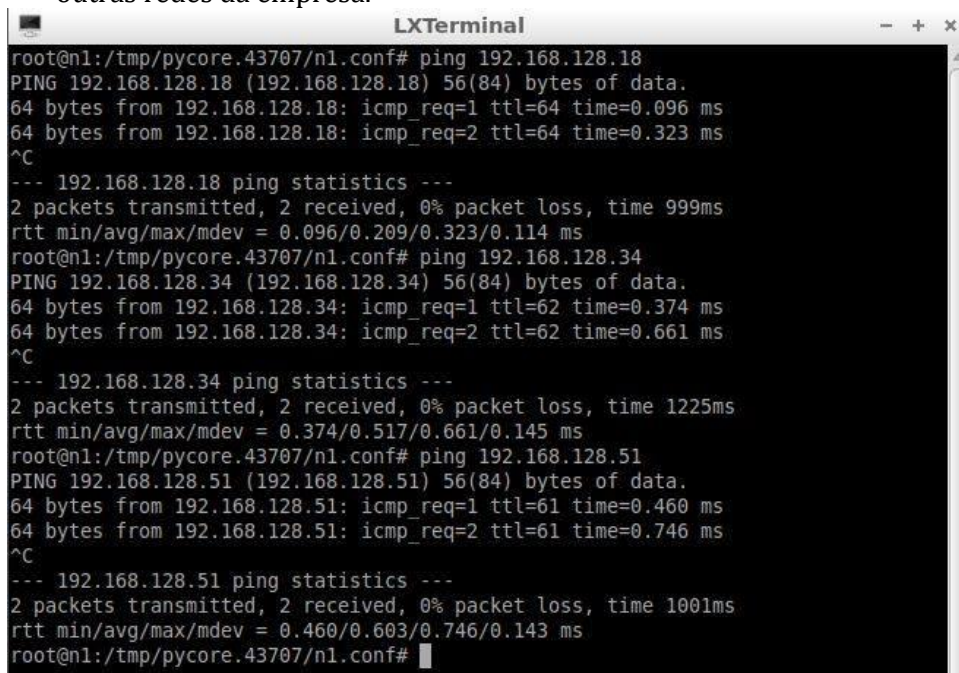
Para cada interface dos router dos departamentos, que terão a sua própria subrede, podem ser ligados até 14 hosts.

Por isso, um departamento pode ter no máximo  $14 \cdot N$ , onde N é o numero de interfaces do router ativas e a ser usadas num dado departamento.

Como cada departamento tem apenas uma interface ligada a uma subrede, e como na nossa divisão cada subrede só pode ter até 14 host ligado a ela, então cada departamento só pode ter 14 host ip interligados na sua rede.

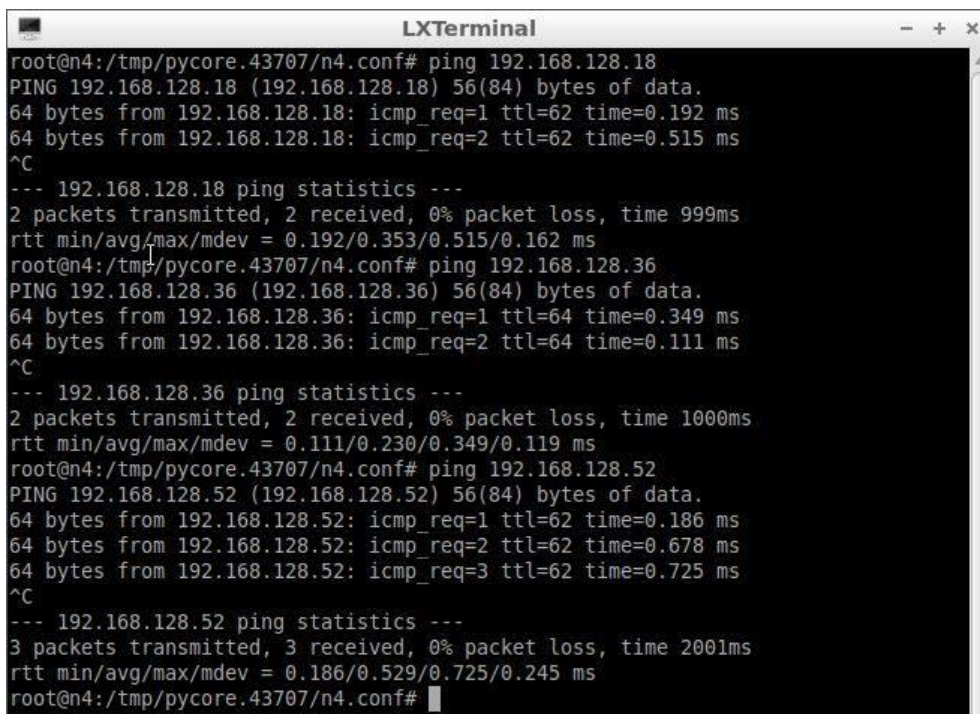
**4) Garanta que conectividade IP entre as várias redes locais da empresa MIEInet é mantida.**

Para garantir a conectividade IP entre as várias redes locais realizamos o comando ping em diferentes computadores (para cada rede) para outros computadores de outras redes da empresa.



```
root@n1:/tmp/pycore.43707/n1.conf# ping 192.168.128.18
PING 192.168.128.18 (192.168.128.18) 56(84) bytes of data.
64 bytes from 192.168.128.18: icmp_req=1 ttl=64 time=0.096 ms
64 bytes from 192.168.128.18: icmp_req=2 ttl=64 time=0.323 ms
^C
--- 192.168.128.18 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.096/0.209/0.323/0.114 ms
root@n1:/tmp/pycore.43707/n1.conf# ping 192.168.128.34
PING 192.168.128.34 (192.168.128.34) 56(84) bytes of data.
64 bytes from 192.168.128.34: icmp_req=1 ttl=62 time=0.374 ms
64 bytes from 192.168.128.34: icmp_req=2 ttl=62 time=0.661 ms
^C
--- 192.168.128.34 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1225ms
rtt min/avg/max/mdev = 0.374/0.517/0.661/0.145 ms
root@n1:/tmp/pycore.43707/n1.conf# ping 192.168.128.51
PING 192.168.128.51 (192.168.128.51) 56(84) bytes of data.
64 bytes from 192.168.128.51: icmp_req=1 ttl=61 time=0.460 ms
64 bytes from 192.168.128.51: icmp_req=2 ttl=61 time=0.746 ms
^C
--- 192.168.128.51 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.460/0.603/0.746/0.143 ms
root@n1:/tmp/pycore.43707/n1.conf#
```

Figura 32 – Do computador n1 do departamento A: Realização de um ping para o servidor do departamento A, para o computador n4 do departamento B e para o computador n9 do departamento C. O comando foi realizado com sucesso e por isso a conectividade foi mantida.



```
root@n4:/tmp/pycore.43707/n4.conf# ping 192.168.128.18
PING 192.168.128.18 (192.168.128.18) 56(84) bytes of data.
64 bytes from 192.168.128.18: icmp_req=1 ttl=62 time=0.192 ms
64 bytes from 192.168.128.18: icmp_req=2 ttl=62 time=0.515 ms
^C
--- 192.168.128.18 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.192/0.353/0.515/0.162 ms
root@n4:/tmp/pycore.43707/n4.conf# ping 192.168.128.36
PING 192.168.128.36 (192.168.128.36) 56(84) bytes of data.
64 bytes from 192.168.128.36: icmp_req=1 ttl=64 time=0.349 ms
64 bytes from 192.168.128.36: icmp_req=2 ttl=64 time=0.111 ms
^C
--- 192.168.128.36 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.111/0.230/0.349/0.119 ms
root@n4:/tmp/pycore.43707/n4.conf# ping 192.168.128.52
PING 192.168.128.52 (192.168.128.52) 56(84) bytes of data.
64 bytes from 192.168.128.52: icmp_req=1 ttl=62 time=0.186 ms
64 bytes from 192.168.128.52: icmp_req=2 ttl=62 time=0.678 ms
64 bytes from 192.168.128.52: icmp_req=3 ttl=62 time=0.725 ms
^C
--- 192.168.128.52 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.186/0.529/0.725/0.245 ms
root@n4:/tmp/pycore.43707/n4.conf#
```

Figura 33 - Do computador n4 do departamento B: Realização de um ping para o servidor do departamento A, para o computador n6 do departamento B e para o computador n7 do departamento C. O comando foi realizado com sucesso e por isso a conectividade foi mantida.



## Conclusão

Com a realização da parte I deste trabalho, foi possível compreender o processo de fragmentação para o envio de Datagramas de grandes dimensões sobre a rede Ethernet, onde o tamanho por pacote é limitado a 1500Bytes.

De forma sucinta, apreendemos que neste processo de fragmentação de Datagramas os fragmentos do mesmo pacote original são identificados pelo mesmo ID (valor do campo Identification do protocolo IPv4) e irão ter no máximo 1480Bytes de dados, pois 20Bytes são ocupados pelo cabeçalho do protocolo referido.

O processo de reconstrução é feito de forma correta recorrendo às informações do campo 'Identification': para juntar todos os fragmentos do mesmo pacote original, sem misturar fragmentos de diferentes Datagramas iniciais; do campo 'Flags': que indica se existem mais fragmentos ou se o fragmento em análise é o último e do campo 'Fragmente offset': que indica onde deve ser a posição a encaixar o fragmento para reconstruir o pacote original.

Para uma configuração do tamanho de mensagem padrão (no nosso caso de estudo 56Bytes mas podia ser qualquer valor que não exceda o limite de 1500Bytes ) não existe a necessidade de realizar fragmentação porque o datagrama pode circular na rede sem problemas.

Para a segunda parte do trabalho, compreendemos a informação listada na tabela de encaminhamento e como é feito o encaminhamento de tráfego, quando um determinado host não conhece o endereço IP destino de um dado pacote ou quando o endereço destino é ele mesmo. Este processo foi descrito de maneira completa na questão 2.a da segunda parte do trabalho e ao longo das questões respondidas.

De igual modo, para a implementação de subnetting no sistema escolhemos realizar a divisão do octeto em 4 bits para gestão de sub-redes e 4 bits para identificação dos hosts. Assim, permitimos que exista um número aceitável de sub-redes e de hosts por sub-rede, visando o futuro crescimento e possibilidade de novas sub-redes na empresa. Este processo foi descrito de forma completa e com imagens na questão 3.

Ana Esmeralda Fernandes A74321  
Bárbara Nadine Freitas Oliveira A75614  
Miguel Dias Miranda A74726

Redes de Computadores  
TP4: Protocolo IPv4  
Novembro de 2016