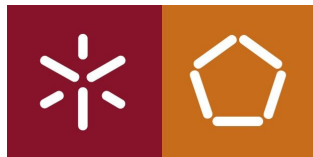


TP4: Protocolo IP

Universidade do Minho
Licenciatura em Ciências da Computação
Sistemas de Comunicações e Redes

Daniel Regado, Maria Barbosa
[a85137,a85290]@alunos.uminho.pt



Captura de Tráfego IP

1.a

```
root@n4: /tmp/pycore.36574/n4.conf
root@n4: /tmp/pycore.36574/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) 0.072 ms 0.048 ms 0.102 ms
 2 10.0.1.1 (10.0.1.1) 0.062 ms 0.058 ms 0.056 ms
 3 A9 (10.0.0.10) 0.074 ms 0.070 ms 0.068 ms
root@n4: /tmp/pycore.36574/n4.conf#
```

1.b

3	3.897523	10.0.2.10	10.0.0.10	ICMP	74 Echo (ping) request id=0x005b, seq=1/256, ttl=1
4	3.897575	10.0.2.1	10.0.2.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
5	3.897789	10.0.2.10	10.0.0.10	ICMP	74 Echo (ping) request id=0x005b, seq=2/512, ttl=1
6	3.897828	10.0.2.1	10.0.2.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
7	3.898008	10.0.2.10	10.0.0.10	ICMP	74 Echo (ping) request id=0x005b, seq=3/768, ttl=1
8	3.898101	10.0.2.1	10.0.2.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
9	3.898282	10.0.2.10	10.0.0.10	ICMP	74 Echo (ping) request id=0x005b, seq=4/1024, ttl=2
10	3.898335	10.0.1.1	10.0.2.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
11	3.898501	10.0.2.10	10.0.0.10	ICMP	74 Echo (ping) request id=0x005b, seq=5/1280, ttl=2
12	3.898550	10.0.1.1	10.0.2.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
13	3.898715	10.0.2.10	10.0.0.10	ICMP	74 Echo (ping) request id=0x005b, seq=6/1536, ttl=2
14	3.898761	10.0.1.1	10.0.2.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
15	3.898928	10.0.2.10	10.0.0.10	ICMP	74 Echo (ping) request id=0x005b, seq=7/1792, ttl=3
16	3.898993	10.0.0.10	10.0.2.10	ICMP	74 Echo (ping) reply id=0x005b, seq=7/1792, ttl=62

Como esperado, vão sendo enviados pacotes ICMP com TTL sucessivamente incrementados por 1.

1.c

O valor inicial mínimo do campo TTL deve ser 3, para alcançar o destino n1.

1.d

O tempo médio de ida e volta (média ponderada dos 3 valores 0.070, 0.074, 0.068) corresponde a 0.0706(6) ms. (valores do print na pergunta 1.a).

2.a

```
Internet Protocol Version 4, Src: 192.168.100.185, Dst: 193.136.9.240
```

É o endereço 192.168.100.185 .

2.b

Tem o valor 01, que corresponde ao protocolo ICMP

Protocol: ICMP (1)

```
00 0c 29 d2 19 f0 50 7b 9d 7f 86 87 08 00 45 00  ..)...P{ .....E.
00 38 1c a1 00 00 ff 01 00 00 c0 a8 64 b9 c1 88  .8...... ....d...
09 f0 08 00 36 3a 00 01 00 03 20 20 20 20 20 20  ....6:.. ..
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20
```

2.c

IPv4 - 20 bytes

Payload - 36 bytes (56 bytes totais - 20 bytes headers IP)

Internet Protocol Version 4 (ip), 20 bytes

Packet size bytes

2.d

Não, porque temos o fragment offset com o valor 0.

...0 0000 0000 0000 = Fragment offset: 0

2.e

Mudam os campos TTL e o identification. Temos 2 exemplos em baixo:

Ex 1:

- ▼ Internet Protocol Version 4, Src: 192.168.100.185, Dst: 193.136.9.240
 - 0100 = Version: 4
 - 0101 = Header Length: 20 bytes (5)
 - > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 56
 - Identification: 0x1ca2 (7330)
 - > Flags: 0x0000
 - > Time to live: 1
 - Protocol: ICMP (1)
 - Header checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
 - Source: 192.168.100.185
 - Destination: 193.136.9.240
- > Internet Control Message Protocol

Ex 2:

```
✓ Internet Protocol Version 4, Src: 192.168.100.185, Dst: 193.136.9.240
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x1ca1 (7329)
  > Flags: 0x0000
    Time to live: 255
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.100.185
    Destination: 193.136.9.240
  > Internet Control Message Protocol
```

2.f

Sim, têm sempre um incremento (+1) simultâneo. (Comprovado pelos prints anteriores).

2.g

O valor TTL nas respostas ICMP (para respostas exceeded) permanece a 64 para o primeiro router. Este valor corresponde a um valor TTL pré-definido, usado para enviar uma resposta caso o ICMP recebido tenha TTL igual a 1 (TTL exceeded). Este TTL pode variar de router em router, e tem um valor suficientemente alto para assegurar a chegada à origem. (Estes prints correspondem a pacotes sucessivos com TTL exceeded)

```
✓ Internet Protocol Version 4, Src: 192.168.100.254, Dst: 192.168.100.185
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xcd2d (52525)
  > Flags: 0x0000
    Time to live: 64
```

```
Internet Protocol Version 4, Src: 192.168.100.254, Dst: 192.168.100.185
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xcd30 (52528)
  > Flags: 0x0000
    Time to live: 64
```


3.a

A mensagem foi fragmentada, visto que 3014 bytes são demasiada informação para ser transportada por um pacote só (limite IPv4 1480 bytes).

Time	Source	Destination	Protocol	Length	Info
1 0.000000	192.168.100.185	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=045c) [Reassembled in #3]
2 0.000013	192.168.100.185	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=045c) [Reassembled in #3]
3 0.000027	192.168.100.185	193.136.9.240	ICMP	68	Echo (ping) request id=0x0001, seq=145/37120, ttl=255 (reply in 6)

[3 IPv4 Fragments (2994 bytes): #1(1480), #2(1480), #3(34)]

[\[Frame: 1, payload: 0-1479 \(1480 bytes\)\]](#)

[\[Frame: 2, payload: 1480-2959 \(1480 bytes\)\]](#)

[\[Frame: 3, payload: 2960-2993 \(34 bytes\)\]](#)

Packet size | 3014 | bytes

3.b

Podemos ver que se trata de um pacote fragmentado pela flag 'More fragments' = 1

Trata-se do primeiro fragmento pelo valor no campo 'Fragment offset' estar a 0.

O tamanho total são 1500 bytes, com 20 bytes para cabeçalhos IPv4.

Internet Protocol Version 4, Src: 192.168.100.185, Dst: 193.136.9.240

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 1500

Identification: 0x045d (1117)

✓ Flags: 0x2000, More fragments

0... = Reserved bit: Not set

.0... = Don't fragment: Not set

..1. = More fragments: Set

...0 0000 0000 0000 = Fragment offset: 0

> Time to live: 1

Protocol: ICMP (1)

Header checksum: 0x0000 [validation disabled]

[Header checksum status: Unverified]

Source: 192.168.100.185

Destination: 193.136.9.240

[Reassembled IPv4 in frame: 10](#)

3.c

Podemos ver que não se trata do primeiro fragmento por ter o campo 'Fragment offset' diferente de 0 (valor dividido por 8, neste caso a 185). Concluimos também que este não é o último pelo campo 'More fragments' se encontrar a 1.

```
Internet Protocol Version 4, Src: 192.168.100.185, Dst: 193.136.9.240
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x045d (1117)
  ✓ Flags: 0x20b9, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    ...0 0000 1011 1001 = Fragment offset: 185
  > Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.100.185
    Destination: 193.136.9.240
    Reassembled IPv4 in frame: 10
```

3.d

Foram criados 3 fragmentos. Deteta-se que este (no print) é o último fragmento por a flag 'More fragments' se encontrar com o valor 0.

```

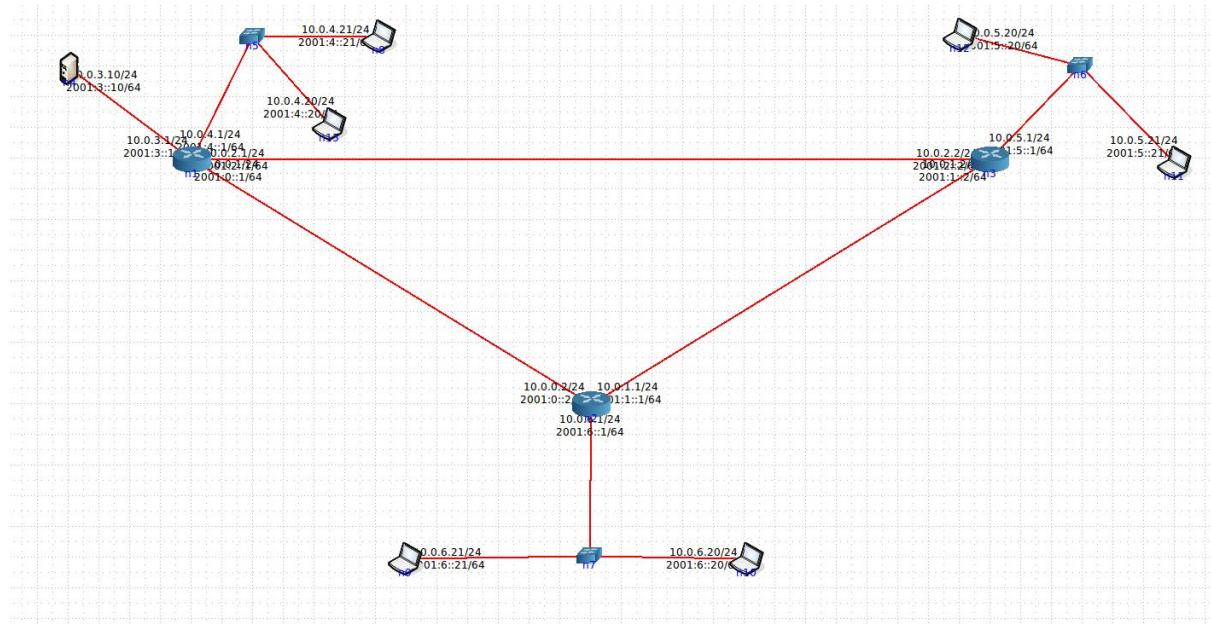
✓ Flags: 0x0172
  0... .. = Reserved bit: Not set
  .0.. .. = Don't fragment: Not set
  ..0. .... = More fragments: Not set
  ...0 0001 0111 0010 = Fragment offset: 370
> Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.100.185
  Destination: 193.136.9.240
✓ [3 IPv4 Fragments (2994 bytes): #8(1480), #9(1480), #10(34)]
  [Frame: 8, payload: 0-1479 (1480 bytes)]
  [Frame: 9, payload: 1480-2959 (1480 bytes)]
  [Frame: 10, payload: 2960-2993 (34 bytes)]
  [Fragment count: 3]
  [Reassembled IPv4 length: 2994]
```

3.e

Pelos prints anteriores, o único campo que se altera no campo IP do cabeçalho é o offset, visto que este valor corresponde à "distância" ao primeiro datagrama fragmentado.

Parte II

1.a



router configuration

Node name: n1 (none)

Type: router Services...

Interface eth0
MAC address auto-assign
IPv4 address 10.0.0.1/24
IPv6 address 2001:0::1/64

Interface eth1
MAC address auto-assign
IPv4 address 10.0.2.1/24
IPv6 address 2001:2::1/64

Interface eth2
MAC address auto-assign
IPv4 address 10.0.3.1/24
IPv6 address 2001:3::1/64

Interface eth3
MAC address auto-assign
IPv4 address 10.0.4.1/24
IPv6 address 2001:4::1/64

Apply Cancel

router configuration

Node name: n2 (none)

Type: router Services...

Interface eth0
MAC address auto-assign
IPv4 address 10.0.0.2/24
IPv6 address 2001:0::2/64

Interface eth1
MAC address auto-assign
IPv4 address 10.0.1.1/24
IPv6 address 2001:1::1/64

Interface eth2
MAC address auto-assign
IPv4 address 10.0.6.1/24
IPv6 address 2001:6::1/64

Apply Cancel

router configuration

Node name: n3 (none)

Type: router Services...

Interface eth0
MAC address auto-assign
IPv4 address 10.0.1.2/24
IPv6 address 2001:1::2/64

Interface eth1
MAC address auto-assign
IPv4 address 10.0.2.2/24
IPv6 address 2001:2::2/64

Interface eth2
MAC address auto-assign
IPv4 address 10.0.5.1/24
IPv6 address 2001:5::1/64

Apply Cancel

1.b

São endereços privados, porque são endereços na gama 10.0.0.0 - 10.255.255.255

1.c

Uma vez que o switch opera em nível 2, não lhe pode ser atribuído um endereço IP, que está relacionado com a camada 3.

1.d

Ping de Laptop de Dept. A para servidor:

```
root@n13: /tmp/pycore.41660/n13.conf
root@n13:/tmp/pycore.41660/n13.conf# ping -c 1 10.0.3.10
PING 10.0.3.10 (10.0.3.10) 56(84) bytes of data.
64 bytes from 10.0.3.10: icmp_req=1 ttl=63 time=0.195 ms

--- 10.0.3.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.195/0.195/0.195/0.000 ms
root@n13:/tmp/pycore.41660/n13.conf#
```

Ping de Laptop de Dept. B para servidor:

```
root@n10: /tmp/pycore.41660/n10.conf
root@n10:/tmp/pycore.41660/n10.conf# ping -c 1 10.0.3.10
PING 10.0.3.10 (10.0.3.10) 56(84) bytes of data.
64 bytes from 10.0.3.10: icmp_req=1 ttl=62 time=0.233 ms

--- 10.0.3.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.233/0.233/0.233/0.000 ms
```

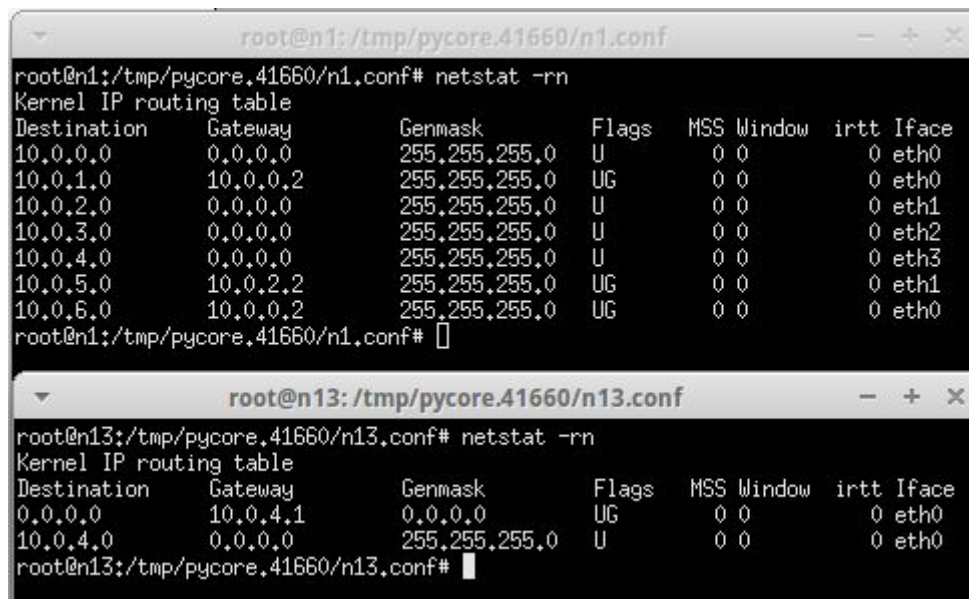
Ping de Laptop de Dept. C para servidor:

```
root@n11: /tmp/pycore.41660/n11.conf
root@n11:/tmp/pycore.41660/n11.conf# ping -c 1 10.0.3.10
PING 10.0.3.10 (10.0.3.10) 56(84) bytes of data.
64 bytes from 10.0.3.10: icmp_req=1 ttl=62 time=0.224 ms

--- 10.0.3.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.224/0.224/0.224/0.000 ms
root@n11:/tmp/pycore.41660/n11.conf#
```

2.a

Sendo n1 o router do Dept.A e n13 um laptop no Dept.A:



The image shows two terminal windows. The top window is titled 'root@n1:/tmp/pycore.41660/n1.conf' and displays the output of 'netstat -rn' for router n1. The bottom window is titled 'root@n13:/tmp/pycore.41660/n13.conf' and displays the output of 'netstat -rn' for laptop n13.

```
root@n1:/tmp/pycore.41660/n1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.0.1.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
10.0.3.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
10.0.4.0 0.0.0.0 255.255.255.0 U 0 0 0 eth3
10.0.5.0 10.0.2.2 255.255.255.0 UG 0 0 0 eth1
10.0.6.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
root@n1:/tmp/pycore.41660/n1.conf#

root@n13:/tmp/pycore.41660/n13.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.4.1 0.0.0.0 UG 0 0 0 eth0
10.0.4.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@n13:/tmp/pycore.41660/n13.conf#
```

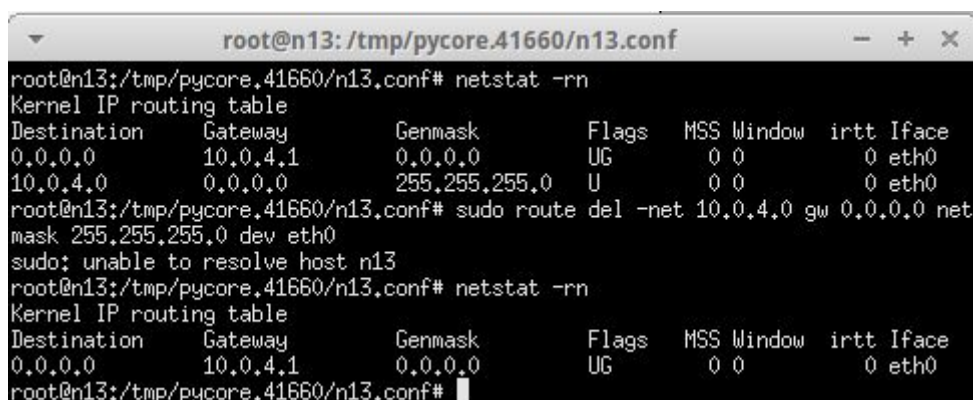
No caso das entradas da tabela com as flags UG, estas compreendem destinos em que o gateway é outro router, ou seja, destinos sem ligação direta (linhas na tipologia CORE). Por outro lado, entradas com o gateway 0.0.0.0 correspondem a destinos diretos, sem necessidade de passar por outro router.

2.b

As rotas com as flags UG correspondem a caminhos dinâmicos, e as restantes rotas, com gateway (0.0.0.0) correspondem a caminhos estáticos.

2.c

Apagar routes 0.0.0.0 no laptop:



The image shows a terminal window titled 'root@n13:/tmp/pycore.41660/n13.conf'. It shows the initial routing table, followed by the command to delete the route for 0.0.0.0, and then the updated routing table.

```
root@n13:/tmp/pycore.41660/n13.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.4.1 0.0.0.0 UG 0 0 0 eth0
10.0.4.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@n13:/tmp/pycore.41660/n13.conf# sudo route del -net 10.0.4.0 gw 0.0.0.0 net
mask 255.255.255.0 dev eth0
sudo: unable to resolve host n13
root@n13:/tmp/pycore.41660/n13.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.4.1 0.0.0.0 UG 0 0 0 eth0
root@n13:/tmp/pycore.41660/n13.conf#
```

Apagar routes 0.0.0.0 no router:

```
root@n1:/tmp/pycore.41660/n1.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask          Flags   MSS Window  irtt Iface
10.0.1.0         10.0.0.2        255.255.255.0    UG      0 0        0 eth0
10.0.2.0         0.0.0.0         255.255.255.0    U       0 0        0 eth1
10.0.3.0         0.0.0.0         255.255.255.0    U       0 0        0 eth2
10.0.4.0         0.0.0.0         255.255.255.0    U       0 0        0 eth3
10.0.5.0         10.0.2.2        255.255.255.0    UG      0 0        0 eth1
10.0.6.0         10.0.0.2        255.255.255.0    UG      0 0        0 eth0
root@n1:/tmp/pycore.41660/n1.conf# sudo route delete -net 10.0.1.0 gw 0.0.0.0 netmask 255.255.255.0 dev eth0
sudo: unable to resolve host n1
root@n1:/tmp/pycore.41660/n1.conf# sudo route delete -net 10.0.2.0 gw 0.0.0.0 netmask 255.255.255.0 dev eth1
sudo: unable to resolve host n1
root@n1:/tmp/pycore.41660/n1.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask          Flags   MSS Window  irtt Iface
10.0.3.0         0.0.0.0         255.255.255.0    U       0 0        0 eth2
10.0.4.0         0.0.0.0         255.255.255.0    U       0 0        0 eth3
10.0.5.0         10.0.2.2        255.255.255.0    UG      0 0        0 eth1
10.0.6.0         10.0.0.2        255.255.255.0    UG      0 0        0 eth0
root@n1:/tmp/pycore.41660/n1.conf# sudo route delete -net 10.0.3.0 gw 0.0.0.0 netmask 255.255.255.0 dev eth2
sudo: unable to resolve host n1
root@n1:/tmp/pycore.41660/n1.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask          Flags   MSS Window  irtt Iface
10.0.4.0         0.0.0.0         255.255.255.0    U       0 0        0 eth3
10.0.5.0         10.0.2.2        255.255.255.0    UG      0 0        0 eth1
10.0.6.0         10.0.0.2        255.255.255.0    UG      0 0        0 eth0
root@n1:/tmp/pycore.41660/n1.conf# sudo route delete -net 10.0.4.0 gw 0.0.0.0 netmask 255.255.255.0 dev eth3
sudo: unable to resolve host n1
root@n1:/tmp/pycore.41660/n1.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask          Flags   MSS Window  irtt Iface
10.0.5.0         10.0.2.2        255.255.255.0    UG      0 0        0 eth1
10.0.6.0         10.0.0.2        255.255.255.0    UG      0 0        0 eth0
root@n1:/tmp/pycore.41660/n1.conf#
```

Tentativa posterior de comunicação entre um laptop (neste caso do dept. B) com o servidor:

```
root@n9:/tmp/pycore.41660/n9.conf
root@n9:/tmp/pycore.41660/n9.conf# ping -c 1 10.0.3.10
PING 10.0.3.10 (10.0.3.10) 56(84) bytes of data:
From 10.0.0.1 icmp_seq=1 Destination Net Unreachable

--- 10.0.3.10 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
root@n9:/tmp/pycore.41660/n9.conf#
```

Depois de apagar os routes que foram eliminados, perde-se a conexão do servidor com o exterior, pois o único gateway deste era o router n1 (do dept. A).

2.d

Usamos o comando: `sudo route add -net 10.0.3.0/24 gw 10.0.3.1 eth2`

```
root@n1:/tmp/pycore.51178/n1.conf# sudo route add -net 10.0.3.0/24 gw 10.0.3.1 eth2
sudo: unable to resolve host n1
root@n1:/tmp/pycore.51178/n1.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask         Flags       MSS Window  irtt Iface
10.0.2.0         10.0.0.2        255.255.255.0   UG          0 0        0 eth0
10.0.3.0         10.0.3.1        255.255.255.0   UG          0 0        0 eth2
10.0.5.0         10.0.0.2        255.255.255.0   UG          0 0        0 eth0
10.0.6.0         10.0.1.2        255.255.255.0   UG          0 0        0 eth1
root@n1:/tmp/pycore.51178/n1.conf#
```

2.e

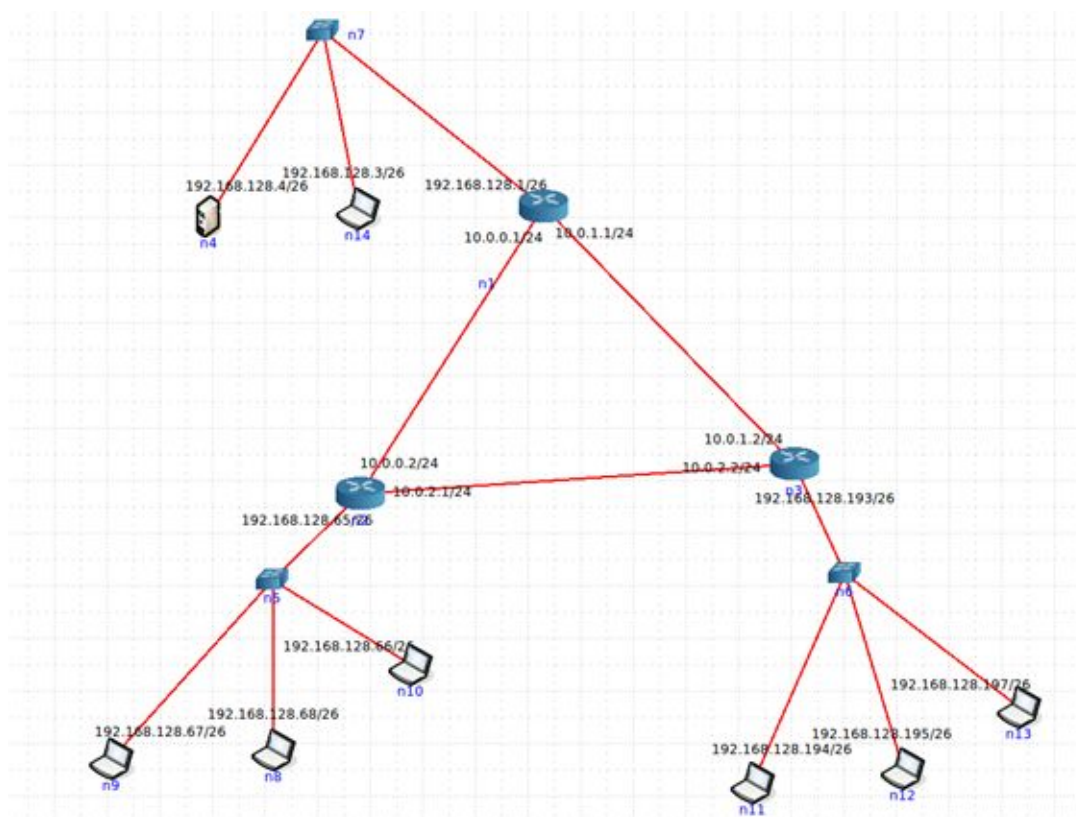
O servidor está novamente acessível, uma vez que é possível realizar, com sucesso, um ping de um laptop do departamento B, para o servidor que se encontra no departamento A (IP 10.0.3.10) .

```
root@n9:/tmp/pycore.51178/n9.conf# ping -c 1 10.0.3.10
PING 10.0.3.10 (10.0.3.10) 56(84) bytes of data.
64 bytes from 10.0.3.10: icmp_req=1 ttl=62 time=0.265 ms

--- 10.0.3.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.265/0.265/0.265/0.000 ms
root@n9:/tmp/pycore.51178/n9.conf#
```


3.a

Uma vez que dispomos apenas de um endereço de rede IP 192.168.128.0/24 mas temos a necessidade de criar 3 redes distintas (uma para cada departamento) e sabendo que a cada bit podemos associar duas redes IP diferentes (isto porque não estamos a contabilizar os IPs reservados) então para as 3 redes será necessário utilizar 2 bits do host. Deste modo temos disponível 4 redes: 192.168.128.0 (00000000), 192.168.128.128 (10000000), 192.168.128. 64 (01000000) e 192.168.128.192 (11000000), às quais esta associada a máscara 255.255.255.192.



3.b

A máscara de rede usada foi 255.255.255.192 (em decimal). Uma vez que nos 3 primeiros octetos todos os bits correspondem a rede $2^7+2^6+2^5+2^4+2^3+2^2+2^1+2^0 = 255$, no último octeto (que em binário corresponde a 11000000) os dois primeiros bits também são usados para rede, logo $2^7+2^6 = 192$ e os restantes correspondem a host.

3.c

Uma vez que a máscara usada foi 255.255.255.192 então dispomos de 63 (255-192) redes IP diferentes, uma vez que não usamos endereços IPs reservados. Assim é possível em cada departamento interligar 63 hosts diferentes.

3.d

É possível assegurar a conectividade dos sistemas, através da realização de ping:

- entre um laptop do departamento A (192.168.128.3) para um de B (192.168.128.66):

```
root@n14:/tmp/pycore.59705/n14.conf# ping 192.168.128.66
PING 192.168.128.66 (192.168.128.66) 56(84) bytes of data.
64 bytes from 192.168.128.66: icmp_req=1 ttl=62 time=0.260 ms
64 bytes from 192.168.128.66: icmp_req=2 ttl=62 time=0.251 ms
64 bytes from 192.168.128.66: icmp_req=3 ttl=62 time=0.264 ms
64 bytes from 192.168.128.66: icmp_req=4 ttl=62 time=0.244 ms
^C
--- 192.168.128.66 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.244/0.254/0.264/0.021 ms
root@n14:/tmp/pycore.59705/n14.conf#
```

- entre um laptop do departamento B (192.168.128.66) para um de C (192.168.128.194):

```
root@n10:/tmp/pycore.59705/n10.conf# ping 192.168.128.194
PING 192.168.128.194 (192.168.128.194) 56(84) bytes of data.
64 bytes from 192.168.128.194: icmp_req=1 ttl=62 time=0.214 ms
64 bytes from 192.168.128.194: icmp_req=2 ttl=62 time=0.248 ms
64 bytes from 192.168.128.194: icmp_req=3 ttl=62 time=0.238 ms
^C
--- 192.168.128.194 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.214/0.233/0.248/0.018 ms
root@n10:/tmp/pycore.59705/n10.conf#
```

- entre um laptop do departamento C (192.168.128.195) para o departamento A (192.168.128.3):

```
root@n12:/tmp/pycore.59705/n12.conf# ping 192.168.128.3
PING 192.168.128.3 (192.168.128.3) 56(84) bytes of data.
64 bytes from 192.168.128.3: icmp_req=1 ttl=62 time=0.228 ms
64 bytes from 192.168.128.3: icmp_req=2 ttl=62 time=0.286 ms
64 bytes from 192.168.128.3: icmp_req=3 ttl=62 time=0.252 ms
^C
--- 192.168.128.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.228/0.255/0.286/0.027 ms
root@n12:/tmp/pycore.59705/n12.conf#
```

Conclusões:

Este trabalho foi realizado no âmbito da UC, Sistemas de Comunicação e Redes, da licenciatura de ciências da computação. A sua realização teve como principal objetivo o estudo do Internet Protocol (IP).

A primeira parte deste trabalho focou-se na captura de tráfego IP, utilizamos para isso o programa traceroute de modo a obter o endereço IP da fonte e do destino para o qual se dirigem os pacotes bem como a identificação dos routers do trajeto. Recorrendo, aos programas pingplotter e wireshark e focando a análise ao nível do cabeçalho IP nas mensagens ICMP, foi possível comprovar que elas são responsáveis, entre outras coisas, por identificar situações anómalas ocorridas no tratamento de datagramas IP permitindo a visualização clara de como se alteram os valores do campo TTL (“tempo de vida”) ao longo do percurso dos pacotes. Conseguimos identificar como se processa o endereçamento e encaminhamento dos datagramas IP de acordo com a ocorrência ou não de fragmentação dos pacotes IP.

Na Segunda parte, focamos o nosso estudo nas técnicas de endereçamento e encaminhamento do internet protocol. Para isso recorremos ao Core, na Máquina Virtual, de modo a construir uma topologia que refletisse a rede local de uma empresa. Permitindo analisar com clareza a relação entre os endereços público e privados numa rede e os vários campos numa tabela de encaminhamento. Pudemos comprovar o quão vantajoso pode ser a utilização de sub-redes para contrariar a falta de endereços IPv4, permitindo melhorar o aproveitamento, organização e gestão do espaço de endereços. No último grupo de questões deste relatório fez-se uma simulação do subnetting na rede core, criada previamente, onde é possível verificar que com apenas um endereço IP se consegue introduzir outro nível hierárquico para routing.

A nossa apreciação global em relação a este trabalho é muito positiva, e consideramos que os objetivos da sua realização foram atingidos, uma vez que com o decorrer da sua execução foi possível não só consolidar conhecimentos adquiridos nas aulas teóricas, mas também experimentar o funcionamento prático de alguns conceitos.