

Universidade do Minho

## **Relatório do Trabalho Prático 2:**

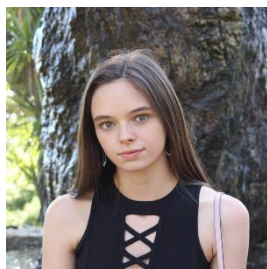
### **Protocolo IPv4 :: Datagramas IP e Fragmentação**

Ano letivo 2021/2022

Abril 2022

Licenciatura em Engenharia Informática

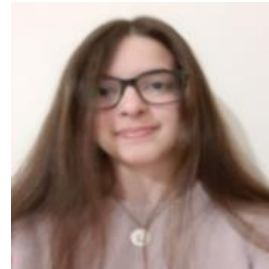
Unidade Curricular de Redes de Computadores



Ana Gonçalves a93259



Luís Faria a93209



Jéssica Fernandes a93318

## Índice

Introdução .....	4
Parte 1 .....	5
Questão 1 (Xubuntu) .....	5
Questão 2 (Máquina Nativa: Ubuntu) .....	8
Questão 3 (Máquina Nativa: Ubuntu com Tamanho de pacote = 4071) .....	12
Parte 2 .....	16
Questão 1 .....	16
Questão 2 .....	20
Questão 3 .....	24
Conclusão .....	26

## Índice de Figuras

FIGURA 1: TOPOLOGIA DE REDE CORE .....	5
FIGURA 2: RESULTADO DE TRACERROUTE NO HOST BELA .....	5
FIGURA 3: TRÁFEGO CAPTURADO APÓS TRACERROUTE .....	6
FIGURA 4: PACOTE SELECIONADO COMO O PRIMEIRO PACOTE ENVIADO PELO HOST DESTINO .....	7
FIGURA 5: COMANDO TRACERROUTE COM MAIOR NÚMERO DE REPETIÇÕES, NO HOST BELA PARA O HOST MONSTRO .....	7
FIGURA 6: PACOTE DE STANDARD QUERY FEITA PELA MÁQUINA NATIVA.....	8
FIGURA 7: PRIMEIRO PACOTE ENVIADO PELA MÁQUINA NATIVA .....	8
FIGURA 8: PRIMEIRO PACOTE ENVIADO PELA MÁQUINA NATIVA .....	9
FIGURA 9: PRIMEIRO PACOTE ENVIADO PELA MÁQUINA NATIVA .....	9
FIGURA 10: PRIMEIRO PACOTE ENVIADO.....	10
FIGURA 11: SEGUNDO PACOTE ENVIADO.....	10
FIGURA 12: TERCEIRO PACOTE ENVIADO .....	10
FIGURA 13: QUARTO PACOTE ENVIADO .....	10
FIGURA 14: PRIMEIRO PACOTE RECEBIDO POR TTL EXCEDIDO .....	11
FIGURA 15: PRIMEIRO FRAGMENTO ENVIADO .....	12
FIGURA 16: PRIMEIRO FRAGMENTO ENVIADO .....	12
FIGURA 17: SEGUNDO FRAGMENTO ENVIADO .....	13
FIGURA 18: TERCEIRO E ÚLTIMO FRAGMENTO DO PRIMEIRO DATAGRAMA ORIGINAL.....	13
FIGURA 19: PRIMEIRO FRAGMENTO.....	14
FIGURA 20: SEGUNDO FRAGMENTO .....	14
FIGURA 21: TERCEIRO E ÚLTIMO FRAGMENTO.....	14
FIGURA 22: TOPOLOGIA DA REDE.....	16
FIGURA 23: COMANDO PING DE BELA PARA SA .....	17
FIGURA 24: COMANDO PING DE JASMIN PARA SB .....	17
FIGURA 25: COMANDO PING DE ERIC PARA SC .....	17
FIGURA 26: COMANDO PING DE NALA PARA SD.....	17
FIGURA 27: COMANDO PING DE BELA PARA SB.....	18
FIGURA 28: COMANDO PING DE BELA PARA SC .....	18
FIGURA 29: COMANDO PING DE BELA PARA SD .....	18
FIGURA 30: COMANDO PING DE ERIC PARA SB .....	18
FIGURA 31: COMANDO PING DE ERIC PARA SD.....	18
FIGURA 32: COMANDO PING DE SIMBA PARA SB .....	19
FIGURA 33: COMANDO PING DE BELA PARA RISP .....	19
FIGURA 34: TABELA DE ENCAMINHAMENTO DE BELA .....	20
FIGURA 35: TABELA DE ENCAMINHAMENTO DE RA.....	20
FIGURA 36: ELIMINAÇÃO DA ROTA DEFAULT .....	22
FIGURA 37: COMANDOS UTILIZADOS PARA REPOR ROTAS NECESSÁRIAS PARA DESFAZER EFEITO DE C).....	22
FIGURA 38: NOVA TABELA DE ENCAMINHAMENTO DO SERVIDOR A .....	22
FIGURA 39: COMANDO PING DO DEP. C PARA O SERVIDOR A.....	23
FIGURA 40: COMANDO PING DO DEP.D PARA O SERVIDOR A .....	23
FIGURA 41: COMANDO PING DO DEP. B PARA O SERVIDOR A .....	23
FIGURA 42: TOPOLOGIA COM SUB-NETTING .....	24
FIGURA 43: COMANDO PING ENTRE DEPARTAMENTO A E B .....	25
FIGURA 44: COMANDO PING ENTRE DEPARTAMENTO A E C.....	25
FIGURA 45: COMANDO PING ENTRE DEPARTAMENTO A E D.....	25
FIGURA 46: COMANDO PING ENTRE DEPARTAMENTO B E C .....	25
FIGURA 47: COMANDO PING ENTRE DEPARTAMENTO B E D.....	25
FIGURA 48: COMANDO PING ENTRE DEPARTAMENTO C E D.....	25

## Introdução

Neste relatório será abordado a resolução do trabalho prático da UC Redes de Computadores, cujo enunciado tinha como objetivo o aprofundamento do conhecimento sobre o protocolo IP, nomeadamente o formato de um pacote, a sua fragmentação, endereçamento e finalmente encaminhamento.

Deste modo, será apresentado neste documento, as resoluções da parte 1 e parte 2 , constituídas pelas perguntas e respostas justificadas, juntamente com uma conclusão do trabalho prático.

## Parte 1

### Questão 1 (Xubuntu)

**a) Active o wireshark ou o tcpdump no host Bela. Numa shell de Bela execute o comando `traceroute -I` para o endereço IP do Monstro**

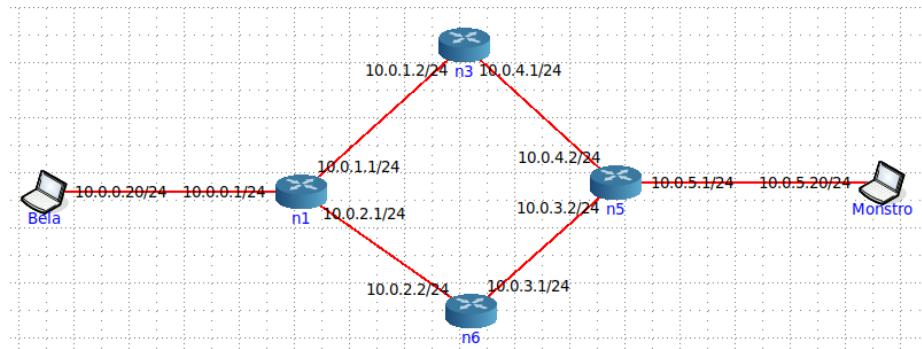


Figura 1: Topologia de Rede Core

```
root@Bela:/tmp/pycore.40631/Bela.conf# traceroute -I 10.0.5.20
traceroute to 10.0.5.20 (10.0.5.20), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.036 ms 0.007 ms 0.006 ms
 2 10.0.1.2 (10.0.1.2) 0.047 ms 0.008 ms 0.010 ms
 3 10.0.3.2 (10.0.3.2) 0.041 ms 0.012 ms 0.011 ms
 4 10.0.5.20 (10.0.5.20) 0.031 ms 0.013 ms 0.013 ms
root@Bela:/tmp/pycore.40631/Bela.conf#
```

Figura 2: Resultado de Traceroute no host Bela

**b) Registe e analise o tráfego ICMP enviado pelo sistema Bela e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.**

2	1.165386657	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=1/256, ttl=1 (no response...
3	1.165414939	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
4	1.165422904	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=2/512, ttl=1 (no response...
5	1.165428428	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
6	1.165433789	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=3/768, ttl=1 (no response...
7	1.165440037	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
8	1.165445814	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=4/1024, ttl=2 (no respons...
9	1.165469568	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
10	1.165474129	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=5/1280, ttl=2 (no respons...
11	1.165482791	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
12	1.165487584	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=6/1536, ttl=2 (no respons...
13	1.165497258	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
14	1.165503008	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=7/1792, ttl=3 (no respons...
15	1.165533248	10.0.3.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
16	1.165537382	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=8/2048, ttl=3 (no respons...
17	1.165547274	10.0.3.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
18	1.165550896	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=9/2304, ttl=3 (no respons...
19	1.165560592	10.0.3.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
20	1.165564729	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=10/2560, ttl=4 (reply in ...
21	1.165584469	10.0.5.20	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x0032, seq=10/2560, ttl=61 (request ...
22	1.165589750	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=11/2816, ttl=4 (reply in ...
23	1.165602024	10.0.5.20	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x0032, seq=11/2816, ttl=61 (request ...
24	1.165605645	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=12/3072, ttl=4 (reply in ...
25	1.165617721	10.0.5.20	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x0032, seq=12/3072, ttl=61 (request ...
26	1.165621930	10.0.0.20	10.0.5.20	ICMP	74 Echo (ping) request	id=0x0032, seq=13/3328, ttl=5 (reply in ...
27	1.165633872	10.0.5.20	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x0032, seq=13/3328, ttl=61 (request ...

```

Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface veth1.0.29, id 0
Ethernet II, Src: 00:00:00:aa:00:00 (00:00:00:aa:00:00), Dst: 00:00:00:aa:00:01 (00:00:00:aa:00:01)
Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.5.20
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 60
  Identification: 0xf869 (63593)
  Flags: 0x0000
  Fragment offset: 0
  Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0xa830 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.0.0.20
  Destination: 10.0.5.20
Internet Control Message Protocol

```

Figura 3: Tráfego capturado após Traceroute

- Correspondem aos 3 pacotes enviados para o host Monstro com TTL de 1
- Correspondem aos 3 pacotes enviados para o host Monstro com TTL de 2
- Correspondem aos 3 pacotes enviados para o host Monstro com TTL de 3

E o ciclo continua até receber o primeiro pacote que confirma que chegou ao host Monstro.

O comportamento esperado do Traceroute, consistia no envio de X pacotes com TTL a crescer linearmente, de modo a chegar ao destino pretendido. E com a figura acima apresentada podemos confirmar que é esse o comportamento que efetivamente acontece. Neste caso específico são enviados 3 pacotes com TTL igual, inicializado a 1.

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

Como verificamos na questão anterior, o host Bela recebeu 3 sequencias de 3 pacotes que não alcançaram o destino. Sendo que cada sequência está associada a um pacote enviado com TTL de 1,2 e 3 correspondentemente, podemos confirmar que o valor mínimo de TTL para alcançar o servidor Monstro é de 4.

10.0.0.1	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=13/3328, ttl=0 (repl..
17.0.547154838	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=14/3584, ttl=5 (repl..
18.0.547156064	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=15/3840, ttl=5 (repl..
19.0.547156855	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=16/4096, ttl=6 (repl..
20.0.567343492	10.0.0.1	10.0.0.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
21.0.567352165	10.0.0.1	10.0.0.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
22.0.567353904	10.0.0.1	10.0.0.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
23.0.568720507	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=17/4352, ttl=6 (repl..
24.0.568734894	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=18/4608, ttl=6 (repl..
25.0.568743055	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=19/4864, ttl=7 (repl..
26.0.587874978	10.0.1.2	10.0.0.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
27.0.587884074	10.0.1.2	10.0.0.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
28.0.587885885	10.0.1.2	10.0.0.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
29.0.588641116	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=20/5120, ttl=7 (repl..
30.0.588654875	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=21/5376, ttl=7 (repl..
31.0.588663879	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=22/5632, ttl=8 (repl..
32.0.628657238	10.0.3.2	10.0.0.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
33.0.628668468	10.0.3.2	10.0.0.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
34.0.628671106	10.0.3.2	10.0.0.10	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)	
35.0.629682584	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=23/5888, ttl=8 (repl..
36.0.629697130	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=24/6144, ttl=8 (repl..
37.0.629705875	10.0.0.10	10.0.0.10	ICMP	74 Echo (ping) request	id=0x0024, seq=25/6400, ttl=9 (repl..
38.0.670597684	10.0.5.10	10.0.0.10	ICMP	74 Echo (ping) reply	id=0x0024, seq=10/2560, ttl=61 (req..
39.0.670606912	10.0.5.10	10.0.0.10	ICMP	74 Echo (ping) reply	id=0x0024, seq=11/2816, ttl=61 (req..
40.0.670608642	10.0.5.10	10.0.0.10	ICMP	74 Echo (ping) reply	id=0x0024, seq=12/3072, ttl=61 (req..
41.0.670619314	10.0.5.10	10.0.0.10	ICMP	74 Echo (ping) reply	id=0x0024, seq=13/3328, ttl=61 (req..

Figura 4: Pacote selecionado como o primeiro pacote enviado pelo host destino

Nesta figura, conseguimos identificar que o pacote selecionado foi enviado com sucesso do host Monstro, em resposta ao pacote com TTL de 4.

**d) Calcule o valor médio do tempo de ida-e-volta (RTT - Round-Trip Time) obtido no acesso ao servidor. Para melhorar a média, poderá alterar o número pacotes de prova com a opção -q.**

```
root@Bela:/tmp/pycore.40631/Bela.conf# traceroute -q 5 10.0.5.20
traceroute to 10.0.5.20 (10.0.5.20), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.037 ms 0.006 ms 0.006 ms 0.005 ms 0.005 ms
 2 10.0.1.2 (10.0.1.2) 0.019 ms 0.010 ms 0.009 ms 0.009 ms 0.009 ms
 3 10.0.3.2 (10.0.3.2) 0.030 ms 0.013 ms 0.015 ms 0.011 ms 0.012 ms
 4 10.0.5.20 (10.0.5.20) 0.045 ms 0.025 ms 0.015 ms 0.027 ms 0.014 ms
```

Figura 5: Comando traceroute com maior número de repetições, no host Bela para o host Monstro

Tratando-se a última linha dos vários RTT calculados, a média é igualada a 0.0198 ms.

**e) O valor médio do atraso num sentido (One-Way Delay) poderia ser calculado com precisão dividindo o RTT por dois? O que torna difícil o cálculo desta métrica?**

Não, o facto de que o caminho percorrido na transmissão Bela->Mostro poder ser diferente do caminho percorrido na transmissão Monstro->Bela implica que a divisão por 2 do RTT não vai ser uma versão fiável do One-Way Delay. Isto porque, se o caminho de ida for muito mais demorado, ao contrário do caminho de volta, o One-Way Delay num sentido vai ser muito mais elevado ao real, e o contrário acontece para o sentido oposto.

## Questão 2 (Máquina Nativa: Ubuntu)

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

4	0.177156627	172.26.3.68	162.159.135.234	TCP	54 59710 → 443 [ACK] Seq=52 Ack=33 Win=1639 Len=
5	2.3642351	43 172.26.3.68	193.137.16.145	DNS	86 Standard query 0xfb44 A marco.uminho.pt OPT
6	2.364346028	172.26.3.68	193.137.16.145	DNS	86 Standard query 0xd7ff AAAA marco.uminho.pt OF

Figura 6: Pacote de Standard Query feita pela máquina nativa

Como podemos verificar o IP da interface do computador pessoal é 172.26.3.68

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

Na seguinte figura verificamos que o valor do campo do protocolo é ICMP, e é identificado pelo valor entre parentesis, que no caso de ICMP é 1.

7	2.369968921	193.137.16.145	172.26.3.68	DNS	140 Standard query response 0xd7ff AAAA marco.umi
8	2.370008010	193.137.16.145	172.26.3.68	DNS	102 Standard query response 0xfb44 A marco.uminho
9	2.370466725	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=1/256, t
10	2.370486834	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=2/512, t
11	2.370492136	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=3/768, t
12	2.370497572	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=4/1024, t
13	2.370502402	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=5/1280, t
14	2.370507073	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=6/1536, t
15	2.370512565	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=7/1792, t
16	2.370517318	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=8/2048, t
17	2.370522075	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=9/2304, t
18	2.370527245	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=10/2560, t
19	2.370532210	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=11/2816, t
20	2.370537040	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=12/3072, t
21	2.370542420	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=13/3328, t
22	2.370548096	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=14/3584, t
23	2.370553087	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=15/3840, t
24	2.370558279	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=16/4096, t
25	2.373466931	172.16.2.1	172.26.3.68	ICMP	70 Time-to-live exceeded (Time to live exceeded
26	2.373516457	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=17/4352, t
27	2.373831811	172.16.2.1	172.26.3.68	ICMP	70 Time-to-live exceeded (Time to live exceeded
28	2.373877780	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=18/4608, t
29	2.374678781	172.16.2.1	172.26.3.68	ICMP	70 Time-to-live exceeded (Time to live exceeded
30	2.374715580	172.26.3.68	193.136.9.240	ICMP	74 Echo (ping) request id=0x0003, seq=19/4864, t
31	2.375488886	172.26.254.254	172.26.3.68	ICMP	70 Time-to-live exceeded (Time to live exceeded

Frame 9: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: f0:77:c3:75:c3:76 (f0:77:c3:75:c3:76), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.3.68, Dst: 193.136.9.240
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 60
Identification: 0x148d (5261)
Flags: 0x0000
Fragment offset: 0
Time to live: 1
Protocol: ICMP (1)
Header checksum: 0x2a5e [validation disabled]

Figura 7: Primeiro Pacote enviado pela máquina nativa



**c) Quantos bytes tem o cabeçalho IPv4? Quantos bytes tem o campo de dados (payload) do datagrama? Como se calcula o tamanho do payload?**

Para poder verificar estas informações é necessário selecionar o pacote ICMP correto, e recolher o tamanho do pacote IP 60 bytes. Sendo o tamanho do overhead de 20 bytes, fica confirmado que o tamanho do campo de dados é de:

$$\text{Tamanho IP} - \text{Overhead} = 60 - 20 = 40 \text{ bytes}$$

No.	Time	Source	Destination	Protocol	Length	Info
7	2.369968921	193.137.16.145	172.26.3.68	DNS	140	Standard query response 0xd7ff AAAA marco.uminho.pt
8	2.370000010	193.137.16.145	172.26.3.68	DNS	102	Standard query response 0xf644 A marco.uminho.pt
9	2.370466725	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=1/256, ttl=64
10	2.370486834	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=2/512, ttl=64
11	2.370492136	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=3/768, ttl=64
12	2.370497572	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=4/1024, ttl=64
13	2.370502402	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=5/1280, ttl=64
14	2.370507073	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=6/1536, ttl=64
15	2.370512565	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=7/1792, ttl=64
16	2.370517318	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=8/2048, ttl=64
17	2.370522075	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=9/2304, ttl=64
18	2.370527245	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=10/2560, ttl=64
19	2.370532210	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=11/2816, ttl=64
20	2.370537040	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=12/3072, ttl=64
21	2.370542420	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=13/3328, ttl=64
22	2.370548096	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=14/3584, ttl=64
23	2.370553087	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=15/3840, ttl=64
24	2.370558279	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=16/4096, ttl=64
25	2.373466931	172.16.2.1	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
26	2.373516457	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=17/4352, ttl=64
27	2.373831811	172.16.2.1	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
28	2.373877780	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=18/4608, ttl=64
29	2.374678781	172.16.2.1	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
30	2.374715580	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=19/4864, ttl=64
31	2.375488886	172.26.254.254	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)

Frame 9: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp0s20f3, id 0  
Ethernet II, Src: f0:77:c3:75:c3:76 (f0:77:c3:75:c3:76), Dst: ComdaEnt\_ff:94:00 (00:d0:03:ff:94:00)  
Internet Protocol Version 4, Src: 172.26.3.68, Dst: 193.136.9.240  
0100 ... = Version: 4  
... 0101 = Header Length: 20 bytes (5)  
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
Total Length: 60  
Identification: 0x148d (5261)  
Flags: 0x0000  
Fragment offset: 0  
Time to live: 1  
Protocol: ICMP (1)  
Header checksum: 0x2a5e [validation disabled]

Figura 8: Primeiro Pacote enviado pela máquina nativa

**d) O datagrama IP foi fragmentado? Justifique**

Não, podemos confirmar através da leitura do wireshark que não existe nenhuma notificação de fragmentação de pacote: através da flag a indicar mais fragmentos nem de offsets diferentes de 0.

No.	Time	Source	Destination	Protocol	Length	Info
8	2.370000010	193.137.16.145	172.26.3.68	DNS	102	Standard query response 0xf644 A marco.uminho.pt
9	2.370466725	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=1/256, ttl=64
10	2.370486834	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=2/512, ttl=64
11	2.370492136	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=3/768, ttl=64
12	2.370497572	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=4/1024, ttl=64
13	2.370502402	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=5/1280, ttl=64
14	2.370507073	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=6/1536, ttl=64
15	2.370512565	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=7/1792, ttl=64
16	2.370517318	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=8/2048, ttl=64
17	2.370522075	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=9/2304, ttl=64
18	2.370527245	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=10/2560, ttl=64
19	2.370532210	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=11/2816, ttl=64
20	2.370537040	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=12/3072, ttl=64
21	2.370542420	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=13/3328, ttl=64
22	2.370548096	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=14/3584, ttl=64
23	2.370553087	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=15/3840, ttl=64
24	2.370558279	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=16/4096, ttl=64
25	2.373466931	172.16.2.1	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
26	2.373516457	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=17/4352, ttl=64
27	2.373831811	172.16.2.1	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
28	2.373877780	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=18/4608, ttl=64
29	2.374678781	172.16.2.1	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
30	2.374715580	172.26.3.68	193.136.9.240	ICMP	74	Echo (ping) request id=0x0003, seq=19/4864, ttl=64
31	2.375488886	172.26.254.254	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)

Frame 9: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp0s20f3, id 0  
Ethernet II, Src: f0:77:c3:75:c3:76 (f0:77:c3:75:c3:76), Dst: ComdaEnt\_ff:94:00 (00:d0:03:ff:94:00)  
Internet Protocol Version 4, Src: 172.26.3.68, Dst: 193.136.9.240  
0100 ... = Version: 4  
... 0101 = Header Length: 20 bytes (5)  
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
Total Length: 60  
Identification: 0x148d (5261)  
Flags: 0x0000  
Fragment offset: 0

Figura 9: Primeiro pacote enviado pela máquina nativa

**e) Ordene os pacotes capturados de acordo com o endereço IP fonte (e.g., selecionando o cabeçalho da coluna Source), e analise a sequência de tráfego ICMP gerado a partir do endereço IP atribuído à interface da sua máquina. Para a sequência de mensagens ICMP enviadas pelo seu computador, indique que campos do cabeçalho IP variam de pacote para pacote.**

```
Total Length: 60  
Identification: 0x148d (5261)  
Flags: 0x00  
Fragment Offset: 0  
Time to Live: 1  
Protocol: ICMP (1)
```

Figura 10: Primeiro pacote enviado

```
Total Length: 60  
Identification: 0x148e (5262)  
Flags: 0x00  
Fragment Offset: 0  
Time to Live: 1  
Protocol: ICMP (1)
```

Figura 11: Segundo pacote enviado

```
Total Length: 60  
Identification: 0x148f (5263)  
Flags: 0x00  
Fragment Offset: 0  
Time to Live: 1  
Protocol: ICMP (1)
```

Figura 12: Terceiro pacote enviado

```
Total Length: 60  
Identification: 0x1490 (5264)  
Flags: 0x00  
Fragment Offset: 0  
Time to Live: 2
```

Figura 13: Quarto pacote enviado

As figuras anteriores apresentam os campos de IP do primeiro, segundo, terceiro e quarto pacote, respectivamente. Como são enviados 3 pacotes por 1 TTL, dentro desses grupos de 3 pacotes (Figura 11, 12 e 13 correspondem ao primeiro grupo enviado) , varia apenas o identificador. De grupo em grupo, o TTL varia adicionalmente.

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

Sim, verificamos que o identificador aumenta linearmente a cada pacote, e o TTL aumenta linearmente de 3 em 3 pacotes, iniciando o ciclo em TTL igual a 1.

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.26.3.68	162.159.135.234	TLSv1.2	105	Application Data
4	0.177156627	172.26.3.68	162.159.135.234	TCP	54	59710 → 443 [ACK] Seq=52 Ack=33 Win=1639 Len=0
2	0.034409580	162.159.135.234	172.26.3.68	TCP	54	443 → 59710 [ACK] Seq=1 Ack=52 Win=76 Len=0
3	0.177135133	162.159.135.234	172.26.3.68	TLSv1.2	86	Application Data
7	2.369968921	193.137.16.145	172.26.3.68	DNS	140	Standard query response 0xd7ff AAAA marco.umi
8	2.370080010	193.137.16.145	172.26.3.68	DNS	102	Standard query response 0xfb44 A marco.uminh
25	2.373466931	172.16.2.1	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
27	2.373831811	172.16.2.1	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
29	2.374678781	172.16.2.1	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
31	2.375488886	172.26.254.254	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
32	2.375634366	172.26.254.254	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
33	2.375911996	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=10/2560,
35	2.376348210	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=13/3328,
36	2.376987848	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=11/2816,
37	2.377219755	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=12/3072,
38	2.377498094	172.16.115.252	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
39	2.378755670	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=14/3584,
40	2.378756847	172.16.115.252	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
41	2.378755743	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=15/3840,
42	2.378756901	172.16.115.252	172.26.3.68	ICMP	70	Time-to-live exceeded (Time to live exceeded)
43	2.378755790	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=16/4096,
44	2.378756994	193.137.16.145	172.26.3.68	DNS	152	Standard query response 0x0541 No such name f
46	2.381356231	193.137.16.145	172.26.3.68	DNS	141	Standard query response 0x0541 No such name f
48	2.383342760	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=17/4352,
49	2.383342937	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=18/4608,
50	2.383342981	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=19/4864,
51	2.383343030	193.136.9.240	172.26.3.68	ICMP	74	Echo (ping) reply id=0x0003, seq=20/5120,
53	2.393887683	193.137.16.145	172.26.3.68	DNS	94	Standard query response 0xda08 Refused PTR 1
55	2.395736679	193.137.16.75	172.26.3.68	DNS	94	Standard query response 0xda08 Refused PTR 1
57	2.397806646	193.137.16.65	172.26.3.68	DNS	94	Standard query response 0xda08 Refused PTR 1
59	2.408132131	193.137.16.145	172.26.3.68	DNS	94	Standard query response 0xda08 Refused PTR 1

▶ Frame 25: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface wlp0s20f3, id 0  
 ▶ Ethernet II, Src: ComdaEnt\_ff:94:00 (00:d0:03:ff:94:00), Dst: f0:77:c3:75:c3:76 (f0:77:c3:75:c3:76)  
 ▶ Internet Protocol Version 4, Src: 172.16.2.1, Dst: 172.26.3.68  
   0100 .... = Version: 4  
   .... 0101 = Header Length: 20 bytes (5)  
   ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
   Total Length: 56  
   Identification: 0xcc66 (52326)  
   ▶ Flags: 0x0000  
   Fragment offset: 0  
   Time to live: **254**  
   Protocol: ICMP (1)  
   Header checksum: 0x92ee [validation disabled]  
   [Header checksum status: Unverified]  
   Source: 172.16.2.1  
   Destination: 172.26.3.68  
 ▶ Internet Control Message Protocol

Figura 14: Primeiro pacote recebido por TTL excedido

Nos vários pacotes ICMP com Time-to-live excedido, foi verificado que os TTL recebidos são todos elevados e maioritariamente constantes (254 / 255). Isto deve-se ao facto de que os routers e o host não terem informação da quantidade de saltos que os pacotes vão ter de percorrer para chegar ao destino. E por isso, é necessário um valor alto de TTL para confirmar a sua chegada ao destino correto.

### Questão 3 (Máquina Nativa: Ubuntu com Tamanho de pacote = 4071)

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

A primeira mensagem ICMP enviada foi a seleciona na seguinte figura:

3 0.005336652	193.137.16.145	172.26.3.68	DNS	140 Standard query response 0x0744 AAAA marco.uminho.pt SOA dns.u...
4 0.005336118	193.137.16.145	172.26.3.68	DNS	102 Standard query response 0xdd14 A marco.uminho.pt A 193.136.9...
5 0.005705150	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26a) [Reasse...
6 0.005718578	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26a) [Rea...
7 0.005721774	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=1/256, ttl=1 (no response...

Figura 15: Primeiro fragmento enviado

Esta fragmentação ocorreu por uma única razão: o tamanho do pacote que foi forçado. O facto de o traceroute enviar um pacote de tamanho tão grande, obrigou a uma fragmentação deste, mais especificamente, em 3 pacotes.

**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?**

Selecionando o primeiro fragmento do enviado, verificamos que podemos confirmar que se trata de um pacote fragmento graças à flag levantada que anuncia a existência de mais pacotes fragmentos por receber.

Identificamos que se trata do primeiro fragmento pois o seu offset é igual a 0, o que implica que na reconstrução do pacote original, ele vai ser inserido no início do pacote construído.

E finalmente, como podemos verificar pela figura, o tamanho do primeiro datagrama fragmentado é de 1500 bytes.

3 0.005336652	193.137.16.145	172.26.3.68	DNS	140 Standard query response 0x0744 AAAA marco.uminho.pt SOA dns.u...
4 0.005336118	193.137.16.145	172.26.3.68	DNS	102 Standard query response 0xdd14 A marco.uminho.pt A 193.136.9...
5 0.005705150	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26a) [Reasse...
6 0.005718578	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26a) [Rea...
7 0.005721774	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=1/256, ttl=1 (no response...
8 0.005733379	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26b) [Reasse...
9 0.005736174	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26b) [Rea...
10 0.005738845	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=2/512, ttl=1 (no response...
11 0.005740813	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26c) [Reasse...
12 0.005751577	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26c) [Rea...
13 0.005754229	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=3/768, ttl=1 (no response...
14 0.005765680	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26d) [Reasse...
15 0.005768529	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26d) [Rea...
16 0.005771268	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=4/1024, ttl=2 (no respons...
17 0.005781071	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26e) [Reasse...
18 0.005783749	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26e) [Rea...
19 0.005786452	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=5/1280, ttl=2 (no respons...
20 0.005796634	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26f) [Reasse...
21 0.005799332	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26f) [Rea...
22 0.005802005	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=6/1536, ttl=2 (no respons...
23 0.005812025	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d270) [Reasse...
24 0.005814789	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d270) [Rea...
25 0.005818290	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=7/1792, ttl=3 (no respons...
26 0.005827934	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d271) [Reasse...
27 0.005830552	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d271) [Rea...
28 0.005833107	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=8/2048, ttl=3 (no respons...
29 0.005842873	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d272) [Reasse...
30 0.005845608	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d272) [Rea...
31 0.005848488	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=9/2304, ttl=3 (no respons...

Frame 5: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: f0:77:c3:75:c3:76 (f0:77:c3:75:c3:76), Dst: ComdaEnt_ff:94:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 172.26.3.68, 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
Flags: 0x2000, More fragments
Fragment offset: 0
Time to live: 1
Protocol: ICMP (1)
Header checksum: 0x46e0 [validation disabled]
[Header checksum status: Unverified]
Source: 172.26.3.68
Destination: 193.136.9.240
Reassembled IPv4 in frame: 7
Data (1480 bytes)

Figura 16: Primeiro fragmento enviado

**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?**

Na seguinte figura, verificamos que não se trata do primeiro fragmento visto que offset já não se encontra igual a 0. No entanto, ainda existem mais fragmentos por enviar pois a flag “more fragments” ainda se encontra levantada.

1	0.000000000	172.26.3.68	193.137.16.145	DNS	86 Standard query 0xdd14 A marco.uminho.pt OPT
2	0.000109236	172.26.3.68	193.137.16.145	DNS	86 Standard query 0x0744 AAAA marco.uminho.pt OPT
3	0.005336652	193.137.16.145	172.26.3.68	DNS	140 Standard query response 0x0744 AAAA marco.uminho.pt SOA dns.uminho.pt OPT
4	0.005336118	193.137.16.145	172.26.3.68	DNS	102 Standard query response 0xdd14 A marco.uminho.pt A 193.136.9.240 OPT
5	0.005705150	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26a) [Reassembled in #7]
6	0.005718578	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26a) [Reassembled in #7]
7	0.005721774	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=1/256, ttl=1 (no response found!)
8	0.005733379	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26b) [Reassembled in #10]
9	0.005736174	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26b) [Reassembled in #10]

```

> Frame 6: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface wlp0s20f3, id 0
> Ethernet II, Src: IntelCor_75:c3:76 (f0:77:c3:75:c3:76), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.3.68, 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: 0xd26a (53866)
  > Flags: 0x20, More fragments
  Fragment Offset: 1480
  > Time to Live: 1
  Protocol: ICMP (1)
  Header Checksum: 0x4627 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.3.68
  Destination Address: 193.136.9.240
  [Reassembled IPv4 in frame: 7]

```

Figura 17: Segundo fragmento enviado

**d) Quantos fragmentos foram criados a partir do datagrama original?**

Como podemos verificar, foram criados 3 fragmentos a partir de um datagrama de tamanho 4071 bytes.

1	0.000000000	172.26.3.68	193.137.16.145	DNS	86 Standard query 0xdd14 A marco.uminho.pt OPT
2	0.000109236	172.26.3.68	193.137.16.145	DNS	86 Standard query 0x0744 AAAA marco.uminho.pt OPT
3	0.005336652	193.137.16.145	172.26.3.68	DNS	140 Standard query response 0x0744 AAAA marco.uminho.pt SOA dns.uminho.pt OPT
4	0.005336118	193.137.16.145	172.26.3.68	DNS	102 Standard query response 0xdd14 A marco.uminho.pt A 193.136.9.240 OPT
5	0.005705150	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26a) [Reassembled in #7]
6	0.005718578	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26a) [Reassembled in #7]
7	0.005721774	172.26.3.68	193.136.9.240	ICMP	1125 Echo (ping) request id=0x0004, seq=1/256, ttl=1 (no response found!)
8	0.005733379	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=d26b) [Reassembled in #10]
9	0.005736174	172.26.3.68	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=d26b) [Reassembled in #10]

```

> Frame 7: 1125 bytes on wire (9000 bits), 1125 bytes captured (9000 bits) on interface wlp0s20f3, id 0
> Ethernet II, Src: IntelCor_75:c3:76 (f0:77:c3:75:c3:76), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.3.68, 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: 1111
  Identification: 0xd26a (53866)
  > Flags: 0x01
  Fragment Offset: 2960
  > Time to Live: 1
  Protocol: ICMP (1)
  Header Checksum: 0x66f3 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.3.68
  Destination Address: 193.136.9.240
  > [3 IPv4 Fragments (4051 bytes): #5(1480), #6(1480), #7(1091)]

```

Figura 18: Terceiro e último fragmento do primeiro datagrama original

**f) 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.**

Considerando um pacote original fragmentado, cada um dos fragmentos vai ter o mesmo identificador e um offset diferente. Para construir o datagrama original, será necessário reunir todos os pacotes recebidos com o mesmo identificador até a flag de “more fragments” já não estiver levantada, e reuni-los de acordo com os seus offsets, começando pelo offset igual a 0.

```
Total Length: 1500
Identification: 0xd26a (53866)
> Flags: 0x20, More fragments
> Fragment Offset: 0
> Time to Live: 1
Protocol: ICMP (1)
Header Checksum: 0x46e0 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.26.3.68
Destination Address: 193.136.9.240
[Reassembled IPv4 in frame: 7]
```

Figura 19: Primeiro fragmento

```
Total Length: 1500
Identification: 0xd26a (53866)
> Flags: 0x20, More fragments
> Fragment Offset: 1480
> Time to Live: 1
Protocol: ICMP (1)
Header Checksum: 0x4627 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.26.3.68
Destination Address: 193.136.9.240
[Reassembled IPv4 in frame: 7]
```

Figura 20: Segundo Fragmento

```
Total Length: 1111
Identification: 0xd26a (53866)
> Flags: 0x01
> Fragment Offset: 2960
> Time to Live: 1
Protocol: ICMP (1)
Header Checksum: 0x66f3 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.26.3.68
Destination Address: 193.136.9.240
> [3 IPv4 Fragments (4051 bytes): #5(1480), #6(1480), #7(1091)]
```

Figura 21: Terceiro e último fragmento

**e) Verifique o processo de fragmentação através de um processo de cálculo.**

Sendo o tamanho original do datagrama de 4071, vamos verificar como ocorreu a fragmentação capturada pelo wireshark.

Se não ocorrer fragmentação, seria enviado um pacote com tamanho de 4071 bytes, em que 20 deles corresponderiam ao overhead e 4051 de conteúdo. No entanto, com a fragmentação, isso não aconteceu.

O que efetivamente aconteceu foi que um dos pacotes foi preenchido com o máximo de conteúdo possível, 1480 bytes, juntamente com 20 bytes, correspondendo a um tamanho total de 1500 bytes, como se pode verificar na figura 20 e 21. Esta situação ocorre 2 vezes, pois o tamanho do pacote obriga a recorrer a 3 pacotes fragmentados. No entanto, no terceiro fragmento, a quantidade de dados restante vai ser igual a  $4051 - 1480 - 1480 = 1091$ . Sendo esta a quantidade de dados, e 20 bytes de overhead, vai resultar num pacote de tamanho de 1111, confirmado pela figura 22.

***h) Escreva uma expressão lógica que permita detetar o último fragmento correspondente ao datagrama original***

Através de pseudocódigo conseguimos fazer esta identificação:

```
While (flag != "more fragments" && wantedID == currentID && offset != 0)
    Fragment = newFragment
LastFragment = Fragment
```



## Parte 2

### Questão 1

**a) Indique que endereços IP e máscaras de rede foram atribuídos pelo CORE a cada equipamento. Para simplificar, pode incluir uma imagem que ilustre de forma clara a topologia definida e o endereçamento usado.**

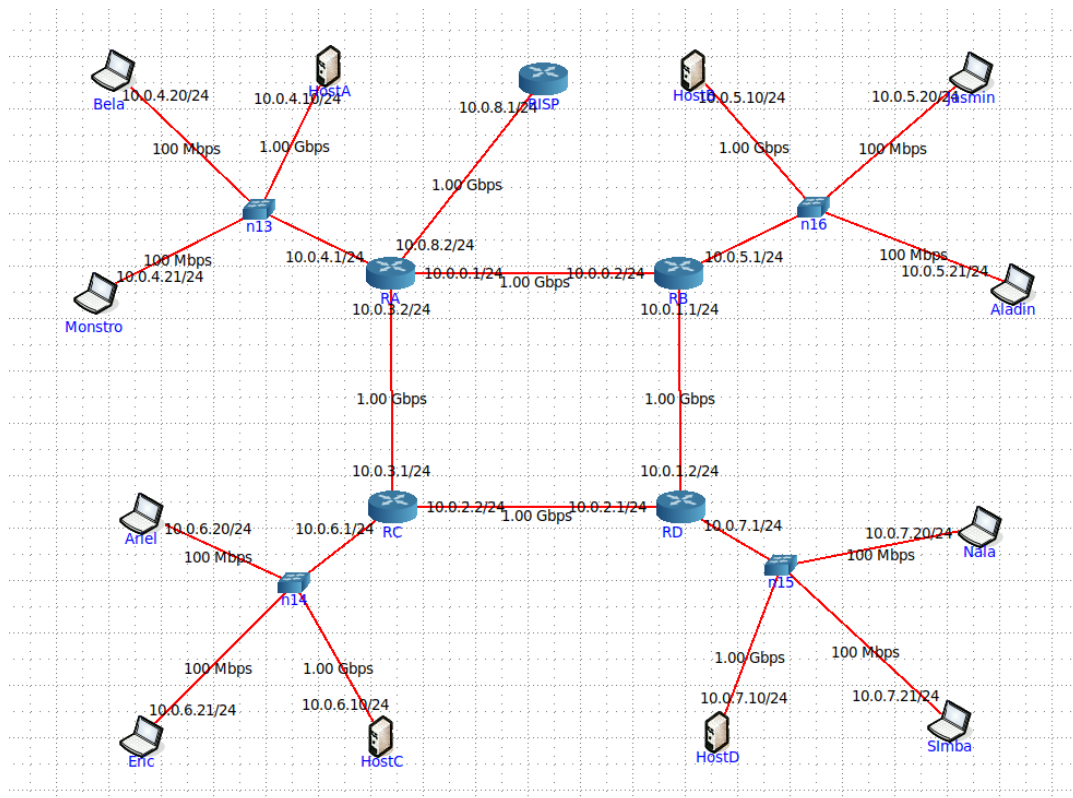


Figura 22: Topologia da Rede

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

Uma das faixas de endereços que está reservada é para os endereços privados, que varia entre os 10.0.0.0/24 e 10.255.255.255/24. E assim confirmamos pelos endereços da figura 23 que estão todos dentro dessa gama, e por isso são privados.

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

Os ethernet switches são instrumentos que operam apenas no nível 2 da pilha protocolar. Isto significando que eles não operam a nível da rede e por isso não precisam de adotar um endereço IP.



**d) Usando o comando ping certifique-se que existe conectividade IP interna a cada departamento (e.g. entre um laptop e o servidor respectivo).**

```
root@Bela:/tmp/pycore.45005/Bela.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=64 time=0.427 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=64 time=0.186 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=64 time=0.207 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=64 time=0.206 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=64 time=0.157 ms
```

*Figura 23: Comando Ping de Bela para SA*

```
root@Jasmin:/tmp/pycore.45005/Jasmin.conf# ping 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=0.600 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=64 time=0.162 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=64 time=0.192 ms
64 bytes from 10.0.1.1: icmp_seq=4 ttl=64 time=0.162 ms
64 bytes from 10.0.1.1: icmp_seq=5 ttl=64 time=0.704 ms
64 bytes from 10.0.1.1: icmp_seq=6 ttl=64 time=0.127 ms
```

*Figura 24: Comando Ping de Jasmin para SB*

```
root@Eric:/tmp/pycore.45005/Eric.conf# ping 10.0.6.1
PING 10.0.6.1 (10.0.6.1) 56(84) bytes of data.
64 bytes from 10.0.6.1: icmp_seq=1 ttl=64 time=0.543 ms
64 bytes from 10.0.6.1: icmp_seq=2 ttl=64 time=0.178 ms
64 bytes from 10.0.6.1: icmp_seq=3 ttl=64 time=0.204 ms
```

*Figura 25: Comando Ping de Eric para SC*

```
root@Nala:/tmp/pycore.45005/Nala.conf# ping 10.0.7.1
PING 10.0.7.1 (10.0.7.1) 56(84) bytes of data.
64 bytes from 10.0.7.1: icmp_seq=1 ttl=64 time=0.638 ms
64 bytes from 10.0.7.1: icmp_seq=2 ttl=64 time=0.171 ms
64 bytes from 10.0.7.1: icmp_seq=3 ttl=64 time=0.167 ms
64 bytes from 10.0.7.1: icmp_seq=4 ttl=64 time=0.177 ms
```

*Figura 26: Comando ping de Nala para SD*

Podemos verificar pelas figuras acima que em todos os departamentos é possível estabelecer conectividade interna.

**e) Execute o número mínimo de comandos ping que lhe permite verificar a existência de conectividade IP entre departamentos.**

```
root@Bela:/tmp/pycore.45005/Bela.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
64 bytes from 10.0.5.10: icmp_seq=1 ttl=62 time=0.666 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=62 time=0.298 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=62 time=0.297 ms
64 bytes from 10.0.5.10: icmp_seq=4 ttl=62 time=0.310 ms
64 bytes from 10.0.5.10: icmp_seq=5 ttl=62 time=0.358 ms
```

Figura 27: Comando ping de Bela para SB

```
root@Bela:/tmp/pycore.45005/Bela.conf# ping 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data.
64 bytes from 10.0.6.10: icmp_seq=1 ttl=62 time=0.638 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=62 time=0.378 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=62 time=0.312 ms
64 bytes from 10.0.6.10: icmp_seq=4 ttl=62 time=0.316 ms
```

Figura 28: Comando Ping de Bela para SC

```
root@Bela:/tmp/pycore.45005/Bela.conf# ping 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data.
64 bytes from 10.0.7.10: icmp_seq=1 ttl=61 time=5.50 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=61 time=0.961 ms
64 bytes from 10.0.7.10: icmp_seq=3 ttl=61 time=0.420 ms
64 bytes from 10.0.7.10: icmp_seq=4 ttl=61 time=2.78 ms
64 bytes from 10.0.7.10: icmp_seq=5 ttl=61 time=1.53 ms
```

Figura 29: Comando Ping de Bela para SD

Com estes 3 comandos confirmamos o estabelecimento da ligação entre A com B,C e D. E o facto de se tratar de um estabelecimento de ligação podemos afirmar que as ligações inversas também estão operacionais. A seguir temos as ligações que faltavam averiguar relativamente ao departamento C

```
root@Eric:/tmp/pycore.45005/Eric.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
64 bytes from 10.0.5.10: icmp_seq=1 ttl=61 time=0.764 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=61 time=0.428 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=61 time=0.536 ms
64 bytes from 10.0.5.10: icmp_seq=4 ttl=61 time=0.397 ms
64 bytes from 10.0.5.10: icmp_seq=5 ttl=61 time=0.418 ms
64 bytes from 10.0.5.10: icmp_seq=6 ttl=61 time=0.539 ms
```

Figura 30: Comando Ping de Eric para SB

```
root@Eric:/tmp/pycore.45005/Eric.conf# ping 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data.
64 bytes from 10.0.7.10: icmp_seq=1 ttl=62 time=0.587 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=62 time=0.279 ms
64 bytes from 10.0.7.10: icmp_seq=3 ttl=62 time=0.331 ms
64 bytes from 10.0.7.10: icmp_seq=4 ttl=62 time=0.423 ms
```

Figura 31: Comando Ping de Eric para SD

E finalmente, temos a confirmação da ligação restante, D e B:

```
root@Simba:/tmp/pycore.45005/Simba.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
64 bytes from 10.0.5.10: icmp_seq=1 ttl=62 time=0.630 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=62 time=0.203 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=62 time=0.270 ms
64 bytes from 10.0.5.10: icmp_seq=4 ttl=62 time=0.327 ms
```

*Figura 32: Comando Ping de Simba para SB*

Concluindo, para realizar a testagem de ligação entre departamentos são necessários no mínimo 6 comandos pings nesta topologia.

***f) Verifique se existe conectividade IP do portátil Bela para o router de acesso RISP***

```
root@Bela:/tmp/pycore.45005/Bela.conf# ping 10.0.8.1
PING 10.0.8.1 (10.0.8.1) 56(84) bytes of data.
64 bytes from 10.0.8.1: icmp_seq=1 ttl=63 time=0.887 ms
64 bytes from 10.0.8.1: icmp_seq=2 ttl=63 time=0.244 ms
64 bytes from 10.0.8.1: icmp_seq=3 ttl=63 time=0.236 ms
64 bytes from 10.0.8.1: icmp_seq=4 ttl=63 time=0.250 ms
64 bytes from 10.0.8.1: icmp_seq=5 ttl=63 time=0.333 ms
```

*Figura 33: Comando Ping de Bela para RISP*

Confirmamos com esta figura que efetivamente existe uma conectividade entre Bela e o Router ISP.

## Questão 2

a) Execute o comando `netstat -rn` por forma a poder consultar a tabela de encaminhamento unicast (IPv4). Inclua no seu relatório as tabelas de encaminhamento obtidas; interprete as várias entradas de cada tabela. Se necessário, consulte o manual respetivo (`man netstat`).

```
root@Bela:/tmp/pycore.45005/Bela.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@Bela:/tmp/pycore.45005/Bela.conf#
```

Figura 34: Tabela de Encaminhamento de Bela

Através do comando `netstat -rn` conseguimos obter a tabela de encaminhamento de Bela, e com isto podemos passar à sua análise sucinta. A tabela tem apenas 2 entradas, a segunda consiste na situação de quando um pacote com destino à sub-rede 10.0.4.0/24 alcançar Bela, ele próprio vai redirecionar esses pacotes a toda a sua sub-rede, e por isso é que o seu próximo salto vai ser ele mesmo. Logo o 0.0.0.0 da segunda entrada vai corresponder a 10.0.4.20. A primeira entrada corresponde a todos os outros destinos que não sejam a sua sub-rede e vai encaminhá-los para o seu router, 10.0.4.1.

```
root@RA:/tmp/pycore.45005/RA.conf# netstat -rn
Kernel IP routing table
Destination    Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0       0.0.0.0         255.255.255.0   U        0 0        0 eth0
10.0.1.0       10.0.0.2        255.255.255.0   UG        0 0        0 eth0
10.0.2.0       10.0.3.1        255.255.255.0   UG        0 0        0 eth1
10.0.3.0       0.0.0.0         255.255.255.0   U        0 0        0 eth1
10.0.4.0       0.0.0.0         255.255.255.0   U        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       10.0.3.1        255.255.255.0   UG        0 0        0 eth1
10.0.7.0       10.0.0.2        255.255.255.0   UG        0 0        0 eth0
10.0.8.0       0.0.0.0         255.255.255.0   U        0 0        0 eth3
root@RA:/tmp/pycore.45005/RA.conf#
```

Figura 35: Tabela de Encaminhamento de RA

Relativamente à tabela de encaminhamento do router A, esta é constituída por 9 entradas. Para facilitar a interpretação foi criada a seguinte tabela que permite melhor visualizar e interpretar a tabela acima apresentada.

Destino	Pacote já se encontra na rede destino?	Próximo Salto	Observações
10.0.0.0	Sim	10.0.0.1	O pacote já se encontra na ligação ponto a ponto destino, por isso, ele próprio vai encaminhar o pacote.
10.0.1.0	Não	10.0.0.2	O pacote tem de chegar a ligação ponto a ponto entre B e D, e sendo B o mais próximo, o pacote vai ser encaminhado para a interface do router de B.
10.0.2.0	Não	10.0.3.1	O pacote tem de chegar a ligação ponto a ponto entre C e D, e sendo C o mais próximo, o pacote vai ser encaminhado para a interface do router de C.
10.0.3.0	Sim	10.0.0.1	O pacote já se encontra na ligação ponto a ponto destino, por isso, ele próprio vai encaminhar o pacote.
10.0.4.0	Sim	10.0.0.1	O pacote já se encontra na ligação ponto a ponto destino, por isso, ele próprio vai encaminhar o pacote.
10.0.5.0	Não	10.0.0.2	O pacote tem de chegar à sub-rede do departamento B, então o pacote vai ser encaminhado para a interface do router B.
10.0.6.0	Não	10.0.3.1	O pacote tem de chegar à sub-rede do departamento C, então o pacote vai ser encaminhado para a interface do router C.
10.0.7.0	Não	10.0.0.2	O pacote tem de chegar à sub-rede do departamento D, então o pacote vai ser encaminhado para a interface do router B, para ser encaminhado por seguida para a interface do router D.
10.0.8.0	Sim	10.0.0.1	O pacote já se encontra na ligação ponto a ponto destino, por isso, ele próprio vai encaminhar o pacote.

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

Como podemos verificar pela seguinte figura, no RouterISP, confirmamos que existe um processo a decorrer, ospfd (open shortest path first daemon). Correspondendo a um processo que procura constantemente o melhor caminho para fazer o encaminhamento, por isso, trata-se de um encaminhamento dinâmico.

```

root@RISP:/tmp/pycore,37079/RISP.conf# ps -ax
  PID TTY          STAT       TIME COMMAND
    1 ?           S          0:00 vncnode -v -c /tmp/pycore,37079/RISP -1 /tmp/pycore,
   39 ?           Ss         0:00 /usr/local/sbin/zebra -d
   47 ?           Ss         0:00 /usr/local/sbin/ospf6d -d
   51 ?           Ss         0:00 /usr/local/sbin/ospfd -d
   58 pts/3       Ss         0:00 /bin/bash
   65 pts/3       R+         0:00 ps -ax
root@RISP:/tmp/pycore,37079/RISP.conf#

```

**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 SA. Use o comando `route delete` para o efeito. Que implicações tem esta medida para os utilizadores da LEI-RC que acedem ao servidor. Justifique.**

```
root@HostA:/tmp/pycore.43025/HostA.conf# route delete default
root@HostA:/tmp/pycore.43025/HostA.conf# netstat -rn
Kernel IP routing table
Destination    Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.4.0        0.0.0.0         255.255.255.0   U        0  0        0 eth0
root@HostA:/tmp/pycore.43025/HostA.conf#
```

Figura 36: Eliminação da rota default

Com a eliminação da rota por defeito, torna-se impossível alguma comunicação feita para fora da rede 10.0.4.0/24 que tenha sido direccionada para o servidor A. Isto acontece, pois, quando algum pacote chega ao servidor A, este não vai saber qual será o seu redireccionamento quando não for da sua própria rede (pois a entrada correspondente à sua rede ainda existe, e esses pacotes, o servidor consegue entregá-los ele próprio).

**d) Não volte a repor a rota por defeito. Adicione todas as rotas estáticas necessárias para restaurar a conectividade para o servidor SA, por forma a contornar a restrição imposta na alínea c). Utilize para o efeito o comando `route add` e registe os comandos que usou.**

```
root@HostA:/tmp/pycore.43025/HostA.conf# route delete default
root@HostA:/tmp/pycore.43025/HostA.conf# route add -net 10.0.5.0 gw 10.0.4.1 netmask 255.255.255.0
root@HostA:/tmp/pycore.43025/HostA.conf# route add -net 10.0.6.0 gw 10.0.4.1 netmask 255.255.255.0
root@HostA:/tmp/pycore.43025/HostA.conf# route add -net 10.0.7.0 gw 10.0.4.1 netmask 255.255.255.0
root@HostA:/tmp/pycore.43025/HostA.conf#
```

Figura 37: Comandos utilizados para repor rotas necessárias para desfazer efeito de c)

**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.**

```
root@HostA:/tmp/pycore.43025/HostA.conf# netstat -rn
Kernel IP routing table
Destination    Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.4.0        0.0.0.0         255.255.255.0   U        0  0        0 eth0
10.0.5.0        10.0.4.1        255.255.255.0   UG        0  0        0 eth0
10.0.6.0        10.0.4.1        255.255.255.0   UG        0  0        0 eth0
10.0.7.0        10.0.4.1        255.255.255.0   UG        0  0        0 eth0
root@HostA:/tmp/pycore.43025/HostA.conf#
```

Figura 38: Nova tabela de encaminhamento do servidor A

Com esta nova tabela de endereçamento, verificamos que está novamente funcional através das seguintes figuras a provar que existe conexão de novo para cada um dos departamentos.

```

root@Eric:/tmp/pycore.43025/Eric.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=0.705 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=0.358 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=0.647 ms
^C
--- 10.0.4.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.358/0.570/0.705/0.151 ms
root@Eric:/tmp/pycore.43025/Eric.conf# █

```

Figura 39: Comando ping do Dep. C para o servidor A

```

root@Simba:/tmp/pycore.43025/Simba.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=61 time=0.839 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=61 time=0.898 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=61 time=0.443 ms
^C
--- 10.0.4.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2057ms
rtt min/avg/max/mdev = 0.443/0.726/0.898/0.202 ms
root@Simba:/tmp/pycore.43025/Simba.conf# █

```

Figura 40: Comando ping do Dep.D para o servidor A

```

root@Jasmin:/tmp/pycore.43025/Jasmin.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=0.683 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=0.325 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=0.302 ms
^C
--- 10.0.4.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.302/0.436/0.683/0.174 ms
root@Jasmin:/tmp/pycore.43025/Jasmin.conf# █

```

Figura 41: Comando ping do Dep. B para o servidor A

### Questão 3

a) Considere que dispõe apenas do endereço de rede IP 192.168.XXX.128/25, em que XXX é o decimal correspondendo ao seu número de grupo (PLXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo as redes de acesso externo e backbone inalteradas), sabendo que o número de departamentos pode vir a aumentar no curto prazo. Atribua endereços às interfaces dos vários sistemas envolvidos. Assuma que todos os endereços de sub-redes são usáveis. Justifique as opções tomadas no planejamento.

Considerando que o endereço IP disponível para realizar a sub-netting em todos os departamentos tem como máscara de 25, sobram 7 bits. Optamos por utilizar 4 bits para sub-redes e 3 bits para hosts.

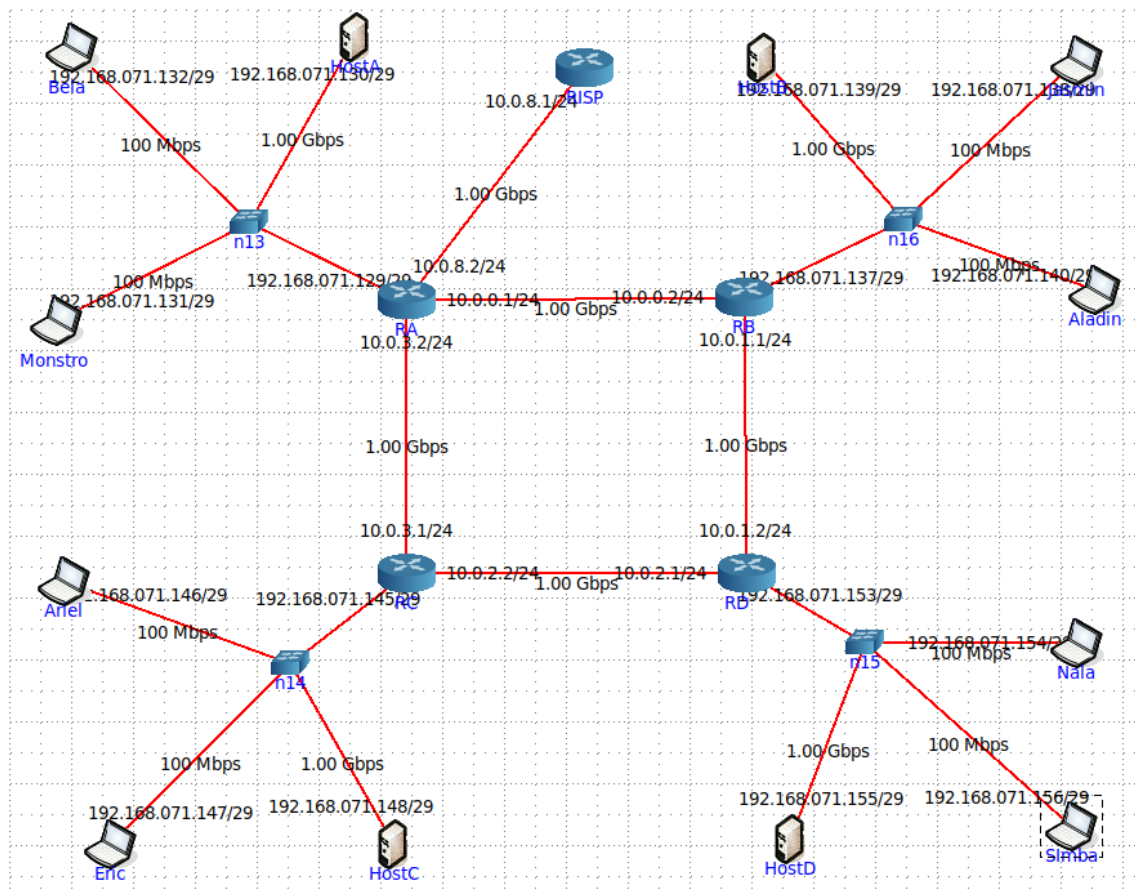


Figura 42: Topologia com sub-netting

b) Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP pode interligar em cada departamento? Quantos prefixos de sub-rede ficam disponíveis para uso futuro? Justifique

Optamos por reservar 3 bits para o host (de cada departamento) possibilitando a existência de  $2^3$  (nr de bits) - 2 (reservados) = 6 hosts conectados a cada departamento. Sobram assim 4 bits para sub-redes, que se constatou que corresponde ao maior número de departamentos possíveis de ser endereçados,  $2^4$  (nr de bits) = 16 departamentos, 12 departamentos para endereçamento futuro. Em relação a máscara, esta aumentou para 29 bits (255.255.255.248).



**c) Verifique e garanta que a conectividade IP interna na rede local LEI-RC é mantida. No caso de não existência de conectividade, reveja a atribuição de endereços efetuada e eventuais erros de encaminhamento por forma a realizar as correções necessárias. Explique como procedeu.**

O grupo verificou que algumas alterações tiveram de ser feitas, mais especificamente a troca de endereçamento dos 4 routers pelo endereço menor da sua sub-rede de modo a criar tabelas de endereçamento cuja entrada default tenha como gateway o router de cada um dos departamentos. Só com esta alteração foi possível a comunicação entre departamentos, pois o endereçamento para a sua própria rede é possível de ambas as formas.

Finalmente, verificamos a existência de conexão com um ping entre departamentos:

```
root@Bela:/tmp/pycore.35169/Bela.conf# ping 192.168.071.140
PING 192.168.071.140 (192.168.57.140) 56(84) bytes of data.
64 bytes from 192.168.57.140: icmp_seq=1 ttl=62 time=0.972 ms
64 bytes from 192.168.57.140: icmp_seq=2 ttl=62 time=0.286 ms
64 bytes from 192.168.57.140: icmp_seq=3 ttl=62 time=0.345 ms
```

Figura 43: Comando ping entre Departamento A e B

```
root@Bela:/tmp/pycore.35169/Bela.conf# ping 192.168.071.147
PING 192.168.071.147 (192.168.57.147) 56(84) bytes of data.
64 bytes from 192.168.57.147: icmp_seq=1 ttl=62 time=0.956 ms
64 bytes from 192.168.57.147: icmp_seq=2 ttl=62 time=0.374 ms
64 bytes from 192.168.57.147: icmp_seq=3 ttl=62 time=0.539 ms
64 bytes from 192.168.57.147: icmp_seq=4 ttl=62 time=0.152 ms
```

Figura 44: Comando ping entre departamento A e C

```
root@Bela:/tmp/pycore.35169/Bela.conf# ping 192.168.071.156
PING 192.168.071.156 (192.168.57.156) 56(84) bytes of data.
64 bytes from 192.168.57.156: icmp_seq=1 ttl=61 time=1.11 ms
64 bytes from 192.168.57.156: icmp_seq=2 ttl=61 time=0.491 ms
64 bytes from 192.168.57.156: icmp_seq=3 ttl=61 time=0.522 ms
64 bytes from 192.168.57.156: icmp_seq=4 ttl=61 time=0.455 ms
```

Figura 45: Comando ping entre departamento A e D

```
root@Jasmin:/tmp/pycore.35169/Jasmin.conf# ping 192.168.071.147
PING 192.168.071.147 (192.168.57.147) 56(84) bytes of data.
64 bytes from 192.168.57.147: icmp_seq=1 ttl=61 time=0.975 ms
64 bytes from 192.168.57.147: icmp_seq=2 ttl=61 time=0.539 ms
64 bytes from 192.168.57.147: icmp_seq=3 ttl=61 time=0.380 ms
64 bytes from 192.168.57.147: icmp_seq=4 ttl=61 time=0.404 ms
```

Figura 46: Comando ping entre departamento B e C

```
root@Jasmin:/tmp/pycore.35169/Jasmin.conf# ping 192.168.071.156
PING 192.168.071.156 (192.168.57.156) 56(84) bytes of data.
64 bytes from 192.168.57.156: icmp_seq=1 ttl=62 time=0.761 ms
64 bytes from 192.168.57.156: icmp_seq=2 ttl=62 time=0.347 ms
64 bytes from 192.168.57.156: icmp_seq=3 ttl=62 time=0.385 ms
64 bytes from 192.168.57.156: icmp_seq=4 ttl=62 time=0.392 ms
```

Figura 47: Comando ping entre departamento B e D

```
root@Eric:/tmp/pycore.35169/Eric.conf# ping 192.168.071.156
PING 192.168.071.156 (192.168.57.156) 56(84) bytes of data.
64 bytes from 192.168.57.156: icmp_seq=1 ttl=62 time=0.826 ms
64 bytes from 192.168.57.156: icmp_seq=2 ttl=62 time=0.314 ms
64 bytes from 192.168.57.156: icmp_seq=3 ttl=62 time=0.356 ms
64 bytes from 192.168.57.156: icmp_seq=4 ttl=62 time=0.336 ms
```

Figura 48: Comando ping entre departamento C e D

## Conclusão

Terminado o trabalho prático 2, foram atingidos todos os objetivos propostos pelos docentes. Os resultados obtidos estão maioritariamente em concordância com os resultados esperados das experiências realizadas. Além disso, encontramos-nos bastante satisfeitos com o mais à-vontade sobre tópicos como endereçamento de pacotes, subnneting e tabelas de endereçamento. As dúvidas que restaram de aulas teóricas foram completamente extintas com a elaboração deste trabalho prático e esperamos que ocorra o mesmo nos próximos trabalhos de modo a construir uma base robusta para eventuais avanços na dificuldade.