

TP2: Protocolo IPv4

Diogo Braga, João Silva, and Ricardo Caçador

University of Minho, Department of Informatics, 4710-057 Braga, Portugal

e-mail: {a82547,a82005,a81064}@alunos.uminho.pt

PL4, Grupo 7

1 Captura de tráfego IP

1.1 Exercício 1

Prepare uma topologia CORE para verificar o comportamento do traceroute. Ligue um host (pc) h1 a um router r2; o router r2 a um router r3, que por sua vez, se liga a um host (servidor) s4. (Note que pode não existir conectividade IP imediata entre h1 e s4 até que o routing estabilize). Ajuste o nome dos equipamentos atribuídos por defeito para a topologia do enunciado.

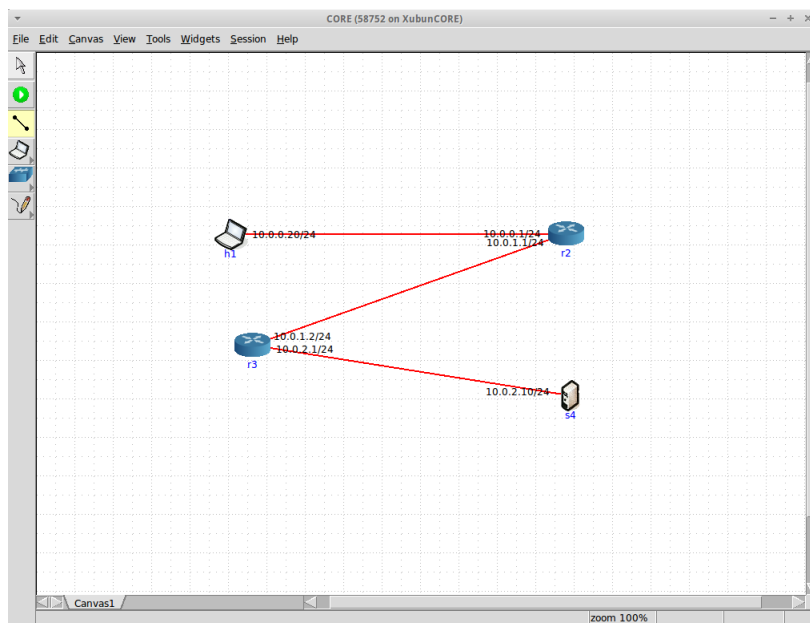


Fig. 1. Rede do exercício 1 do enunciado.

- a. Active o wireshark ou o tcpdump no pc h1. Numa shell de h1, execute o comando `traceroute-I` para o endereço IP do host s4.

```

root@h1:/tmp/pycore.58752/h1.conf# traceroute -I 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  A0 (10.0.0.1)  0.053 ms  0.008 ms  0.006 ms
 2  10.0.1.2 (10.0.1.2)  0.024 ms  0.008 ms  0.007 ms
 3  10.0.2.10 (10.0.2.10)  0.026 ms  0.010 ms  0.010 ms
root@h1:/tmp/pycore.58752/h1.conf#

```

Fig. 2. Shell de h1 com comando traceroute.

R: Como se pode observar na figura 2, os pacotes passam por 2 routers, cujos IPs das interfaces ativas de cada um são, respetivamente, 10.0.0.1 e 10.0.1.2, até chegarem ao destino cujo IP da sua interface ativa é 10.0.2.10.

b. Registe e analise o tráfego ICMP enviado por h1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.

5	18.782172	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x007f, seq=1/256, ttl=1
6	18.782194	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
7	18.782223	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x007f, seq=2/512, ttl=1
8	18.782227	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
9	18.782233	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x007f, seq=3/768, ttl=1
10	18.782236	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
11	18.782241	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x007f, seq=4/1024, ttl=2
12	18.782253	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
13	18.782258	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x007f, seq=5/1280, ttl=2
14	18.782263	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
15	18.782267	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x007f, seq=6/1536, ttl=2
16	18.782274	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
17	18.782278	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x007f, seq=7/1792, ttl=3
18	18.782291	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x007f, seq=7/1792, ttl=62

Fig. 3. Tráfego capturado pelo Wireshark.

R: Como predefinição o traceroute envia 3 datagramas com o mesmo TTL pelo que capturamos 3 vezes mais mensagens ICMP. De modo a simplificar utilizamos o comando "traceroute -I 10.0.2.10 -q 1" que apenas envia um datagrama, facilitando a análise do tráfego.

7	15.361875	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0080, seq=1/256, ttl=1
8	15.361893	10.0.0.1	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
9	15.361929	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0080, seq=2/512, ttl=2
10	15.361961	10.0.1.2	10.0.0.20	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
11	15.361971	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request id=0x0080, seq=3/768, ttl=3
12	15.361999	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply id=0x0080, seq=3/768, ttl=62

Fig. 4. Tráfego capturado pelo Wireshark.

Analisando o tráfego percebe-se que são enviadas duas mensagens do tipo *Time-to-live exceeded* de cada um dos routers para h1. Já se esperava este resultado uma vez que o traceroute é um processo iterativo que permite descobrir a rota até ao destino "salto-a-salto". As mensagens ICMP são enviadas para h1 quando o TTL é 1 e 2, respetivamente, pelo primeiro router (10.0.0.1) e pelo segundo router (10.0.1.2). Neste caso o TTL é excedido. Como esperado quando o TTL é 3, o datagrama já chega ao destino e o destino responde como se pode verificar na última linha da figura 4.

c. Qual deve ser o valor inicial mínimo do campo TTL para alcançar o destino s 4 ? Verifique na prática que a sua resposta está correta.

R: O valor mínimo do campo TTL para alcançar o destino s4 deve ser 3, uma vez que o TTL é decrementado aquando da passagem em cada router. Para verificar este valor utilizamos o comando "*tracert -I 10.0.2.10 -f 3*" que define o valor do primeiro TTL a ser usado como 3. Pela análise de tráfego da figura 5 percebe-se que o datagrama atinge o destino logo na primeira tentativa.

```
root@h1:/tmp/pycore.58752/h1.conf# traceroute -I 10.0.2.10 -f 3
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 3 10.0.2.10 (10.0.2.10)  0.039 ms  0.009 ms  0.007 ms
root@h1:/tmp/pycore.58752/h1.conf#
```

Fig. 5. Shell de h1 com comando traceroute.

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

```
root@h1:/tmp/pycore.58752/h1.conf# ping 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data:
64 bytes from 10.0.2.10: icmp_req=1 ttl=62 time=0.048 ms
64 bytes from 10.0.2.10: icmp_req=2 ttl=62 time=0.050 ms
64 bytes from 10.0.2.10: icmp_req=3 ttl=62 time=0.048 ms
64 bytes from 10.0.2.10: icmp_req=4 ttl=62 time=0.049 ms
64 bytes from 10.0.2.10: icmp_req=5 ttl=62 time=0.208 ms
64 bytes from 10.0.2.10: icmp_req=6 ttl=62 time=0.049 ms
64 bytes from 10.0.2.10: icmp_req=7 ttl=62 time=0.046 ms
64 bytes from 10.0.2.10: icmp_req=8 ttl=62 time=0.045 ms
64 bytes from 10.0.2.10: icmp_req=9 ttl=62 time=0.146 ms
64 bytes from 10.0.2.10: icmp_req=10 ttl=62 time=0.058 ms
64 bytes from 10.0.2.10: icmp_req=11 ttl=62 time=0.051 ms
64 bytes from 10.0.2.10: icmp_req=12 ttl=62 time=0.155 ms
64 bytes from 10.0.2.10: icmp_req=13 ttl=62 time=0.049 ms
64 bytes from 10.0.2.10: icmp_req=14 ttl=62 time=0.081 ms
64 bytes from 10.0.2.10: icmp_req=15 ttl=62 time=0.270 ms
64 bytes from 10.0.2.10: icmp_req=16 ttl=62 time=0.220 ms
64 bytes from 10.0.2.10: icmp_req=17 ttl=62 time=0.194 ms
64 bytes from 10.0.2.10: icmp_req=18 ttl=62 time=0.059 ms
64 bytes from 10.0.2.10: icmp_req=19 ttl=62 time=0.099 ms
64 bytes from 10.0.2.10: icmp_req=20 ttl=62 time=0.052 ms
64 bytes from 10.0.2.10: icmp_req=21 ttl=62 time=0.233 ms
64 bytes from 10.0.2.10: icmp_req=22 ttl=62 time=0.215 ms
^C
--- 10.0.2.10 ping statistics ---
22 packets transmitted, 22 received, 0% packet loss, time 20996ms
rtt min/avg/max/mdev = 0.045/0.110/0.270/0.076 ms
root@h1:/tmp/pycore.58752/h1.conf#
```

Fig. 6. Shell de h1 com comando ping.

R: Como se pode verificar pela figura 6, na penúltima linha estão os valores do Round-Trip Time (rtt), e o valor correspondente ao valor médio é o avg (*average*), que é **0.110 milissegundos**.

1.2 Exercício 2

Procedimento a seguir: Usando o wireshark capture o tráfego gerado pelo traceroute para os seguintes tamanhos de pacote: (i) sem especificar, i.e., usando o tamanho por defeito; e (ii) 35XX bytes, em que XX é o seu número de grupo. Utilize como máquina destino o host marco.uminho.pt. Pare a captura. Com base no tráfego capturado, identifique os pedidos ICMP Echo Request e o conjunto de mensagens devolvidas em resposta a esses

pedidos. Selecione a primeira mensagem ICMP capturada (referente a (i) tamanho por defeito) e centre a análise no nível protocolar IP (expanda o tab correspondente na janela de detalhe do wireshark). Através da análise do cabeçalho IP diga:

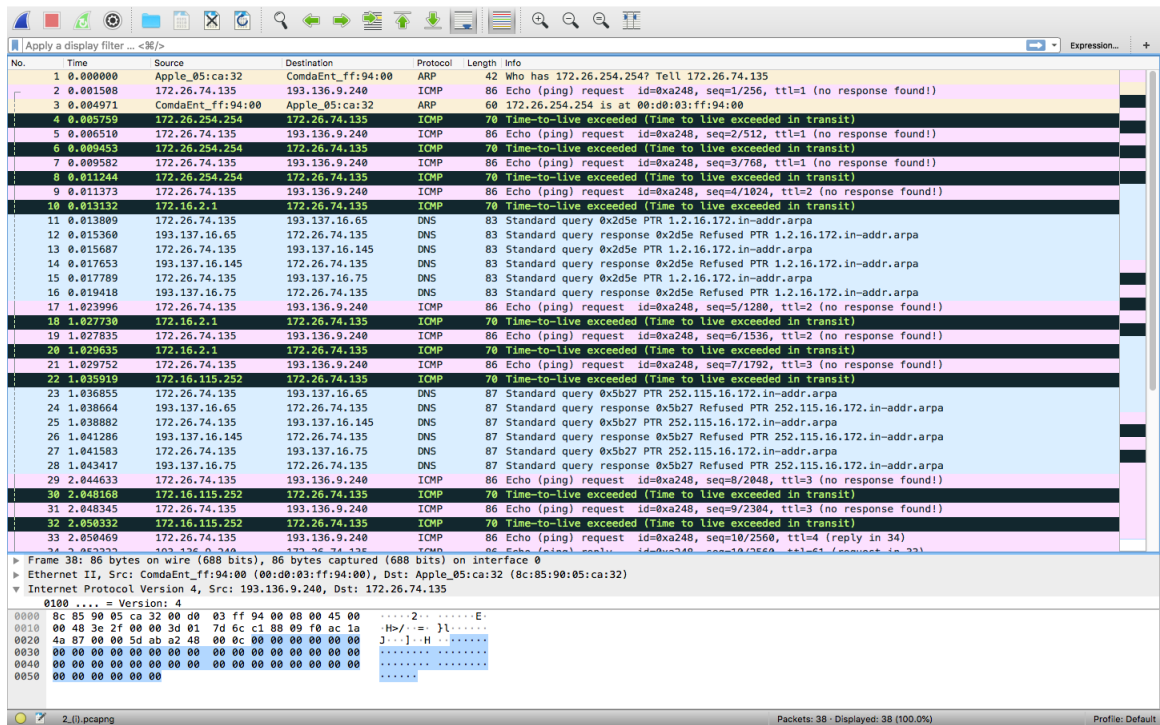


Fig. 7. Visão geral do tráfego gerado pelo traceroute usando o tamanho por defeito.

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

R: Selecionando a primeira mensagem ICMP capturada, é atingida a janela presente na figura 8. Nesta janela é possível observar a secção do IPv4 com o endereço IP da interface ativa do nosso computador, **172.26.74.135**.

Concluimos que é este o endereço pois na secção do ICMP é possível visualizar que esta mensagem possui tipo 8. O tipo 8 descreve a mensagem de **echo request**. Este género de mensagem é um utilitário que usa o protocolo ICMP para testar a conectividade entre equipamentos. Consiste no envio de pacotes para o equipamento de destino e na captura das respostas. Se o equipamento de destino estiver ativo, uma "resposta" será devolvida ao computador solicitante. Por isso, concluimos que o nosso computador é o Source desta mensagem.

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

R: Como se pode observar na figura 9, o valor do campo protocolo é **ICMP (1)**. Este campo identifica o protocolo usado para a transmissão do datagrama. Neste caso, o ICMP (Internet Control Message Protocol) é usado para testar a conectividade entre a origem e o destino.

```

▶ Frame 2: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.74.135, 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: 72
    Identification: 0xa249 (41545)
    ▶ Flags: 0x0000
    ▶ Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x5552 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.74.135
    Destination: 193.136.9.240
▼ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x55b6 [correct]
    [Checksum Status: Good]
0000 00 d0 03 ff 94 00 8c 85 90 05 ca 32 08 00 45 00 ..... 2..E.
0010 00 48 a2 49 00 00 01 01 55 52 ac 1a 4a 87 c1 88 ..H.I....UR..J...
0020 09 f0 08 00 55 b6 a2 48 00 01 00 00 00 00 00 00 ...U..H.....
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Fig. 8. Endereço IP da interface ativa do computador, sublinhado a vermelho.

```

▶ Frame 2: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.74.135, 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: 72
    Identification: 0xa249 (41545)
    ▶ Flags: 0x0000
    ▶ Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x5552 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.74.135
    Destination: 193.136.9.240
▼ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x55b6 [correct]
    [Checksum Status: Good]
0000 00 d0 03 ff 94 00 8c 85 90 05 ca 32 08 00 45 00 ..... 2..E.
0010 00 48 a2 49 00 00 01 01 55 52 ac 1a 4a 87 c1 88 ..H.I....UR..J...
0020 09 f0 08 00 55 b6 a2 48 00 01 00 00 00 00 00 00 ...U..H.....
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Fig. 9. Valor do campo protocolo, sublinhado a vermelho.

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 ?

R: Os bytes do cabeçalho IP(v4) podem ser visualizados na figura 10 sublinhado a vermelho, neste caso **20 bytes**. O payload é parte principal dos dados transmitidos, da qual se excluem as informações utilizadas para, por exemplo, realizar a entrega, como cabeçalhos. Os bytes do payload são calculados subtraindo o número de bytes totais do datagrama (na figura 10 sublinhado a laranja) pelo número de bytes do cabeçalho antes mencionado. Portanto, o número de bytes do payload são $(72 - 20 =) 52$ bytes.

```

▶ Frame 2: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.74.135, 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: 72
    Identification: 0xa249 (41545)
    ▶ Flags: 0x0000
    ▶ Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x5552 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.74.135
    Destination: 193.136.9.240
▼ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x55b6 [correct]
    [Checksum status: Good]
0000 00 d0 03 ff 94 00 8c 85 90 05 ca 32 08 00 45 00 .....2..E.
0010 00 48 a2 49 00 00 01 01 55 52 ac 1a 4a 87 c1 88 .H.I....UR..J..
0020 09 f0 08 00 55 b6 a2 48 00 01 00 00 00 00 00 00 ...U..H.....
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Fig. 10. Bytes do cabeçalho IP(v4) sublinhado a vermelho; Bytes totais do datagrama sublinhado a laranja.

d. O datagrama IP foi fragmentado ? Justifique.

```

▶ Frame 2: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.74.135, 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: 72
    Identification: 0xa249 (41545)
    ▼ Flags: 0x0000
        0... .. = Reserved bit: Not set
        .0.. .. = Don't fragment: Not set
        ..0. .... = More fragments: Not set
        ...0 0000 0000 0000 = Fragment offset: 0
    ▶ Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x5552 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.74.135
    Destination: 193.136.9.240
▼ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x55b6 [correct]
    [Checksum status: Good]
0000 00 d0 03 ff 94 00 8c 85 90 05 ca 32 08 00 45 00 .....2..E.
0010 00 48 a2 49 00 00 01 01 55 52 ac 1a 4a 87 c1 88 .H.I....UR..J..
0020 09 f0 08 00 55 b6 a2 48 00 01 00 00 00 00 00 00 ...U..H.....
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Fig. 11. Parâmetro de fragmentação sublinhado a vermelho; Offset do fragmento sublinhado a laranja.

R: Como se pode observar na figura 7, não se verificam ações de fragmentação do datagrama. Caso existisse fragmentação seria possível de visualizar em ações consecutivas com datagramas fragmentados. Este facto pode ser confirmado mais especificamente na figura 11, pois verifica-se que o parâmetro **More fragments** sublinhado a vermelho não está estabelecido (Not set). Visto não existir fragmentação, também o parâmetro offset do fragmento está a 0.

2_01.pcapng
WireShark - Packet 2 - 2_01.pcapng
WireShark - Packet 5 - 2_01.pcapng
WireShark - Packet 7 - 2_01.pcapng
WireShark - Packet 9 - 2_01.pcapng

Apply a display filter... < %>
Expression...

No.	Time	Source	Destination	Protocol	Length	Info
22	1.6635919	172.26.115.252	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
32	2.040168	172.16.115.252	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
32	2.050332	172.16.115.252	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
0	0.013132	172.16.2.1	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
18	1.027730	172.16.2.1	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
20	1.029635	172.16.2.1	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
4	0.005759	172.26.254.254	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
6	0.009453	172.26.254.254	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
8	0.011244	172.26.254.254	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2	0.001508	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=1/256, ttl=1 (no response found!)
5	0.006510	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=2/512, ttl=1 (no response found!)
7	0.009582	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=3/768, ttl=1 (no response found!)
9	0.011373	193.136.9.240	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=4/1024, ttl=2 (no response found!)
11	0.013809	172.26.74.135	193.137.16.65	DNS	83	Standard query 0x2d5e PTR 1.2.16.172.in-addr.arpa
13	0.015687	172.26.74.135	193.137.16.145	DNS	83	Standard query 0x2d5e PTR 1.2.16.172.in-addr.arpa
15	0.017789	172.26.74.135	193.137.16.75	DNS	83	Standard query 0x2d5e PTR 1.2.16.172.in-addr.arpa
17	1.023996	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=5/1280, ttl=2 (no response found!)
19	1.027835	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=6/1536, ttl=2 (no response found!)
21	1.029752	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=7/1792, ttl=3 (no response found!)
23	1.036855	172.26.74.135	193.137.16.65	DNS	87	Standard query 0x5b27 PTR 252.115.16.172.in-addr.arpa
25	1.038882	172.26.74.135	193.137.16.145	DNS	87	Standard query 0x5b27 PTR 252.115.16.172.in-addr.arpa
27	1.041583	172.26.74.135	193.137.16.75	DNS	87	Standard query 0x5b27 PTR 252.115.16.172.in-addr.arpa
29	2.044633	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=8/2048, ttl=3 (no response found!)
31	2.046345	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=9/2304, ttl=3 (no response found!)
32	2.050409	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=10/2560, ttl=4 (reply in 34)
35	2.053182	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=11/2816, ttl=4 (reply in 36)
37	2.055288	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=12/3072, ttl=4 (reply in 38)
34	2.052322	193.136.9.240	172.26.74.135	ICMP	86	Echo (ping) reply id=0xa248, seq=10/2560, ttl=61 (request in 33)
36	2.055142	193.136.9.240	172.26.74.135	ICMP	86	Echo (ping) reply id=0xa248, seq=11/2816, ttl=61 (request in 35)
38	2.057826	193.136.9.240	172.26.74.135	ICMP	86	Echo (ping) reply id=0xa248, seq=12/3072, ttl=61 (request in 37)
14	0.017653	193.137.16.145	172.26.74.135	DNS	83	Standard query response 0x2d5e Refused PTR 1.2.16.172.in-addr.arpa
16	1.041286	193.137.16.65	172.26.74.135	DNS	87	Standard query response 0x5b27 Refused PTR 252.115.16.172.in-addr.arpa
12	0.015360	193.137.16.65	172.26.74.135	DNS	83	Standard query response 0x2d5e Refused PTR 1.2.16.172.in-addr.arpa
14	0.036664	193.137.16.65	172.26.74.135	DNS	87	Standard query response 0x5b27 Refused PTR 252.115.16.172.in-addr.arpa
16	1.019418	193.137.16.75	172.26.74.135	DNS	83	Standard query response 0x2d5e Refused PTR 1.2.16.172.in-addr.arpa
18	1.043417	193.137.16.75	172.26.74.135	DNS	87	Standard query response 0x5b27 Refused

```

▶ Frame 2: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▶ Internet Protocol Version 4, Src: 172.26.74.135, 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: 72
    Identification: 0xa249 (41545)
▶ Flags: 0x0000
▶ Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x5552 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.74.135
    Destination: 193.136.9.240

```

Fig. 13. Frame 2.


```

▶ Frame 5: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.74.135, 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: 72
    Identification: 0xa24a (41546)
    ▶ Flags: 0x0000
    ▶ Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0x5551 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.74.135
    Destination: 193.136.9.240

```

Fig. 14. Frame 5.

```

▶ Frame 9: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.74.135, 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: 72
    Identification: 0xa24c (41548)
    ▶ Flags: 0x0000
    ▶ Time to live: 2
    Protocol: ICMP (1)
    Header checksum: 0x544f [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.74.135
    Destination: 193.136.9.240

```

Fig. 15. Frame 9.

R: Tendo em vista os sublinhados das figuras 13, 14 e 15, é possível concluir que os campos do cabeçalho IP que variam de pacote para pacote são o campo **Identification**, o campo **Header Checksum** e o campo **Time to live** na secção do IP.

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

O campo de identificação do datagrama IP aumenta uma unidade por cada datagrama enviado pelo nosso computador. Relativamente ao campo TTL, como usamos o comando traceroute padrão, ele envia 3 datagramas com o mesmo TTL pelo que o TTL aumenta uma unidade em cada 3 datagramas enviados pelo nosso computador. Por exemplo, tendo em conta os frames 2,5,7 e 9, é possível verificar nas imagens 13, 14 e 15, que o 2 e 5 têm o mesmo TTL, já o 9 tem um TTL maior uma unidade.

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 cons tante para todas as mensagens de resposta ICMP TTL exceeded enviados ao seu host ? Porquê ?*

No.	Time	Source	Destination	Protocol	Length	Info
4	0.005759	172.26.254.254	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
6	0.008453	172.26.254.254	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
8	0.011244	172.26.254.254	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
10	0.013132	172.16.2.1	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
12	0.015360	193.137.16.65	172.26.74.135	DNS	83	Standard query response 0x2d5e Refused PTR 1.2.16.172.in-addr.arpa
14	0.017653	193.137.16.145	172.26.74.135	DNS	83	Standard query response 0x2d5e Refused PTR 1.2.16.172.in-addr.arpa
16	0.019418	193.137.16.75	172.26.74.135	DNS	83	Standard query response 0x2d5e Refused PTR 1.2.16.172.in-addr.arpa
18	1.027730	172.16.2.1	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
20	1.029635	172.16.2.1	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
22	1.035919	172.16.115.252	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
24	1.038664	193.137.16.65	172.26.74.135	DNS	87	Standard query response 0x5b27 Refused PTR 252.115.16.172.in-addr.arpa
26	1.041286	193.137.16.145	172.26.74.135	DNS	87	Standard query response 0x5b27 Refused PTR 252.115.16.172.in-addr.arpa
28	1.043417	193.137.16.75	172.26.74.135	DNS	87	Standard query response 0x5b27 Refused PTR 252.115.16.172.in-addr.arpa
30	2.048168	172.16.115.252	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
32	2.050322	172.16.115.252	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
34	2.052322	193.136.9.240	172.26.74.135	ICMP	86	Echo (ping) reply id=0xa248, seq=10/2560, ttl=61 (request in 33)
36	2.055142	193.136.9.240	172.26.74.135	ICMP	86	Echo (ping) reply id=0xa248, seq=11/2816, ttl=61 (request in 35)
38	2.057282	193.136.9.240	172.26.74.135	ICMP	86	Echo (ping) reply id=0xa248, seq=12/3072, ttl=61 (request in 37)
2	0.001508	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=1/256, ttl=1 (no response found!)
5	0.006510	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=2/512, ttl=1 (no response found!)
7	0.009582	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=3/768, ttl=1 (no response found!)
9	0.011373	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=4/1024, ttl=2 (no response found!)
17	1.023996	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=5/1280, ttl=2 (no response found!)
19	1.027835	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=6/1536, ttl=2 (no response found!)
21	1.029752	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=7/1792, ttl=3 (no response found!)
29	2.044633	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=8/2048, ttl=3 (no response found!)
31	2.048345	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=9/2304, ttl=3 (no response found!)
33	2.050469	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=10/2560, ttl=4 (reply in 34)
35	2.053182	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=11/2816, ttl=4 (reply in 36)
37	2.055288	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=12/3072, ttl=4 (reply in 38)
13	0.015687	172.26.74.135	193.137.16.145	DNS	83	Standard query 0x2d5e PTR 1.2.16.172.in-addr.arpa
25	1.038882	172.26.74.135	193.137.16.145	DNS	87	Standard query 0x5b27 PTR 252.115.16.172.in-addr.arpa
11	0.013009	172.26.74.135	193.137.16.65	DNS	83	Standard query 0x2d5e PTR 1.2.16.172.in-addr.arpa
23	1.036855	172.26.74.135	193.137.16.65	DNS	87	Standard query 0x5b27 PTR 252.115.16.172.in-addr.arpa
15	0.017789	172.26.74.135	193.137.16.75	DNS	83	Standard query 0x2d5e PTR 1.2.16.172.in-addr.arpa
27	1.041583	172.26.74.135	193.137.16.75	DNS	87	Standard query 0x5b27 PTR 252.115.16.172.in-addr.arpa
3	0.004971	ComdaEnt_ff:94:00	Apple_05:ca:32	ARP	60	172.26.254.254 is at 00:d0:03:ff:94:00
1	0.000000	Apple_05:ca:32	ComdaEnt_ff:94:00	ARP	42	Who has 172.26.254.254? Tell 172.26.74.135

▶ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
 ▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
 ▶ Address Resolution Protocol (request)

2 (i) pcapng Packets: 38 - Displayed: 38 (100.0%) Profile: Default

Fig. 16. Visão geral do tráfego gerado pelo traceroute ordenado por endereço destino.

```

▶ Frame 4: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▶ Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: Apple_05:ca:32 (8c:85:90:05:ca:32)
▼ Internet Protocol Version 4, Src: 172.26.254.254, Dst: 172.26.74.135
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    ▶ Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x1815 (6165)
    ▶ Flags: 0x0000
    Time to live: 255
    Protocol: ICMP (1)
    Header checksum: 0x0135 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.254.254
    Destination: 172.26.74.135
▶ Internet Control Message Protocol
  
```

Fig. 17. Frame 4.

```

▶ Frame 10: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▶ Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: Apple_05:ca:32 (8c:85:90:05:ca:32)
▼ Internet Protocol Version 4, Src: 172.16.2.1, Dst: 172.26.74.135
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 56
      Identification: 0x18f2 (6386)
    ▶ Flags: 0x0000
      Time to live: 254
      Protocol: ICMP (1)
      Header checksum: 0xff1f [validation disabled]
      [Header checksum status: Unverified]
      Source: 172.16.2.1
      Destination: 172.26.74.135
▶ Internet Control Message Protocol

```

Fig. 18. Frame 10.

```

▶ Frame 22: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▶ Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: Apple_05:ca:32 (8c:85:90:05:ca:32)
▼ Internet Protocol Version 4, Src: 172.16.115.252, Dst: 172.26.74.135
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 56
      Identification: 0x04af (1199)
    ▶ Flags: 0x0000
      Time to live: 253
      Protocol: ICMP (1)
      Header checksum: 0xa267 [validation disabled]
      [Header checksum status: Unverified]
      Source: 172.16.115.252
      Destination: 172.26.74.135
▶ Internet Control Message Protocol

```

Fig. 19. Frame 22.

R: O primeiro valor do campo TTL é 255, e mantém-se enquanto a fonte for a mesma. Quando a fonte variar este campo é decrementado uma unidade. Isto deve-se ao facto de que cada router tem que assegurar que a mensagem ICMP chega ao seu destino, daí ter um valor tão alto para o TTL. O facto de ser decrementado deve-se ao normal funcionamento de trânsito de datagramas entre routers, cada router diferente decrementa uma unidade ao TTL até este chegar ao destino. Como podemos ver nas imagens 17, 18 e 19, o TTL diminui consoante a Source.

1.3 Exercício 3

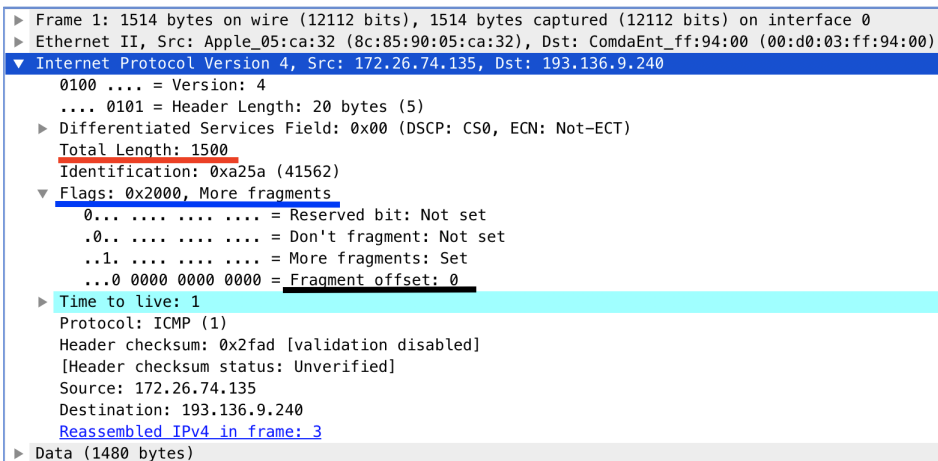
Pretende-se agora analisar a fragmentação de pacotes IP. Reponha a ordem do tráfego capturado usando a coluna do tempo de captura. Observe o tráfego depois do tamanho de pacote ter sido definido para 35XX bytes.

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

R: A quantidade máxima de dados que um frame da camada Física consegue transportar chama-se **MTU** (maximum transmission unit). Como os frames Ethernet só conseguem

carregar até 1500 bytes de dados, e como nós queríamos capturar tráfego para pacotes com 3547 bytes, então ter-se-ia que se dividir o pacote inicial em 3 fragmentos.

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?*



```
▶ Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.74.135, 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: 0xa25a (41562)
      ▼ 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: 0x2fad [validation disabled]
      [Header checksum status: Unverified]
      Source: 172.26.74.135
      Destination: 193.136.9.240
      Reassembled IPv4 in frame: 3
    ▶ Data (1480 bytes)
```

Fig. 20. Primeiro segmento do datagrama fragmentado.

R: Como se pode observar na figura 20, sublinhado a azul escuro, existe uma flag no cabeçalho que indica a existência de mais fragmentos, se existem mais é porque o que está a ser analisado é um fragmento.

Trata-se do primeiro fragmento uma vez que o campo sublinhado a preto **Fragment offset**, indica o offset do datagrama e este está a 0. Segmentos consecutivos terão o campo do offset com valores maiores.

Como se pode observar na figura, sublinhado a vermelho, encontra-se o tamanho do datagrama que é **1500 bytes**.

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?*

R: Como se pode observar na figura 22, no quadrado com contorno a vermelho, foram criados 3 fragmentos, aliás como se pode constatar no campo **Fragments count**, mas esta é informação que é fornecida pelo Wireshark e não está presente no cabeçalho IP.

O último fragmento do datagrama original pode ser identificado quando o campo **More fragments** não está assinalado (Not set), e quando o campo **Fragment offset** é diferente de 0. Observando a figura, assinalado a preto temos esses dois campos, que estão de acordo com o descrito. O campo **More fragments** não está assinalado e o campo **Fragment offset** é 370, que é logicamente diferente de 0.

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.

```
▶ Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.74.135, 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: 0xa25a (41562)
  ▼ 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: 0x2fad [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.74.135
    Destination: 193.136.9.240
    Reassembled IPv4 in frame: 3
▶ Data (1480 bytes)
```

Fig. 23. Primeiro segmento do datagrama fragmentado.

```
▶ Frame 2: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.74.135, 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: 0xa25a (41562)
  ▼ 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: 0x2ef4 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.74.135
    Destination: 193.136.9.240
    Reassembled IPv4 in frame: 3
▶ Data (1480 bytes)
```

Fig. 24. Segundo segmento do datagrama fragmentado.

```

▶ Frame 3: 601 bytes on wire (4808 bits), 601 bytes captured (4808 bits) on interface 0
▶ Ethernet II, Src: Apple_05:ca:32 (8c:85:90:05:ca:32), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.74.135, 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: 587
    Identification: 0xa25a (41562)
▼ 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: 0x51cc [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.74.135
    Destination: 193.136.9.240
▼ [3 IPv4 Fragments (3527 bytes): #1(1480), #2(1480), #3(567)]
    [Frame: 1, payload: 0-1479 (1480 bytes)]
    [Frame: 2, payload: 1480-2959 (1480 bytes)]
    [Frame: 3, payload: 2960-3526 (567 bytes)]
    [Fragment count: 3]
    [Reassembled IPv4 length: 3527]
    [Reassembled IPv4 data: 080055a5a259000100000000000000000000000000000000...]
▶ Internet Control Message Protocol

```

Fig. 25. Terceiro segmento do datagrama fragmentado.

R: Uma vez que existem 3 fragmentos relativos ao datagrama original podemos comparar os primeiros dois devido às suas semelhanças e por fim comparar as principais diferenças de um dos dois primeiros fragmentos para com o último fragmento.

Comparando os dois primeiros fragmentos, figura 23 e 24, podemos notar que as principais diferenças estão nos campos **Fragment offset** e no **Header checksum**, que se encontram sublinhados em cada figura a preto e vermelho, respetivamente.

Comparando o segundo fragmento com o último, figura 24 e 25, as principais diferenças estão nos campos **Total Length**, **More fragments**, **Fragment offset** e no **Header checksum**, que se encontram sublinhados em cada figura a azul, roxo, preto e vermelho, respetivamente.

O campo **Identification** é igual em todos os fragmentos pelo que no destino é possível saber quais dos fragmentos vão originar um datagrama. O campo **More fragments** dita se ainda existem mais fragmentos, sabe-se então que se recebe o último fragmento quando este campo não está assinalado (Not set). Este último fragmento permite observar, como se pode constatar na figura 25, no campo **Fragment count** sublinhado a verde, em quantos fragmentos foi dividido o datagrama original, pelo que se consegue saber se já foram recebidos todos os fragmentos ou não. Por último, depois de todos os fragmentos chegarem ao destino, para reconstruir o datagrama original por ordem usa-se o campo **Fragment offset**, que dita em que posição cada fragmento se encontra no datagrama original.

2 Endereçamento e Encaminhamento IP

Considere que a organização MIEI-RC é constituída por três departamentos (A, B, e C) e cada departamento possui um router de acesso à sua rede local. Estes routers de acesso (Ra, Rb, e Rc) estão interligados entre si por ligações Ethernet a 1Gbps, formando um anel. Por sua vez, existe um servidor (S1) na rede do departamento C e, pelo menos, três laptops por departamento, interligados ao router respetivo através de um comutador (switch). S1 tem uma ligação a 1Gbps e os laptops ligações a 100Mbps. Considere apenas a existência de um comutador por departamento.

A conectividade IP externa da organização é assegurada através de um router de acesso Rext conectado a Rc por uma ligação ponto-a-ponto a 10 Gbps.

Construa uma topologia CORE que reflita a rede local da empresa. Para facilitar a visualização pode ocultar o endereçamento IPv6.

2.1 Exercício 1

Atenda aos endereços IP atribuídos automaticamente pelo CORE aos diversos equipamentos da topologia.

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.

R: Na seguinte figura encontra-se a topologia de rede definida bem como todos os endereços IP atribuídos a cada equipamento da rede.

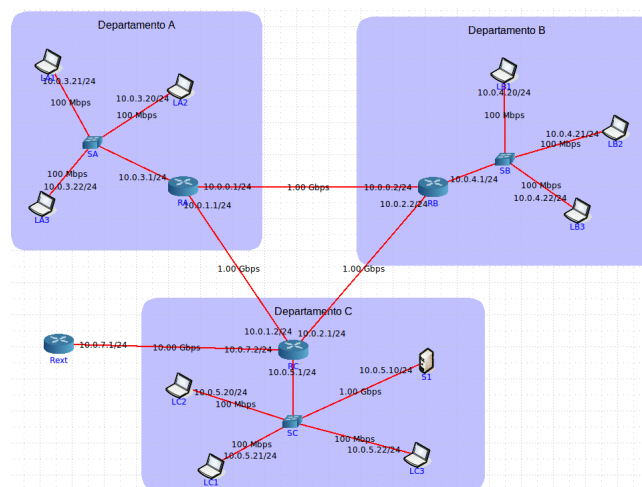


Fig. 26. Topologia definida.

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

R: Segundo o **RFC1918**, a IANA (Internet Assigned Numbers Authority) reservou o espaço de endereçamento 10.0.0.0 - 10.255.255.255, para internets privadas. Como se pode observar na figura 26, todos os equipamentos estão endereçados dentro deste espaço, conclui-se então que os endereços são **privados**.

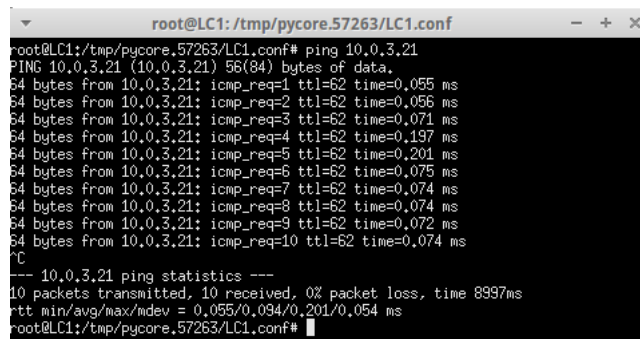
c) *Porque razão não é atribuído um endereço IP aos switches?*

R: Um endereço IP é um requisito para a camada de rede (nível 3), os switches trabalham estritamente na camada link (nível 2) e, portanto, não precisam de um endereço IP.

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

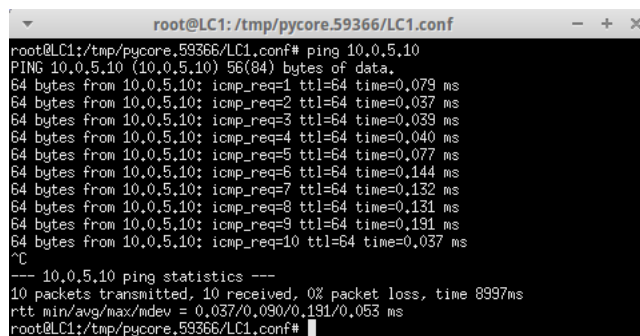
R: Usando o comando ping testamos a existência de conectividade entre departamentos, e dos departamentos com o servidor S1.

Existe conectividade entre o Departamento C e o Departamento A, podemos verificar isso na figura 27 onde conectamos o os laptops LC1 e LA1 (10.0.3.21). O departamento C também tem conectividade com o servidor S1, como se pode verificar na figura 28.



```
root@LC1: /tmp/pycore.57263/LC1.conf# ping 10.0.3.21
PING 10.0.3.21 (10.0.3.21) 56(84) bytes of data:
64 bytes from 10.0.3.21: icmp_req=1 ttl=62 time=0.055 ms
64 bytes from 10.0.3.21: icmp_req=2 ttl=62 time=0.056 ms
64 bytes from 10.0.3.21: icmp_req=3 ttl=62 time=0.071 ms
64 bytes from 10.0.3.21: icmp_req=4 ttl=62 time=0.197 ms
64 bytes from 10.0.3.21: icmp_req=5 ttl=62 time=0.201 ms
64 bytes from 10.0.3.21: icmp_req=6 ttl=62 time=0.075 ms
64 bytes from 10.0.3.21: icmp_req=7 ttl=62 time=0.074 ms
64 bytes from 10.0.3.21: icmp_req=8 ttl=62 time=0.074 ms
64 bytes from 10.0.3.21: icmp_req=9 ttl=62 time=0.072 ms
64 bytes from 10.0.3.21: icmp_req=10 ttl=62 time=0.074 ms
^C
--- 10.0.3.21 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8997ms
rtt min/avg/max/mdev = 0.055/0.094/0.201/0.054 ms
root@LC1: /tmp/pycore.57263/LC1.conf#
```

Fig. 27. Conexão entre departamento C e departamento A.



```
root@LC1: /tmp/pycore.59366/LC1.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_req=1 ttl=64 time=0.079 ms
64 bytes from 10.0.5.10: icmp_req=2 ttl=64 time=0.037 ms
64 bytes from 10.0.5.10: icmp_req=3 ttl=64 time=0.039 ms
64 bytes from 10.0.5.10: icmp_req=4 ttl=64 time=0.040 ms
64 bytes from 10.0.5.10: icmp_req=5 ttl=64 time=0.077 ms
64 bytes from 10.0.5.10: icmp_req=6 ttl=64 time=0.144 ms
64 bytes from 10.0.5.10: icmp_req=7 ttl=64 time=0.132 ms
64 bytes from 10.0.5.10: icmp_req=8 ttl=64 time=0.131 ms
64 bytes from 10.0.5.10: icmp_req=9 ttl=64 time=0.191 ms
64 bytes from 10.0.5.10: icmp_req=10 ttl=64 time=0.037 ms
^C
--- 10.0.5.10 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8997ms
rtt min/avg/max/mdev = 0.037/0.090/0.191/0.053 ms
root@LC1: /tmp/pycore.59366/LC1.conf#
```

Fig. 28. Conexão entre departamento C e o servidor S1.

Existe conectividade entre o Departamento C e o Departamento B, podemos verificar isso na figura 29 onde conectamos o os laptops LC1 e LB1 (10.0.4.20).

```
root@LC1:/tmp/pycore.57263/LC1.conf# ping 10.0.4,20
PING 10.0.4,20 (10.0.4,20) 56(84) bytes of data.
64 bytes from 10.0.4,20: icmp_req=1 ttl=62 time=0,049 ms
64 bytes from 10.0.4,20: icmp_req=2 ttl=62 time=0,068 ms
64 bytes from 10.0.4,20: icmp_req=3 ttl=62 time=0,272 ms
64 bytes from 10.0.4,20: icmp_req=4 ttl=62 time=0,069 ms
64 bytes from 10.0.4,20: icmp_req=5 ttl=62 time=0,063 ms
64 bytes from 10.0.4,20: icmp_req=6 ttl=62 time=0,062 ms
64 bytes from 10.0.4,20: icmp_req=7 ttl=62 time=0,050 ms
64 bytes from 10.0.4,20: icmp_req=8 ttl=62 time=0,065 ms
64 bytes from 10.0.4,20: icmp_req=9 ttl=62 time=0,078 ms
64 bytes from 10.0.4,20: icmp_req=10 ttl=62 time=0,061 ms
^C
--- 10.0.4,20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9001ms
rtt min/avg/max/ndev = 0,049/0,083/0,272/0,064 ms
root@LC1:/tmp/pycore.57263/LC1.conf#
```

Fig. 29. Conexão entre departamento C e o departamento B.

Existe conectividade entre o Departamento A e o Departamento B, podemos verificar isso na figura 30 onde conectamos o os laptops LA1 e LB1 (10.0.4.20). O departamento A também tem conectividade com o servidor S1, como se pode verificar na figura 31.

```
root@LA1:/tmp/pycore.57263/LA1.conf# ping 10.0.4,20
PING 10.0.4,20 (10.0.4,20) 56(84) bytes of data.
64 bytes from 10.0.4,20: icmp_req=1 ttl=62 time=0,056 ms
64 bytes from 10.0.4,20: icmp_req=2 ttl=62 time=0,060 ms
64 bytes from 10.0.4,20: icmp_req=3 ttl=62 time=0,061 ms
64 bytes from 10.0.4,20: icmp_req=4 ttl=62 time=0,061 ms
64 bytes from 10.0.4,20: icmp_req=5 ttl=62 time=0,063 ms
64 bytes from 10.0.4,20: icmp_req=6 ttl=62 time=0,234 ms
64 bytes from 10.0.4,20: icmp_req=7 ttl=62 time=0,060 ms
64 bytes from 10.0.4,20: icmp_req=8 ttl=62 time=0,075 ms
64 bytes from 10.0.4,20: icmp_req=9 ttl=62 time=0,060 ms
64 bytes from 10.0.4,20: icmp_req=10 ttl=62 time=0,058 ms
^C
--- 10.0.4,20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8999ms
rtt min/avg/max/ndev = 0,056/0,078/0,234/0,053 ms
root@LA1:/tmp/pycore.57263/LA1.conf#
```

Fig. 30. Conexão entre departamento A e o departamento B.

```
root@LA1:/tmp/pycore.59366/LA1.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_req=1 ttl=62 time=0,112 ms
64 bytes from 10.0.5,10: icmp_req=2 ttl=62 time=0,075 ms
64 bytes from 10.0.5,10: icmp_req=3 ttl=62 time=0,139 ms
64 bytes from 10.0.5,10: icmp_req=4 ttl=62 time=0,062 ms
64 bytes from 10.0.5,10: icmp_req=5 ttl=62 time=0,233 ms
64 bytes from 10.0.5,10: icmp_req=6 ttl=62 time=0,071 ms
64 bytes from 10.0.5,10: icmp_req=7 ttl=62 time=0,074 ms
64 bytes from 10.0.5,10: icmp_req=8 ttl=62 time=0,060 ms
64 bytes from 10.0.5,10: icmp_req=9 ttl=62 time=0,075 ms
64 bytes from 10.0.5,10: icmp_req=10 ttl=62 time=0,076 ms
^C
--- 10.0.5,10 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9001ms
rtt min/avg/max/ndev = 0,060/0,097/0,233/0,062 ms
root@LA1:/tmp/pycore.59366/LA1.conf#
```

Fig. 31. Conexão entre departamento A e o servidor S1.

O departamento B tem conectividade com o servidor S1, como se pode verificar na figura 32.

```
root@LB1: /tmp/pycore.59366/LB1.conf
root@LB1:/tmp/pycore.59366/LB1.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_req=1 ttl=62 time=0.092 ms
64 bytes from 10.0.5.10: icmp_req=2 ttl=62 time=0.059 ms
64 bytes from 10.0.5.10: icmp_req=3 ttl=62 time=0.057 ms
64 bytes from 10.0.5.10: icmp_req=4 ttl=62 time=0.081 ms
64 bytes from 10.0.5.10: icmp_req=5 ttl=62 time=0.060 ms
64 bytes from 10.0.5.10: icmp_req=6 ttl=62 time=0.065 ms
64 bytes from 10.0.5.10: icmp_req=7 ttl=62 time=0.063 ms
64 bytes from 10.0.5.10: icmp_req=8 ttl=62 time=0.075 ms
64 bytes from 10.0.5.10: icmp_req=9 ttl=62 time=0.162 ms
64 bytes from 10.0.5.10: icmp_req=10 ttl=62 time=0.062 ms
^C
--- 10.0.5.10 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8998ms
rtt min/avg/max/mdev = 0.057/0.077/0.162/0.031 ms
root@LB1:/tmp/pycore.59366/LB1.conf#
```

Fig. 32. Conexão entre departamento B e o servidor S1.

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

R: Existe conectividade entre o servidor S1 e o router externo Rext, como se pode ver na figura 33, onde se testa a conectividade do router de acesso para o servidor.

```
root@Rext: /tmp/pycore.59366/Rext.conf
root@Rext:/tmp/pycore.59366/Rext.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_req=1 ttl=63 time=0.111 ms
64 bytes from 10.0.5.10: icmp_req=2 ttl=63 time=0.052 ms
64 bytes from 10.0.5.10: icmp_req=3 ttl=63 time=0.098 ms
64 bytes from 10.0.5.10: icmp_req=4 ttl=63 time=0.052 ms
64 bytes from 10.0.5.10: icmp_req=5 ttl=63 time=0.228 ms
64 bytes from 10.0.5.10: icmp_req=6 ttl=63 time=0.036 ms
64 bytes from 10.0.5.10: icmp_req=7 ttl=63 time=0.061 ms
64 bytes from 10.0.5.10: icmp_req=8 ttl=63 time=0.065 ms
64 bytes from 10.0.5.10: icmp_req=9 ttl=63 time=0.055 ms
64 bytes from 10.0.5.10: icmp_req=10 ttl=63 time=0.065 ms
^C
--- 10.0.5.10 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9002ms
rtt min/avg/max/mdev = 0.036/0.082/0.228/0.053 ms
root@Rext:/tmp/pycore.59366/Rext.conf#
```

Fig. 33. Conexão entre router de acesso Rext e o servidor S1.

2.2 Exercício 2

Para o router e um laptop do departamento A:

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`).

R: Pode-se ver na figura 34 a tabela de encaminhamento do laptop LA1, e na figura 35 a tabela de encaminhamento do router RA.

```
root@LA1:/tmp/pycore.57263/LA1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.3.1 0.0.0.0 UG 0 0 0 eth0
10.0.3.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@LA1:/tmp/pycore.57263/LA1.conf#
```

Fig. 34. Tabela de encaminhamento do laptop LA1.

```
root@RA:/tmp/pycore.59384/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 0.0.0.0 255.255.255.0 U 0 0 0 eth1
10.0.2.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
10.0.3.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
10.0.4.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
10.0.5.0 10.0.1.2 255.255.255.0 UG 0 0 0 eth1
10.0.7.0 10.0.1.2 255.255.255.0 UG 0 0 0 eth1
root@RA:/tmp/pycore.59384/RA.conf#
```

Fig. 35. Tabela de encaminhamento do router RA.

Na tabela de encaminhamento do laptop LA1, os únicos endereços de saída são o respectivo router do departamento por onde saem todos os packets destinados a outros departamentos ou outras redes, e no caso de querer comunicar com laptops no próprio departamento, os datagramas serão entregues na interface de endereço próprio que corresponde ao gateway 0.0.0.0, sendo isto feito com auxílio do Ethernet Switch.

Na tabela de encaminhamento do router do departamento A, dependendo para onde se quer enviar os datagramas usamos endereços de saída diferentes. No caso de se querer enviar datagramas para o departamento B (10.0.4.0) usa-se interface de endereço 10.0.0.2. No caso de se querer enviar datagramas para o departamento C (10.0.5.0) usa-se interface de endereço 10.0.1.2. No caso de se querer comunicar com as redes adjacentes usa-se a interface de endereço default (0.0.0.0). Por último, no caso de se querer comunicar com redes exteriores à topologia criada para os departamentos (10.0.7.0) usa-se a interface de endereço 10.0.1.2.

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

R: Este sistema possui encaminhamento dinâmico e estático. Na topologia de rede em anel (Router A - Router B - Router C) o encaminhamento é dinâmico. Desta forma, as rotas são atualizadas ao longo do tempo e, caso exista alguma falha num link entre routers, existe adaptação a um novo caminho.

Em cada departamento, o encaminhamento entre os equipamentos é realizado de forma estática. Este tipo de encaminhamento é baseado em rotas pré-definidas, visto estarmos perante topologias de rede em que existe maior conhecimento entre os intervenientes. O mesmo acontece na ligação entre o Router C e o Router RExt.

```
root@LA1: /tmp/pycore.49430/LA1.conf
root@LA1:/tmp/pycore.49430/LA1.conf# ps -ef
UID      PID  PPID  C  STIME TTY          TIME CMD
root      1    0  0  12:10 ?        00:00:00 /usr/sbin/vnoded -v -c /tmp/pyco
root     131    1  1  12:25 pts/5    00:00:00 /bin/bash
root     185   131  0  12:25 pts/5    00:00:00 ps -ef
root@LA1:/tmp/pycore.49430/LA1.conf#
```

Fig. 36. Processos a decorrer no laptop LA1.

```
root@RA: /tmp/pycore.49430/RA.conf
root@RA:/tmp/pycore.49430/RA.conf# ps -ef
UID      PID  PPID  C  STIME TTY          TIME CMD
root      1    0  0  12:10 ?        00:00:00 /usr/sbin/vnoded -v -c /tmp/pyco
root      41    1  0  12:10 ?        00:00:00 /usr/lib/quagga/zebra -u root -g
root      47    1  0  12:10 ?        00:00:00 /usr/lib/quagga/ospfd -u root -g
root      52    1  0  12:10 ?        00:00:00 /usr/lib/quagga/ospf6d -u root -
root     128    1  3  12:25 pts/5    00:00:00 /bin/bash
root     182   128  0  12:25 pts/5    00:00:00 ps -ef
root@RA:/tmp/pycore.49430/RA.conf#
```

Fig. 37. Processos a decorrer no router RA.

Podemos verificar na imagem 36 que apenas temos 3 processos a decorrer na máquina: o processo-pai, a bash e o comando ps.

Podemos verificar na imagem 37 que temos 6 processos a decorrer na máquina: o processo-pai, a bash, o comando ps e 3 deamons, 2 dos quais são protocolos de roteamento para IPv4 e IPv6.

Desta forma, concluímos que nos routers existe encaminhamento dinâmico, e nos laptops encaminhamento estático.

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

R: Pode-se ver na figura 38 a aplicação do comando `route delete` no servidor S1, assim como a tabela de encaminhamento antes e depois do sucedido, que comprova o sucesso do comando realizado.

```
root@S1: /tmp/pycore.59384/S1.conf
root@S1:/tmp/pycore.59384/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.5.1 0.0.0.0 UG 0 0 0 eth0
10.0.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1:/tmp/pycore.59384/S1.conf# route del -net 0.0.0.0 gw 10.0.5.1 netmask 0.0.0.0 dev eth0
root@S1:/tmp/pycore.59384/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1:/tmp/pycore.59384/S1.conf#
```

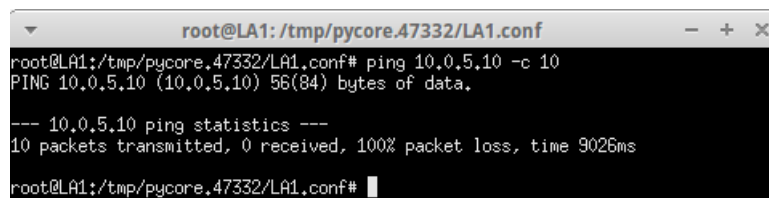
Fig. 38. Utilização do comando `route delete`.

Ao retirar a rota por defeito da tabela de encaminhamento do servidor S1, os utilizadores da empresa que por norma acedem ao servidor deixam de poder fazer tal acesso, pois foi retirado o endereço de saída 10.0.5.1 com que o servidor se ligava ao switch que por sua vez se ligava ao Router C.

Sempre que os utilizadores tentam comunicar com o servidor S1, enviam os seus packets mas não recebem resposta do servidor uma vez que este já não tem como comunicar com o exterior do departamento C, logo a rota não é definida e a comunicação não é estabelecida.

Um exemplo disso é o teste de conexão do laptop LA1 ao servidor S1, como se pode ver na figura 39.

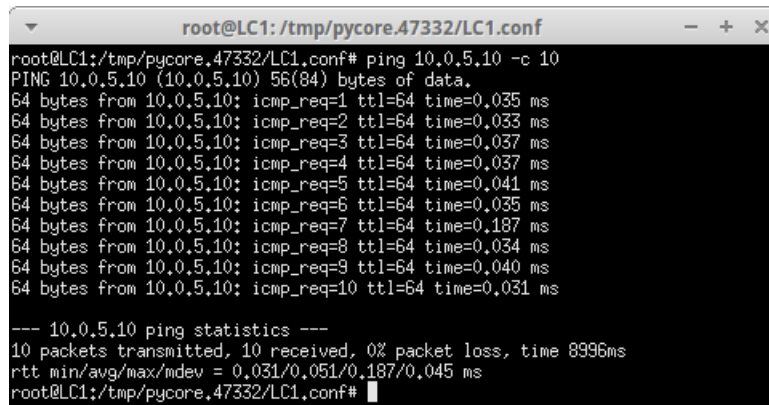
O servidor apenas consegue conectar com laptops dentro do departamento C uma vez que estes tem o endereço que combina com 10.0.5.0, essa conexão pode ser vista na figura 40.



```
root@LA1: /tmp/pycore.47332/LA1.conf
root@LA1:/tmp/pycore.47332/LA1.conf# ping 10.0.5.10 -c 10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data:

--- 10.0.5.10 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9026ms
root@LA1:/tmp/pycore.47332/LA1.conf#
```

Fig. 39. Teste de conexão entre LA1 e o S1.



```
root@LC1: /tmp/pycore.47332/LC1.conf
root@LC1:/tmp/pycore.47332/LC1.conf# ping 10.0.5.10 -c 10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data:
64 bytes from 10.0.5.10: icmp_req=1 ttl=64 time=0.035 ms
64 bytes from 10.0.5.10: icmp_req=2 ttl=64 time=0.033 ms
64 bytes from 10.0.5.10: icmp_req=3 ttl=64 time=0.037 ms
64 bytes from 10.0.5.10: icmp_req=4 ttl=64 time=0.037 ms
64 bytes from 10.0.5.10: icmp_req=5 ttl=64 time=0.041 ms
64 bytes from 10.0.5.10: icmp_req=6 ttl=64 time=0.035 ms
64 bytes from 10.0.5.10: icmp_req=7 ttl=64 time=0.187 ms
64 bytes from 10.0.5.10: icmp_req=8 ttl=64 time=0.034 ms
64 bytes from 10.0.5.10: icmp_req=9 ttl=64 time=0.040 ms
64 bytes from 10.0.5.10: icmp_req=10 ttl=64 time=0.031 ms

--- 10.0.5.10 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8996ms
rtt min/avg/max/mdev = 0.031/0.051/0.187/0.045 ms
root@LC1:/tmp/pycore.47332/LC1.conf#
```

Fig. 40. Teste de conexão entre LC1 e o S1.

d) Adicione as rotas estáticas necessárias para restaurar a conectividade para o servidor S1, por forma a contornar a restrição imposta na alínea c). Utilize para o efeito o comando `route add` e registe os comandos que usou.

R: Pode-se ver na figura 41 que a aplicação do comando Route Add no servidor S1, assim como a tabela de encaminhamento antes e depois do sucedido, que comprova o sucesso do comando realizado.

```
root@S1: /tmp/pycore.59384/S1.conf
root@S1:/tmp/pycore.59384/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1:/tmp/pycore.59384/S1.conf# route add default gw 10.0.5.1 eth0
root@S1:/tmp/pycore.59384/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.5.1 0.0.0.0 UG 0 0 0 eth0
10.0.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1:/tmp/pycore.59384/S1.conf#
```

Fig. 41. Utilização do comando route add.

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.*

R: Pode se ver na figura 42 que a nova política de encaminhamento funciona corretamente e que o servidor está novamente acessível, através do teste de conectividade com o laptop de outro departamento, neste caso o laptop 1 do departamento B. Pode-se ver na figura 43 a nova tabela de encaminhamento do servidor S1.

```
root@S1: /tmp/pycore.59384/S1.conf
root@S1:/tmp/pycore.59384/S1.conf# ping 10.0.4.20
PING 10.0.4.20 (10.0.4.20) 56(84) bytes of data:
64 bytes from 10.0.4.20: icmp_req=1 ttl=62 time=0.203 ms
64 bytes from 10.0.4.20: icmp_req=2 ttl=62 time=0.088 ms
64 bytes from 10.0.4.20: icmp_req=3 ttl=62 time=0.076 ms
64 bytes from 10.0.4.20: icmp_req=4 ttl=62 time=0.078 ms
64 bytes from 10.0.4.20: icmp_req=5 ttl=62 time=0.072 ms
64 bytes from 10.0.4.20: icmp_req=6 ttl=62 time=0.077 ms
64 bytes from 10.0.4.20: icmp_req=7 ttl=62 time=0.076 ms
64 bytes from 10.0.4.20: icmp_req=8 ttl=62 time=0.073 ms
64 bytes from 10.0.4.20: icmp_req=9 ttl=62 time=0.171 ms
64 bytes from 10.0.4.20: icmp_req=10 ttl=62 time=0.130 ms
^C
--- 10.0.4.20 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 899ms
rtt min/avg/max/mdev = 0.072/0.104/0.203/0.045 ms
root@S1:/tmp/pycore.59384/S1.conf#
```

Fig. 42. Conexão entre o laptop LB1 e o servidor S1.

```
root@S1: /tmp/pycore.59384/S1.conf
root@S1:/tmp/pycore.59384/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.5.1 0.0.0.0 UG 0 0 0 eth0
10.0.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1:/tmp/pycore.59384/S1.conf#
```

Fig. 43. Nova tabela de encaminhamento do servidor S1.

3 Definição de Sub-redes

Considere a topologia definida anteriormente. Assuma que o endereçamento entre os routers se mantém inalterado, contudo, o endereçamento em cada departamento deve ser redefinido.

3.1 1)

Considere que dispõe apenas do endereço de rede IP 172.XX.48.0/20, em que XX é o decimal correspondendo ao seu número de grupo (PLXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo a rede de acesso e core inalteradas) e atribua endereços às interfaces dos vários sistemas envolvidos. Deve justificar as opções usadas.

R: Pode se ver na figura 44 o novo de esquema de endereçamento para as redes dos departamentos usando apenas o endereço de rede 172.47.48.0/20.

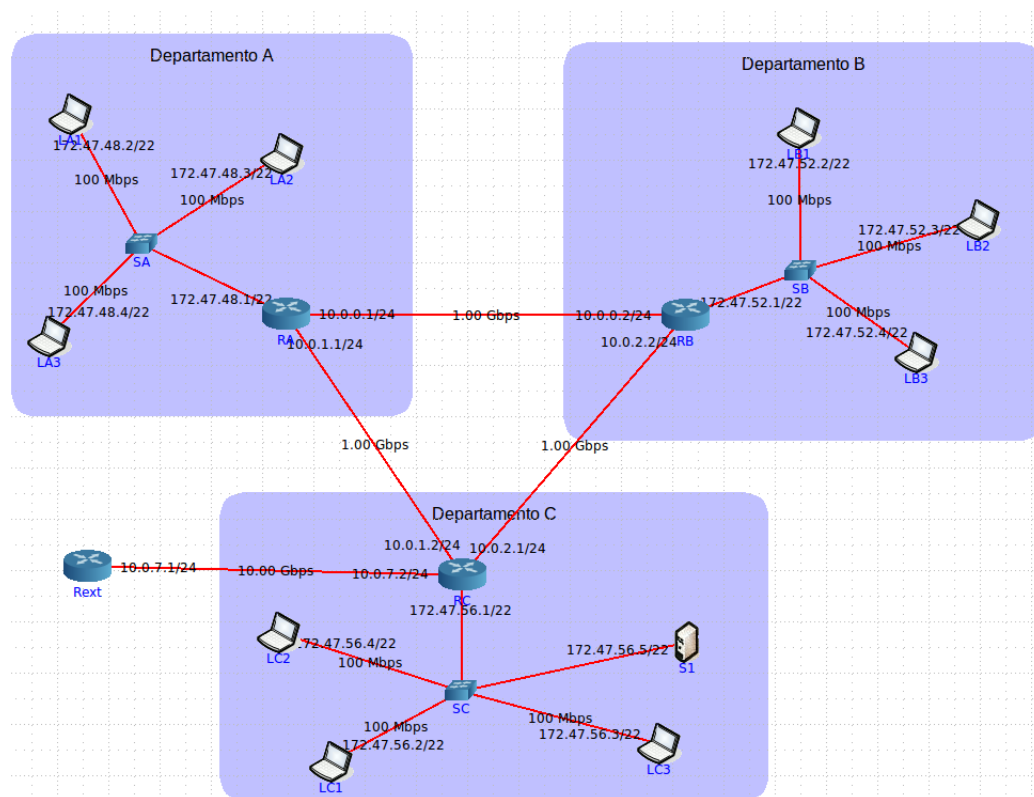


Fig. 44. Novo esquema de endereçamento.

O endereço 172.47.48.0/20 corresponde a 10101100.00101111.00111000.00000000. Os bits antes da barra representam a rede, enquanto os bits posteriores representam os hosts.

Como necessitávamos de definir 3 sub-redes basta apenas alocar 2 bits para tal, ficando assim a máscara de rede igual a /22. Desta forma conseguimos definir 4 novas sub-redes.

Para o departamento A, usamos a subrede 00, pelo que os hosts são endereçados entre o endereço 172.47.48.1/22 e o endereço 172.47.51.254/22.

Para o departamento B, usamos a subrede 01, pelo que os hosts são endereçados entre o endereço 172.47.52.1/22 e o endereço 172.47.55.254/22.

Para o departamento C, usamos a subrede 10, pelo que os hosts são endereçados entre o endereço 172.47.56.1/22 e o endereço 172.47.59.254/22.

3.2 2)

Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP pode interligar em cada departamento? Justifique.

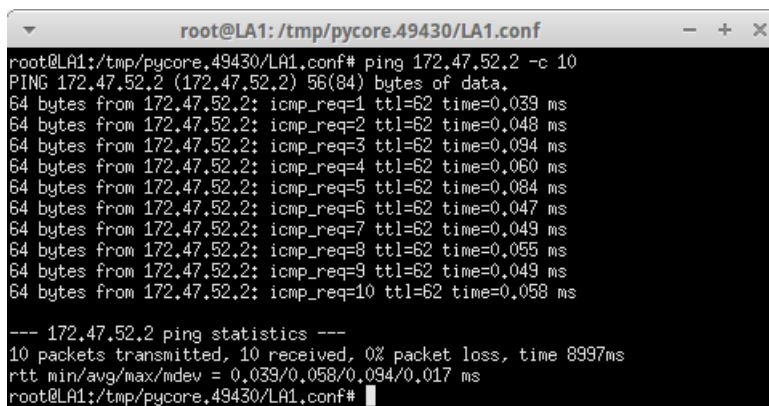
R: A máscara de rede usada foi /22. Como o endereço na totalidade possui 32 bits, e visto que os bits de identificação de rede são 20 e os de identificação de subrede são 2, ficam a sobrar 10 bits para endereçamento de hosts. Portanto, é possível em cada departamento endereçar $((2 \text{ elevado a } 10) - 2 =)$ 1022 hosts. Os 2 endereços retirados estão reservados para broadcast e identificador de subrede.

Usamos 2 bits para definição de subredes pois é o menor número possível para criação de 3 subredes.

3.3 3)

Garanta e verifique que conectividade IP entre as várias redes locais da organização MIEI-RC é mantida. Explique como procedeu.

R: Pode se verificar nas figuras 45, 46 e 47 que a conectividade entre os departamentos da organização é mantida.



```
root@LA1: /tmp/pycore.49430/LA1.conf
root@LA1:/tmp/pycore.49430/LA1.conf# ping 172.47.52.2 -c 10
PING 172.47.52.2 (172.47.52.2) 56(84) bytes of data:
64 bytes from 172.47.52.2: icmp_req=1 ttl=62 time=0.039 ms
64 bytes from 172.47.52.2: icmp_req=2 ttl=62 time=0.048 ms
64 bytes from 172.47.52.2: icmp_req=3 ttl=62 time=0.094 ms
64 bytes from 172.47.52.2: icmp_req=4 ttl=62 time=0.060 ms
64 bytes from 172.47.52.2: icmp_req=5 ttl=62 time=0.084 ms
64 bytes from 172.47.52.2: icmp_req=6 ttl=62 time=0.047 ms
64 bytes from 172.47.52.2: icmp_req=7 ttl=62 time=0.049 ms
64 bytes from 172.47.52.2: icmp_req=8 ttl=62 time=0.055 ms
64 bytes from 172.47.52.2: icmp_req=9 ttl=62 time=0.049 ms
64 bytes from 172.47.52.2: icmp_req=10 ttl=62 time=0.058 ms
--- 172.47.52.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 899ms
rtt min/avg/max/mdev = 0.039/0.058/0.094/0.017 ms
root@LA1:/tmp/pycore.49430/LA1.conf#
```

Fig. 45. Conectividade entre o equipamento LA1 e o LB1.

```
root@LB1: /tmp/pycore.49430/LB1.conf# ping 172.47.56.2 -c 10
PING 172.47.56.2 (172.47.56.2) 56(84) bytes of data.
64 bytes from 172.47.56.2: icmp_req=1 ttl=62 time=0.076 ms
64 bytes from 172.47.56.2: icmp_req=2 ttl=62 time=0.067 ms
64 bytes from 172.47.56.2: icmp_req=3 ttl=62 time=0.066 ms
64 bytes from 172.47.56.2: icmp_req=4 ttl=62 time=0.067 ms
64 bytes from 172.47.56.2: icmp_req=5 ttl=62 time=0.058 ms
64 bytes from 172.47.56.2: icmp_req=6 ttl=62 time=0.058 ms
64 bytes from 172.47.56.2: icmp_req=7 ttl=62 time=0.066 ms
64 bytes from 172.47.56.2: icmp_req=8 ttl=62 time=0.079 ms
64 bytes from 172.47.56.2: icmp_req=9 ttl=62 time=0.057 ms
64 bytes from 172.47.56.2: icmp_req=10 ttl=62 time=0.066 ms

--- 172.47.56.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9000ms
rtt min/avg/max/mdev = 0.057/0.066/0.079/0.007 ms
root@LB1: /tmp/pycore.49430/LB1.conf#
```

Fig. 46. Conectividade entre o equipamento LB1 e o LC1.

```
root@LC1: /tmp/pycore.49430/LC1.conf# ping 172.47.48.2 -c 10
PING 172.47.48.2 (172.47.48.2) 56(84) bytes of data.
64 bytes from 172.47.48.2: icmp_req=1 ttl=62 time=0.067 ms
64 bytes from 172.47.48.2: icmp_req=2 ttl=62 time=0.048 ms
64 bytes from 172.47.48.2: icmp_req=3 ttl=62 time=0.046 ms
64 bytes from 172.47.48.2: icmp_req=4 ttl=62 time=0.078 ms
64 bytes from 172.47.48.2: icmp_req=5 ttl=62 time=0.050 ms
64 bytes from 172.47.48.2: icmp_req=6 ttl=62 time=0.044 ms
64 bytes from 172.47.48.2: icmp_req=7 ttl=62 time=0.050 ms
64 bytes from 172.47.48.2: icmp_req=8 ttl=62 time=0.051 ms
64 bytes from 172.47.48.2: icmp_req=9 ttl=62 time=0.065 ms
64 bytes from 172.47.48.2: icmp_req=10 ttl=62 time=0.051 ms

--- 172.47.48.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8996ms
rtt min/avg/max/mdev = 0.044/0.055/0.078/0.010 ms
root@LC1: /tmp/pycore.49430/LC1.conf#
```

Fig. 47. Conectividade entre o equipamento LC1 e o LA1.

4 Conclusões

Neste trabalho prático dividido em duas partes, abordamos inicialmente questões relacionadas com o formato dos datagramas que são enviados entre uma origem e um destino com ajuda do comando traceroute. Na segunda fase tratamos de assuntos tais como o endereçamento e encaminhamento IP, e subnetting.

Na parte 1 do trabalho trabalhamos com o wireshark que captura o tráfego entre duas entidades protocolares e com uma topologia CORE muito básica. Analisamos a fundo o campo TTL do datagrama e conseqüentemente as mensagens ICMP que essencialmente são utilizadas para fornecer relatórios de erro à fonte original. Baseando-nos nos datagramas capturados tratamos do assunto da fragmentação, analisando os campos que nos informam de tal e que nos possibilitam a construção do datagrama original na origem.

Na parte 2 do trabalho criamos uma topologia CORE mais complexa baseada em 3 departamentos onde analisamos a fundo as tabelas de endereçamento tanto dos laptops como dos routers e quais as conseqüência da alteração ou da eliminação de rotas nestas tabelas. Praticamos o conceito de subnetting no último exercício desta parte, na medida em que foi necessário criar uma subrede para cada departamento.

Concluindo abordamos todos os assuntos relacionados com IP, tal como foram expostos nas teóricas, desta forma aplicamos todo o conhecimento teórico a um nível mais prático ficando desta forma melhor consolidado.