

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.

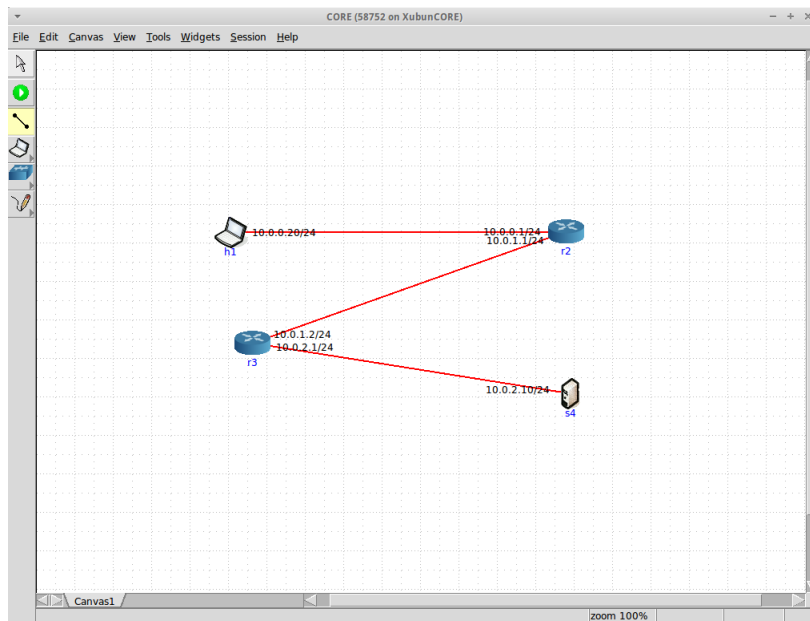


Fig. 1. Rede do exercício 1 do enunciado.

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:

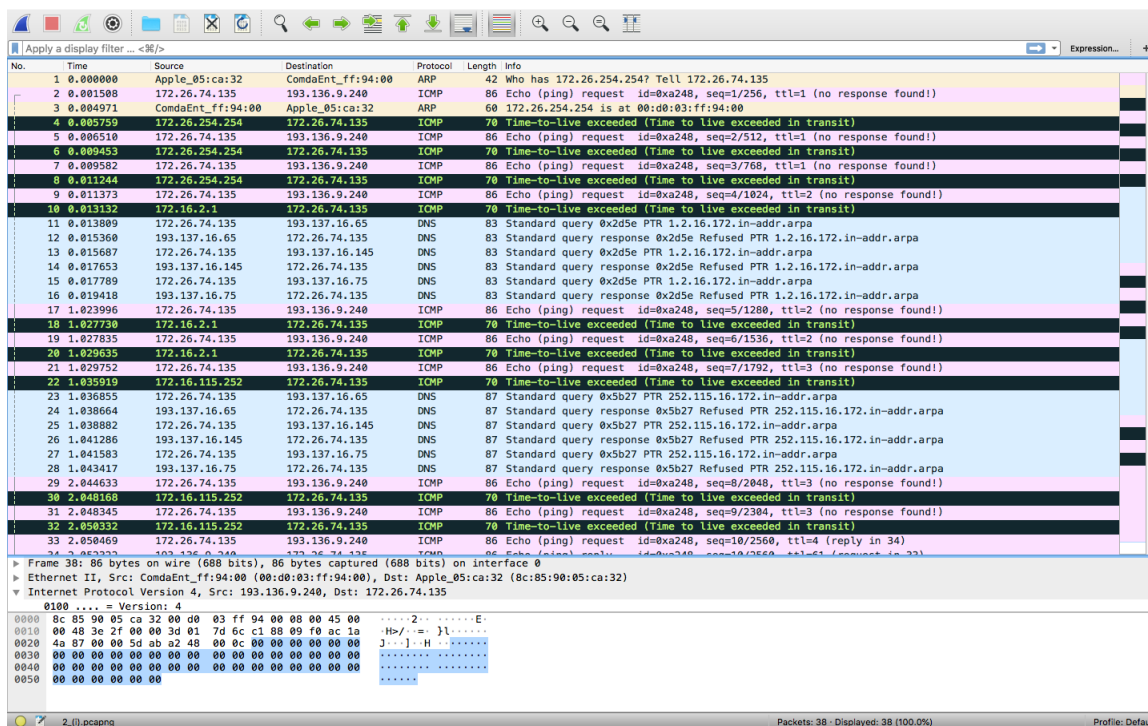


Fig. 7. Visão geral do tráfego gerado pelo traceroute usando o tamanho por defeito.

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.

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.

No.	Time	Source	Destination	Protocol	Length	Info
22	1.035919	172.16.115.252	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
30	2.048168	172.16.115.252	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
32	2.050332	172.16.115.252	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)
19	1.027720	172.16.2.1	172.26.74.135	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
20	1.028635	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	0.000510	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!)
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	0.015807	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	2.048345	172.26.74.135	193.136.9.240	ICMP	86	Echo (ping) request id=0xa248, seq=9/2304, ttl=3 (no response found!)
33	2.050460	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	2.057202	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	0.041286	193.137.16.145	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
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
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
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

Fig. 12. Visão geral do tráfego gerado pelo traceroute ordenado de acordo com o endereço IP fonte.

```

▶ 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

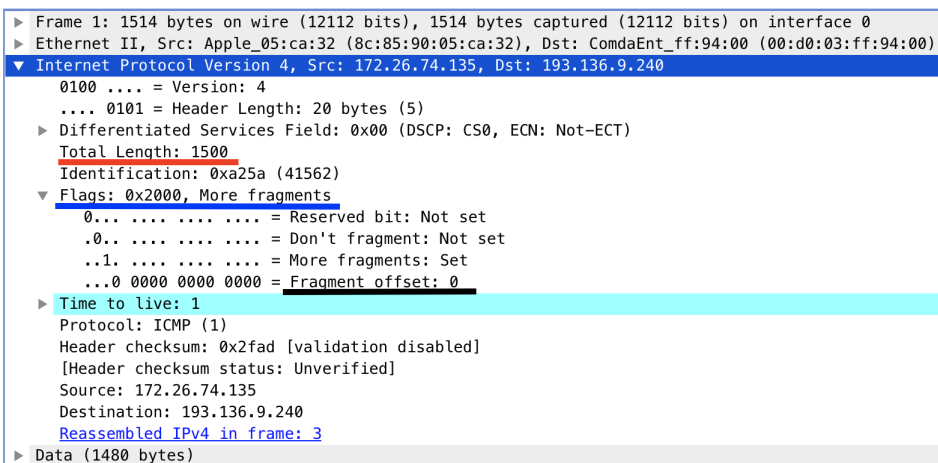
Pretende-se agora analisar a fragmentação de pacotes IP. Reponha a ordem do tráfego capturado usando a coluna do tempo de captura. Observe o tráfego depois do tamanho de pacote ter sido definido para 35XX bytes.

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.

