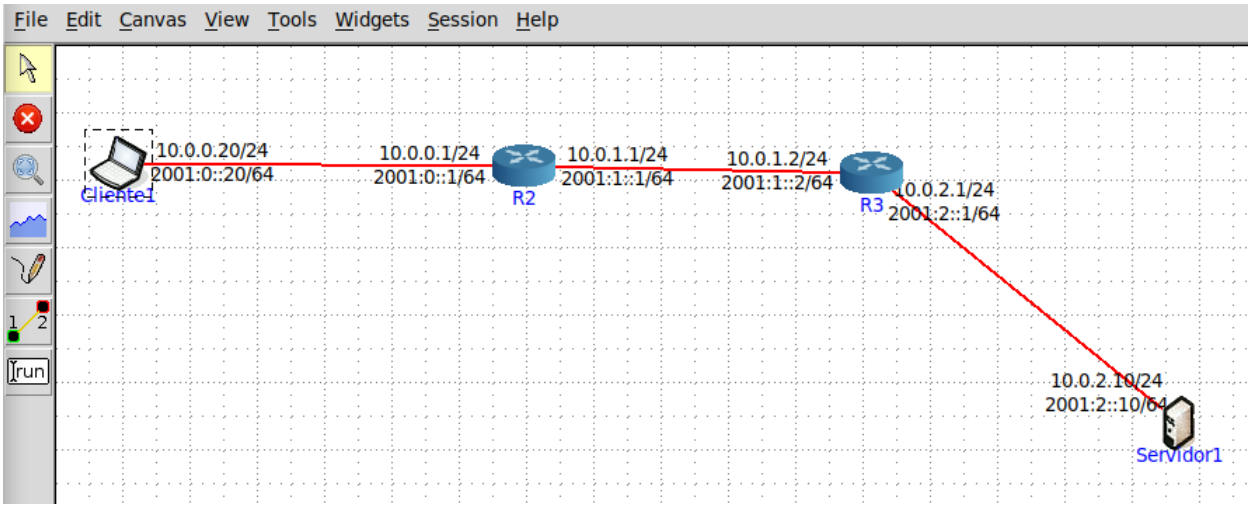
TP3 - PL19

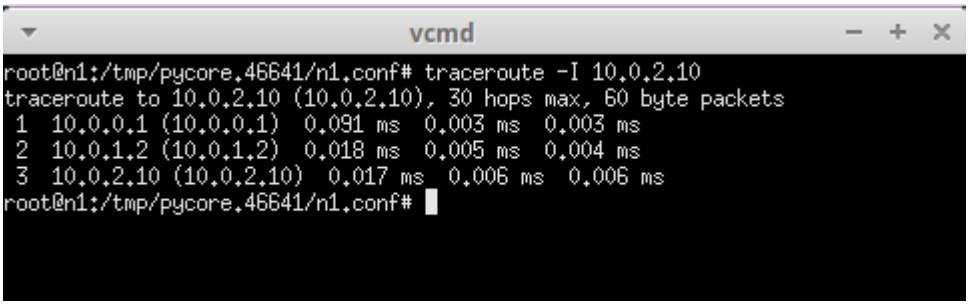
Eduardo Manuel Sousa Pereira - A70619

Questões e respostas

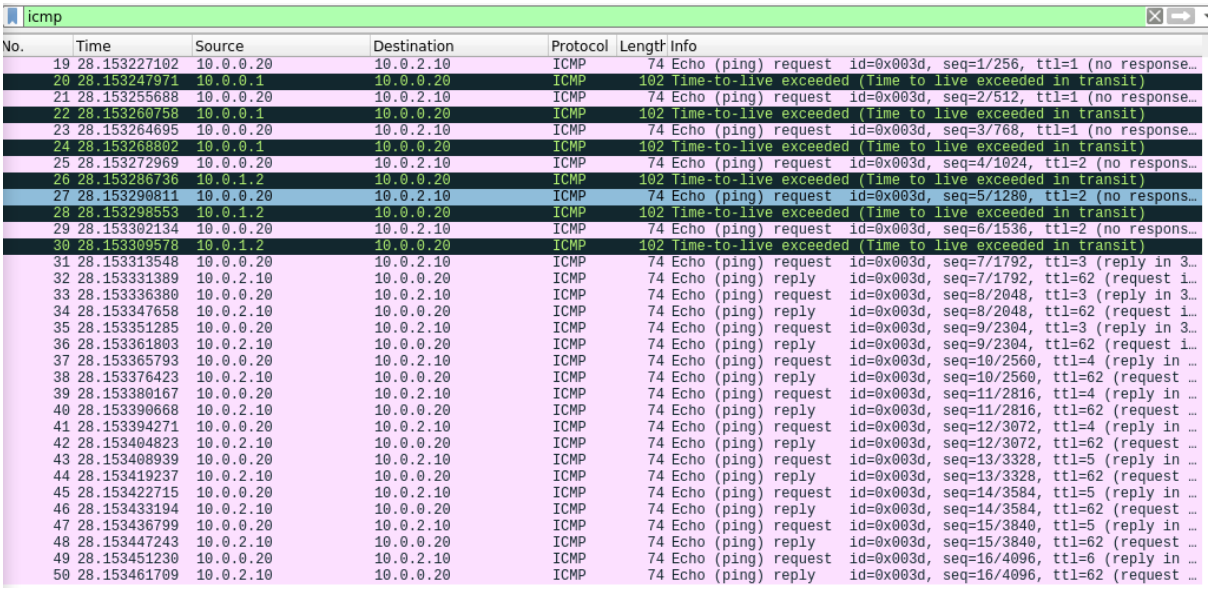
1. Prepare uma topologia CORE para verificar o comportamento do traceroute. Ligue um host (pc) *Cliente1* a um router *R2*; o router *R2* a um router *R3*, que por sua vez, se liga a um host (servidor) *Servidor1*. Ajuste o nome dos equipamentos atribuídos por defeito para a topologia do enunciado.



a) Ative o Wireshark no *Cliente1*. Numa shell do *Cliente1*, execute o comando traceroute -I para o endereço IP do *Servidor1*.



b) Registe e analise o tráfego ICMP enviado pelo *Cliente1* e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.



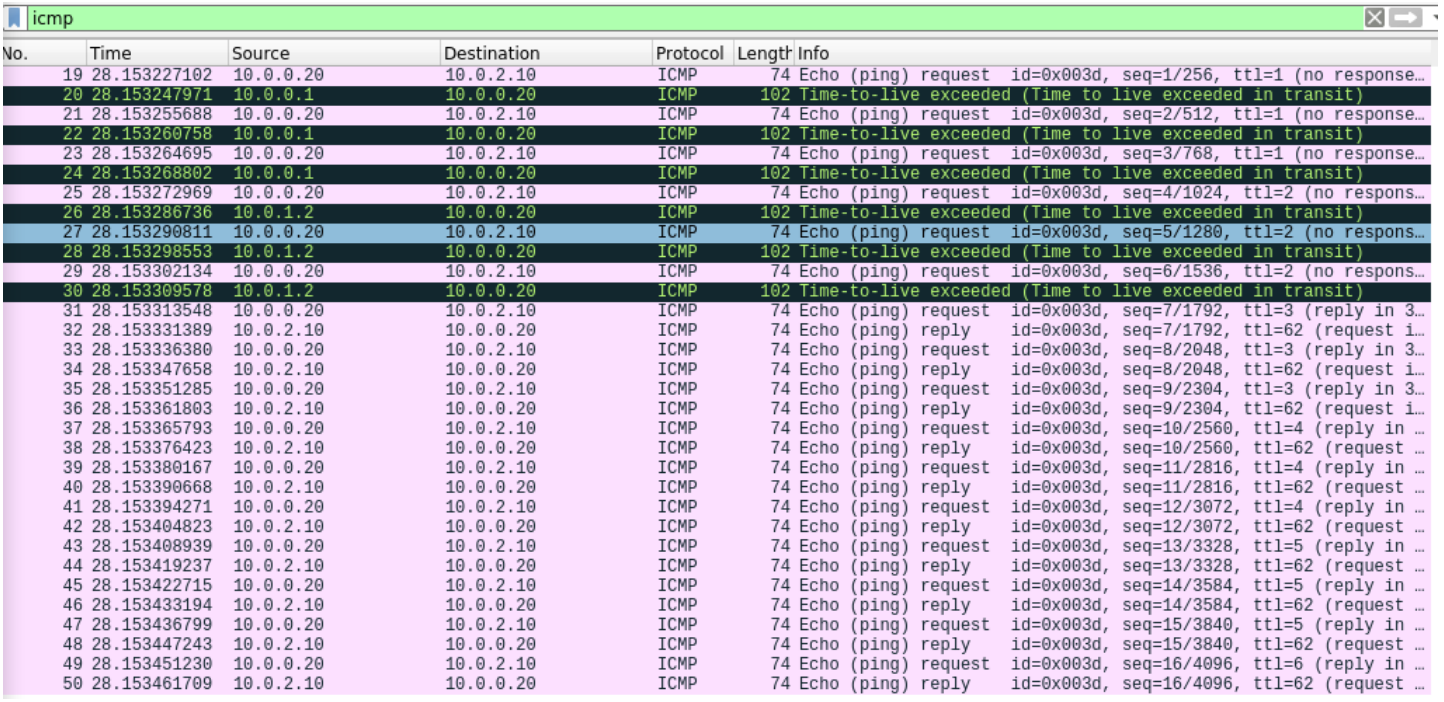
icmp

No.	Time	Source	Destination	Protocol	Length	Info
19	28.153227102	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=1/256, ttl=1 (no response…
20	28.153247971	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
21	28.153255688	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=2/512, ttl=1 (no response…
22	28.153260758	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
23	28.153264695	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=3/768, ttl=1 (no response…
24	28.153268802	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
25	28.153272969	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=4/1024, ttl=2 (no respons…
26	28.153286736	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
27	28.153290811	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=5/1280, ttl=2 (no respons…
28	28.153298553	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
29	28.153302134	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=6/1536, ttl=2 (no respons…
30	28.153309578	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
31	28.153313548	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=7/1792, ttl=3 (reply in 3…
32	28.153331389	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=7/1792, ttl=62 (request i…
33	28.153336380	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=8/2048, ttl=3 (reply in 3…
34	28.153347658	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=8/2048, ttl=62 (request i…
35	28.153351285	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=9/2304, ttl=3 (reply in 3…
36	28.153361803	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=9/2304, ttl=62 (request i…
37	28.153365793	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=10/2560, ttl=4 (reply in …
38	28.153376423	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=10/2560, ttl=62 (request …
39	28.153380167	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=11/2816, ttl=4 (reply in …
40	28.153390668	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=11/2816, ttl=62 (request …
41	28.153394271	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=12/3072, ttl=4 (reply in …
42	28.153404823	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=12/3072, ttl=62 (request …
43	28.153408939	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=13/3328, ttl=5 (reply in …
44	28.153419237	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=13/3328, ttl=62 (request …
45	28.153422715	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=14/3584, ttl=5 (reply in …
46	28.153433194	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=14/3584, ttl=62 (request …
47	28.153436799	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=15/3840, ttl=5 (reply in …
48	28.153447243	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=15/3840, ttl=62 (request …
49	28.153451230	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=16/4096, ttl=6 (reply in …
50	28.153461709	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=16/4096, ttl=62 (request …

Como era esperado, o resultado é a comunicação entre o Cliente1 e o Servidor1, sendo que o Cliente1 faz requests ao Servidor1 e, quando o TTL é maior que 3, este dá reply.

c) Qual deve ser o valor inicial mínimo do campo TTL para alcançar o *Servidor1*? Verifique na prática que a sua resposta está correta.

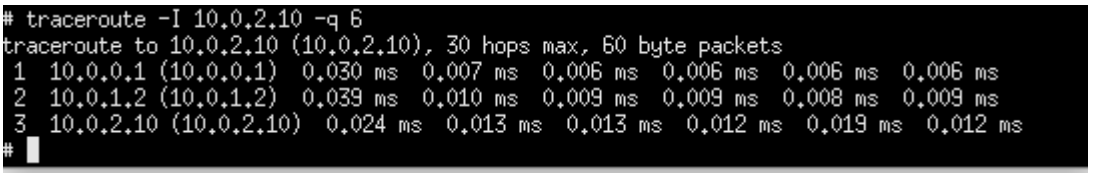
O valor inicial de TTL deve ser TTL = 3, porque como se pode verificar na imagem abaixo, com TTL = 1 ou com TTL = 2, o pacote não chega ao Servidor1, que, por sua vez não retorna resposta. Ou seja, o pacote só consegue chegar ao Servidor1, com TTL igual ou superior a 3.



icmp

No.	Time	Source	Destination	Protocol	Length	Info
19	28.153227102	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=1/256, ttl=1 (no response…
20	28.153247971	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
21	28.153255688	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=2/512, ttl=1 (no response…
22	28.153260758	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
23	28.153264695	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=3/768, ttl=1 (no response…
24	28.153268802	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
25	28.153272969	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=4/1024, ttl=2 (no respons…
26	28.153286736	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
27	28.153290811	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=5/1280, ttl=2 (no respons…
28	28.153298553	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
29	28.153302134	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=6/1536, ttl=2 (no respons…
30	28.153309578	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
31	28.153313548	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=7/1792, ttl=3 (reply in 3…
32	28.153331389	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=7/1792, ttl=62 (request i…
33	28.153336380	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=8/2048, ttl=3 (reply in 3…
34	28.153347658	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=8/2048, ttl=62 (request i…
35	28.153351285	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=9/2304, ttl=3 (reply in 3…
36	28.153361803	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=9/2304, ttl=62 (request i…
37	28.153365793	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=10/2560, ttl=4 (reply in …
38	28.153376423	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=10/2560, ttl=62 (request …
39	28.153380167	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=11/2816, ttl=4 (reply in …
40	28.153390668	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=11/2816, ttl=62 (request …
41	28.153394271	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=12/3072, ttl=4 (reply in …
42	28.153404823	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=12/3072, ttl=62 (request …
43	28.153408939	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=13/3328, ttl=5 (reply in …
44	28.153419237	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=13/3328, ttl=62 (request …
45	28.153422715	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=14/3584, ttl=5 (reply in …
46	28.153433194	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=14/3584, ttl=62 (request …
47	28.153436799	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=15/3840, ttl=5 (reply in …
48	28.153447243	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=15/3840, ttl=62 (request …
49	28.153451230	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x003d, seq=16/4096, ttl=6 (reply in …
50	28.153461709	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x003d, seq=16/4096, ttl=62 (request …

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



```
# traceroute -I 10.0.2.10 -q 6
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.030 ms  0.007 ms  0.006 ms  0.006 ms  0.006 ms  0.006 ms
 2  10.0.1.2 (10.0.1.2)  0.039 ms  0.010 ms  0.009 ms  0.009 ms  0.008 ms  0.009 ms
 3  10.0.2.10 (10.0.2.10)  0.024 ms  0.013 ms  0.013 ms  0.012 ms  0.019 ms  0.012 ms
#
```

O tempo médio de ida-e-volta pode-se saber calculando:
(0.024+0.013+0.013+0.012+0.019+0.012)/6 = 0.016ms

2. Usando o Wireshark, capture o tráfego gerado pelo traceroute para os seguintes tamanhos de pacote: (i) sem especificar, ou seja, o tamanho por defeito; (ii) 3219 bytes. 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 como resposta.

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.87	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=31831/22396, ttl=10 (no response found!)
2	0.015953	193.136.4.100	192.168.1.87	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
3	0.040544	192.168.1.87	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=31832/22652, ttl=11 (no response found!)
4	0.080936	192.168.1.87	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=31833/22908, ttl=12 (no response found!)
5	0.121178	192.168.1.87	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=31834/23164, ttl=13 (no response found!)
6	0.162125	192.168.1.87	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=31835/23420, ttl=14 (reply in 7)
7	0.178655	193.136.9.240	192.168.1.87	ICMP	70	Echo (ping) reply id=0x0001, seq=31835/23420, ttl=51 (request in 6)
8	0.202156	192.168.1.87	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=31836/23676, ttl=255 (reply in 9)
9	0.218917	193.136.9.240	192.168.1.87	ICMP	70	Echo (ping) reply id=0x0001, seq=31836/23676, ttl=51 (request in 8)

Selecione a primeira mensagem ICMP capturada referente a (i) e centre a análise no nível protocolar IP.

```
▾ Internet Protocol Version 4, Src: 192.168.1.87, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0xa2d2 (41682)
  ▸ Flags: 0x00
    Fragment Offset: 0
    Time to Live: 10
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.87
    Destination Address: 193.136.9.240
```

```
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xb9e1 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 31835 (0x7c5b)
    Sequence Number (LE): 23420 (0x5b7c)
    [Response frame: 7]
  ▾ Data (28 bytes)
      Data: 20202020202020202020202020202020202020202020202020202020
      [Length: 28]
```

Através da análise do cabeçalho IP diga:

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

O endereço IP é 192.168.1.87.

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

O valor do campo protocolo é 0100, em binário.

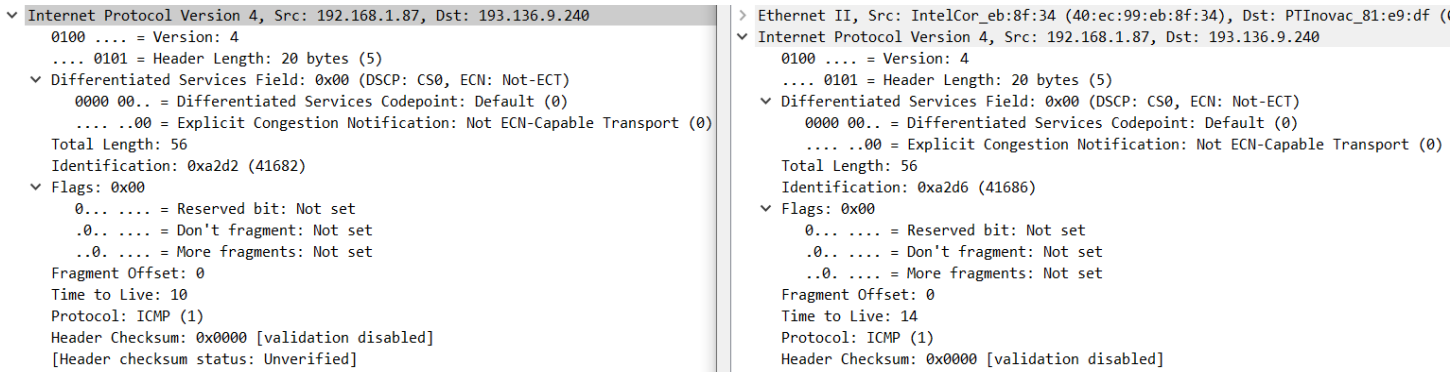
c) Quantos bytes tem o cabeçalho IP(v4)? Quantos bytes tem o campo de dados (payload) do datagrama? Como se calcula o tamanho do payload?

O cabeçalho IP(v4) tem 20 bytes e o campo de dados do datagrama tem 28 bytes.
O tamanho do payload calcula-se subtraíndo o *Header Length* ao *Total Length.* Neste caso, seria 56-20=36.

d) O datagrama IP foi fragmentado? Justifique.

Não, porque o tamanho do datagrama não excede 1480 bytes. Logo, não é necessário fragmentar o datagrama.

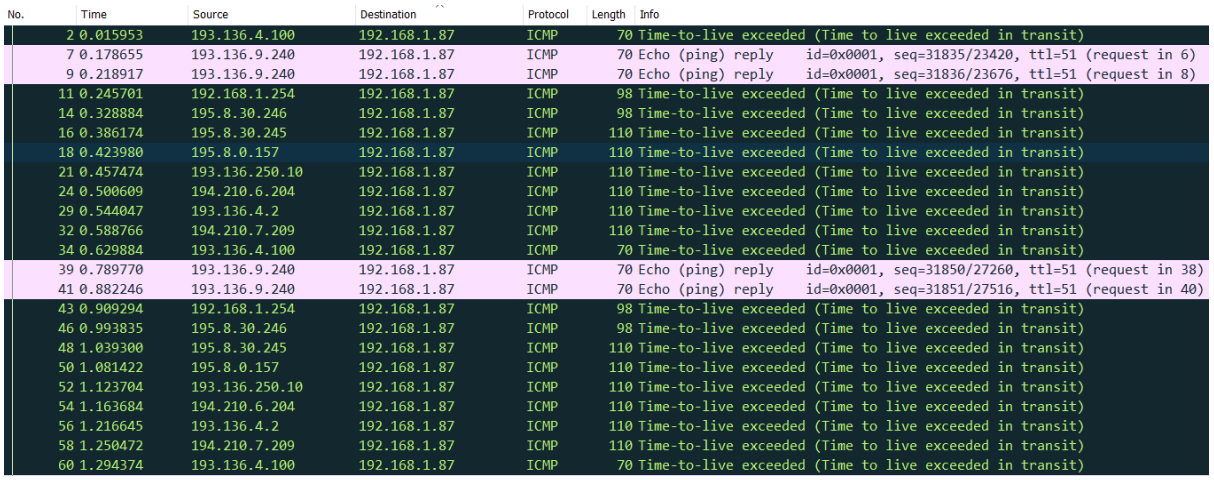
e) Ordene os pacotes capturados de acordo com o endereço IP fonte e analise a sequência de tráfego ICMP gerado a partir do endereço IP atribuído à interface da sua máquina. Para a sequência de mensagens ICMP enviadas pelo seu computador, indique que campos do cabeçalho IP variam de pacote para pacote.



Os únicos campos do cabeçalho IP que variam de pacote para pacote são o Time to Live e a identificação do datagrama IP.

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

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



No.	Time	Source	Destination	Protocol	Length	Info
2	0.015953	193.136.4.100	192.168.1.87	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
7	0.178655	193.136.9.240	192.168.1.87	ICMP	70	Echo (ping) reply id=0x0001, seq=31835/23420, ttl=51 (request in 6)
9	0.218917	193.136.9.240	192.168.1.87	ICMP	70	Echo (ping) reply id=0x0001, seq=31836/23676, ttl=51 (request in 8)
11	0.245701	192.168.1.254	192.168.1.87	ICMP	98	Time-to-live exceeded (Time to live exceeded in transit)
14	0.328884	195.8.30.246	192.168.1.87	ICMP	98	Time-to-live exceeded (Time to live exceeded in transit)
16	0.386174	195.8.30.245	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
18	0.423980	195.8.0.157	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
21	0.457474	193.136.250.10	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
24	0.500609	194.210.6.204	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
29	0.544047	193.136.4.2	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
32	0.588766	194.210.7.209	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
34	0.629884	193.136.4.100	192.168.1.87	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
39	0.789770	193.136.9.240	192.168.1.87	ICMP	70	Echo (ping) reply id=0x0001, seq=31850/27260, ttl=51 (request in 38)
41	0.882246	193.136.9.240	192.168.1.87	ICMP	70	Echo (ping) reply id=0x0001, seq=31851/27516, ttl=51 (request in 40)
43	0.909294	192.168.1.254	192.168.1.87	ICMP	98	Time-to-live exceeded (Time to live exceeded in transit)
46	0.993835	195.8.30.246	192.168.1.87	ICMP	98	Time-to-live exceeded (Time to live exceeded in transit)
48	1.039300	195.8.30.245	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
50	1.081422	195.8.0.157	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
52	1.123704	193.136.250.10	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
54	1.163684	194.210.6.204	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
56	1.216645	193.136.4.2	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
58	1.250472	194.210.7.209	192.168.1.87	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
60	1.294374	193.136.4.100	192.168.1.87	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

O valor do TTL é 51. Este valor não é constante para todas as mensagens ICMP TTL exceeded enviadas ao host, pois depende de onde são enviadas. Cada vez que passa por um *node*, o TTL é alterado.

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 3219 bytes.

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

Sempre que o tamanho do datagrama é superior a 1480 bytes, o datagrama é fragmentado. Neste caso o tamanho do datagrama está definido para 3219 bytes, daí a necessidade de fragmentar o pacote inicial.

b)

Imprima o primeiro fragmento do datagrama IP segmentado.

```
Internet Protocol Version 4, Src: 192.168.1.87, 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: 0x78a5 (30885)
    Flags: 0x20, More fragments
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..1. .... = More fragments: Set
    Fragment Offset: 0
    Time to Live: 255
```

Que informação no cabeçalho indica que o datagrama foi fragmentado?

Na imagem acima, em *More Fragments*, o último bit MF, está configurado em 1, o que significa que este não é o último fragmento (o último teria de ter o bit MF configurado em 0). Logo, o datagrama foi fragmentado.

Que informação no cabeçalho IP indica que se trata do primeiro fragmento?

Quando se trata do primeiro fragmento, o *Fragment Offset* tem valor 0. Pela imagem acima pode-se comprovar que este é o caso, logo este é o primeiro fragmento.

Qual é o tamanho deste datagrama IP?

O tamanho deste datagrama IP 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?

```
Internet Protocol Version 4, Src: 192.168.1.87, 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: 0x78a5 (30885)
    Flags: 0x20, More fragments
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..1. .... = More fragments: Set
    Fragment Offset: 1480
    Time to Live: 255
```

Quando se trata do 1º fragmento, o *Fragment Offset* tem valor 0, como podemos ver em cima, este tem valor 1480, logo este não pode ser o 1º fragmento.
Existem mais fragmentos, pois quando se trata do último fragmento, o último bit *More Fragments* é configurado como 0. Como se vê na imagem, o último bit está configurado como 1.

d) Quantos fragmentos foram criados a partir do datagrama original? Como se detecta o último fragmento correspondente ao datagrama original?

```
Source Address: 192.168.1.87
Destination Address: 193.136.9.240
[3 IPv4 Fragments (3199 bytes): #1(1480), #2(1480), #3(239)]
    [Frame: 1, payload: 0-1479 (1480 bytes)]
    [Frame: 2, payload: 1480-2959 (1480 bytes)]
    [Frame: 3, payload: 2960-3198 (239 bytes)]
    [Fragment count: 3]
    [Reassembled IPv4 length: 3199]
    [Reassembled IPv4 data: 08005d8a000452492020202020202020202020202020202020202020202020202020…]
```

Foram criados 3 fragmentos a partir do datagrama original.
É possível determinar qual é o último fragmento a partir do campo MF (More Fragments), se este tiver o último bit a 1, não é o último fragmento, se, pelo contrário, tiver o último bit a 0, então este é o último fragmento correspondente ao datagrama original.

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.

```
Ethernet II, Src: IntelCor_eb:8f:34 (40:ec:99:eb:8f:34), Dst: PTInovac_81:e9:df (00:06:91:81:e9:df)
Internet Protocol Version 4, Src: 192.168.1.87, 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: 0x78a5 (30885)
    Flags: 0x20, More fragments
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..1. .... = More fragments: Set
    Fragment Offset: 0
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.87
    Destination Address: 193.136.9.240
    [Reassembled IPv4 in frame: 3]
```

```
Internet Protocol Version 4, Src: 192.168.1.87, 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: 0x78a5 (30885)
    Flags: 0x20, More fragments
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..1. .... = More fragments: Set
    Fragment Offset: 1480
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.87
    Destination Address: 193.136.9.240
    [Reassembled IPv4 in frame: 3]
```

```
Internet Protocol Version 4, Src: 192.168.1.87, 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: 259
    Identification: 0x78a5 (30885)
    Flags: 0x01
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment Offset: 2960
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.87
    Destination Address: 193.136.9.240
    [3 IPv4 Fragments (3199 bytes): #1(1480), #2(1480), #3(239)]
        [Frame: 1, payload: 0-1479 (1480 bytes)]
        [Frame: 2, payload: 1480-2959 (1480 bytes)]
        [Frame: 3, payload: 2960-3198 (239 bytes)]
        [Fragment count: 3]
        [Reassembled IPv4 length: 3199]
```

O que muda, essencialmente, entre os 3 fragmentos, é o Fragment Offset e o valor do último bit de MF. Quando o Offset não é 0 e o último bit de MF é 0, o que significa que está no último fragmento, usa a informação do Offset para remontar o datagrama.