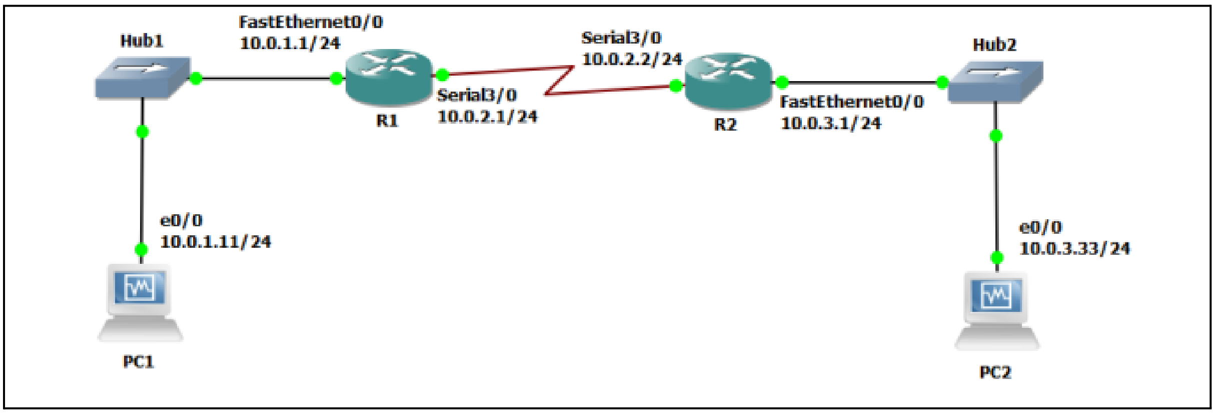
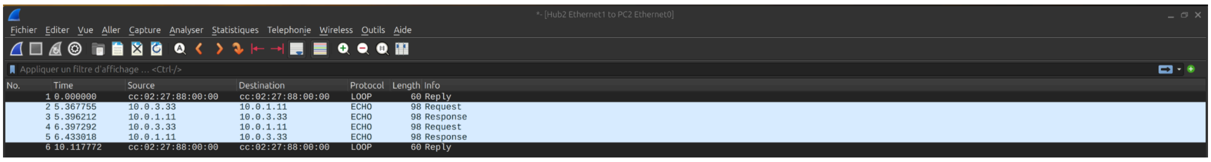
Tristan BRAU
Léo PHAV

Interconnexion des réseaux - Lab 5

PART 1: Comparing UDP and TCP Performance

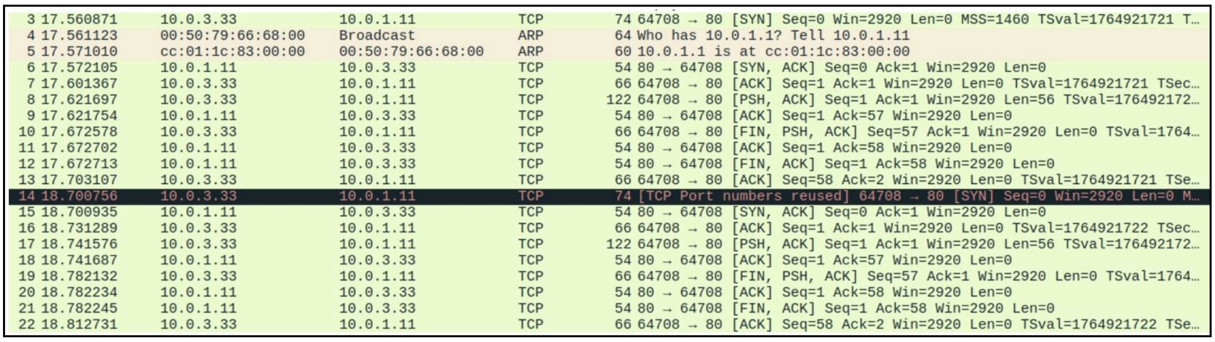


Exercise 1. Network setup and transmitting data with TCP and UDP


*- [Hub2 Ethernet1 to PC2 Ethernet0]
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
Appliquer un filtre d'affichage ... <Ctrl-/>

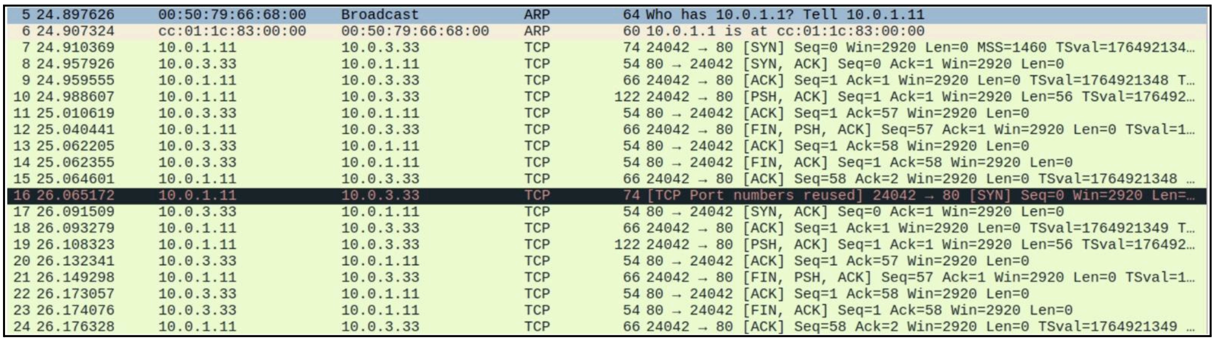
```
No.      Time        Source              Destination         Protocol  Length Info
       1 0.000000    cc:02:27:88:00:00   cc:02:27:88:00:00   LOOP        60 Reply
       2 5.367755    10.0.3.33           10.0.1.11           ECHO        98 Request
       3 5.396212    10.0.1.11           10.0.3.33           ECHO        98 Response
       4 6.397292    10.0.3.33           10.0.1.11           ECHO        98 Request
       5 6.433018    10.0.1.11           10.0.3.33           ECHO        98 Response
       6 10.117772   cc:02:27:88:00:00   cc:02:27:88:00:00   LOOP        60 Reply
```

PC1 -> PC2:



```
 3 17.560871   10.0.3.33           10.0.1.11           TCP        74 64708 → 80 [SYN] Seq=0 Win=2920 Len=0 MSS=1460 TSval=1764921721 T…
 4 17.561123   00:50:79:66:68:00   Broadcast           ARP        64 Who has 10.0.1.1? Tell 10.0.1.11
 5 17.571010   cc:01:1c:83:00:00   00:50:79:66:68:00   ARP        60 10.0.1.1 is at cc:01:1c:83:00:00
 6 17.572105   10.0.1.11           10.0.3.33           TCP        54 80 → 64708 [SYN, ACK] Seq=0 Ack=1 Win=2920 Len=0
 7 17.601367   10.0.3.33           10.0.1.11           TCP        66 64708 → 80 [ACK] Seq=1 Ack=1 Win=2920 Len=0 TSval=1764921721 TSec…
 8 17.621697   10.0.3.33           10.0.1.11           TCP       122 64708 → 80 [PSH, ACK] Seq=1 Ack=1 Win=2920 Len=56 TSval=176492172…
 9 17.621754   10.0.1.11           10.0.3.33           TCP        54 80 → 64708 [ACK] Seq=1 Ack=57 Win=2920 Len=0
10 17.672578   10.0.3.33           10.0.1.11           TCP        66 64708 → 80 [FIN, PSH, ACK] Seq=57 Ack=1 Win=2920 Len=0 TSval=1764…
11 17.672702   10.0.1.11           10.0.3.33           TCP        54 80 → 64708 [ACK] Seq=1 Ack=58 Win=2920 Len=0
12 17.672713   10.0.1.11           10.0.3.33           TCP        54 80 → 64708 [FIN, ACK] Seq=1 Ack=58 Win=2920 Len=0
13 17.703107   10.0.3.33           10.0.1.11           TCP        66 64708 → 80 [ACK] Seq=58 Ack=2 Win=2920 Len=0 TSval=1764921721 TSe…
14 18.700756   10.0.3.33           10.0.1.11           TCP        74 [TCP Port numbers reused] 64708 → 80 [SYN] Seq=0 Win=2920 Len=0 M…
15 18.700935   10.0.1.11           10.0.3.33           TCP        54 80 → 64708 [SYN, ACK] Seq=0 Ack=1 Win=2920 Len=0
16 18.731289   10.0.3.33           10.0.1.11           TCP        66 64708 → 80 [ACK] Seq=1 Ack=1 Win=2920 Len=0 TSval=1764921722 TSec…
17 18.741576   10.0.3.33           10.0.1.11           TCP       122 64708 → 80 [PSH, ACK] Seq=1 Ack=1 Win=2920 Len=56 TSval=176492172…
18 18.741687   10.0.1.11           10.0.3.33           TCP        54 80 → 64708 [ACK] Seq=1 Ack=57 Win=2920 Len=0
19 18.782132   10.0.3.33           10.0.1.11           TCP        66 64708 → 80 [FIN, PSH, ACK] Seq=57 Ack=1 Win=2920 Len=0 TSval=1764…
20 18.782234   10.0.1.11           10.0.3.33           TCP        54 80 → 64708 [ACK] Seq=1 Ack=58 Win=2920 Len=0
21 18.782245   10.0.1.11           10.0.3.33           TCP        54 80 → 64708 [FIN, ACK] Seq=1 Ack=58 Win=2920 Len=0
22 18.812731   10.0.3.33           10.0.1.11           TCP        66 64708 → 80 [ACK] Seq=58 Ack=2 Win=2920 Len=0 TSval=1764921722 TSe…
```

PC2 -> PC1:



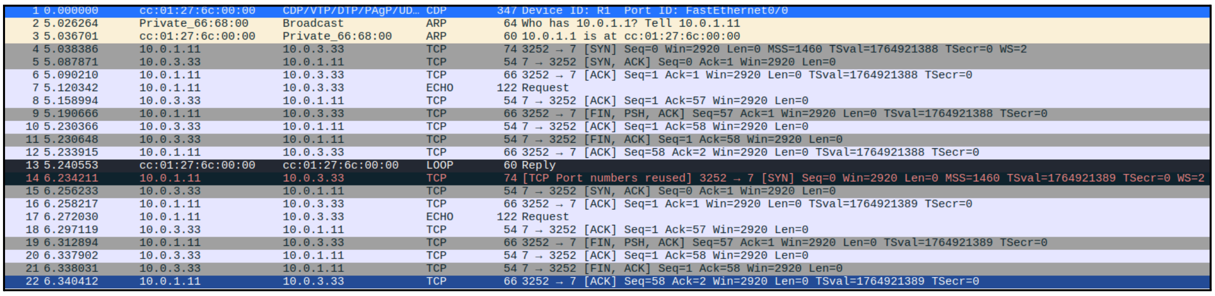
```
 5 24.897626   00:50:79:66:68:00   Broadcast           ARP        64 Who has 10.0.1.1? Tell 10.0.1.11
 6 24.907324   cc:01:1c:83:00:00   00:50:79:66:68:00   ARP        60 10.0.1.1 is at cc:01:1c:83:00:00
 7 24.910369   10.0.1.11           10.0.3.33           TCP        74 24042 → 80 [SYN] Seq=0 Win=2920 Len=0 MSS=1460 TSval=176492134…
 8 24.957926   10.0.3.33           10.0.1.11           TCP        54 80 → 24042 [SYN, ACK] Seq=0 Ack=1 Win=2920 Len=0
 9 24.959555   10.0.1.11           10.0.3.33           TCP        66 24042 → 80 [ACK] Seq=1 Ack=1 Win=2920 Len=0 TSval=1764921348 T…
10 24.988607   10.0.1.11           10.0.3.33           TCP       122 24042 → 80 [PSH, ACK] Seq=1 Ack=1 Win=2920 Len=56 TSval=176492…
11 25.010619   10.0.3.33           10.0.1.11           TCP        54 80 → 24042 [ACK] Seq=1 Ack=57 Win=2920 Len=0
12 25.040441   10.0.1.11           10.0.3.33           TCP        66 24042 → 80 [FIN, PSH, ACK] Seq=57 Ack=1 Win=2920 Len=0 TSval=1…
13 25.062205   10.0.3.33           10.0.1.11           TCP        54 80 → 24042 [ACK] Seq=1 Ack=58 Win=2920 Len=0
14 25.062355   10.0.3.33           10.0.1.11           TCP        54 80 → 24042 [FIN, ACK] Seq=1 Ack=58 Win=2920 Len=0
15 25.064601   10.0.1.11           10.0.3.33           TCP        66 24042 → 80 [ACK] Seq=58 Ack=2 Win=2920 Len=0 TSval=1764921348 …
16 26.065172   10.0.1.11           10.0.3.33           TCP        74 [TCP Port numbers reused] 24042 → 80 [SYN] Seq=0 Win=2920 Len=…
17 26.091509   10.0.3.33           10.0.1.11           TCP        54 80 → 24042 [SYN, ACK] Seq=0 Ack=1 Win=2920 Len=0
18 26.093279   10.0.1.11           10.0.3.33           TCP        66 24042 → 80 [ACK] Seq=1 Ack=1 Win=2920 Len=0 TSval=1764921349 T…
19 26.108323   10.0.1.11           10.0.3.33           TCP       122 24042 → 80 [PSH, ACK] Seq=1 Ack=1 Win=2920 Len=56 TSval=176492…
20 26.132341   10.0.3.33           10.0.1.11           TCP        54 80 → 24042 [ACK] Seq=1 Ack=57 Win=2920 Len=0
21 26.149298   10.0.1.11           10.0.3.33           TCP        66 24042 → 80 [FIN, PSH, ACK] Seq=57 Ack=1 Win=2920 Len=0 TSval=1…
22 26.173057   10.0.3.33           10.0.1.11           TCP        54 80 → 24042 [ACK] Seq=1 Ack=58 Win=2920 Len=0
23 26.174076   10.0.3.33           10.0.1.11           TCP        54 80 → 24042 [FIN, ACK] Seq=1 Ack=58 Win=2920 Len=0
24 26.176328   10.0.1.11           10.0.3.33           TCP        66 24042 → 80 [ACK] Seq=58 Ack=2 Win=2920 Len=0 TSval=1764921349 …
```

Tristan BRAU
Léo PHAV

Lab Questions:

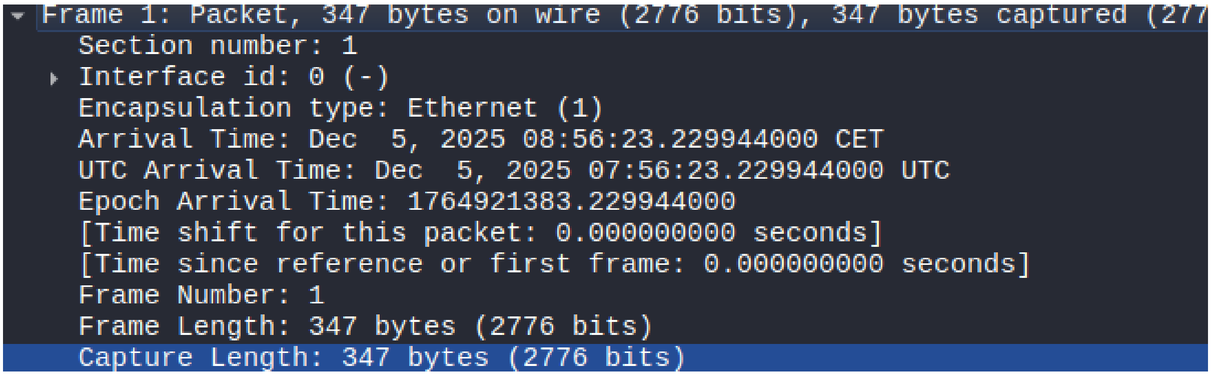
Use the data captured with Wireshark to answer the following questions.

- How many packets are exchanged in the data transfer? What are the sizes of the TCP segments?



```
 1 0.000000   cc:01:27:6c:00:00  CDP/VTP/DTP/PAgP/UD… CDP    347 Device ID: R1  Port ID: FastEthernet0/0
 2 5.026264   Private_66:68:00   Broadcast            ARP     64 Who has 10.0.1.1? Tell 10.0.1.11
 3 5.036701   cc:01:27:6c:00:00  Private_66:68:00     ARP     60 10.0.1.1 is at cc:01:27:6c:00:00
 4 5.038386   10.0.1.11          10.0.3.33            TCP     74 3252 → 7 [SYN] Seq=0 Win=2920 Len=0 MSS=1460 TSval=1764921388 TSecr=0 WS=2
 5 5.087871   10.0.3.33          10.0.1.11            TCP     54 7 → 3252 [SYN, ACK] Seq=0 Ack=1 Win=2920 Len=0
 6 5.090210   10.0.1.11          10.0.3.33            TCP     66 3252 → 7 [ACK] Seq=1 Ack=1 Win=2920 Len=0 TSval=1764921388 TSecr=0
 7 5.120342   10.0.1.11          10.0.3.33            ECHO   122 Request
 8 5.158994   10.0.3.33          10.0.1.11            TCP     54 7 → 3252 [ACK] Seq=1 Ack=57 Win=2920 Len=0
 9 5.190666   10.0.1.11          10.0.3.33            TCP     66 3252 → 7 [FIN, PSH, ACK] Seq=57 Ack=1 Win=2920 Len=0 TSval=1764921388 TSecr=0
10 5.230366   10.0.3.33          10.0.1.11            TCP     54 7 → 3252 [ACK] Seq=1 Ack=58 Win=2920 Len=0
11 5.230648   10.0.3.33          10.0.1.11            TCP     54 7 → 3252 [FIN, ACK] Seq=1 Ack=58 Win=2920 Len=0
12 5.233915   10.0.1.11          10.0.3.33            TCP     66 3252 → 7 [ACK] Seq=58 Ack=2 Win=2920 Len=0 TSval=1764921388 TSecr=0
13 5.240553   cc:01:27:6c:00:00  cc:01:27:6c:00:00    LOOP    60 Reply
14 6.234211   10.0.1.11          10.0.3.33            TCP     74 [TCP Port numbers reused] 3252 → 7 [SYN] Seq=0 Win=2920 Len=0 MSS=1460 TSval=1764921389 TSecr=0 WS=2
15 6.256233   10.0.3.33          10.0.1.11            TCP     54 7 → 3252 [SYN, ACK] Seq=0 Ack=1 Win=2920 Len=0
16 6.258217   10.0.1.11          10.0.3.33            TCP     66 3252 → 7 [ACK] Seq=1 Ack=1 Win=2920 Len=0 TSval=1764921389 TSecr=0
17 6.272030   10.0.1.11          10.0.3.33            ECHO   122 Request
18 6.297119   10.0.3.33          10.0.1.11            TCP     54 7 → 3252 [ACK] Seq=1 Ack=57 Win=2920 Len=0
19 6.312894   10.0.1.11          10.0.3.33            TCP     66 3252 → 7 [FIN, PSH, ACK] Seq=57 Ack=1 Win=2920 Len=0 TSval=1764921389 TSecr=0
20 6.337902   10.0.3.33          10.0.1.11            TCP     54 7 → 3252 [ACK] Seq=1 Ack=58 Win=2920 Len=0
21 6.338031   10.0.3.33          10.0.1.11            TCP     54 7 → 3252 [FIN, ACK] Seq=1 Ack=58 Win=2920 Len=0
22 6.340412   10.0.1.11          10.0.3.33            TCP     66 3252 → 7 [ACK] Seq=58 Ack=2 Win=2920 Len=0 TSval=1764921389 TSecr=0
```

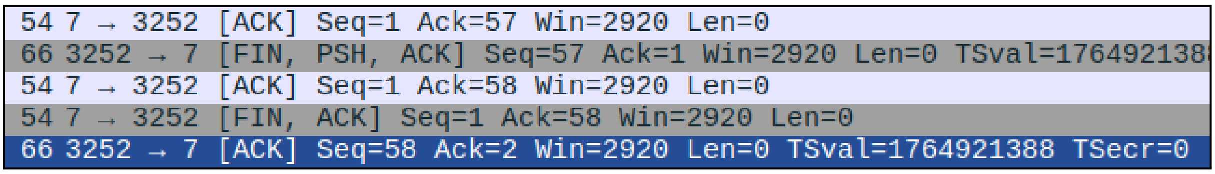
During the exchange between the two hosts, Wireshark shows that a total of 15 packets were transmitted. At the beginning, only one SYN packet is seen for the connection establishment, and the remaining packets are similar in structure, which corresponds to normal TCP communication.



```
▾ Frame 1: Packet, 347 bytes on wire (2776 bits), 347 bytes captured (277
    Section number: 1
  ▸ Interface id: 0 (-)
    Encapsulation type: Ethernet (1)
    Arrival Time: Dec  5, 2025 08:56:23.229944000 CET
    UTC Arrival Time: Dec  5, 2025 07:56:23.229944000 UTC
    Epoch Arrival Time: 1764921383.229944000
    [Time shift for this packet: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 347 bytes (2776 bits)
    Capture Length: 347 bytes (2776 bits)
```

The size of the packet is 347 bytes.

- What is the range of the sequence numbers?



```
54 7 → 3252 [ACK] Seq=1 Ack=57 Win=2920 Len=0
66 3252 → 7 [FIN, PSH, ACK] Seq=57 Ack=1 Win=2920 Len=0 TSval=176492138
54 7 → 3252 [ACK] Seq=1 Ack=58 Win=2920 Len=0
54 7 → 3252 [FIN, ACK] Seq=1 Ack=58 Win=2920 Len=0
66 3252 → 7 [ACK] Seq=58 Ack=2 Win=2920 Len=0 TSval=1764921388 TSecr=0
```

The range is [0,58]

- How many packets do not carry a payload, that is, how many packets are control packets?

To determine whether a TCP packet carries payload, we examine each packet and calculate: Total Length – IP header – TCP header. If the result is 0, the packet contains no application data and is considered a control packet. If the result is greater than 0, the packet carries payload.

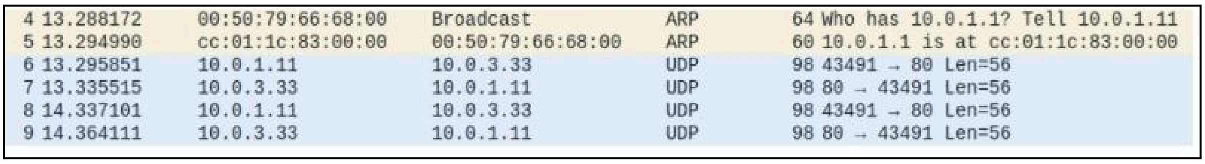
SYN → pas de payload → control
SYN/ACK → pas de payload → control
ACK → pas de payload → control
ACK → pas de payload → control
ACK → pas de payload → control
FIN/ACK → pas de payload → control
ACK → pas de payload → control
PSH ACK → contient du **PSH** (données poussées) → **payload présent**

- Compare the total number of bytes transmitted, in both directions, including Ethernet, IP, and TCP headers, to the amount of application data transmitted.

Couches	Size
Ethernet	Frame Length: 122 bytes
IP	Header Length: 20 bytes (5)
TCP	Header Length: 32 bytes (8)
Payload	56 bytes

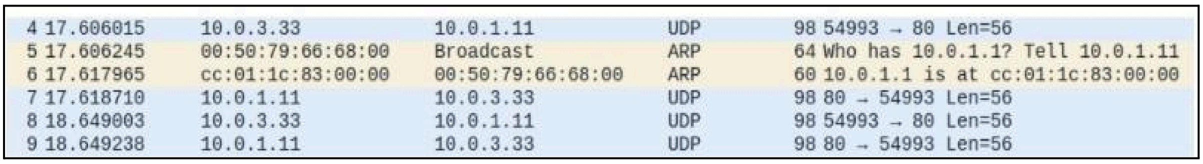
- Compare the amount of data transmitted in the TCP and the UDP data transfers.

PC1 -> PC2:



```
4 13.288172   00:50:79:66:68:00   Broadcast           ARP   64 Who has 10.0.1.1? Tell 10.0.1.11
5 13.294990   cc:01:1c:83:00:00   00:50:79:66:68:00   ARP   60 10.0.1.1 is at cc:01:1c:83:00:00
6 13.295851   10.0.1.11           10.0.3.33           UDP   98 43491 → 80 Len=56
7 13.335515   10.0.3.33           10.0.1.11           UDP   98 80 → 43491 Len=56
8 14.337101   10.0.1.11           10.0.3.33           UDP   98 43491 → 80 Len=56
9 14.364111   10.0.3.33           10.0.1.11           UDP   98 80 → 43491 Len=56
```

PC2 -> PC1 :



```
4 17.606015   10.0.3.33           10.0.1.11           UDP   98 54993 → 80 Len=56
5 17.606245   00:50:79:66:68:00   Broadcast           ARP   64 Who has 10.0.1.1? Tell 10.0.1.11
6 17.617965   cc:01:1c:83:00:00   00:50:79:66:68:00   ARP   60 10.0.1.1 is at cc:01:1c:83:00:00
7 17.618710   10.0.1.11           10.0.3.33           UDP   98 80 → 54993 Len=56
8 18.649003   10.0.3.33           10.0.1.11           UDP   98 54993 → 80 Len=56
9 18.649238   10.0.1.11           10.0.3.33           UDP   98 80 → 54993 Len=56
```

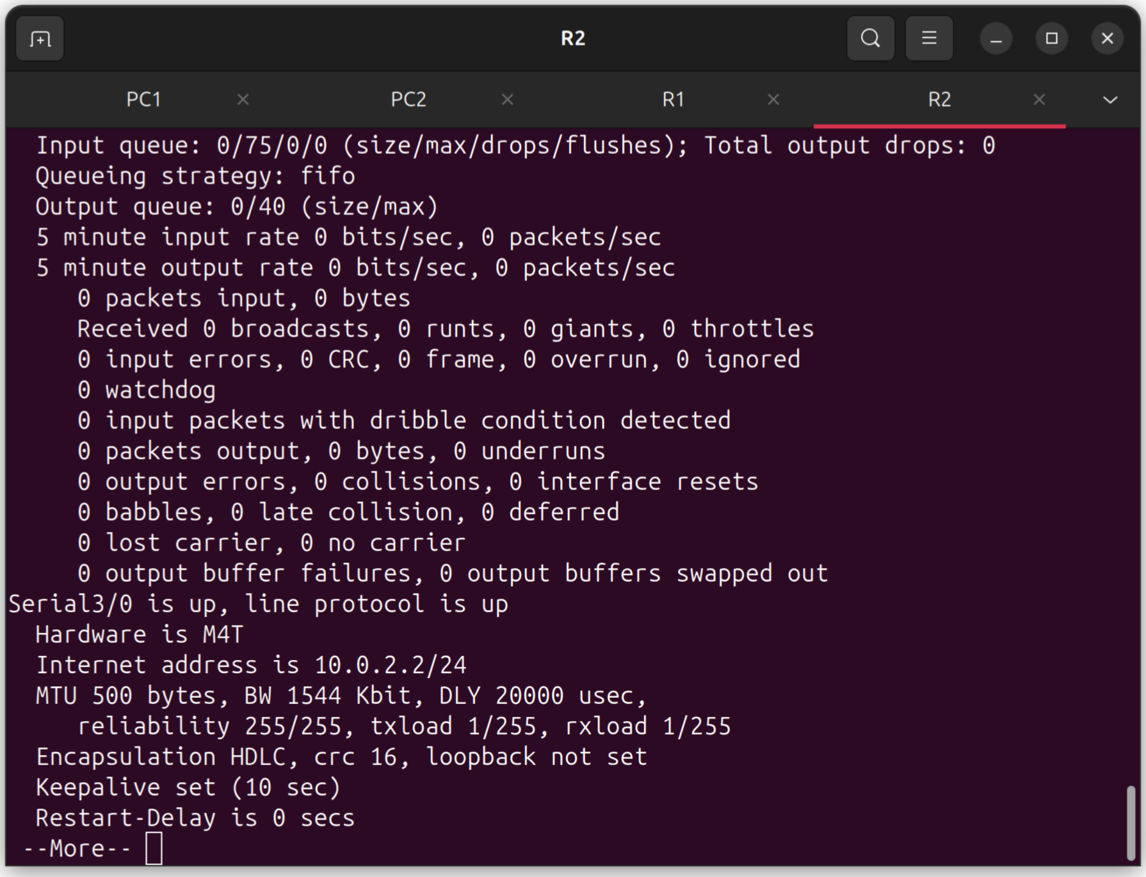
In our UDP capture, each frame has a total length of 98 bytes, of which 56 bytes are actual payload. This illustrates that in UDP, unlike TCP, there is no handshake or control overhead, so the payload-to-frame ratio is higher. In TCP, the total frame length often includes additional bytes for connection management, making the payload a smaller portion of the frame.

- Take the largest UDP segment and the largest TCP segment that you observed, and compare the amount of application data that is transmitted in the UDP segment and the TCP segment.

Comparing the largest UDP and TCP segments observed: the largest UDP segment has a total length of 98 bytes with 56 bytes of payload, while the largest TCP segment has a total length of 122 bytes with the same 56 bytes of payload. This shows that although both segments carry the same amount of application data, the TCP segment includes additional bytes for headers and control information, making TCP less efficient in terms of payload-to-total-frame ratio compared to UDP.

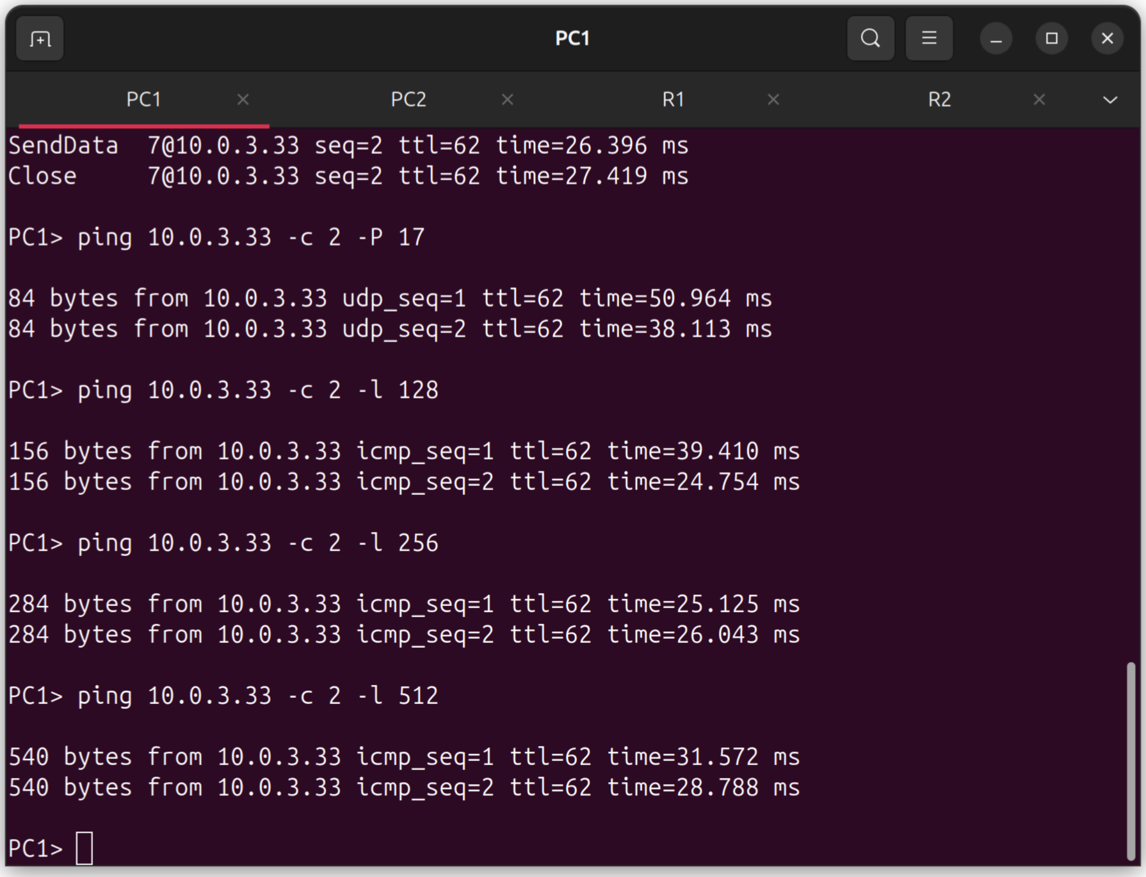
PART 2: IP Fragmentation

Exercise 2. IP fragmentation



```
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog
     0 input packets with dribble condition detected
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
Serial3/0 is up, line protocol is up
  Hardware is M4T
  Internet address is 10.0.2.2/24
  MTU 500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
 --More--
```

Tristan BRAU
Léo PHAV



```
SendData  7@10.0.3.33 seq=2 ttl=62 time=26.396 ms
Close     7@10.0.3.33 seq=2 ttl=62 time=27.419 ms

PC1> ping 10.0.3.33 -c 2 -P 17

84 bytes from 10.0.3.33 udp_seq=1 ttl=62 time=50.964 ms
84 bytes from 10.0.3.33 udp_seq=2 ttl=62 time=38.113 ms

PC1> ping 10.0.3.33 -c 2 -l 128

156 bytes from 10.0.3.33 icmp_seq=1 ttl=62 time=39.410 ms
156 bytes from 10.0.3.33 icmp_seq=2 ttl=62 time=24.754 ms

PC1> ping 10.0.3.33 -c 2 -l 256

284 bytes from 10.0.3.33 icmp_seq=1 ttl=62 time=25.125 ms
284 bytes from 10.0.3.33 icmp_seq=2 ttl=62 time=26.043 ms

PC1> ping 10.0.3.33 -c 2 -l 512

540 bytes from 10.0.3.33 icmp_seq=1 ttl=62 time=31.572 ms
540 bytes from 10.0.3.33 icmp_seq=2 ttl=62 time=28.788 ms

PC1> 
```

Lab Questions:

- Determine the exact packet size at which fragmentation occurs.

IP Header: 20 bytes

ICMP Header: 8 bytes

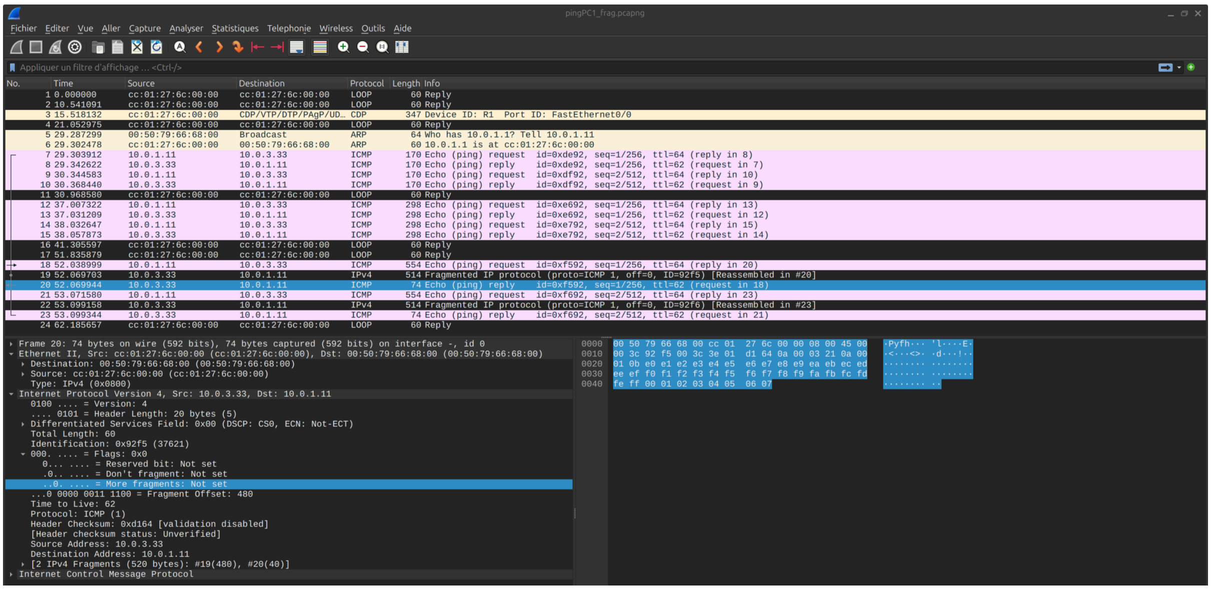
MTU: 500 bytes

$\Rightarrow Segment\ 1\ Payload\ Size\ =\ MTU\ -\ Overhead\ =\ 500\ -\ 28\ =\ 472\ bytes$

$\Rightarrow Segment\ 2\ Payload\ Size\ =\ 512\ -\ 472\ =\ 40\ bytes$

Tristan BRAU
Léo PHAV

- From the saved Wireshark data, select one IP packet that is fragmented. Look at the complete packet before fragmentation and include all fragments after fragmentation. For each fragment of this packet, determine the values of the fields in the IP header that are used for fragmentation (Identification, Fragment Offset, Don't Fragment Bit, More Fragments Bit).


pingPC1_frag.pcapng
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
Appliquer un filtre d'affichage … <Ctrl-/>

```
No.   Time       Source             Destination          Protocol Length Info
    1 0.000000   cc:01:27:6c:00:00  cc:01:27:6c:00:00    LOOP       60 Reply
    2 10.541091  cc:01:27:6c:00:00  cc:01:27:6c:00:00    LOOP       60 Reply
    3 15.518132  cc:01:27:6c:00:00  CDP/VTP/DTP/PAgP/UD… CDP       347 Device ID: R1  Port ID: FastEthernet0/0
    4 21.052975  cc:01:27:6c:00:00  cc:01:27:6c:00:00    LOOP       60 Reply
    5 29.287299  00:50:79:66:68:00  Broadcast            ARP        64 Who has 10.0.1.1? Tell 10.0.1.11
    6 29.302478  cc:01:27:6c:00:00  00:50:79:66:68:00    ARP        60 10.0.1.1 is at cc:01:27:6c:00:00
    7 29.303912  10.0.1.11          10.0.3.33            ICMP      170 Echo (ping) request  id=0xde92, seq=1/256, ttl=64 (reply in 8)
    8 29.342622  10.0.3.33          10.0.1.11            ICMP      170 Echo (ping) reply    id=0xde92, seq=1/256, ttl=62 (request in 7)
    9 30.344583  10.0.1.11          10.0.3.33            ICMP      170 Echo (ping) request  id=0xdf92, seq=2/512, ttl=64 (reply in 10)
   10 30.368440  10.0.3.33          10.0.1.11            ICMP      170 Echo (ping) reply    id=0xdf92, seq=2/512, ttl=62 (request in 9)
   11 30.968580  cc:01:27:6c:00:00  cc:01:27:6c:00:00    LOOP       60 Reply
   12 37.007322  10.0.1.11          10.0.3.33            ICMP      298 Echo (ping) request  id=0xe692, seq=1/256, ttl=64 (reply in 13)
   13 37.031209  10.0.3.33          10.0.1.11            ICMP      298 Echo (ping) reply    id=0xe692, seq=1/256, ttl=62 (request in 12)
   14 38.032647  10.0.1.11          10.0.3.33            ICMP      298 Echo (ping) request  id=0xe792, seq=2/512, ttl=64 (reply in 15)
   15 38.057873  10.0.3.33          10.0.1.11            ICMP      298 Echo (ping) reply    id=0xe792, seq=2/512, ttl=62 (request in 14)
   16 41.305597  cc:01:27:6c:00:00  cc:01:27:6c:00:00    LOOP       60 Reply
   17 51.835879  cc:01:27:6c:00:00  cc:01:27:6c:00:00    LOOP       60 Reply
   18 52.038999  10.0.1.11          10.0.3.33            ICMP      554 Echo (ping) request  id=0xf592, seq=1/256, ttl=64 (reply in 20)
   19 52.069703  10.0.3.33          10.0.1.11            IPv4      514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=92f5) [Reassembled in #20]
   20 52.069944  10.0.3.33          10.0.1.11            ICMP       74 Echo (ping) reply    id=0xf592, seq=1/256, ttl=62 (request in 18)
   21 53.071580  10.0.1.11          10.0.3.33            ICMP      554 Echo (ping) request  id=0xf692, seq=2/512, ttl=64 (reply in 23)
   22 53.099158  10.0.3.33          10.0.1.11            IPv4      514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=92f6) [Reassembled in #23]
   23 53.099344  10.0.3.33          10.0.1.11            ICMP       74 Echo (ping) reply    id=0xf692, seq=2/512, ttl=62 (request in 21)
   24 62.185657  cc:01:27:6c:00:00  cc:01:27:6c:00:00    LOOP       60 Reply
```

```
› Frame 20: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface -, id 0
˅ Ethernet II, Src: cc:01:27:6c:00:00 (cc:01:27:6c:00:00), Dst: 00:50:79:66:68:00 (00:50:79:66:68:00)
  › Destination: 00:50:79:66:68:00 (00:50:79:66:68:00)
  › Source: cc:01:27:6c:00:00 (cc:01:27:6c:00:00)
    Type: IPv4 (0x0800)
˅ Internet Protocol Version 4, Src: 10.0.3.33, Dst: 10.0.1.11
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  › Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x92f5 (37621)
  ˅ 000. .... = Flags: 0x0
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    ...0 0000 0011 1100 = Fragment Offset: 480
    Time to Live: 62
    Protocol: ICMP (1)
    Header Checksum: 0xd164 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.0.3.33
    Destination Address: 10.0.1.11
  › [2 IPv4 Fragments (520 bytes): #19(480), #20(40)]
› Internet Control Message Protocol
```

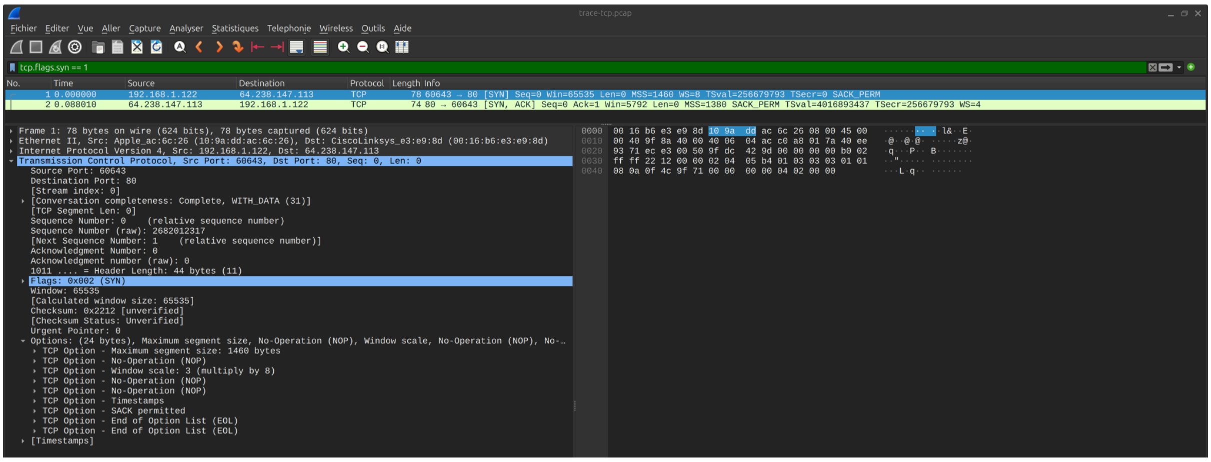
```
0000  00 50 79 66 68 00 cc 01  27 6c 00 00 08 00 45 00   ·Pyfh··· 'l····E·
0010  00 3c 92 f5 00 3c 3e 01  d1 64 0a 00 03 21 0a 00   ·<···<>· ·d···!··
0020  01 0b e0 e1 e2 e3 e4 e5  e6 e7 e8 e9 ea eb ec ed   ········ ········
0030  ee ef f0 f1 f2 f3 f4 f5  f6 f7 f8 f9 fa fb fc fd   ········ ········
0040  fe ff 00 01 02 03 04 05  06 07                     ········ ··
```

Both packets have the same identification (0x92f5) to ensure they belong to the same frame.
The "Don't fragment" bit is set to 0 in order to allow the fragmentation.
The first packet has a flag (bit to 1) "More fragment" because it has been fragmented in contrast to the last fragment.
Furthermore, the last fragment has an offset, equals to precedent segment payload size divided by 8, here it's 472/8 = 59.

Tristan BRAU
Léo PHAV

PART 3: TCP Protocol

Exercise 3(A): Connection Options



```
tcp.flags.syn == 1
```

```
No.  Time       Source          Destination     Protocol Length Info
   1 0.000000   192.168.1.122   64.238.147.113  TCP         78 60643 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=256679793 TSecr=0 SACK_PERM
   2 0.088010   64.238.147.113  192.168.1.122   TCP         74 80 → 60643 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1380 SACK_PERM TSval=4016893437 TSecr=256679793 WS=4
```

```
› Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
› Ethernet II, Src: Apple_ac:6c:26 (10:9a:dd:ac:6c:26), Dst: CiscoLinksys_e3:e9:8d (00:16:b6:e3:e9:8d)
› Internet Protocol Version 4, Src: 192.168.1.122, Dst: 64.238.147.113
⌄ Transmission Control Protocol, Src Port: 60643, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 60643
    Destination Port: 80
    [Stream index: 0]
  › [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 2682012317
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1011 .... = Header Length: 44 bytes (11)
  › Flags: 0x002 (SYN)
    Window: 65535
    [Calculated window size: 65535]
    Checksum: 0x2212 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ⌄ Options: (24 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-…
    › TCP Option - Maximum segment size: 1460 bytes
    › TCP Option - No-Operation (NOP)
    › TCP Option - Window scale: 3 (multiply by 8)
    › TCP Option - No-Operation (NOP)
    › TCP Option - No-Operation (NOP)
    › TCP Option - Timestamps
    › TCP Option - SACK permitted
    › TCP Option - End of Option List (EOL)
    › TCP Option - End of Option List (EOL)
  › [Timestamps]
```

```
0000  00 16 b6 e3 e9 8d 10 9a  dd ac 6c 26 08 00 45 00   ········ ··l&··E·
0010  00 40 9f 8a 40 00 40 06  04 ac c0 a8 01 7a 40 ee   ·@··@·@· ·····z@·
0020  93 71 ec e3 00 50 9f dc  42 9d 00 00 00 00 b0 02   ·q···P·· B·······
0030  ff ff 22 12 00 00 02 04  05 b4 01 03 03 03 01 01   ··"····· ········
0040  08 0a 0f 4c 9f 71 00 00  00 00 04 02 00 00        ···L·q·· ······
```

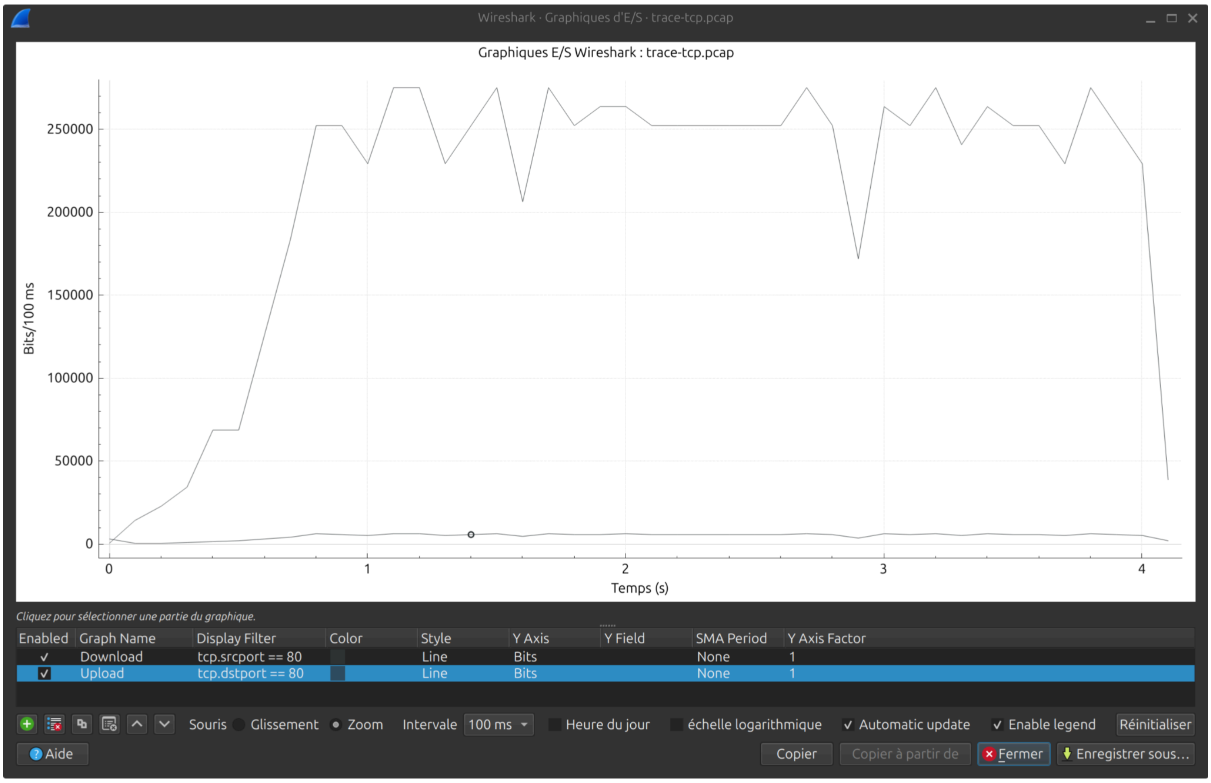
Lab Questions:

• What TCP Options are carried on the SYN packets for the trace provided

There are 4 options in the trace:

- Maximum segment size
- Window scale
- Timestamps
- SACK permitted

Exercise 3(B): TCP Data Transfer



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
✓	Download	tcp.srcport == 80		Line	Bits		None	1
☑	Upload	tcp.dstport == 80		Line	Bits		None	1

Intervale 100 ms

○ Souris ● Zoom ☐ Heure du jour ☐ échelle logarithmique ✓ Automatic update ✓ Enable legend

Tristan BRAU
Léo PHAV

Lab Questions:

• Identify the Slow Start, Additive Increase, Multiplicative Decrease events in the download traffic curve.

Slow Start	0 à 0.4s	Exponential increase in data rate.
Additive Increase	0.4 à 1.0s	Gradual/Linear increase in data rate.
Multiplicative Decrease	Multiple points: 1.5s, 2.5s, 3.5s	Sudden, sharp drops in data rate, indicating congestion detection (packet loss).

• What is the rough data rate in the download direction in packets/second and bits/second once the TCP connection is running “well”?

The stabilized maximum download rate is ≈ 250,000 Bits/100 ms
Rate in Bits/s: 250,000 Bits/100 ms×10 = 2.5 Mbps

• What percentage of this download rate is content? Show your calculation. To find out, look at a typical download packet; there should be many similar, large download packets. You can see how long it is, and how many bytes of TCP payload it contains.

Typical Packet Size: 1514 bytes
Total Headers:

- Ethernet Header: 14 bytes
- IP Header: 20 bytes
- TCP Header: 32 bytes (with options)
- **Total Overhead:** 14+20+32=66 bytes

TCP Payload: Total Size − Total Overhead = 1514−66 = 1448 bytes
$Content\ Percentage\ = \frac{TCP\ Payload}{Total\ Packet\ Size} \times\ 100\ = \frac{1448}{1514} \times\ 100\ =\ 95.6\%$

• What is the rough data rate in the upload direction in packets/second and bits/second due to the ACK packets?

The upload line is very low, ≈ 1,500 Bits/100 ms
Rate in Bits/s: 1,500 Bits/100 ms×10 = 15 Kbps

• If the most recently received TCP segment from the server has a sequence number of X, then what ACK number does the next transmitted TCP segment carry?

If the most recently received TCP segment from the server has a sequence number of **X**, the next transmitted TCP segment from the client will carry the ACK number:
$ACK\ Number\ =\ X\ +\ Received\ Segment\ Payload\ Size$