Started on	Tuesday, 10 November 2020, 16:32
State	Finished
Completed on	Wednesday, 11 November 2020, 08:00
Time taken	15 hours 28 mins
Feedback	Your overall score for this part is 100%. Congratulations ! The exact grade as well as details for each question will be made available as soon as the submission period for the quiz closes.

Question **1**

Complete

Marked out of 1.00

Flag question

Lab3 Part2 Question 1:

How many messages do you receive when you send the command

```
CMD short:0
```

?

15

How many messages do you receive when you send the command

```
 CMD short:1
```

?

15

Watch the terms used in the question.

We ask you to count PMU messages. What is a PMU message ?

Question **2**

Complete

Marked out of 1.00

Flag question

Lab3 Part2 Question 2:

In Wireshark, how many packets FROM the server with a PAYLOAD length STRICTLY positive do you observe:

With the command

```
CMD_short:1
```

?

15

With the command

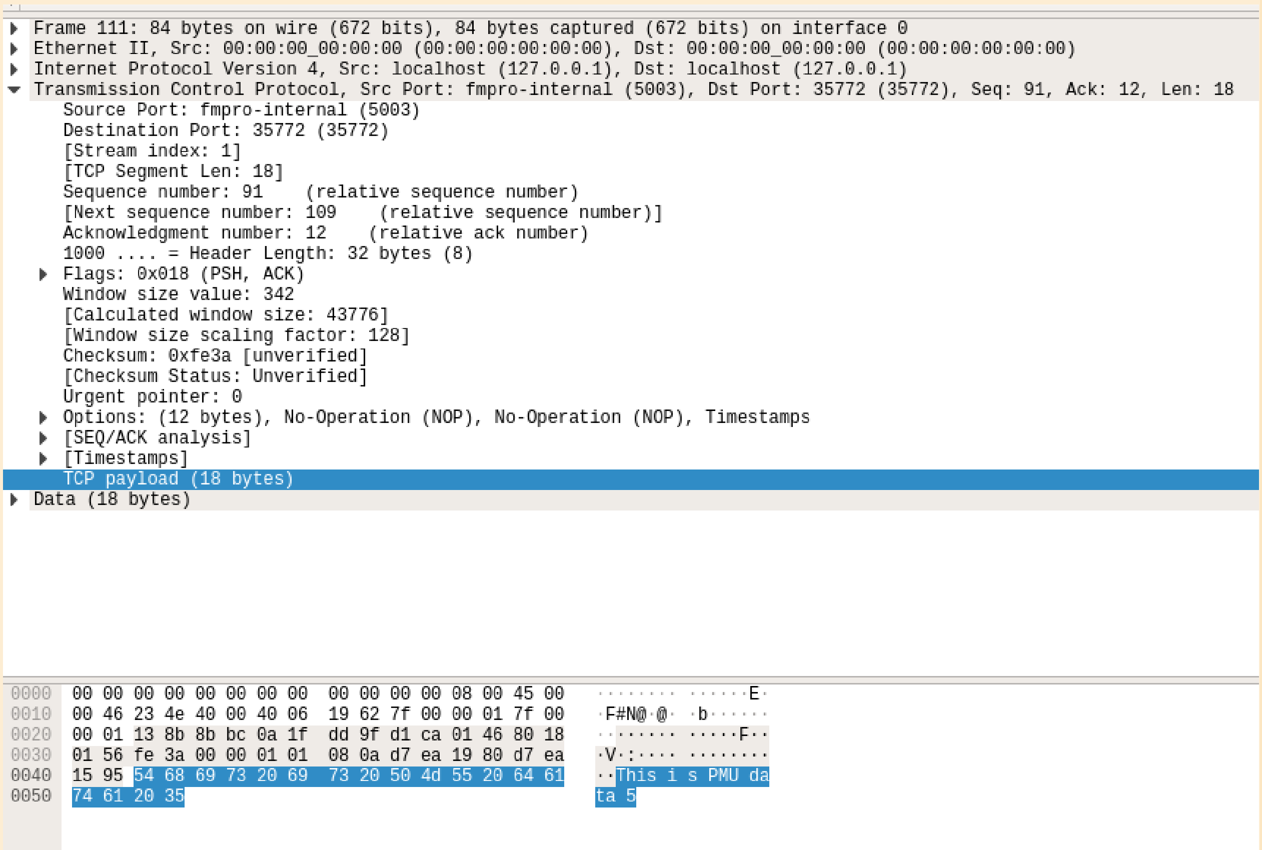
`CMD_short:0`

?

11

We specifically ask you to consider ONLY the packets (=TCP segments) that have a non-empty payload.

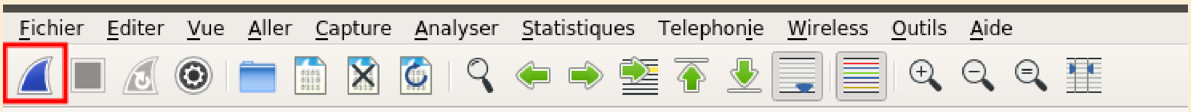
In Wireshark, once you select a packet, in the packet details, expand the tree corresponding to the TCP protocol, and look for the payload field to get its length.



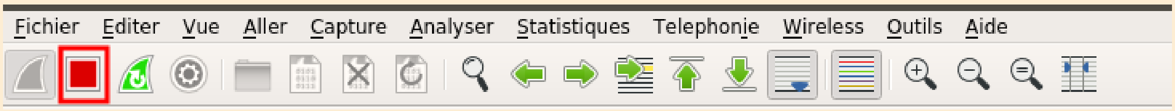
```
▶ Frame 111: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)
▼ Transmission Control Protocol, Src Port: fmpro-internal (5003), Dst Port: 35772 (35772), Seq: 91, Ack: 12, Len: 18
     Source Port: fmpro-internal (5003)
     Destination Port: 35772 (35772)
     [Stream index: 1]
     [TCP Segment Len: 18]
     Sequence number: 91    (relative sequence number)
     [Next sequence number: 109    (relative sequence number)]
     Acknowledgment number: 12    (relative ack number)
     1000 .... = Header Length: 32 bytes (8)
   ▶ Flags: 0x018 (PSH, ACK)
     Window size value: 342
     [Calculated window size: 43776]
     [Window size scaling factor: 128]
     Checksum: 0xfe3a [unverified]
     [Checksum Status: Unverified]
     Urgent pointer: 0
   ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
   ▶ [SEQ/ACK analysis]
   ▶ [Timestamps]
     TCP payload (18 bytes)
▶ Data (18 bytes)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 46 23 4e 40 00 40 06  19 62 7f 00 00 01 7f 00   ·F#N@·@· ·b······
0020  00 01 13 8b 8b bc 0a 1f  dd 9f d1 ca 01 46 80 18   ········ ·····F··
0030  01 56 fe 3a 00 00 01 01  08 0a d7 ea 19 80 d7 ea   ·V·:···· ········
0040  15 95 54 68 69 73 20 69  73 20 50 4d 55 20 64 61   ··This i s PMU da
0050  74 61 20 35                                        ta 5
```

You have lot of packets displayed in Wireshark and you don't really know where are the packets you're looking for ? Here are several tips for you:

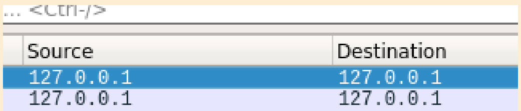
TIP1: Try to launch the Wireshark capture JUST BEFORE you launch the client scrip.



And to stop it JUST AFTER the client script is completed (with CMD_sort:0, it's very fast!)



TIP2: In the list of packets, IPs displayed in the "Source"/"Destination" columns are difficult to interpret (learning by heart all the IPv4 addresses is not mandatory for networking students).

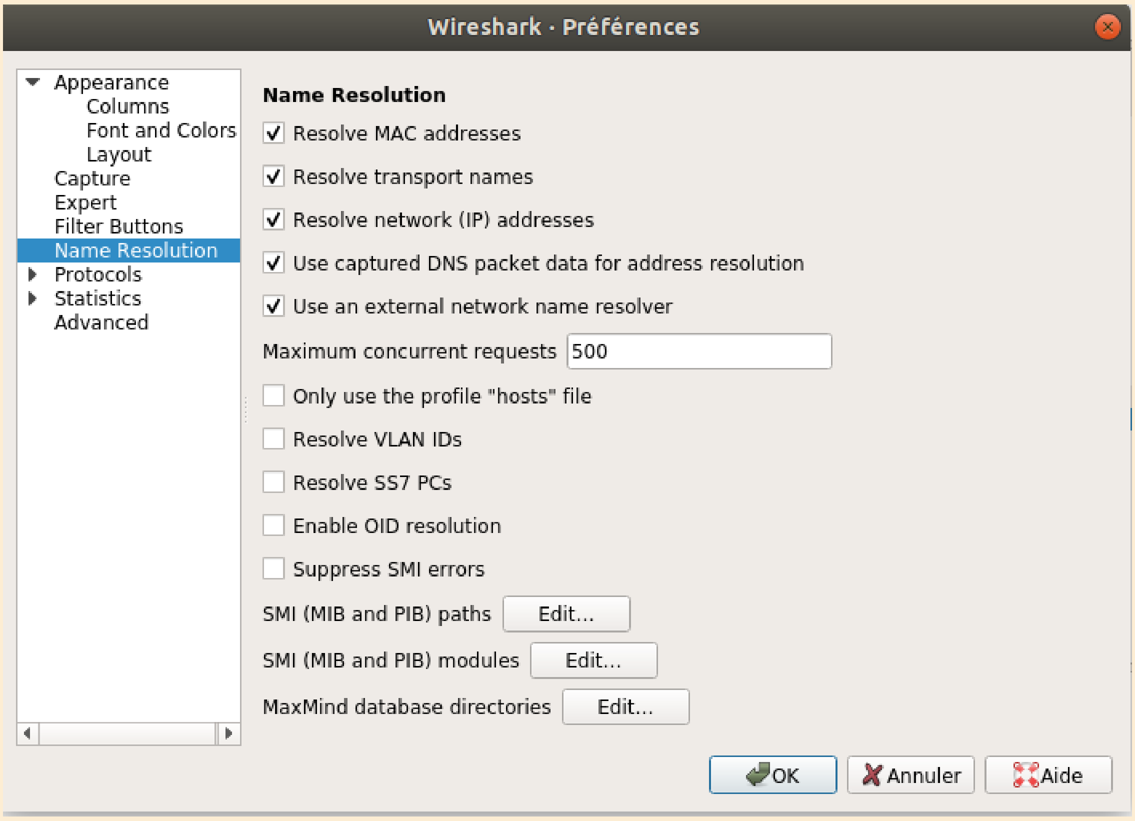


... <Ctrl-/>

Source	Destination
127.0.0.1	127.0.0.1
127.0.0.1	127.0.0.1

We can ask Wireshark to resolve for us the IP addresses into (domain) names.

Open the preferences window (Edit>Preferences). In the "Name Resolution" section, make sure the boxes are checked as follows:



Name Resolution

☑ Resolve MAC addresses

☑ Resolve transport names

☑ Resolve network (IP) addresses

☑ Use captured DNS packet data for address resolution

☑ Use an external network name resolver

Maximum concurrent requests: 500

□ Only use the profile "hosts" file

□ Resolve VLAN IDs

□ Resolve SS7 PCs

□ Enable OID resolution

□ Suppress SMI errors

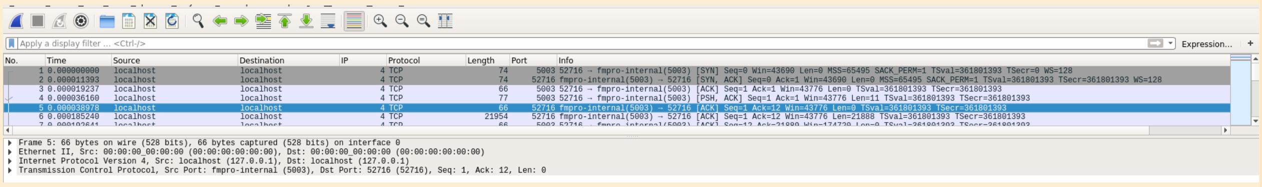
SMI (MIB and PIB) paths Edit...

SMI (MIB and PIB) modules Edit...

MaxMind database directories Edit...

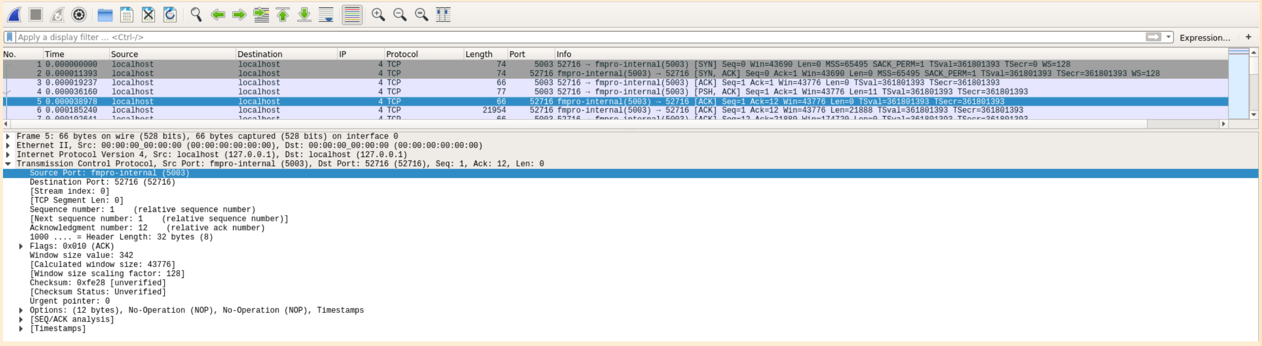
And now the IPs should be resolved, which will help you identify traffic coming from tcpip.epfl.ch (you might need to restart the capture).

TIP3: Filters ! Filters in Wireshark are a powerful tool that you need to master. However, you are not required to learn by heart all the names of the filters. So here is a tip for you to identify the filter you need. Let's assume we want to display only the packets sent FROM the port 5003. The first step is to find and select one of these packets.



```
No.   Time         Source      Destination    IP  Protocol  Length   Port   Info
    1 0.000000000  localhost   localhost       4  TCP           74    5003 52716 → fmpro-internal(5003) [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=361801393 TSecr=0 WS=128
    2 0.000011393  localhost   localhost       4  TCP           74   52716 fmpro-internal(5003) → 52716 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=361801393 TSecr=361801393 WS=128
    3 0.000019237  localhost   localhost       4  TCP           66    5003 52716 → fmpro-internal(5003) [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=361801393 TSecr=361801393
    4 0.000036160  localhost   localhost       4  TCP           77    5003 52716 → fmpro-internal(5003) [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=11 TSval=361801393 TSecr=361801393
    5 0.000038978  localhost   localhost       4  TCP           66   52716 fmpro-internal(5003) → 52716 [ACK] Seq=1 Ack=12 Win=43776 Len=0 TSval=361801393 TSecr=361801393
    6 0.000185240  localhost   localhost       4  TCP        21954   52716 fmpro-internal(5003) → 52716 [ACK] Seq=1 Ack=12 Win=43776 Len=21888 TSval=361801393 TSecr=361801393
    7 0.000192641  localhost   localhost       4  TCP           66    5003 52716 → fmpro-internal(5003) [ACK] Seq=12 Ack=21889 Win=174720 Len=0 TSval=361801393 TSecr=361801393

▸ Frame 5: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)
▸ Transmission Control Protocol, Src Port: fmpro-internal (5003), Dst Port: 52716 (52716), Seq: 1, Ack: 12, Len: 0
```

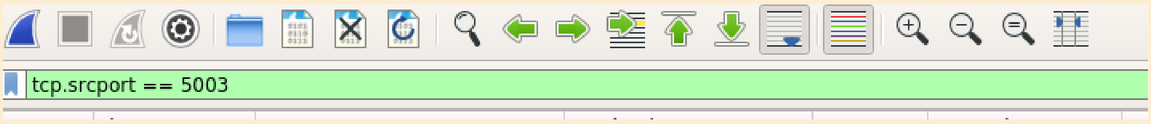
In the above capture, we found and selected one of the packets sent by the server. Then, if we want to filter through the port number, we know that port numbers are located in the TCP header, so we expand the TCP field in the packet inspector.



```
No.   Time         Source      Destination    IP  Protocol  Length   Port   Info
    1 0.000000000  localhost   localhost       4  TCP           74    5003 52716 → fmpro-internal(5003) [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=361801393 TSecr=0 WS=128
    2 0.000011393  localhost   localhost       4  TCP           74   52716 fmpro-internal(5003) → 52716 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=361801393 TSecr=361801393 WS=128
    3 0.000019237  localhost   localhost       4  TCP           66    5003 52716 → fmpro-internal(5003) [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=361801393 TSecr=361801393
    4 0.000036160  localhost   localhost       4  TCP           77    5003 52716 → fmpro-internal(5003) [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=11 TSval=361801393 TSecr=361801393
    5 0.000038978  localhost   localhost       4  TCP           66   52716 fmpro-internal(5003) → 52716 [ACK] Seq=1 Ack=12 Win=43776 Len=0 TSval=361801393 TSecr=361801393
    6 0.000185240  localhost   localhost       4  TCP        21954   52716 fmpro-internal(5003) → 52716 [ACK] Seq=1 Ack=12 Win=43776 Len=21888 TSval=361801393 TSecr=361801393
    7 0.000192641  localhost   localhost       4  TCP           66    5003 52716 → fmpro-internal(5003) [ACK] Seq=12 Ack=21889 Win=174720 Len=0 TSval=361801393 TSecr=361801393

▸ Frame 5: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)
▾ Transmission Control Protocol, Src Port: fmpro-internal (5003), Dst Port: 52716 (52716), Seq: 1, Ack: 12, Len: 0
    Source Port: fmpro-internal (5003)
    Destination Port: 52716 (52716)
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 12    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 342
    [Calculated window size: 43776]
    [Window size scaling factor: 128]
    Checksum: 0xfe28 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
```

In the above capture, we identified the "source port" field. Now, use a right click, select "Prepare a filter" and "Selected". In the case you already had a filter prepared in the filter field, you can choose to combine the current filter using one of the option "...and selected", "...or selected", etc.

This will create for you a filter in the filter field. Notice the green highlighting that indicates a correct format. Your cursor will also be placed in the filter field, and with the cursor IN the filter field, you need to press ENTER to apply the filter.



```
tcp.srcport == 5003
```

Question **3**

Complete

Marked out of 3.00

Flag question

Lab3 Part2 Question 3:

The goal of this question is to explain the observations made in the two previous answers.

With

`CMD_short:1`

, you receive exactly 1 message per packet .

With

`CMD_short:0`

, you receive sometimes more than 1 message per packet .

If we take a look at the server code, we find the following line which is executed whatever the

`CMD_short`

command it receives:

```
c.send(message.encode())
```

where:

- `message`
  is one of the PMU message you identified in Question 1
- `c`
  is a socket provided by the
  `accept()`
  call when your PDC client connects to the PMU. See part 1.2 as well as the lecture.

The

`send()`

call made by the server's application:

○ asks the server OS to create a TCP segment containing
`message`
and to send it via the connection to the client

● puts
`message`
in a queue and asks the server OS to send the message (via TCP), but without specifying how

○ puts
`message`
in a queue but asks the server OS to wait a bit before sending the message (via TCP)

. The TCP segment is created by the server OS

when the server OS decides it, according to some internal algorithm

.

Usually, the chosen algorithm is the

Nagle

's algorithm.

Check the lecture:

Question **4**

Complete

Marked out of 2.00

Flag question

Lab3 Part2 Question 4:

From a programmer's perspective, how can your application be sure it received all the messages from the PMU server ?

○ Your application should catch the `SocketNoMoreDataException` exception. When raised, it means that the socket has no more data to read

○ Your application should wait 30 seconds. If more than 30 seconds are elapsed since the last byte has been received, it means the server has closed the connection and has no more data to send

● Your application should test the value returned by `recv()` . When this value has a `False` boolean representation, it means that the connection has been orderly closed by the remote end and there is no more data to be read.

○ Your application should wait for the user to terminate the program (`CTRL+C`).

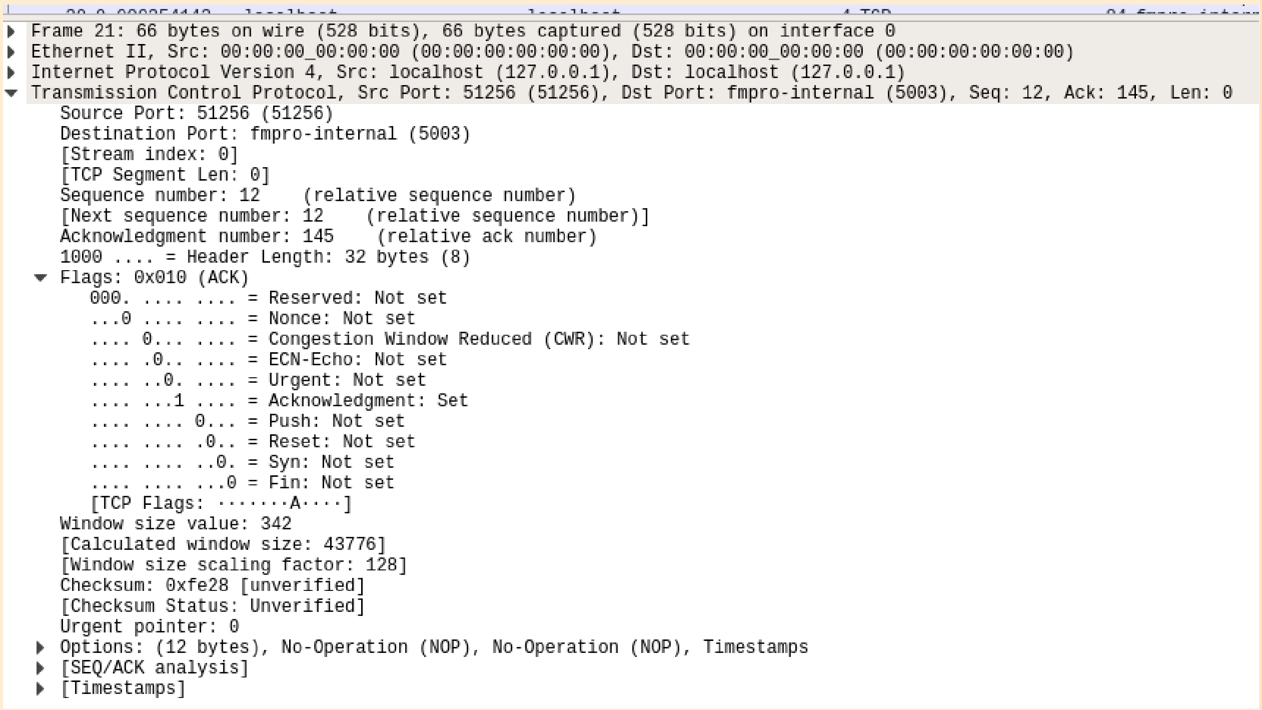
○ Your application can never be sure to have received all the data, packets might have been lost in the network.

From a network engineer's perspective, which of the following flags is set on the TCP header when the sender informs that it closes the connection and has nothing more to send ?

Answer: Fin

For the first part of the question, refer to Question6 in part 1.2 of the lab.

For the second part of the question, in Wireshark, you can see the flags by extending the corresponding field in the packet details:



```
▶ Frame 21: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)
▼ Transmission Control Protocol, Src Port: 51256 (51256), Dst Port: fmpro-internal (5003), Seq: 12, Ack: 145, Len: 0
    Source Port: 51256 (51256)
    Destination Port: fmpro-internal (5003)
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 12    (relative sequence number)
    [Next sequence number: 12    (relative sequence number)]
    Acknowledgment number: 145    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▼ Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·······A····]
    Window size value: 342
    [Calculated window size: 43776]
    [Window size scaling factor: 128]
    Checksum: 0xfe28 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
  ▶ [Timestamps]
```

Question **5**

Complete

Marked out of 1.00

Flag question

Lab3 Part2 Question 5:

In Wireshark, how may packets with a NON EMPTY payload have you seen coming from the PMU TCP server ?

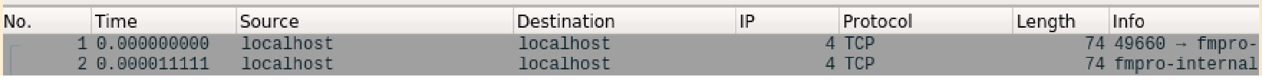
Answer: 16

See help of Question2.

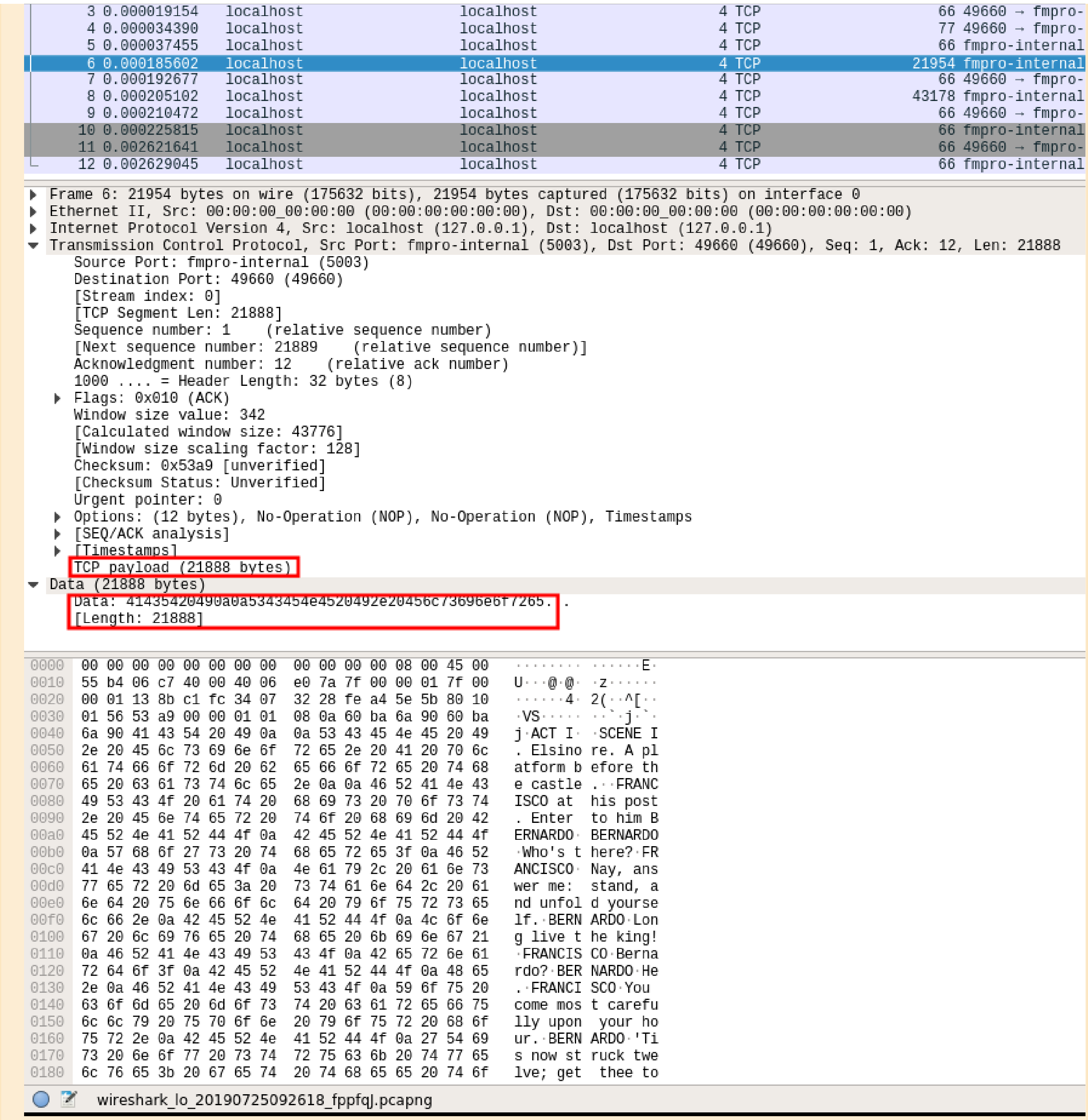
Here is an additional tip when using Wireshark:

You potentially have many packets with a non-empty payload and you don't want to count all of them ? In this case, we can use Wireshark filters !

First, let's have a look at one packet with a non empty payload, we extend the TCP and the "Data" tree in the packet detail field:



```
No.  Time         Source      Destination   IP  Protocol  Length  Info
   1 0.000000000  localhost   localhost      4  TCP           74  49660 → fmpro-
   2 0.000011111  localhost   localhost      4  TCP           74  fmpro-internal
```



```
   3 0.000019154   localhost              localhost                 4 TCP                66 49660 → fmpro-
   4 0.000034390   localhost              localhost                 4 TCP                77 49660 → fmpro-
   5 0.000037455   localhost              localhost                 4 TCP                66 fmpro-internal
   6 0.000185602   localhost              localhost                 4 TCP             21954 fmpro-internal
   7 0.000192677   localhost              localhost                 4 TCP                66 49660 → fmpro-
   8 0.000205102   localhost              localhost                 4 TCP             43178 fmpro-internal
   9 0.000210472   localhost              localhost                 4 TCP                66 49660 → fmpro-
  10 0.000225815   localhost              localhost                 4 TCP                66 fmpro-internal
  11 0.002621641   localhost              localhost                 4 TCP                66 49660 → fmpro-
  12 0.002629045   localhost              localhost                 4 TCP                66 fmpro-internal
```

```
▶ Frame 6: 21954 bytes on wire (175632 bits), 21954 bytes captured (175632 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)
▼ Transmission Control Protocol, Src Port: fmpro-internal (5003), Dst Port: 49660 (49660), Seq: 1, Ack: 12, Len: 21888
    Source Port: fmpro-internal (5003)
    Destination Port: 49660 (49660)
    [Stream index: 0]
    [TCP Segment Len: 21888]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 21889    (relative sequence number)]
    Acknowledgment number: 12    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▶ Flags: 0x010 (ACK)
    Window size value: 342
    [Calculated window size: 43776]
    [Window size scaling factor: 128]
    Checksum: 0x53a9 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
  ▶ [Timestamps]
    TCP payload (21888 bytes)
▼ Data (21888 bytes)
    Data: 414354204490a0a5343454e4520492e20456c73696e6f7265...
    [Length: 21888]
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  55 b4 06 c7 40 00 40 06  e0 7a 7f 00 00 01 7f 00   U···@·@· ·z······
0020  00 01 13 8b c1 fc 34 07  32 28 fe a4 5e 5b 80 10   ······4· 2(··^[··
0030  01 56 53 a9 00 00 01 01  08 0a 60 ba 6a 90 60 ba   ·VS····· ··`·j·`·
0040  6a 90 41 43 54 20 49 0a  0a 53 43 45 4e 45 20 49   j·ACT I· ·SCENE I
0050  2e 20 45 6c 73 69 6e 6f  72 65 2e 20 41 20 70 6c   . Elsino re. A pl
0060  61 74 66 6f 72 6d 20 62  65 66 6f 72 65 20 74 68   atform b efore th
0070  65 20 63 61 73 74 6c 65  2e 0a 0a 46 52 41 4e 43   e castle .··FRANC
0080  49 53 43 4f 20 61 74 20  68 69 73 20 70 6f 73 74   ISCO at  his post
0090  2e 20 45 6e 74 65 72 20  74 6f 20 68 69 6d 20 42   . Enter  to him B
00a0  45 52 4e 41 52 44 4f 0a  42 45 52 4e 41 52 44 4f   ERNARDO· BERNARDO
00b0  0a 57 68 6f 27 73 20 74  68 65 72 65 3f 0a 46 52   ·Who's t here?·FR
00c0  41 4e 43 49 53 43 4f 0a  4e 61 79 2c 20 61 6e 73   ANCISCO· Nay, ans
00d0  77 65 72 20 6d 65 3a 20  73 74 61 6e 64 2c 20 61   wer me:  stand, a
00e0  6e 64 20 75 6e 66 6f 6c  64 20 79 6f 75 72 73 65   nd unfol d yourse
00f0  6c 66 2e 0a 42 45 52 4e  41 52 44 4f 0a 4c 6f 6e   lf.·BERN ARDO·Lon
0100  67 20 6c 69 76 65 20 74  68 65 20 6b 69 6e 67 21   g live t he king!
0110  0a 46 52 41 4e 43 49 53  43 4f 0a 42 65 72 6e 61   ·FRANCIS CO·Berna
0120  72 64 6f 3f 0a 42 45 52  4e 41 52 44 4f 0a 48 65   rdo?·BER NARDO·He
0130  2e 0a 46 52 41 4e 43 49  53 43 4f 0a 59 6f 75 20   .·FRANCI SCO·You 
0140  63 6f 6d 65 20 6d 6f 73  74 20 63 61 72 65 66 75   come mos t carefu
0150  6c 6c 79 20 75 70 6f 6e  20 79 6f 75 72 20 68 6f   lly upon  your ho
0160  75 72 2e 0a 42 45 52 4e  41 52 44 4f 0a 27 54 69   ur.·BERN ARDO·'Ti
0170  73 20 6e 6f 77 20 73 74  72 75 63 6b 20 74 77 65   s now st ruck twe
0180  6c 76 65 3b 20 67 65 74  20 74 68 65 65 20 74 6f   lve; get  thee to
```

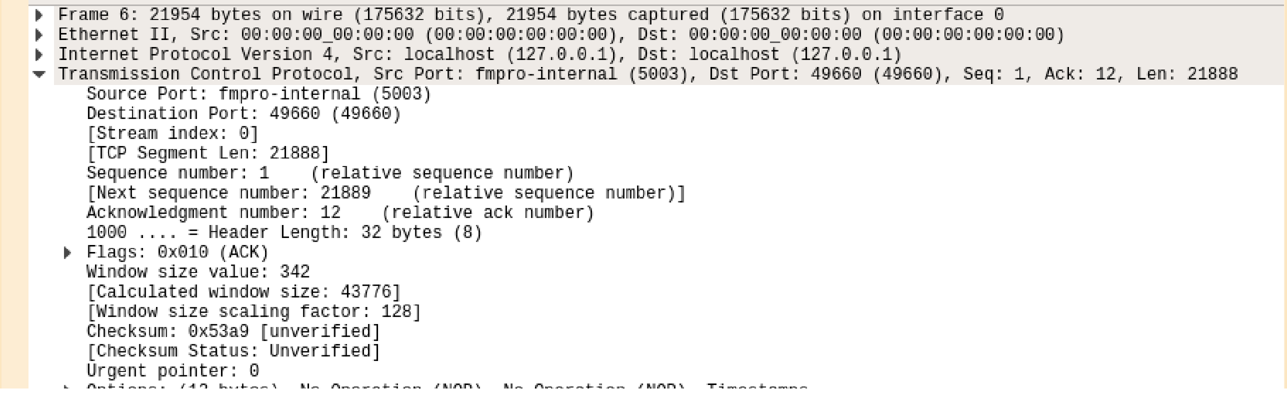
wireshark_lo_20190725092618_fppfqJ.pcapng

Three fields are of interest in terms of payload (see red boxes above):

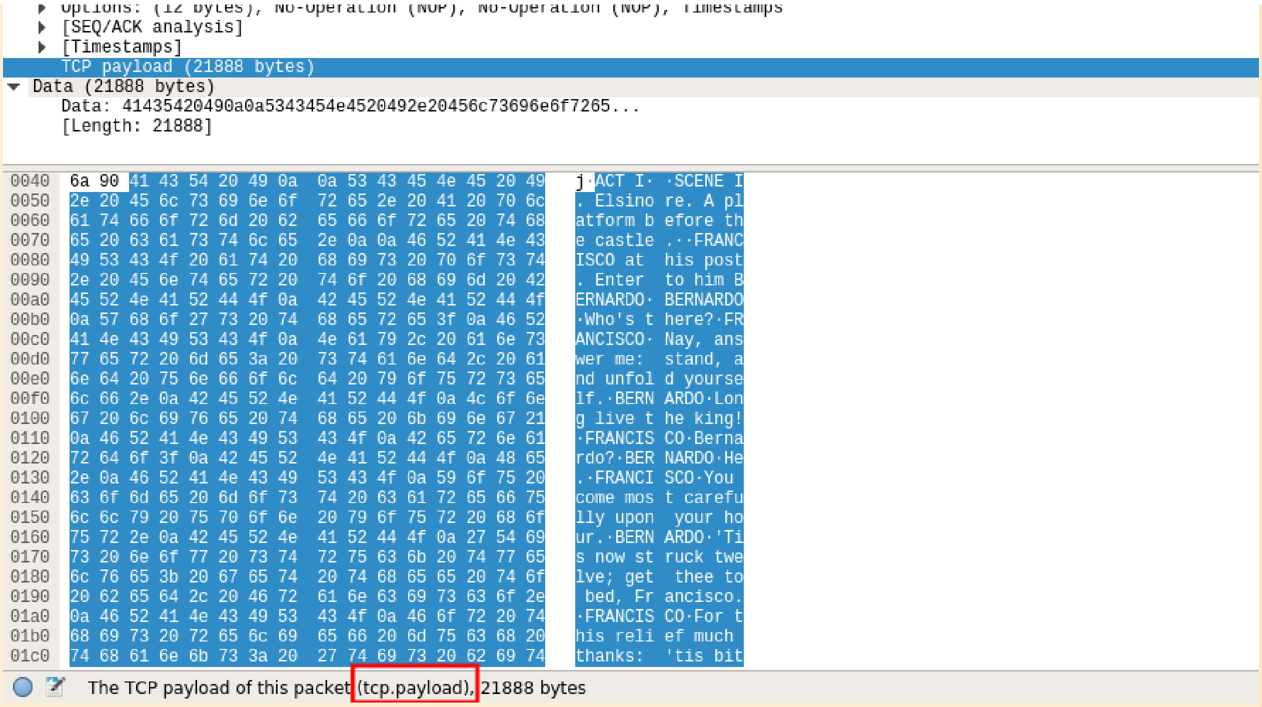
- The "TCP payload" field in the section "TCP"
- The "Data" field in the section "Data"
- The "Length" field in the section "Data". Note that the "Length" field is shown in "[]" brackets.

Brackets indicate a value computed by Wireshark but not directly readable in the packet. It usually concerns length (Wireshark counts the bytes for you), sequence numbers, Acknowledgment analysis, etc.

Now let's retrieve the name of the three fields (rather than their DISPLAY name). If you select one of the field, its name will be prompted at the bottom of the Wireshark window:



```
▶ Frame 6: 21954 bytes on wire (175632 bits), 21954 bytes captured (175632 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)
▼ Transmission Control Protocol, Src Port: fmpro-internal (5003), Dst Port: 49660 (49660), Seq: 1, Ack: 12, Len: 21888
    Source Port: fmpro-internal (5003)
    Destination Port: 49660 (49660)
    [Stream index: 0]
    [TCP Segment Len: 21888]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 21889    (relative sequence number)]
    Acknowledgment number: 12    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▶ Flags: 0x010 (ACK)
    Window size value: 342
    [Calculated window size: 43776]
    [Window size scaling factor: 128]
    Checksum: 0x53a9 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
```



```
    ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▸ [SEQ/ACK analysis]
    ▸ [Timestamps]
      TCP payload (21888 bytes)
▾ Data (21888 bytes)
      Data: 41435420490a0a5343454e4520492e20456c73696e6f7265...
      [Length: 21888]

0040  6a 90 41 43 54 20 49 0a  0a 53 43 45 4e 45 20 49   j·ACT I· ·SCENE I
0050  2e 20 45 6c 73 69 6e 6f  72 65 2e 20 41 20 70 6c   . Elsino re. A pl
0060  61 74 66 6f 72 6d 20 62  65 66 6f 72 65 20 74 68   atform b efore th
0070  65 20 63 61 73 74 6c 65  2e 0a 0a 46 52 41 4e 43   e castle .··FRANC
0080  49 53 43 4f 20 61 74 20  68 69 73 20 70 6f 73 74   ISCO at  his post
0090  2e 20 45 6e 74 65 72 20  74 6f 20 68 69 6d 20 42   . Enter  to him B
00a0  45 52 4e 41 52 44 4f 0a  42 45 52 4e 41 52 44 4f   ERNARDO· BERNARDO
00b0  0a 57 68 6f 27 73 20 74  68 65 72 65 3f 0a 46 52   ·Who's t here?·FR
00c0  41 4e 43 49 53 43 4f 0a  4e 61 79 2c 20 61 6e 73   ANCISCO· Nay, ans
00d0  77 65 72 20 6d 65 3a 20  73 74 61 6e 64 2c 20 61   wer me:  stand, a
00e0  6e 64 20 75 6e 66 6f 6c  64 20 79 6f 75 72 73 65   nd unfol d yourse
00f0  6c 66 2e 0a 42 45 52 4e  41 52 44 4f 0a 4c 6f 6e   lf.·BERN ARDO·Lon
0100  67 20 6c 69 76 65 20 74  68 65 20 6b 69 6e 67 21   g live t he king!
0110  0a 46 52 41 4e 43 49 53  43 4f 0a 42 65 72 6e 61   ·FRANCIS CO·Berna
0120  72 64 6f 3f 0a 42 45 52  4e 41 52 44 4f 0a 48 65   rdo?·BER NARDO·He
0130  2e 0a 46 52 41 4e 43 49  53 43 4f 0a 59 6f 75 20   .·FRANCI SCO·You
0140  63 6f 6d 65 20 6d 6f 73  74 20 63 61 72 65 66 75   come mos t carefu
0150  6c 6c 79 20 75 70 6f 6e  20 79 6f 75 72 20 68 6f   lly upon  your ho
0160  75 72 2e 0a 42 45 52 4e  41 52 44 4f 0a 27 54 69   ur.·BERN ARDO·'Ti
0170  73 20 6e 6f 77 20 73 74  72 75 63 6b 20 74 77 65   s now st ruck twe
0180  6c 76 65 3b 20 67 65 74  20 74 68 65 65 20 74 6f   lve; get  thee to
0190  20 62 65 64 2c 20 46 72  61 6e 63 69 73 63 6f 2e    bed, Fr ancisco.
01a0  0a 46 52 41 4e 43 49 53  43 4f 0a 46 6f 72 20 74   ·FRANCIS CO·For t
01b0  68 69 73 20 72 65 6c 69  65 66 20 6d 75 63 68 20   his reli ef much
01c0  74 68 61 6e 6b 73 3a 20  27 74 69 73 20 62 69 74   thanks:  'tis bit
```

The TCP payload of this packet (tcp.payload), 21888 bytes

Now our goal is to create a DISPLAY FILTER that displays only the packets with a non empty payload, using one of the three options.

What to write in the display filter and which of the three fields to use ?

TCP Payload and Data fields hold the (binary) value of the payload while Data Length equals their length.

At first glance, Data Length could be seen as easier to use, and our display filter would look like "We want the data length to be strictly positive", ie:

```
data.len&gt;0
```

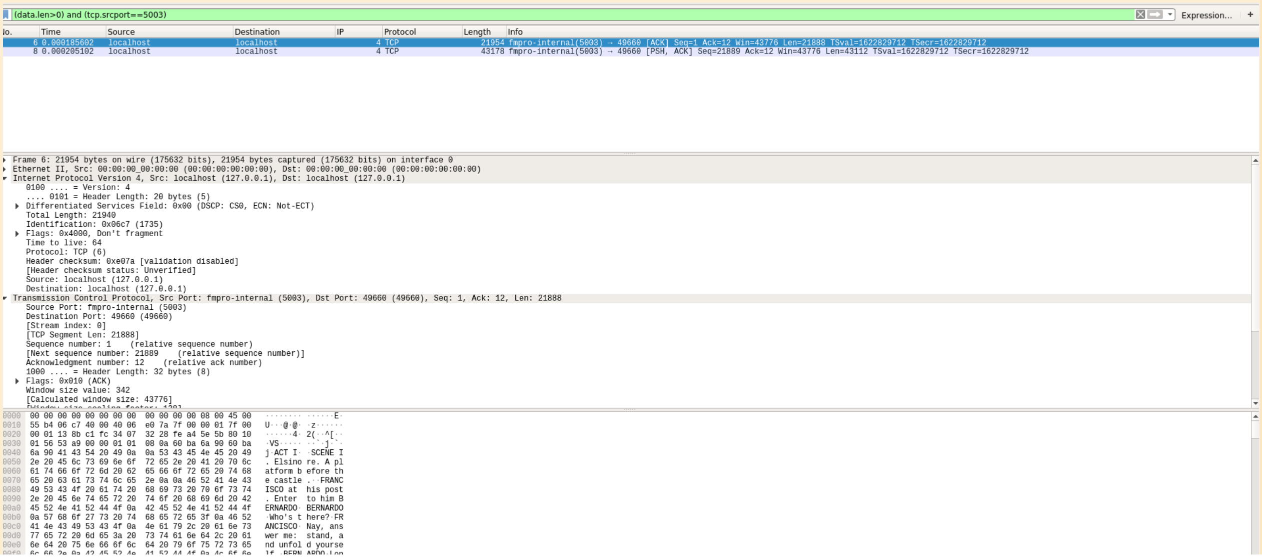
However Wireshark manages booleans the same way a programming language does. Specially, anything that is different to "null" or "0" or "non-existent" is considered as "True". It means that

tcp.payload
is also a valid display filter and will display all packets with an existing TCP payload.

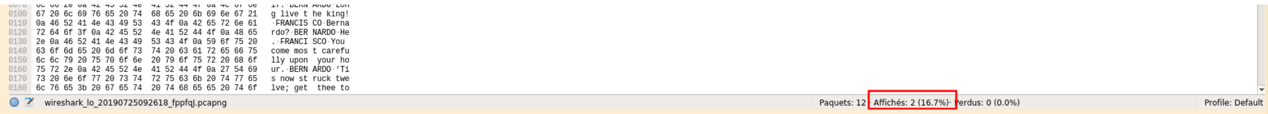
Don't forget to combine one of the possible filters with a filter selecting the traffic coming FROM the PMU. In Wireshark, display filters can be combined with boolean operators.

Once the display filter is green, is means the filter can be accepted (otherwise you have a synthax issue, check the field names with the above method). Don't forget to press ENTER with your cursor in the display filter field to apply the filter.

Only the packets that match the filter will be displayed, you can see their count on the bottom right corner of the window:



```
(data.len>0) and (tcp.srcport==5003)                                         Expression...  +

No.  Time         Source      Destination  IP  Protocol  Length  Info
   6 0.000185602  localhost   localhost     4  TCP        21954  fmpro-internal(5003) → 49660 [ACK] Seq=1 Ack=12 Win=43776 Len=21888 TSval=1622829712 TSecr=1622829712
   8 0.000205102  localhost   localhost     4  TCP        43178  fmpro-internal(5003) → 49660 [PSH, ACK] Seq=21889 Ack=12 Win=43776 Len=43112 TSval=1622829712 TSecr=1622829712

▸ Frame 6: 21954 bytes on wire (175632 bits), 21954 bytes captured (175632 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▾ Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 21940
    Identification: 0x06c7 (1735)
  ▸ Flags: 0x4000, Don't fragment
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0xe07a [validation disabled]
    [Header checksum status: Unverified]
    Source: localhost (127.0.0.1)
    Destination: localhost (127.0.0.1)
▾ Transmission Control Protocol, Src Port: fmpro-internal (5003), Dst Port: 49660 (49660), Seq: 1, Ack: 12, Len: 21888
    Source Port: fmpro-internal (5003)
    Destination Port: 49660 (49660)
    [Stream index: 0]
    [TCP Segment Len: 21888]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 21889    (relative sequence number)]
    Acknowledgment number: 12    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 342
    [Calculated window size: 43776]
    [Window size scaling factor: 128]

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  55 b4 06 c7 40 00 40 06  e0 7a 7f 00 00 01 7f 00   U···@·@· ·z······
0020  00 01 13 8b c1 fc 34 07  32 28 fe a4 5e 5b 80 10   ······4· 2(··^[··
0030  01 56 53 a9 00 00 01 01  08 0a 60 ba 6a 90 60 ba   ·VS····· ··`·j·`·
0040  6a 90 41 43 54 20 49 0a  0a 53 43 45 4e 45 20 49   j·ACT I· ·SCENE I
0050  2e 20 45 6c 73 69 6e 6f  72 65 2e 20 41 20 70 6c   . Elsino re. A pl
0060  61 74 66 6f 72 6d 20 62  65 66 6f 72 65 20 74 68   atform b efore th
0070  65 20 63 61 73 74 6c 65  2e 0a 0a 46 52 41 4e 43   e castle .··FRANC
0080  49 53 43 4f 20 61 74 20  68 69 73 20 70 6f 73 74   ISCO at  his post
0090  2e 20 45 6e 74 65 72 20  74 6f 20 68 69 6d 20 42   . Enter  to him B
00a0  45 52 4e 41 52 44 4f 0a  42 45 52 4e 41 52 44 4f   ERNARDO· BERNARDO
00b0  0a 57 68 6f 27 73 20 74  68 65 72 65 3f 0a 46 52   ·Who's t here?·FR
00c0  41 4e 43 49 53 43 4f 0a  4e 61 79 2c 20 61 6e 73   ANCISCO· Nay, ans
00d0  77 65 72 20 6d 65 3a 20  73 74 61 6e 64 2c 20 61   wer me:  stand, a
00e0  6e 64 20 75 6e 66 6f 6c  64 20 79 6f 75 72 73 65   nd unfol d yourse
00f0  6c 66 2e 0a 42 45 52 4e  41 52 44 4f 0a 4c 6f 6e   lf.·BERN ARDO·Lon
```



```
0100  67 20 6c 69 76 65 20 74  68 65 20 6b 69 6e 67 21   g live t he king!
0110  0a 46 52 41 4e 43 49 53  43 4f 0a 42 65 72 6e 61   ·FRANCIS CO·Berna
0120  72 64 6f 3f 0a 42 45 52  4e 41 52 44 4f 0a 48 65   rdo?·BER NARDO·He
0130  2e 0a 46 52 41 4e 43 49  53 43 4f 0a 59 6f 75 20   .·FRANCI SCO·You 
0140  63 6f 6d 65 20 6d 6f 73  74 20 63 61 72 65 66 75   come mos t carefu
0150  6c 6c 79 20 75 70 6f 6e  20 79 6f 75 72 20 68 6f   lly upon  your ho
0160  75 72 2e 0a 42 45 52 4e  41 52 44 4f 0a 27 54 69   ur.·BERN ARDO·'Ti
0170  73 20 6e 6f 77 20 73 74  72 75 63 6b 20 74 77 65   s now st ruck twe
0180  6c 76 65 3b 20 67 65 74  20 74 68 65 65 20 74 6f   lve; get  thee to
```

wireshark_lo_20190725092618_fppfqJ.pcapng   Paquets: 12 · Affichés: 2 (16.7%) · Perdus: 0 (0.0%)   Profile: Default

Question **6**

Complete

Marked out of 2.00

Flag question

Lab3 Part2 Question 6:

How many times was the

`recv()`

call invoked at the client side ?

21

Is the number of packets you see in Wireshark the same as the number of

`recv()`

invocations at your client ?

○ Yes

● No

What can you change IN YOUR CLIENT CODE to change the number of

`recv()`

invocations ? Answer with a short sentence:

We can change the recv argument to receive more bytes at onc

Refer to Question 5 of Part1.2.

For the last question, we are mostly expecting 2 keywords that are the name of the value you can change.

Question **7**

Complete

Marked out of 3.00

Flag question

Lab3 Part2 Question 7:

This question will summarize your findings of part 2. Complete the following text

TCP is a stream-oriented protocol.

From the programmer's perspective, TCP provides a service which is analogous to:

○ A post system distributing post cards. Each postcard contains fields that the sender is required to provide: a date, an opening "Dear...", a message of known maximum size (the size of a postcard), a closing form that encodes the relationship between the sender and the recipient and a signature. The post office also adds the date and distribute the postcard to the recipient.

● A garden hose without any hole: anything that enters the hose will exit it on the remote end in the same order, irrespective of whether the hose is transporting water, fertilizer, pesticides or soda.

Who decides how the data transported by TCP must be formatted and interpreted ? The application layer

.

In which of the following applications is a stream-oriented transport protocol a good solution ?

☑ Downloading a large file using FTP (File Transfert Protocol)

☐ Sending homemade control messages of a small size from a PMU to a PDC, each one spaced by 1second

☑ Loading a web page containing one unique large image using HTTP (HyperText Transfert Protocol)

For the last part of the question, consider the following:

What are HTTP, FTP protocols ? What is their purpose ?

When you create homemade control messages, do you use HTTP or FTP or any equivalent protocol ?

You were asked to prompt one PMU message per line, did you achieve it ? Was is easy ?

Finish review

◄ [Graded] Lab3 - Part 1.2

Jump to...

[Graded] Lab3 - Part 3 ►

Quiz navigation

1 2 3 4 5 6 7

Show one page at a time

Finish review

EPFL

Contact EPFL CH-1015 Lausanne +41 21 693 11 11

Follow EPFL on social media