SharkFest '17 Europe

Custom LUA dissectors to the rescue in RCA

How to write a LUA dissector for a proprietary protocol to assist in troubleshooting

9 november 2017

Sake Blok

relational therapist for computer systems

sake.blok@SYN-bit.nl

SYN-bit
deep traffic analysis

Hands-on files:

http://www.SYN-bit.nl/files/sf17eu-lua.zip

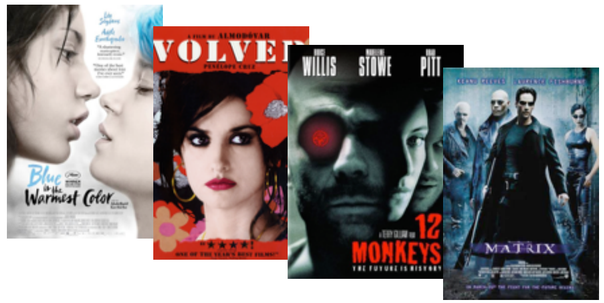
About me…

EURONET INTERNET

ABN·AMRO

ion ip

SYN-bit
deep traffic analysis



SYN-bit

deep traffic analysis

Application and network troubleshooting

Protocol and packet analysis

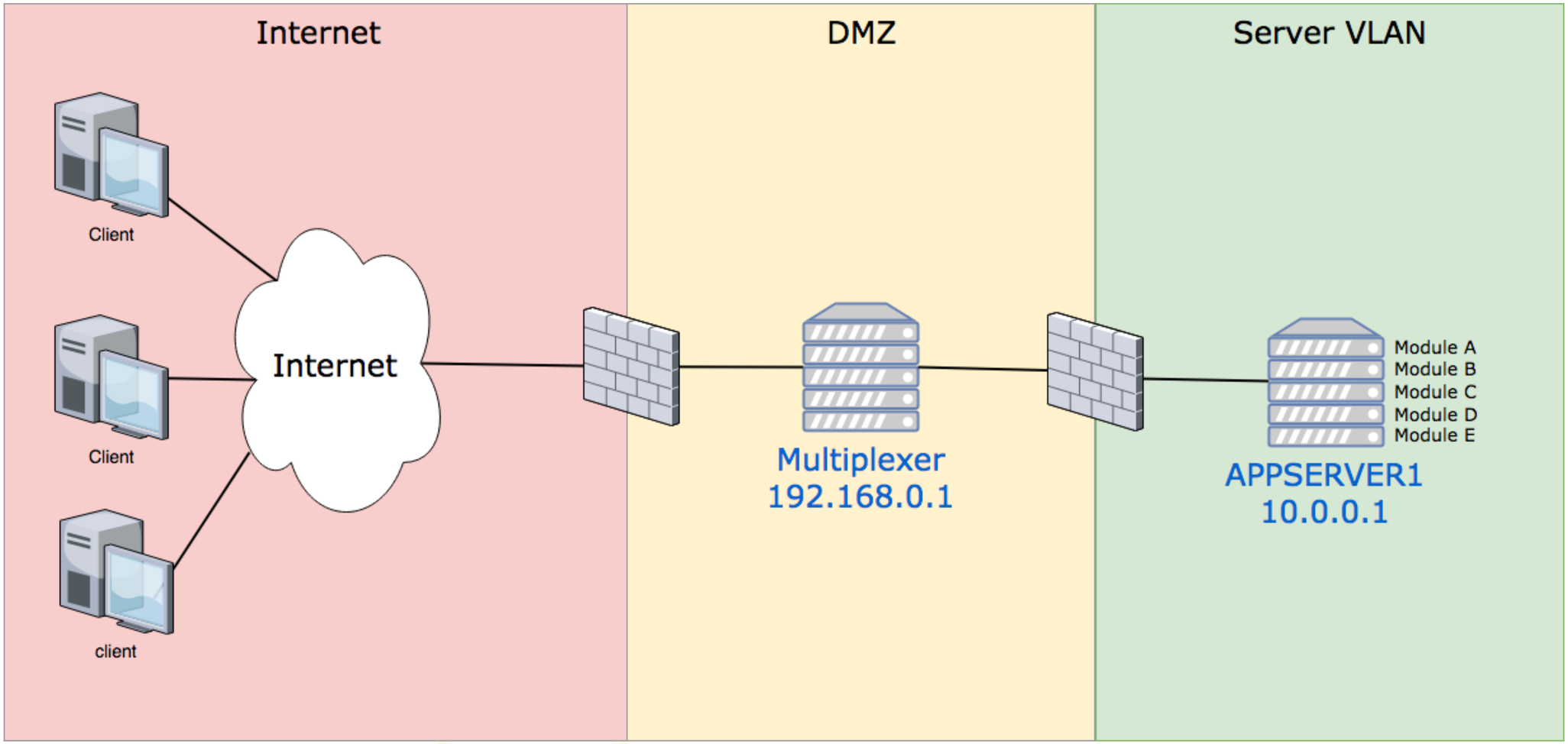
Training (Wireshark, TCP, SSL)

www.SYN-bit.nl

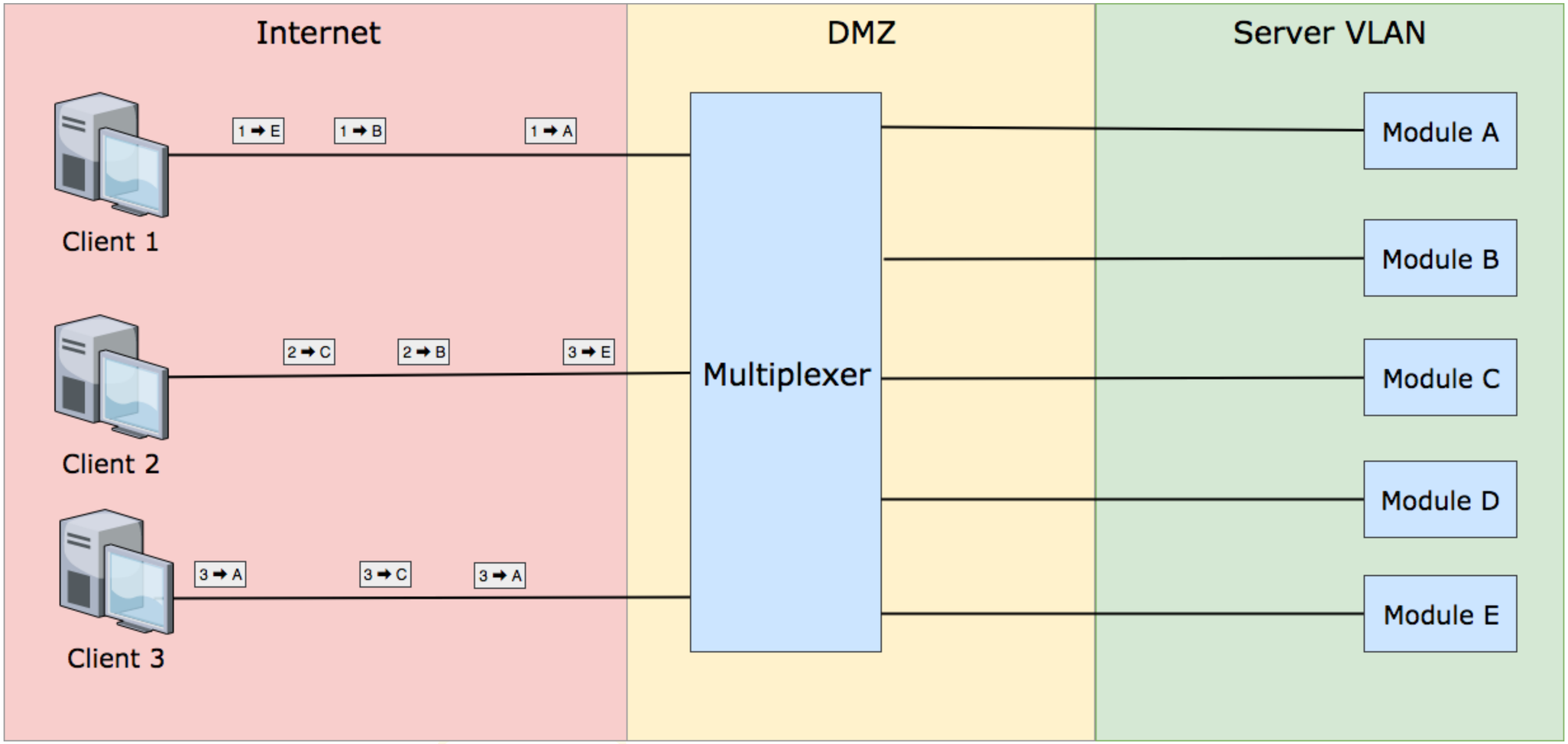
Houston, we have a problem…

- Trading platform
- Connections being reset
- Logfiles point to network
- Software supplier blames the network (duh!)
- Custom protocol, so just TCP analysis possible
- What now?

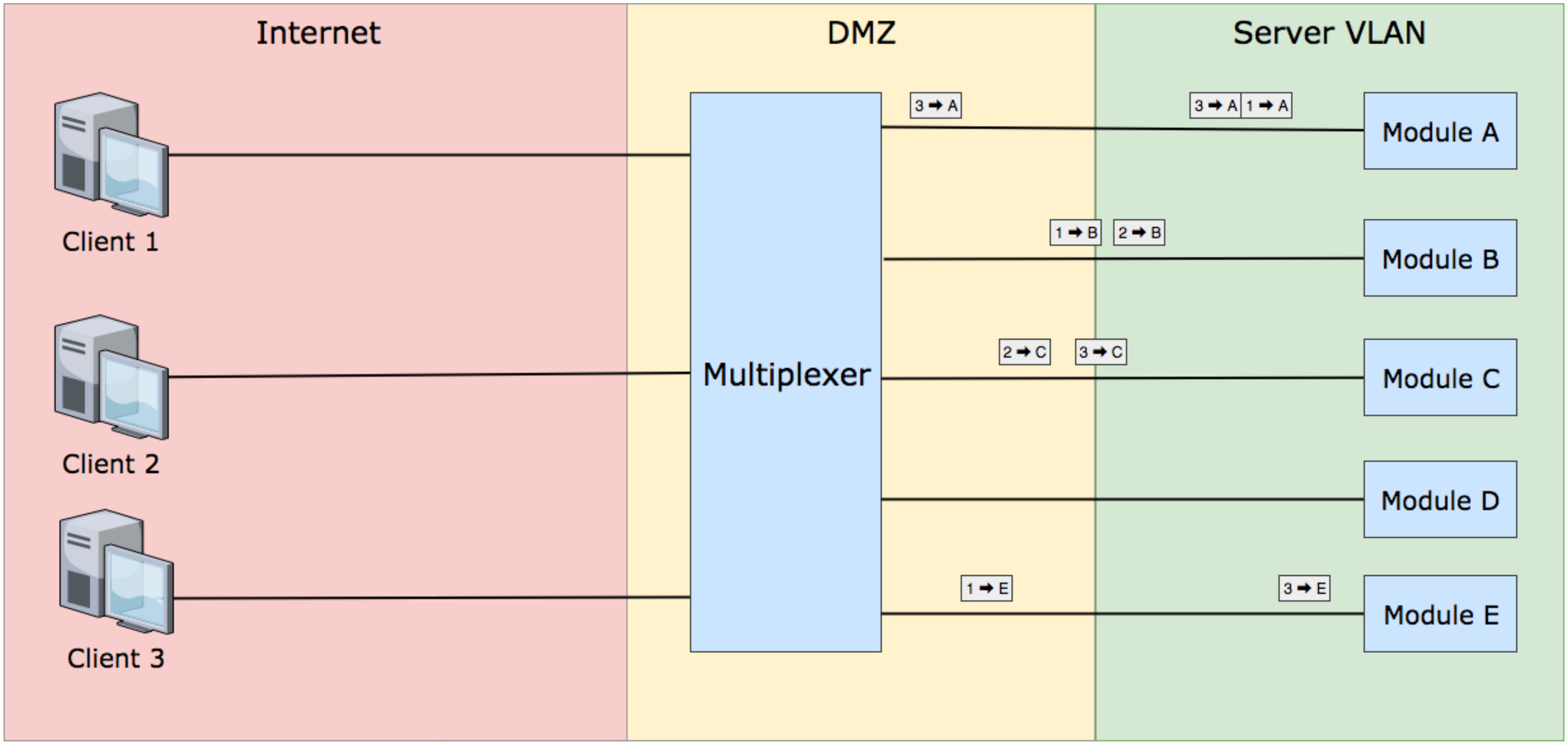
So how does the application work?



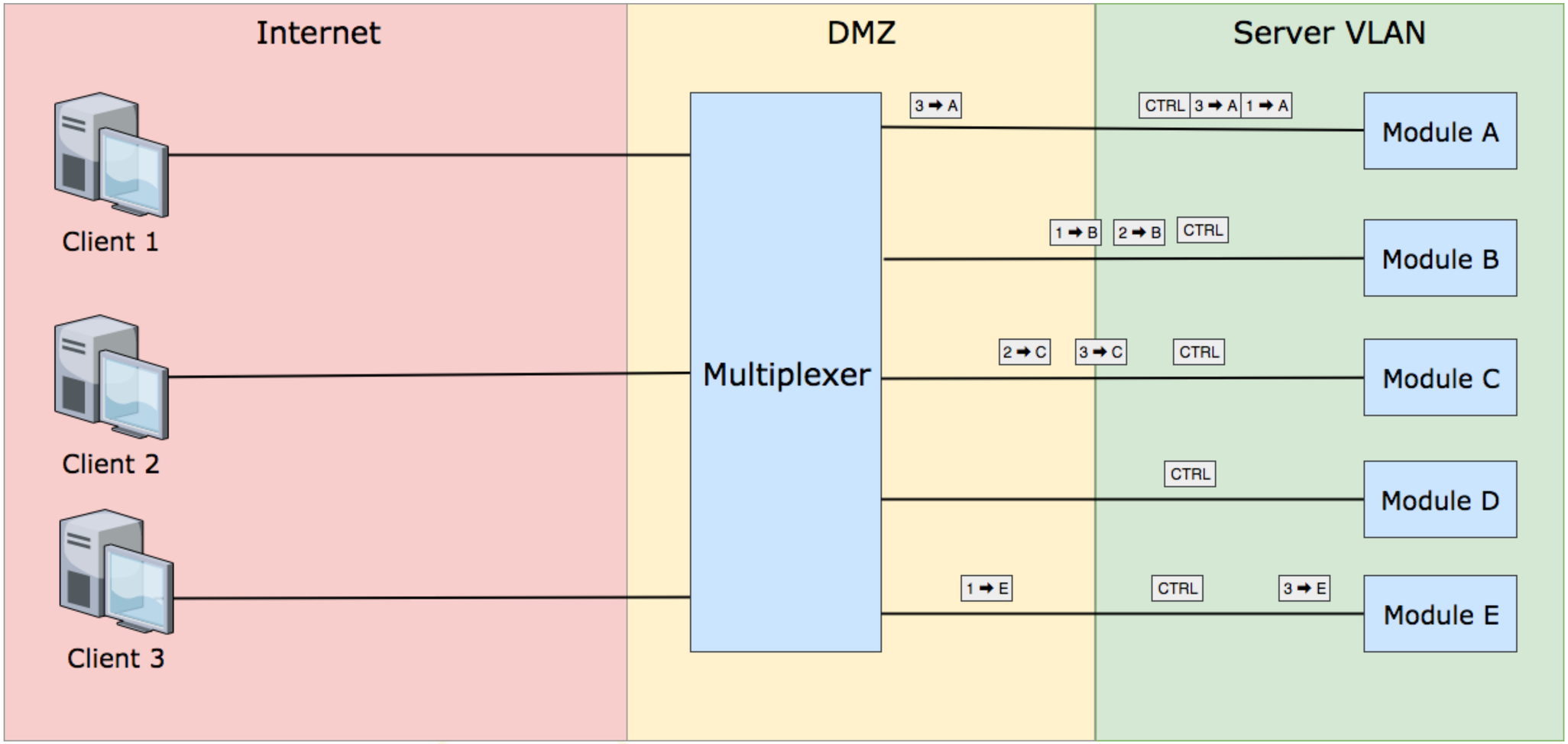
PDU's coming in



PDU's going out



… plus Control PDU's



Trader module Log

```
2014-06-05 09:27:51,883 NOTICE - Creating clientConnection: 2654 user: XXXXXX0020 on: Trader
2014-06-05 09:28:57,779 NOTICE - Creating clientConnection: 2655 user: XXXXXX0019 on: Trader
2014-06-05 09:30:12,755 NOTICE - Closing clientConnection: 2652 from: Trader
2014-06-05 09:30:48,043 NOTICE - Creating clientConnection: 2656 user: XXXXXX0013 on: Trader
2014-06-05 09:31:58,978 ERROR - No response from multiplexer: 192.168.0.1:2107 within 1 minutes 30 seconds. Terminating connection.
2014-06-05 09:31:58,978 NOTICE - Shutdown connection from: Trader
2014-06-05 09:31:58,978 WARN - ModuleServer Trader lost connection to multiplexer on 192.168.0.1:2107
2014-06-05 09:37:40,252 NOTICE - Creating clientConnection: 1129 user: APPSERVER1 on: Trader
2014-06-05 09:42:03,556 WARN - Connection lost in receive loop: Trader
2014-06-05 09:42:03,556 NOTICE - Shutdown connection from: Trader
2014-06-05 09:42:03,556 WARN - ModuleServer Trader lost connection to multiplexer on 192.168.0.1:2107
2014-06-05 09:43:54,147 NOTICE - Creating clientConnection: 2003 user: XXXXXX0016 on: Trader
```

Multiplexer Log

```
2014-06-05 09:34:41,006 NOTICE - Logging in clientConnection: 2678 user: XXXXXX0009 to: MarketConfiguration
2014-06-05 09:34:41,006 NOTICE - Logging in clientConnection: 2678 user: XXXXXX0009 to: FinancialModule
2014-06-05 09:34:52,816 NOTICE - Disabling client logins on port: '443'
2014-06-05 09:34:52,816 NOTICE - Client accept loop stopped
2014-06-05 09:34:53,362 NOTICE - Logging in clientConnection: 2671 user: XXXXXX0000 to: BackOffice
2014-06-05 09:34:53,580 NOTICE - Logging in clientConnection: 2671 user: XXXXXX0000 to: Trader
2014-06-05 09:34:53,580 NOTICE - Logging in clientConnection: 2671 user: XXXXXX0000 to: MarketConfiguration
2014-06-05 09:34:53,580 NOTICE - Logging in clientConnection: 2671 user: XXXXXX0000 to: FinancialModule
2014-06-05 09:34:56,622 NOTICE - Logging in clientConnection: 2679 user: XXXXXX0021 to: BackOffice
2014-06-05 09:34:56,825 NOTICE - Logging in clientConnection: 2679 user: XXXXXX0021 to: Trader
2014-06-05 09:34:56,825 NOTICE - Logging in clientConnection: 2679 user: XXXXXX0021 to: MarketConfiguration
2014-06-05 09:34:56,825 NOTICE - Logging in clientConnection: 2679 user: XXXXXX0021 to: FinancialModule
2014-06-05 09:35:01,458 sss  127.000.000.001              Operator      2014-06-05 09:35:01.458      Multiplexer stopped.
Stopped logging 2014-06-05 09:35:01,458
Started logging 2014-06-05 09:35:16,060
2014-06-05 09:35:16,060 sss  127.000.000.001              Operator      2014-06-05 09:35:16.060      Multiplexer started.
2014-06-05 09:35:32,924 NOTICE - Received start id: 2001
2014-06-05 09:35:32,940 NOTICE - Received security parameters
2014-06-05 09:41:26,864 sss  127.000.000.001              Operator      2014-06-05 09:41:26.864      Multiplexer stopped.
Stopped logging 2014-06-05 09:41:26,864
Started logging 2014-06-05 09:42:46,004
2014-06-05 09:42:46,004 sss  127.000.000.001              Operator      2014-06-05 09:42:46.004      Multiplexer started.
2014-06-05 09:42:50,107 NOTICE - Received start id: 2001
2014-06-05 09:42:50,138 NOTICE - Received security parameters
2014-06-05 09:43:28,531 NOTICE - Enabling client logins on port: '443'
2014-06-05 09:43:28,531 NOTICE - Client accept loop started
2014-06-05 09:43:53,647 NOTICE - Logging in clientConnection: 2003 user: XXXXXX0016 to: BackOffice
2014-06-05 09:43:53,865 NOTICE - Logging in clientConnection: 2003 user: XXXXXX0016 to: Trader
```

Timeline

```
2014-06-05 09:31:58,978 ERROR - No response from multiplexer: 192.168.0.1:2107 within 1 minutes 30 seconds. Terminating connection.
2014-06-05 09:31:58,978 WARN - ModuleServer Trader lost connection to multiplexer on 192.168.0.1:2107

2014-06-05 09:34:52,816 NOTICE - Disabling client logins on port: '443'
2014-06-05 09:34:52,816 NOTICE - Client accept loop stopped

2014-06-05 09:35:01,458 sss             127.000.000.001          Operator     2014-06-05 09:35:01.458    Multiplexer stopped.
2014-06-05 09:35:16,060 sss             127.000.000.001          Operator     2014-06-05 09:35:16.060    Multiplexer started.
2014-06-05 09:35:32,924 NOTICE - Received start id: 2001

2014-06-05 09:41:26,864 sss             127.000.000.001          Operator     2014-06-05 09:41:26.864    Multiplexer stopped.
2014-06-05 09:42:46,004 sss             127.000.000.001          Operator     2014-06-05 09:42:46.004    Multiplexer started.
2014-06-05 09:42:50,107 NOTICE - Received start id: 2001
2014-06-05 09:43:28,531 NOTICE - Enabling client logins on port: '443'
2014-06-05 09:43:28,531 NOTICE - Client accept loop started

2014-06-05 09:43:53,647 NOTICE - Logging in clientConnection: 2003 user: XXXXXX0016 to: BackOffice
```

Packet timestamps don’t match

No.	Time	Delta	Source	Destination	Protocol	Length	Info
31362	09:31:49.982437	0.000559	10.0.0.1	192.168.0.1	TCP	60	1195 → 2107 [ACK] Seq=21251242 Ack=397458 Win=7747840 Len=0
31363	09:31:50.158695	0.176258	192.168.0.1	10.0.0.1	TCP	1434	[TCP ZeroWindow] 2107 → 1195 [ACK] Seq=397458 Ack=21251242 Win=0 Len=1380
31364	09:31:50.158699	0.000004	192.168.0.1	10.0.0.1	TCP	1434	[TCP ZeroWindow] 2107 → 1195 [ACK] Seq=398838 Ack=21251242 Win=0 Len=1380
31365	09:31:50.158703	0.000004	192.168.0.1	10.0.0.1	TCP	1434	[TCP ZeroWindow] 2107 → 1195 [ACK] Seq=400218 Ack=21251242 Win=0 Len=1380
31366	09:31:50.158707	0.000004	192.168.0.1	10.0.0.1	TCP	1434	[TCP ZeroWindow] 2107 → 1195 [ACK] Seq=401598 Ack=21251242 Win=0 Len=1380
31367	09:31:50.158711	0.000004	192.168.0.1	10.0.0.1	TCP	875	[TCP ZeroWindow] 2107 → 1195 [PSH, ACK] Seq=402978 Ack=21251242 Win=0 Len=821
31368	09:31:50.159241	0.000530	10.0.0.1	192.168.0.1	TCP	60	1195 → 2107 [ACK] Seq=21251242 Ack=403799 Win=7741440 Len=0
31369	09:31:53.888822	3.729581	10.0.0.1	192.168.0.1	TCP	65	1170 → 2107 [PSH, ACK] Seq=148616 Ack=180900 Win=130816 Len=11
31370	09:31:54.091036	0.202214	192.168.0.1	10.0.0.1	TCP	60	2107 → 1170 [ACK] Seq=180900 Ack=148627 Win=130816 Len=0
31371	09:31:54.170051	0.079015	10.0.0.1	192.168.0.1	TCP	65	1170 → 2107 [PSH, ACK] Seq=148627 Ack=180900 Win=130816 Len=11
31372	09:31:54.371839	0.201788	192.168.0.1	10.0.0.1	TCP	60	2107 → 1170 [ACK] Seq=180900 Ack=148638 Win=130816 Len=0
31373	09:31:54.888059	0.516220	10.0.0.1	192.168.0.1	TCP	65	1170 → 2107 [PSH, ACK] Seq=148638 Ack=180900 Win=130816 Len=11
31374	09:31:55.105074	0.217015	192.168.0.1	10.0.0.1	TCP	60	2107 → 1170 [ACK] Seq=180900 Ack=148649 Win=130816 Len=0
31375	09:31:59.226292	4.121218	10.0.0.1	192.168.0.1	TCP	65	1170 → 2107 [PSH, ACK] Seq=148649 Ack=180900 Win=130816 Len=11
31376	09:31:59.441887	0.215595	192.168.0.1	10.0.0.1	TCP	60	2107 → 1170 [ACK] Seq=180900 Ack=148660 Win=130816 Len=0
31377	09:32:03.343767	3.901880	10.0.0.1	192.168.0.1	TCP	66	1592 → 2107 [SYN] Seq=0 Win=8192 Len=0 MSS=1380 WS=256 SACK_PERM=1
31378	09:32:03.344095	0.000328	192.168.0.1	10.0.0.1	TCP	66	2107 → 1592 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
31379	09:32:03.344383	0.000288	10.0.0.1	192.168.0.1	TCP	60	1592 → 2107 [ACK] Seq=1 Ack=1 Win=131072 Len=0
31380	09:32:03.344877	0.000494	10.0.0.1	192.168.0.1	TCP	63	1592 → 2107 [PSH, ACK] Seq=1 Ack=1 Win=131072 Len=9
31381	09:32:03.349625	0.004748	192.168.0.1	10.0.0.1	TCP	71	2107 → 1170 [PSH, ACK] Seq=180900 Ack=148660 Win=130816 Len=17
31382	09:32:03.544694	0.195069	192.168.0.1	10.0.0.1	TCP	60	2107 → 1592 [ACK] Seq=1 Ack=10 Win=131072 Len=0

No.	Time	Delta	Source	Destination	Protocol	Length	Info
31488	09:32:12.351357	0.000004	192.168.0.1	10.0.0.1	TCP	1434	[TCP ZeroWindow] 2107 → 1195 [ACK] Seq=440839 Ack=21251242 Win=0 Len=1380
31489	09:32:12.351362	0.000005	192.168.0.1	10.0.0.1	TCP	1434	[TCP ZeroWindow] 2107 → 1195 [ACK] Seq=442219 Ack=21251242 Win=0 Len=1380
31490	09:32:12.351366	0.000004	192.168.0.1	10.0.0.1	TCP	1214	[TCP ZeroWindow] 2107 → 1195 [PSH, ACK] Seq=443599 Ack=21251242 Win=0 Len=1160
31491	09:32:12.351912	0.000546	10.0.0.1	192.168.0.1	TCP	60	1195 → 2107 [RST, ACK] Seq=21251242 Ack=405179 Win=0 Len=0

The Blaming Game

- Zero Window on Multiplexer
  - But why does the Trader module report it did not receive a response while it is still receiving lots of data?
- TCP/RST from APPSERVER1
  - But not at the time of the log message!
- ~~The network is the problem~~
  - But what is? And how to convince the software vendor?
  - Client (APPSERVER1) or server (Multiplexer) side problem?

LUA to the rescue…

- Different LUA dissection types
- Skeleton Dissector
- Parsing the PDU header
- Reassembly

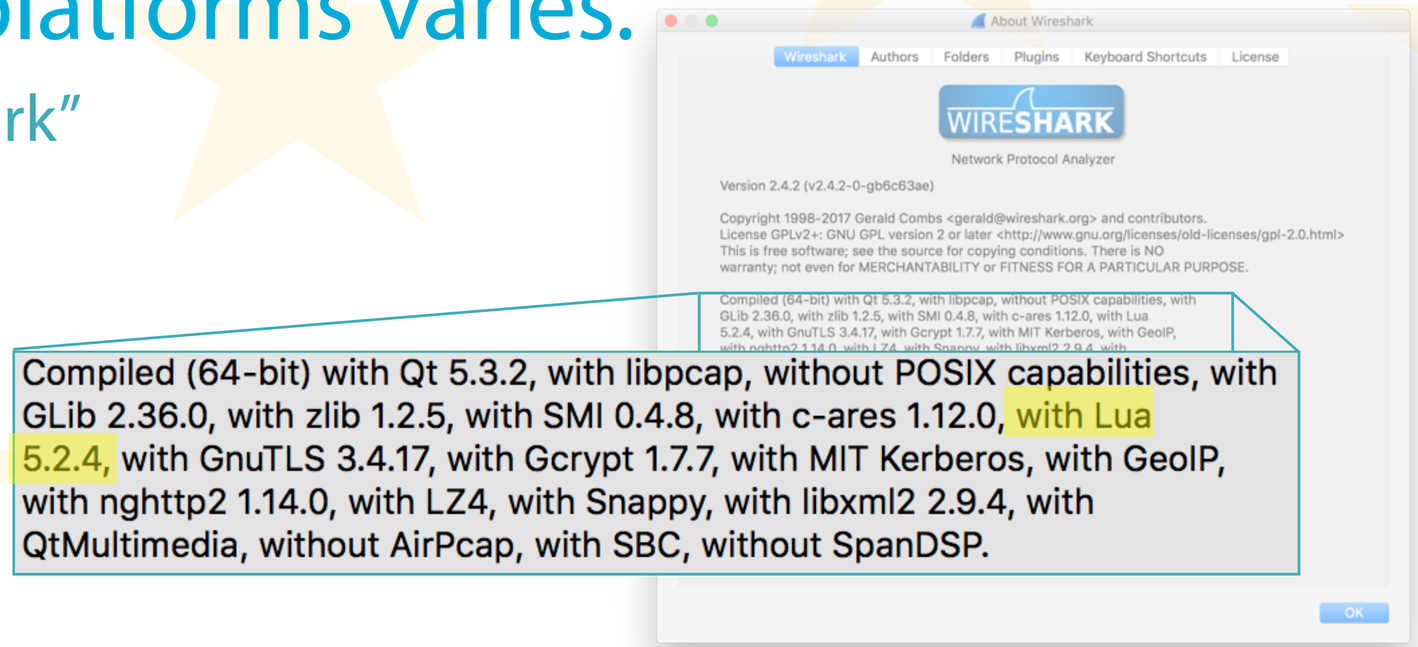
- Starting point:
  https://wiki.wireshark.org/Lua

LUA in Wireshark

- Lua can be used to write dissectors, post-dissectors and taps.
  - Although it's possible to write **dissectors in Lua**, Wireshark dissectors are written in C, as C is several times faster than Lua. Lua is ok for prototyping dissectors, during Reverse Engineering you can use your time for finding out how things work instead of compiling and debugging your C dissector.
  - **Post-dissectors** are dissectors meant to run after every other dissector has run. They can add items to the dissection tree so they can be used to create your own extensions to the filtering mechanism.
  - **Taps** are used to collect information after the packet has been dissected.
- So for our purpose, a normal Lua dissector is the best fit

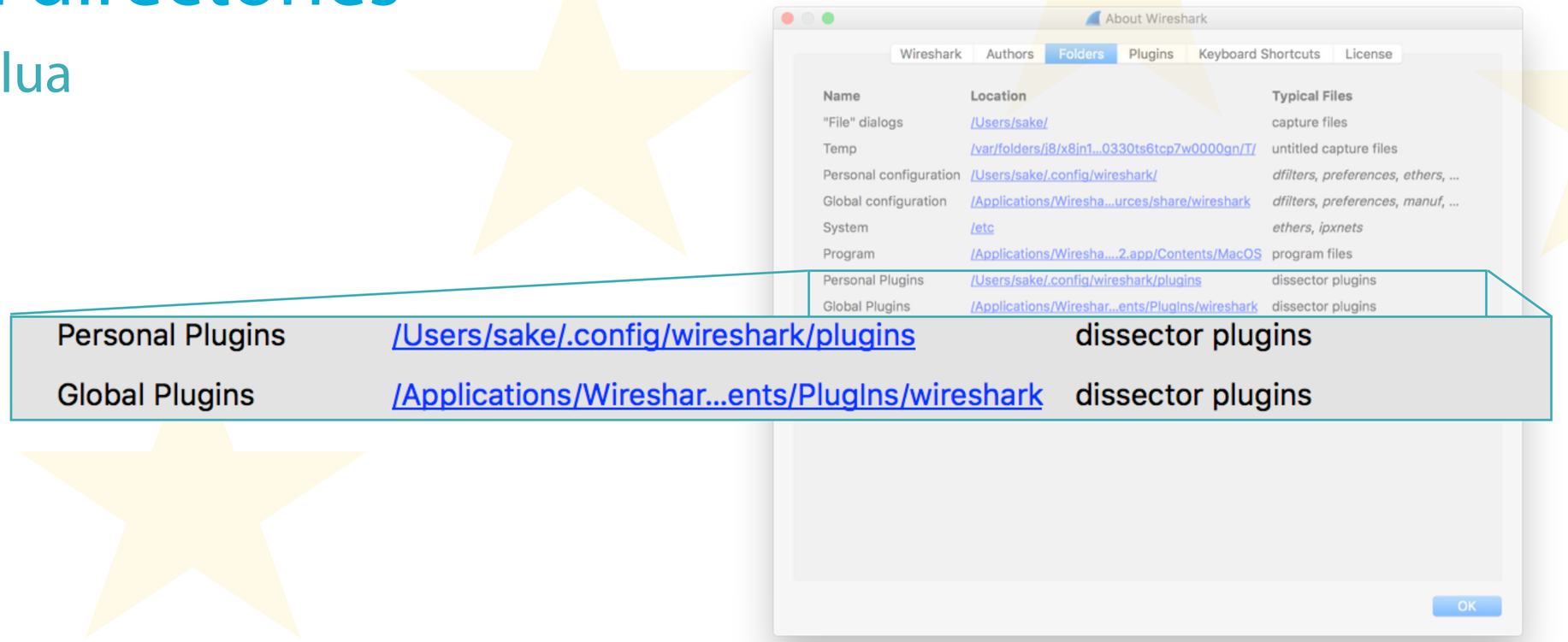
LUA supported?

- LUA has shipped with the Windows version of Wireshark since 0.99.4.
- Availability on other platforms varies.
  - Check: “Help -> About Wireshark”



How to execute?

- CLI option: -X lua_script:<script-file>
- In the plugin directories
  - Need extension .lua



Hello World!

```
sake@MacSake:~$ cat ~/.config/wireshark/plugins/hello.lua
-- hello.lua
-- Lua's implementation of D. Ritchie's hello world program.
print("hello world!")
sake@MacSake:~$
```

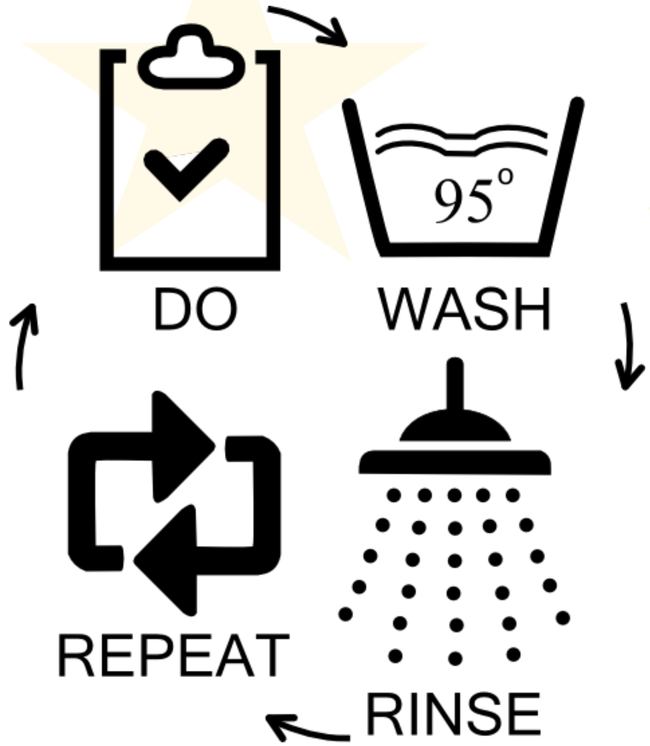
```
sake@MacSake:~$ tshark -r icmp.pcap -c 1
    1   0.000000   172.16.0.34 → 8.8.8.8      ICMP 98 Echo (ping) request  id=0xfa0e, seq=0/0, ttl=64
sake@MacSake:~$
```

```
sake@MacSake:~$ tshark -r icmp.pcap -c 1 -X lua_script:hello.lua
hello world!
    1   0.000000   172.16.0.34 → 8.8.8.8      ICMP 98 Echo (ping) request  id=0xfa0e, seq=0/0, ttl=64
sake@MacSake:~$
```

```
sake@MacSake:~$ mv hello.lua ~/.config/wireshark/plugins/
sake@MacSake:~$ tshark -r icmp.pcap -c 1
hello world!
    1   0.000000   172.16.0.34 → 8.8.8.8      ICMP 98 Echo (ping) request  id=0xfa0e, seq=0/0, ttl=64
sake@MacSake:~$
```

Writing: trade.lua

- Write a basic LUA dissector
- Get a protocol description
- Add some reverse engineering
  - Interpret packet data
  - Write code
  - Test
  - Rinse
  - Repeat



PDU format

- Each packet has a 5 byte header:
- 1: sender
  - List of module ID's provided
- 2-3: total size including header
- 4-5: receiver
  - In case of control packets: The receiver will be ff:ff
- The header is followed by a keyword and a null termination
- Optional data

Examples

- Keepalive packets from module (DocumentSystem) to multiplexer contain the following bytes:
  - 25:00:0a:ff:ff:50:69:6e:67:00
- The Multiplexer replies with:
  - 01:00:0a:ff:ff:50:6f:6e:67:00
- Roundtrip packets from the Multiplexer to module(DocumentSystem) contain the following bytes:
  - 01:00:17:ff:ff:52:6f:75:6e:64:74:72:69:70:00:07:1b:88:aa:db:46:b6:0c
- The module replies with:
  - 25:00:17:ff:ff:52:6f:75:6e:64:74:72:69:70:00:07:1b:88:aa:db:46:b6:0c

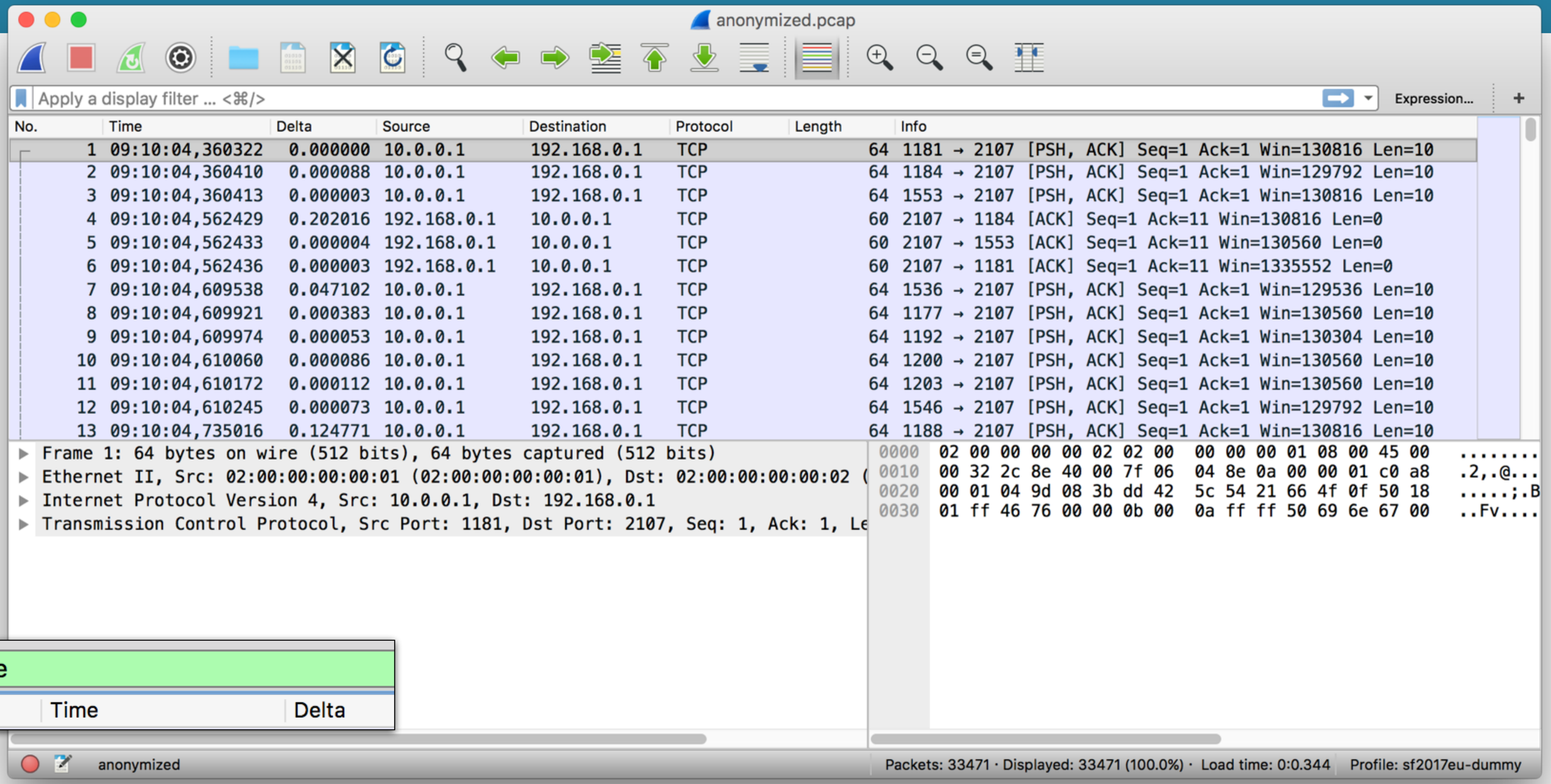
Example Drilldown

```
25                                  Module ID
00:17                               Size (Big endian)
ff:ff                               Receiver
52:6f:75:6e:64:74:72:69:70          Keyword: ”Roundtrip" (ASCII)
00                                  Keyword termination
07                                  Size of following data
1b:88:aa:db:46:b6:0c                Microsecond count (Little endian)
```

01.lua: Register the dissector

```
do
    local trade = Proto("trade","TRADE");

    function trade.dissector(tvb,pinfo,tree)
    end

    local tcp_encap_table = DissectorTable.get("tcp.port")
    tcp_encap_table:add(2107,trade)
end
```

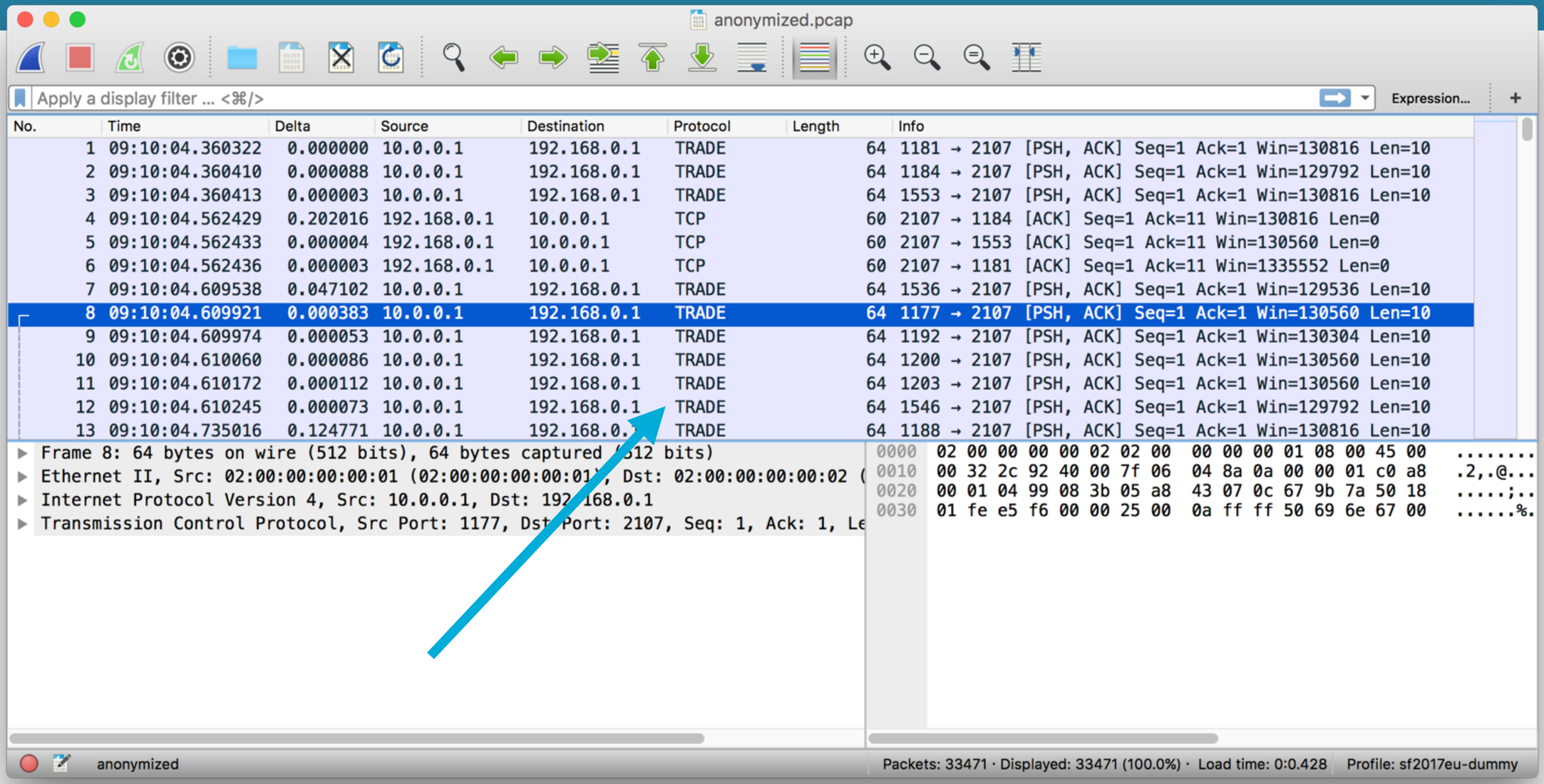
Result of 01.lua



02.lua: Update the protocol column

```
do
    local trade = Proto("trade","TRADE");

    function trade.dissector(tvb,pinfo,tree)

        -- info("Entering dissector for frame " .. pinfo.number)
        pinfo.cols.protocol = "TRADE"

    end

    local tcp_encap_table = DissectorTable.get("tcp.port")
    tcp_encap_table:add(2107,trade)
end
```

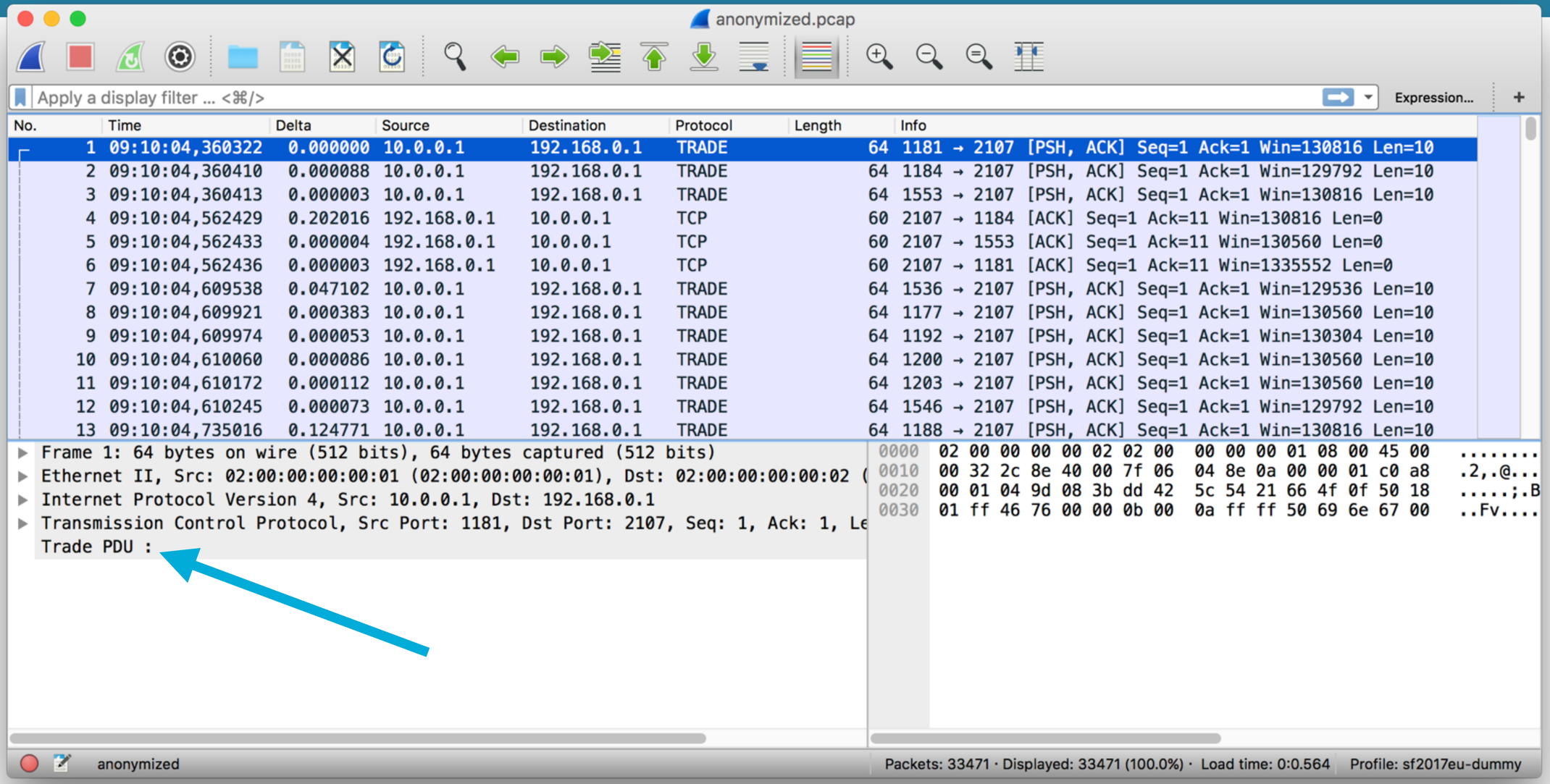
Result of 02.lua



03.lua: Add a subtree

```
do
    local trade = Proto("trade","TRADE");

    function trade.dissector(tvb,pinfo,tree)

        -- info("Entering dissector for frame " .. pinfo.number)
        pinfo.cols.protocol = "TRADE"

        local subtree

        subtree = tree:add(trade,tvb(0,tvb:len()),"Trade PDU : ")
    end

    local tcp_encap_table = DissectorTable.get("tcp.port")
    tcp_encap_table:add(2107,trade)
end
```

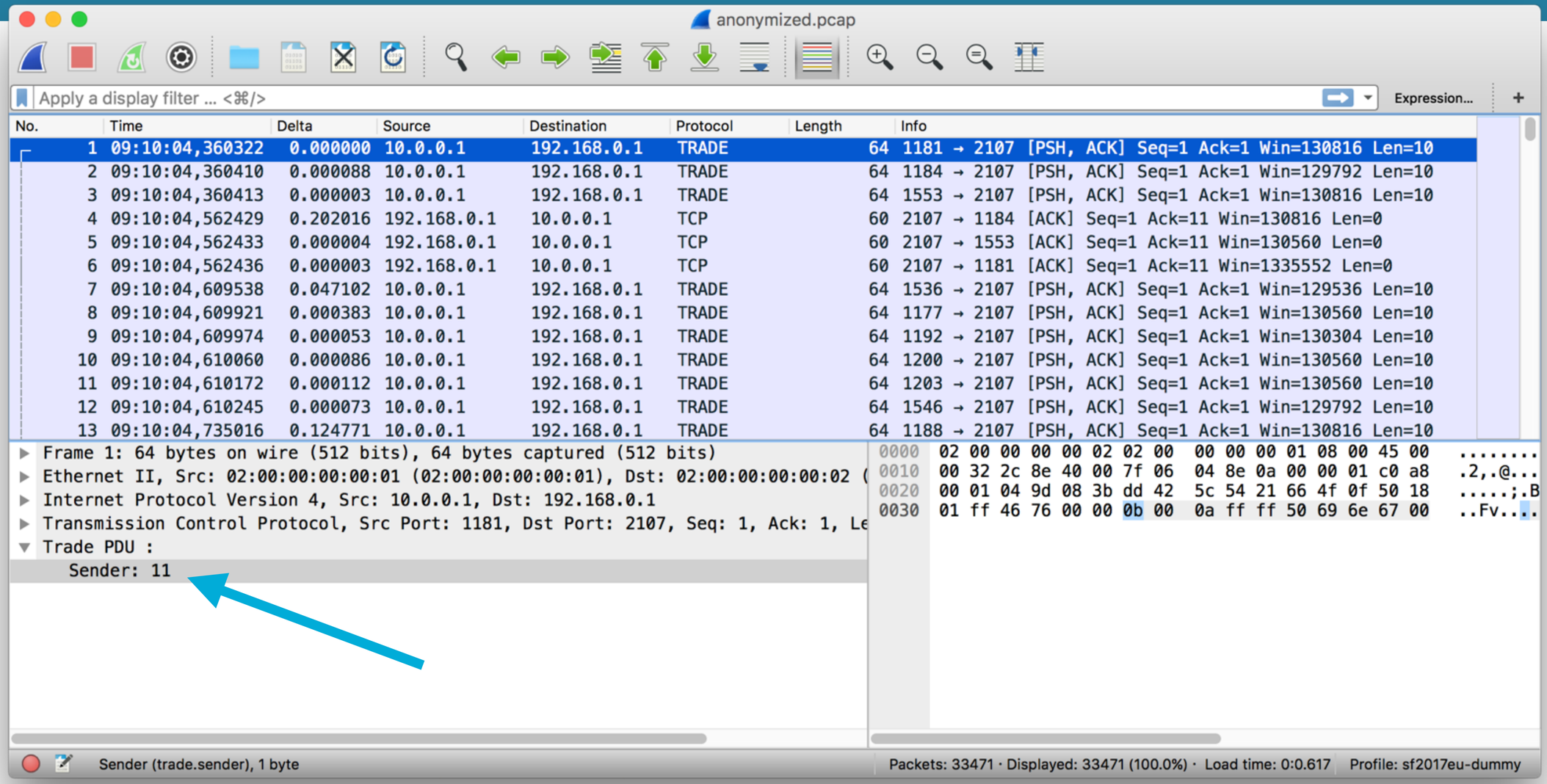
Result of 03.lua



04.lua: Add a field

```
sake@MacSake:~$ diff -b 03.lua 04.lua
2a3,5
>     local f = trade.fields
>
>     f.sender     = ProtoField.uint8("trade.sender","Sender",base.DEC)
9a13,20
>         local offset = 0
>         local info = ""
>
>         subtree = tree:add(trade,tvb(offset,tvb:len()),"Trade PDU : ")
>
>         -- Add sender
>         subtree:add(f.sender,tvb(offset,1))
>         offset = offset + 1
11c22,23
<         subtree = tree:add(trade,tvb(0,tvb:len()),"Trade PDU : ")
---
>         -- Tell the calling dissector how many bytes we dissected
>         return tvb:len()
sake@MacSake:~$
```

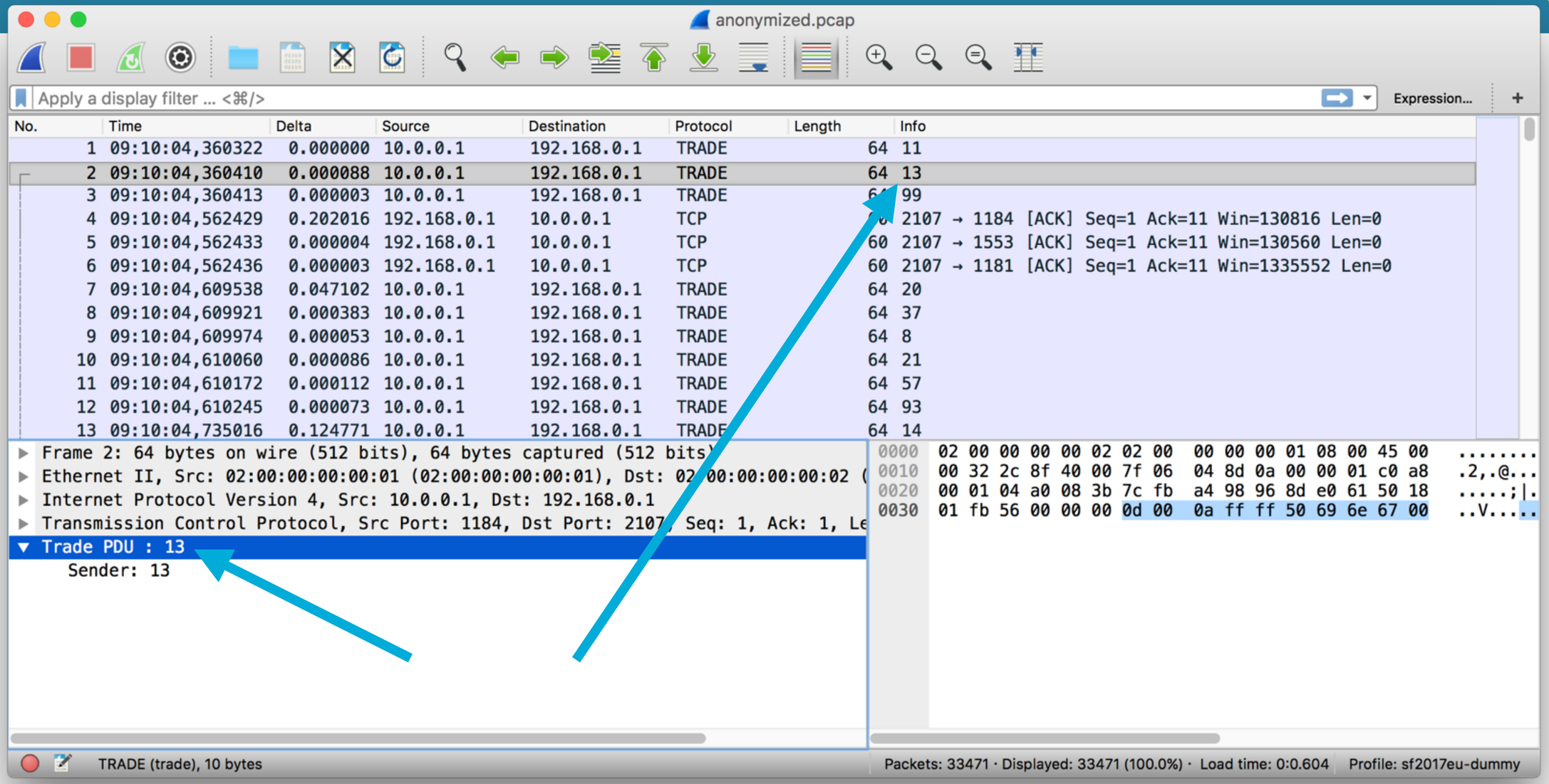
Result of 04.lua



05.lua: Fill the info column

```
-- Add sender
local sender = tvb(offset,1):uint()
info = tostring(sender)

subtree:add(f.sender,tvb(offset,1))
offset = offset + 1

subtree:append_text(info)
pinfo.cols.info = info
```

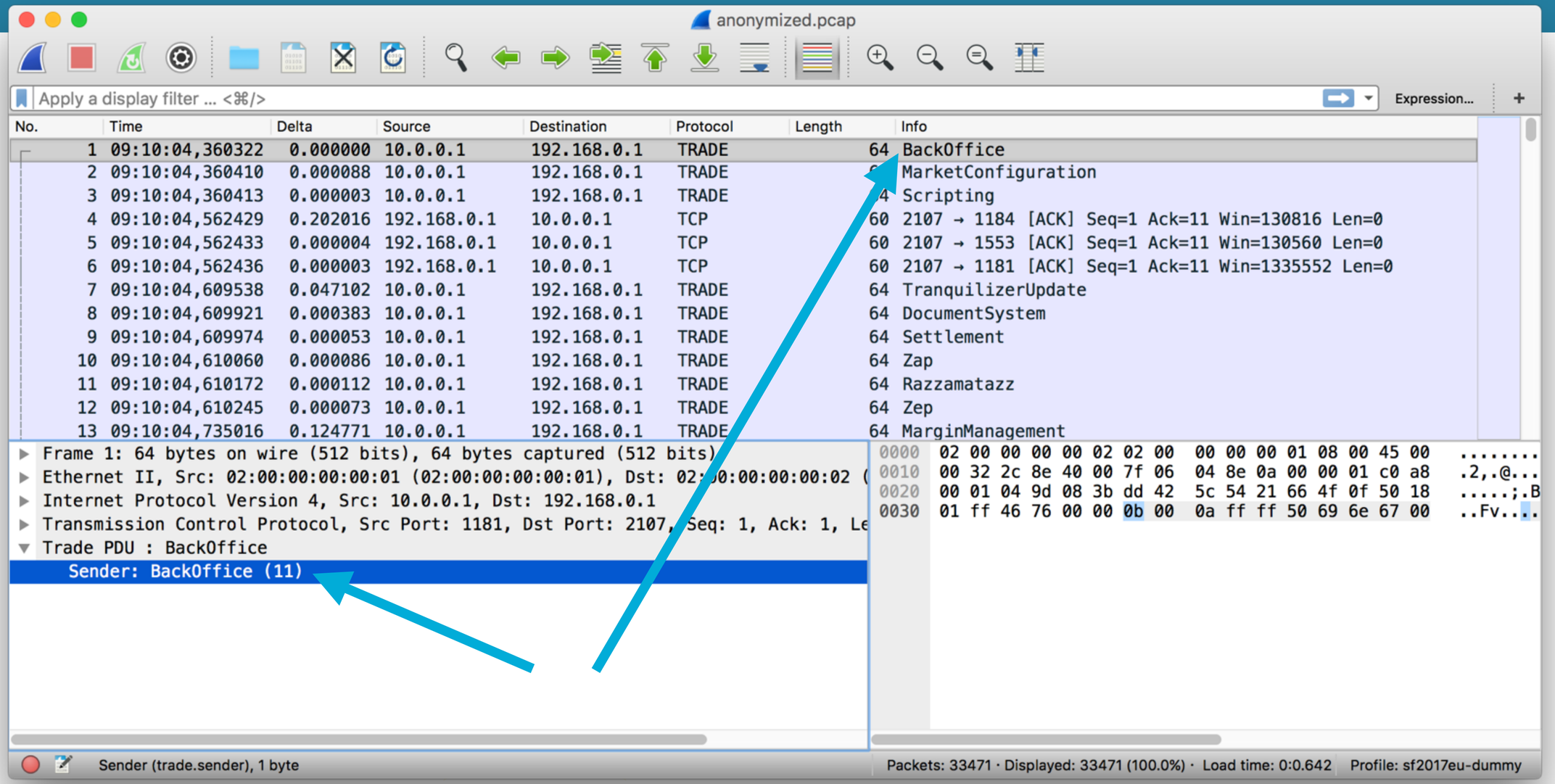
Result of 05.lua



06.lua: Interpret field values

```
local modules = {
    [1]   = 'Multiplexer',
    [6]   = 'CarpenterUpdate',
    [7]   = 'Tradesite',
    [. . . . . . . . . ]
    [93]  = 'Zep',
    [99]  = 'Scripting'
}

f.sender     = ProtoField.uint8("trade.sender","Sender",base.DEC,modules)

    -- Add sender
    local sender = tvb(offset,1):uint()
    local mod = modules[sender]
    assert( mod ~= nil,  "Unknown Sender!")
    info = mod
```

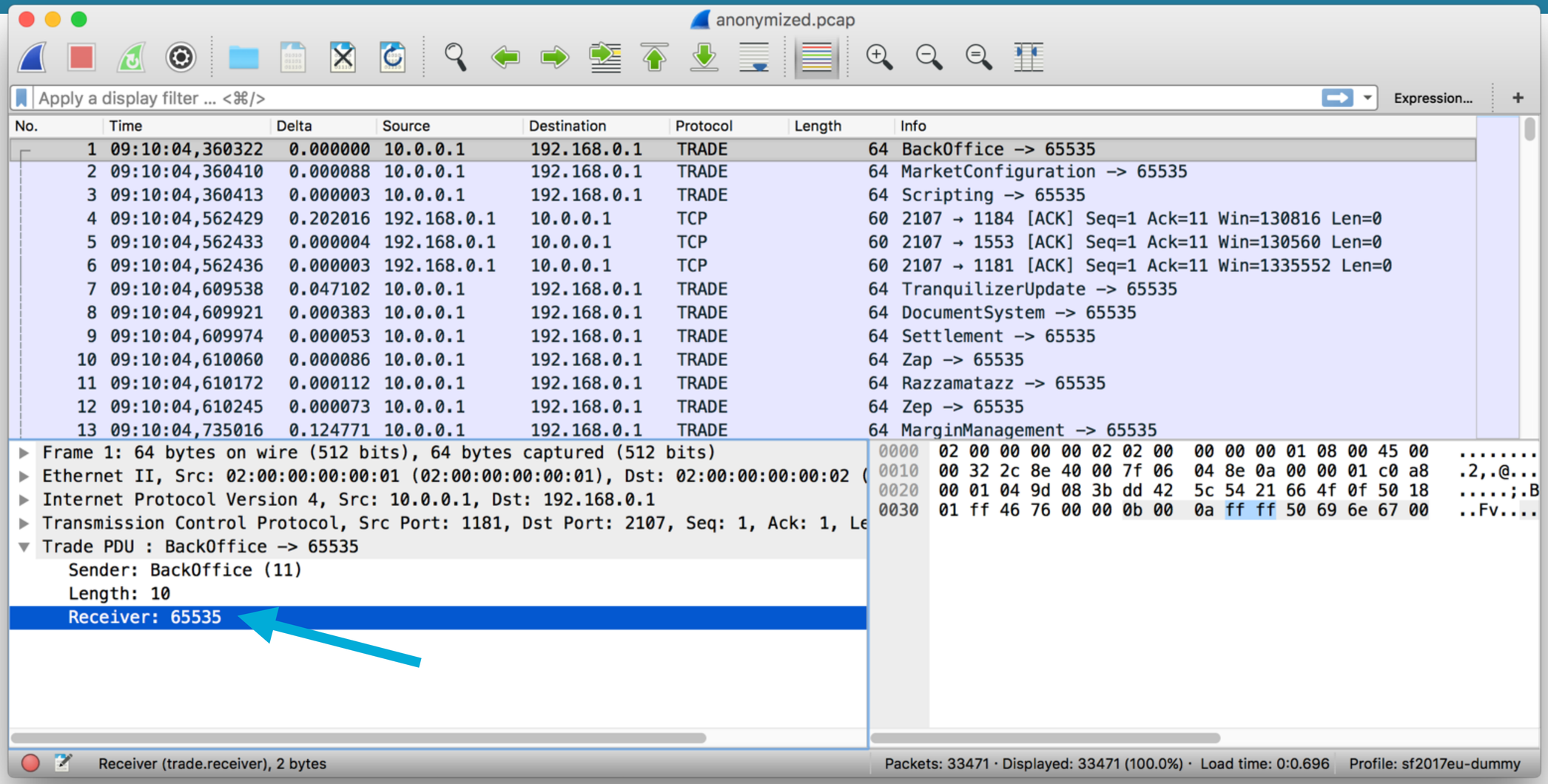
Result of 06.lua



07.lua: Add more fields

```
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$ diff 06.lua 07.lua
23a24,25
>     f.len         = ProtoField.uint16("trade.length","Length",base.DEC)
>     f.receiver    = ProtoField.uint16("trade.receiver","Receiver",base.DEC)
44a47,57
>         -- Add length
>         subtree:add(f.len,tvb(offset,2))
>         offset = offset + 2
>
>         -- Add receiver
>         local receiver = tvb(offset,2):uint()
>         info = info .. " -> " .. receiver
>
>         subtree:add(f.receiver,tvb(offset,2))
>         offset = offset + 2
>
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$
```

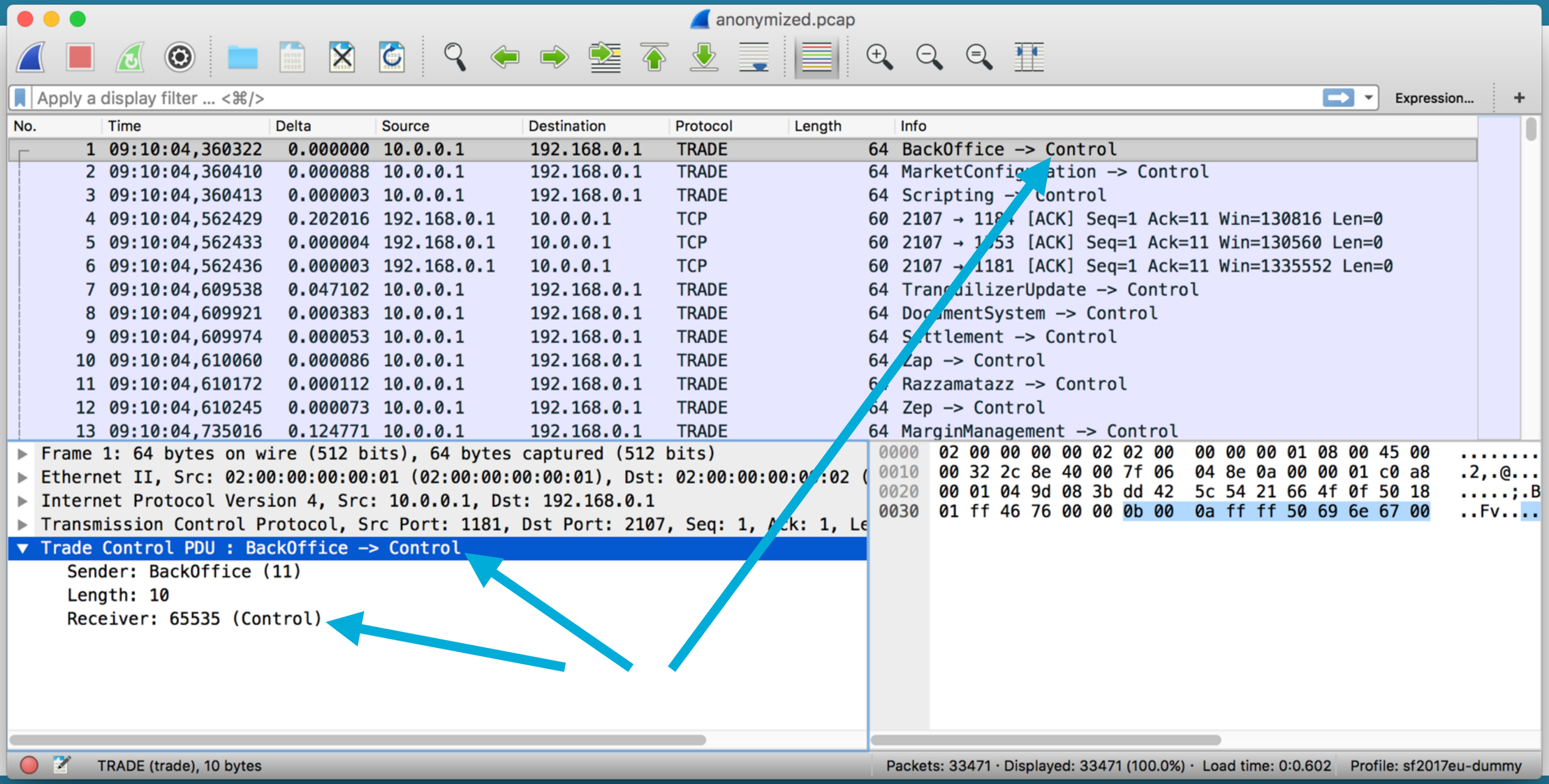
Result of 07.lua



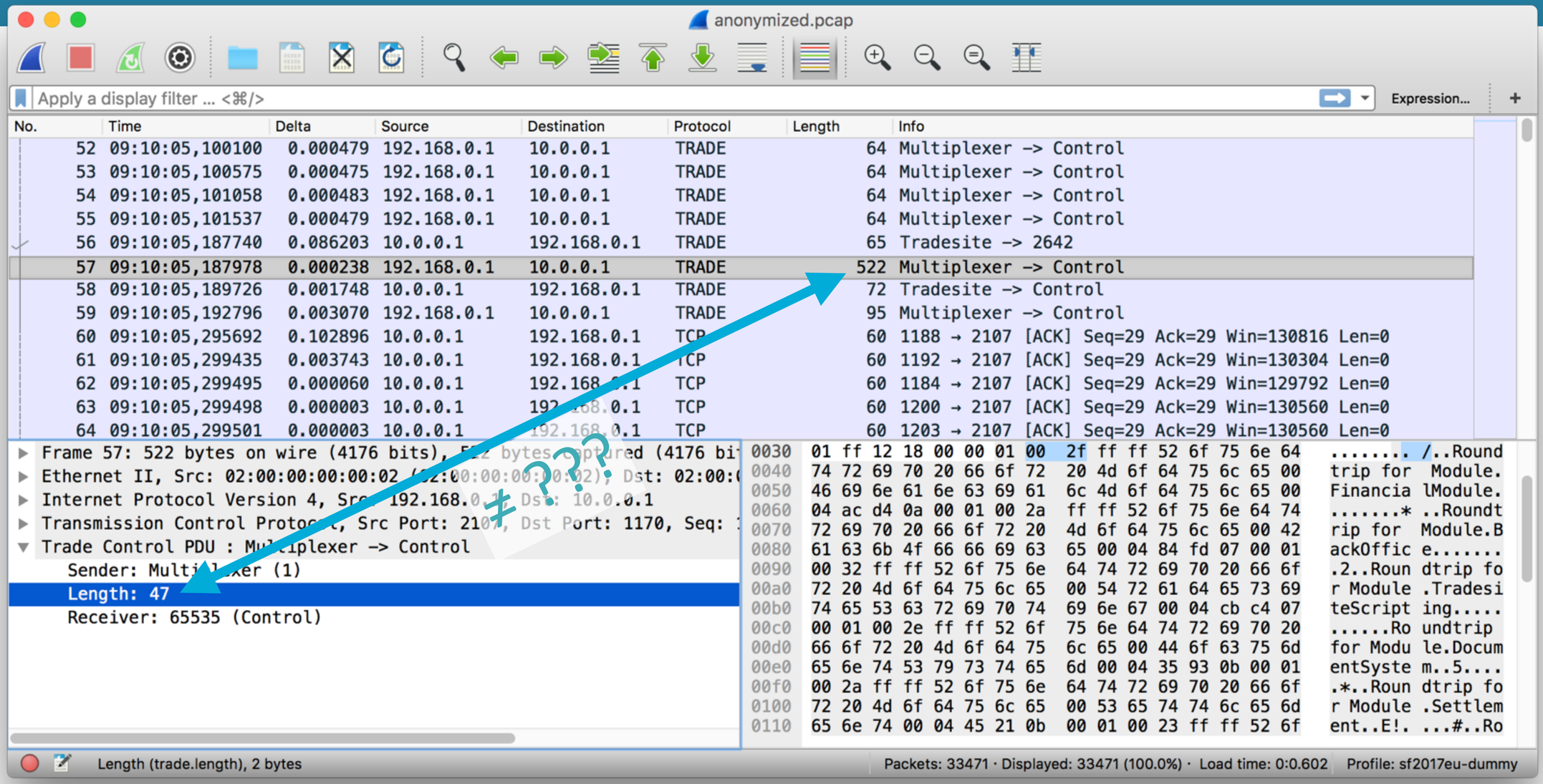
08.lua: Differentiate type of packets

```
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$ diff -b 07.lua 08.lua
36,38d35
<         subtree = tree:add(trade,tvb(offset,tvb:len()),"Trade PDU : ")
<
<         -- Add sender
43a41,49
>         local receiver = tvb(offset+3,2):uint()
>
>         if receiver == 65535 then
>             subtree = tree:add(trade,tvb(offset,tvb:len()),"Trade Control PDU : ")
>         else
>             subtree = tree:add(trade,tvb(offset,tvb:len()),"Trade Data PDU : ")
>         end
>
>         -- Add sender
52,55c58
<         local receiver = tvb(offset,2):uint()
<         info = info .. " -> " .. receiver
<
<         subtree:add(f.receiver,tvb(offset,2))
---
>         local t = subtree:add(f.receiver,tvb(offset,2))
57a61,69
>         if receiver == 65535 then
>             -- Dissect control PDU
>             t:append_text(" (Control)")
>             info = info .. " -> Control"
>         else
>             -- Dissect data PDU
>             info = info .. " -> " .. receiver
>         end
>
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$
```

Result of 08.lua



Multiple PDU's in one packet?



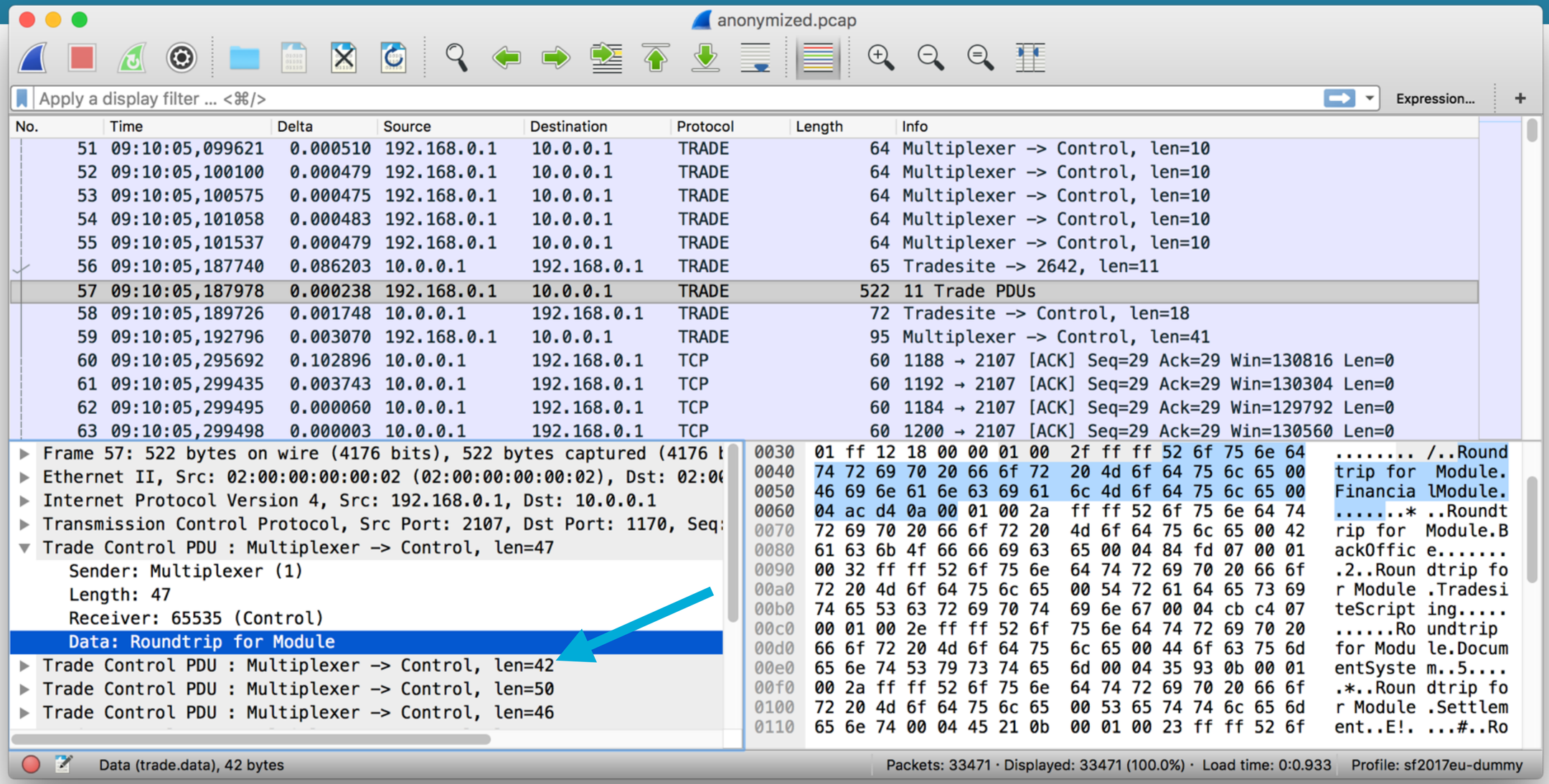
09.lua: Support multiple PDU's

```
repeat
    -- Add sender
    subtree:add(f.sender,tvb(offset,1))
    offset = offset + 1
    -- Add length
    -- Add receiver
    -- Add optional data
until offset >= tvb:len() or count >= 100

if count == 1 then
    pinfo.cols.info = info
else
    pinfo.cols.info = count .. " Trade PDUs"
end
```

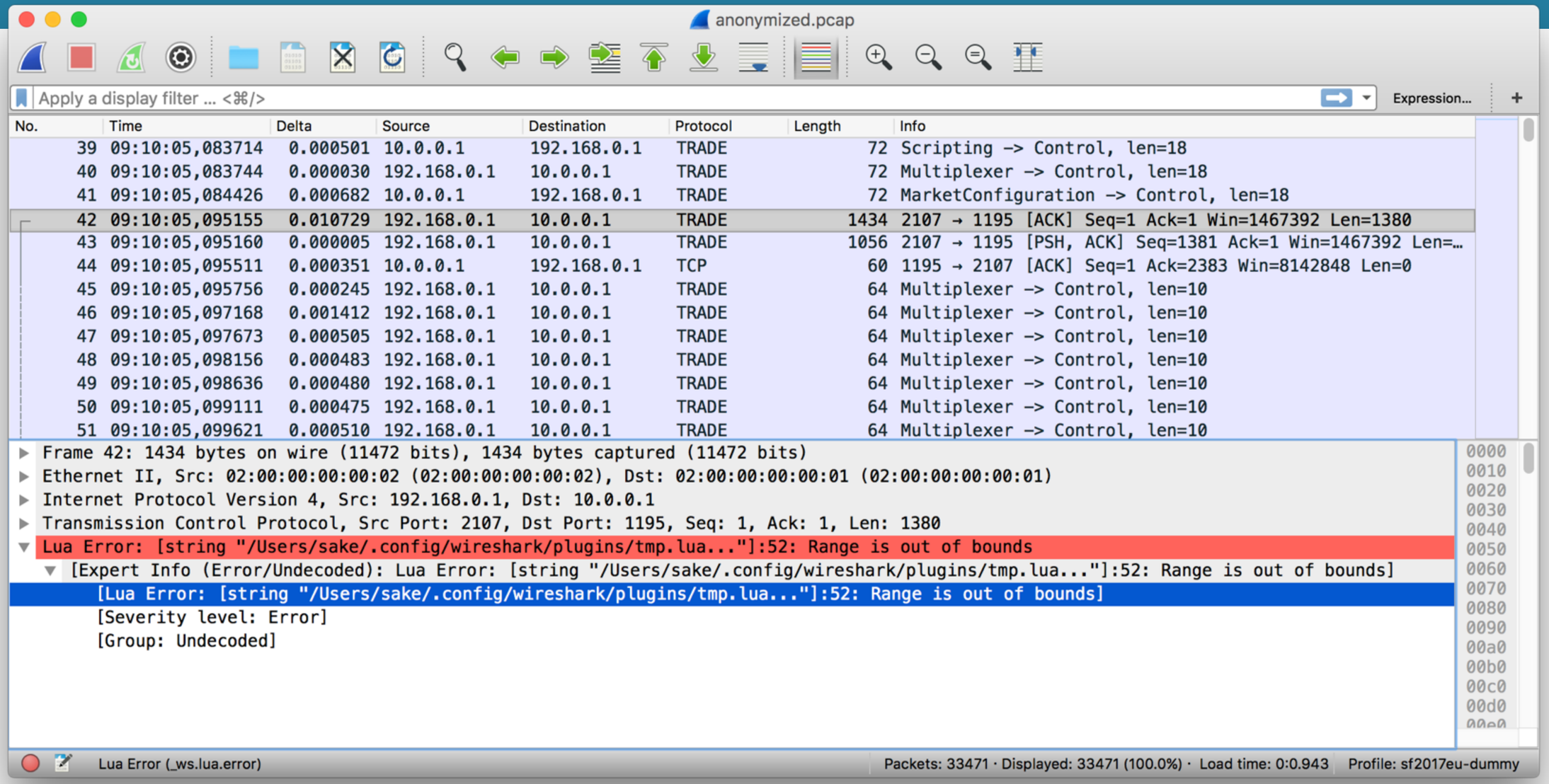
09.lua

```
repeat
    len = tvb(offset+1,2):uint()
    -- Add sender
    subtree:add(f.sender,tvb(offset,1))
    offset = offset + 1
    len = len - 1
    -- Add length etc.

    if len > 0 then
        -- Add data field for unknown data
        local data = tvb(offset,len):bytes()
        subtree:add(f.data,tvb(offset,data:len()))
        offset = offset + data:len()
        len = len - data:len()
    end
  count = count + 1
until offset >= tvb:len() or count >= 100
```

Result of 09.lua



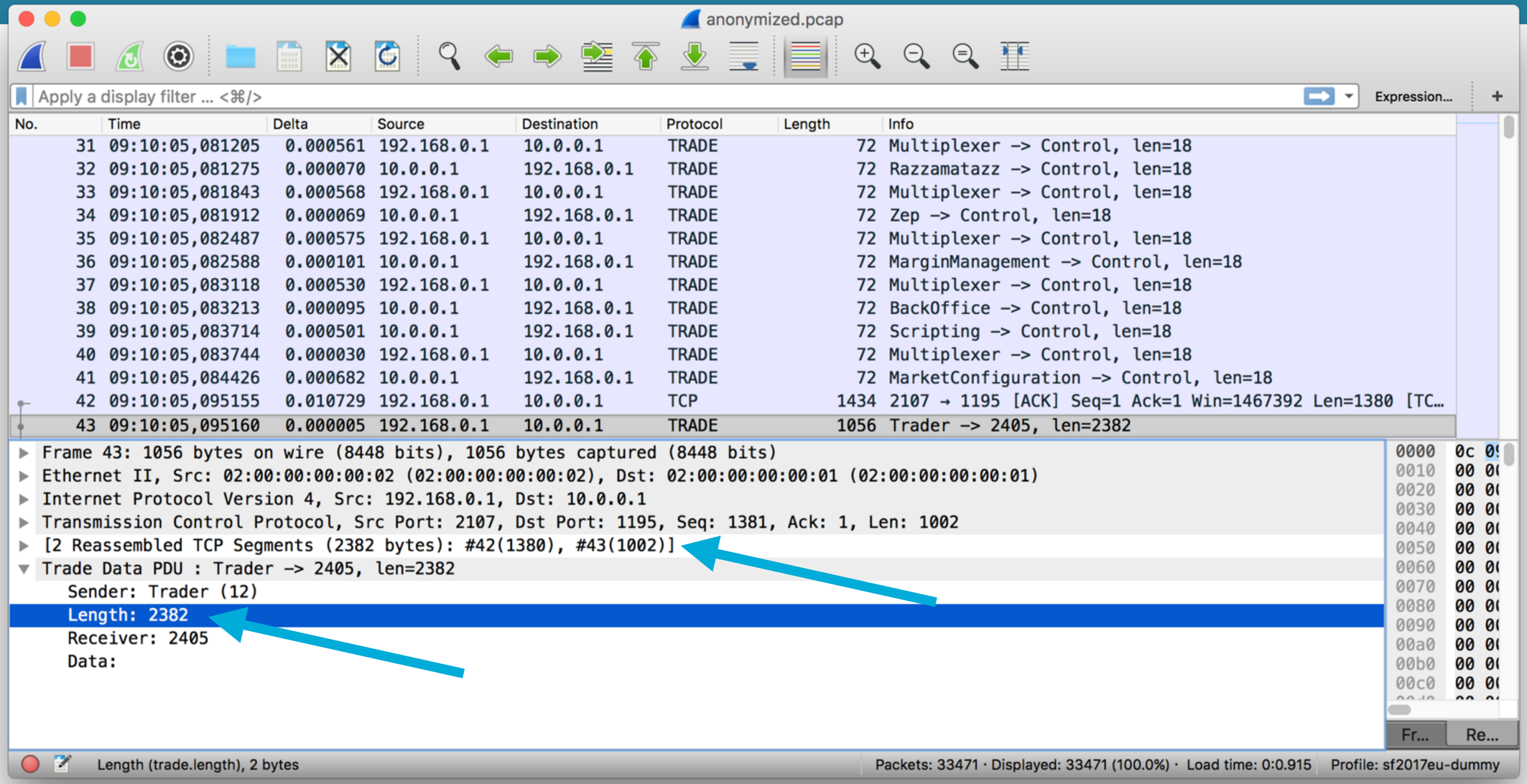
PDU's spanning multiple packets



10.lua: Add reassembly

```
local offset = pinfo.desegment_offset or 0
repeat
    if tvb:len() >= offset + 3 then
        -- we have enough bytes for the length field of the PDU
        len = tvb(offset+1,2):uint()
        if offset + len > tvb:len() then
            -- we don't have all the data we need for the full PDU yet
            pinfo.desegment_offset = offset
            pinfo.desegment_len = len - tvb:len()
            return
        end
    else
        -- we don't have all of length field yet
        pinfo.desegment_offset = offset
        pinfo.desegment_len = DESEGMENT_ONE_MORE_SEGMENT
        return
    end
```

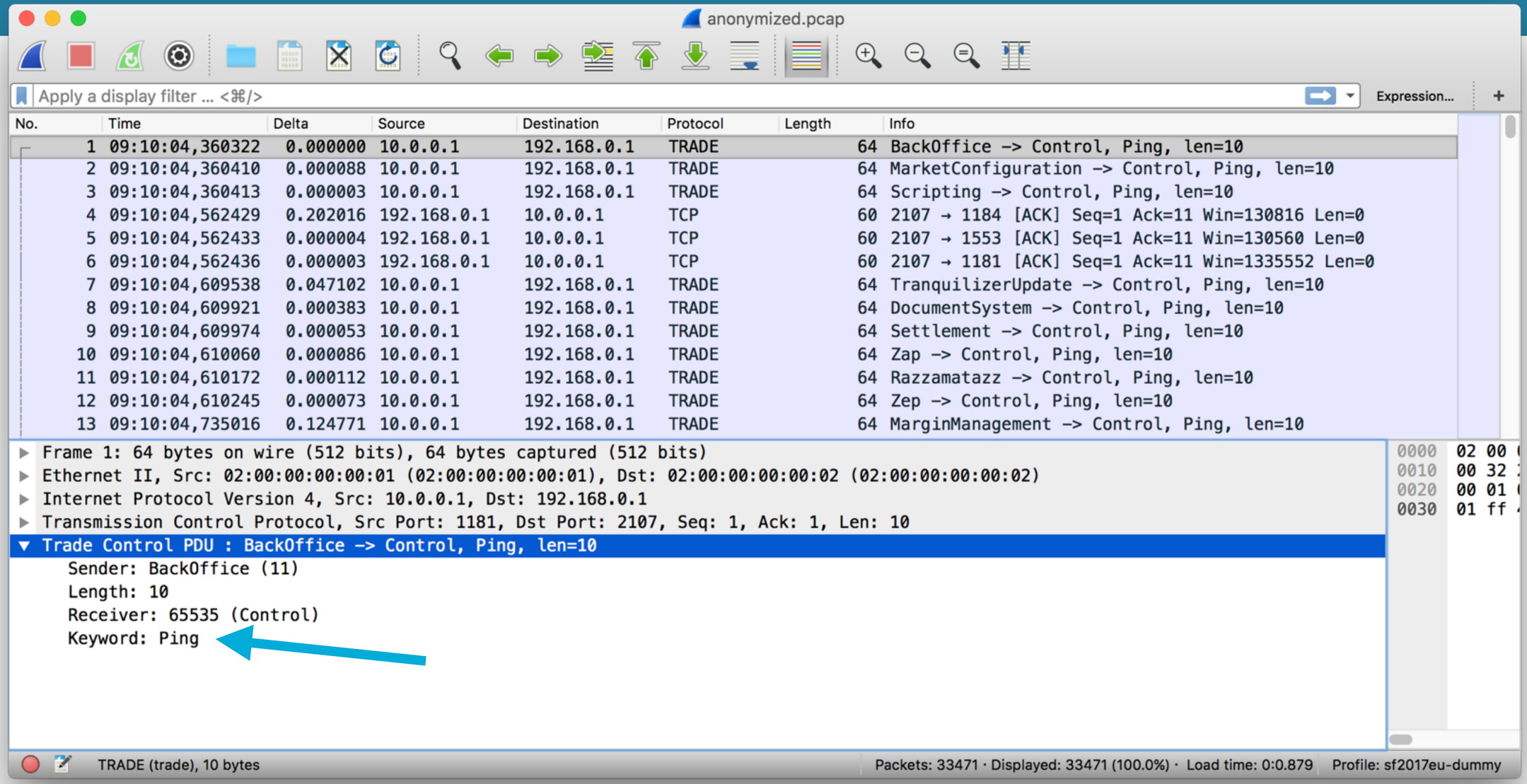
Result of 10.lua



11.lua: Add more fields

```
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$ diff 10.lua 11.lua
25a26
>     f.keyword     = ProtoField.stringz("trade.keyword","Keyword")
88a90,101
>
>                 -- Add keyword
>                 local keyword = tvb(offset):stringz()
>                 if keyword:len() > 0 then
>                     subtree:add(f.keyword,tvb(offset,keyword:len()))
>                     info = info .. ", " .. keyword
>                     offset = offset + keyword:len() + 1
>                     len = len - keyword:len() - 1
>                 else
>                     -- there was no keyword (first byte was 0x00)
>                     subtree:add_expert_info(PI_MALFORMED,PI_ERROR,"There was no
keyword in this control frame")
>                 end
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$
```

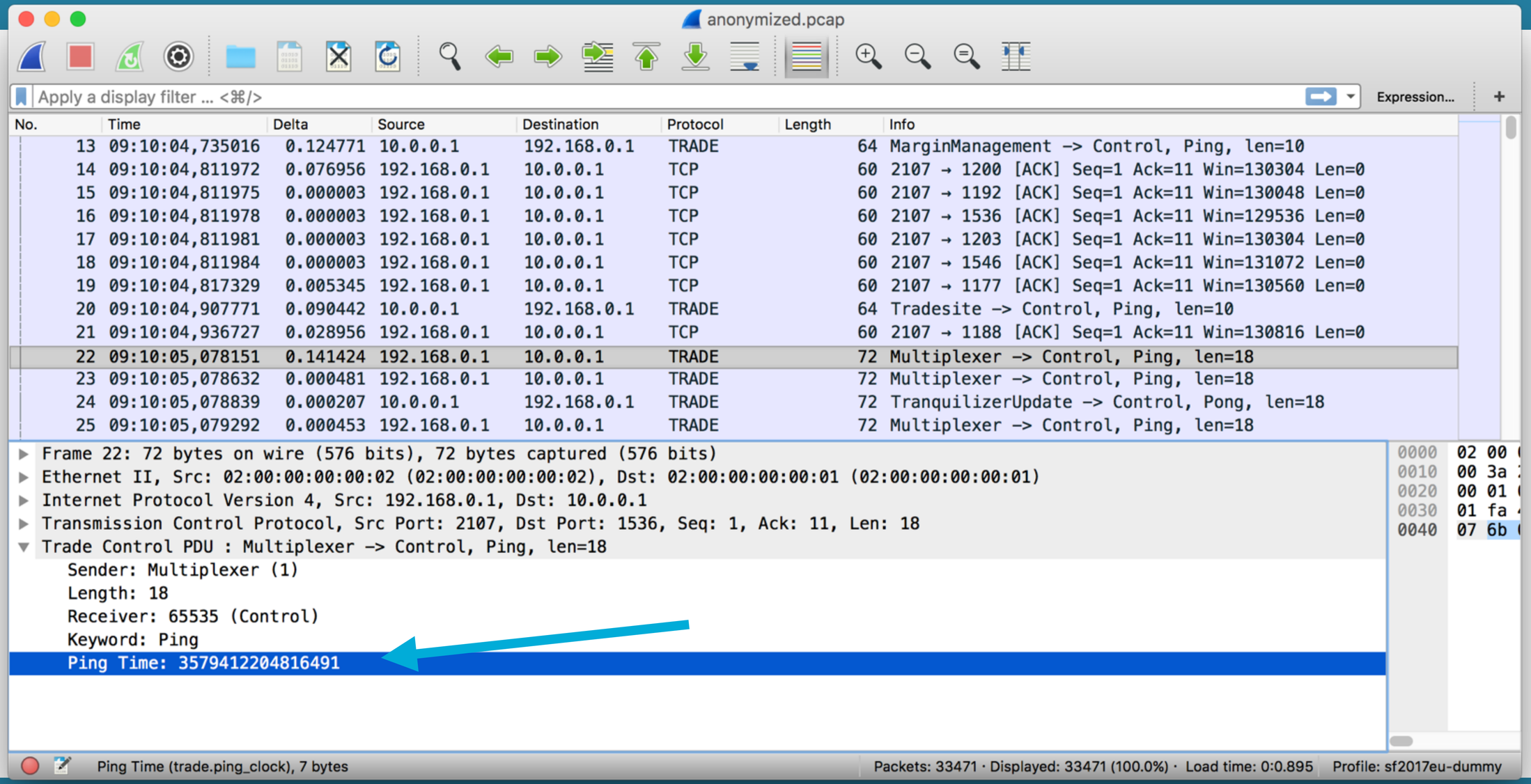
Result of 11.lua



12.lua: Add more fields

```
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$ diff -b 11.lua 12.lua
26a27
>     f.ping_clock = ProtoField.uint64("trade.ping_clock","Ping Time")
95c96
<                     info = info .. ", " .. keyword
---
>                     info = info .. ", "
97a99,114
>                     if len > 0 then
>                         if keyword == "Ping" or
>                            keyword == "Pong" then
>                             -- length (1 byte)
>                             -- time in microseconds (since Jan 1, 1901, see http://en.wikipedia.org/wiki/System_time)
>                             local clocklen = tvb(offset,1):uint()
>                             subtree:add_le(f.ping_clock,tvb(offset+1,clocklen))
>                             offset = offset + clocklen + 1
>                             len = len - clocklen - 1
>                             info = info .. keyword
>                         else
>                             info = info .. keyword .. " (Unknown keyword)"
>                         end
>                     else
>                         info = info .. keyword
>                     end
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$
```

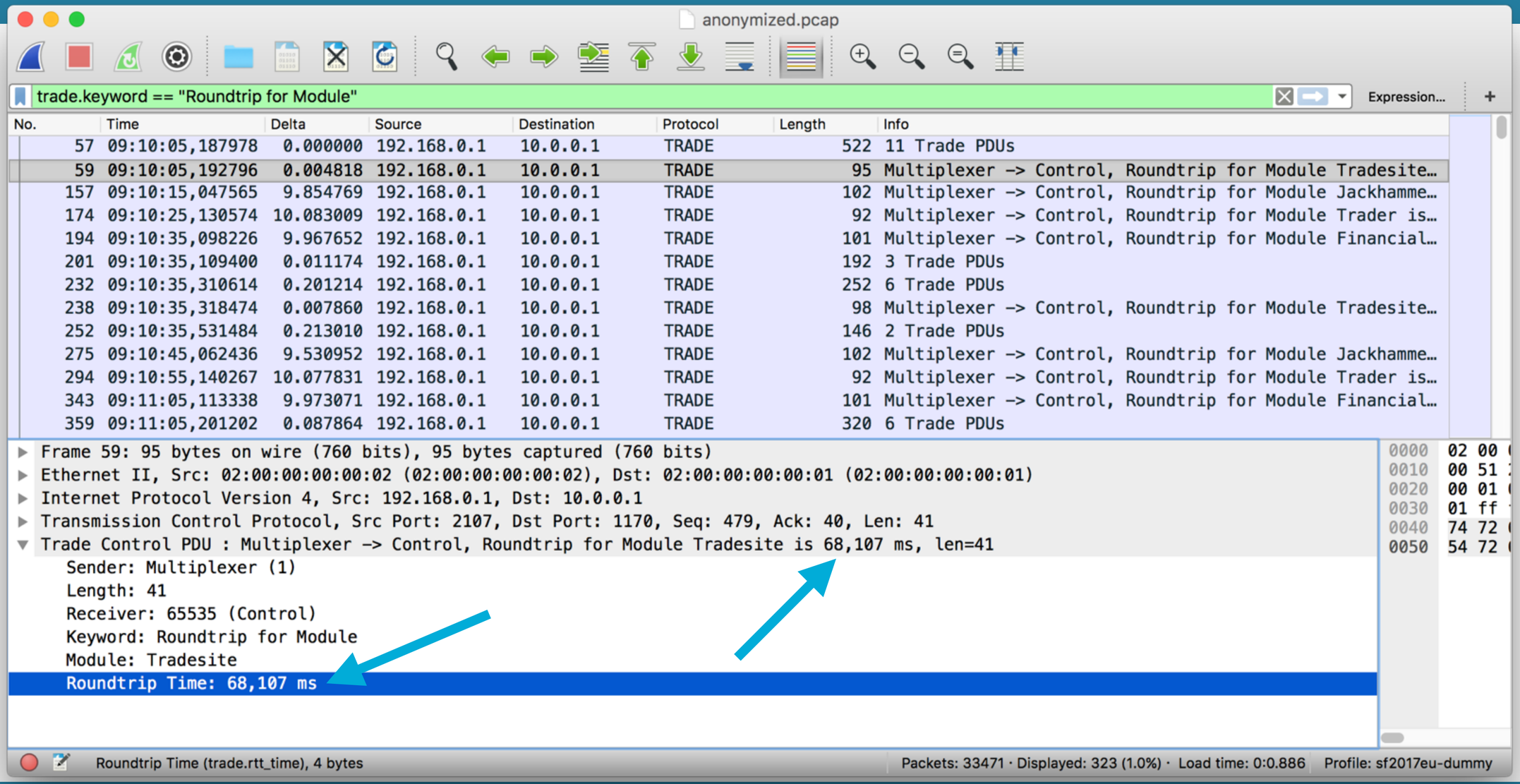
Result of 12.lua



13.lua: Add more fields

```
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$ diff -b 12.lua 13.lua
27a28,29
>     f.rtt_time    = ProtoField.double("trade.rtt_time","Roundtrip Time")
>     f.mod         = ProtoField.stringz("trade.module","Module")
108a111,131
>
>                         elseif keyword == "Roundtrip for Module" then
>                             -- string with module name
>                             -- length (1 byte)
>                             -- time in milliseconds
>                             local mod = tvb(offset):stringz()
>                             subtree:add(f.mod,tvb(offset,mod:len()))
>                             offset = offset + mod:len() + 1
>                             len = len - mod:len() - 1
>
>                             local clocklen = tvb(offset,1):uint()
>                             assert(clocklen<=4, "clocklength too big")
>
>                             local rtt_time = tvb(offset+1,clocklen):le_uint()
>                             local t = subtree:add(f.rtt_time,tvb(offset+1,clocklen),rtt_time/1000)
>                             t:append_text(" ms")
>                             offset = offset + clocklen + 1
>                             len = len - clocklen - 1
>
>                             info = info .. "Roundtrip for Module " .. mod .. " is " .. rtt_time/1000 .. " ms"
>
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$
```

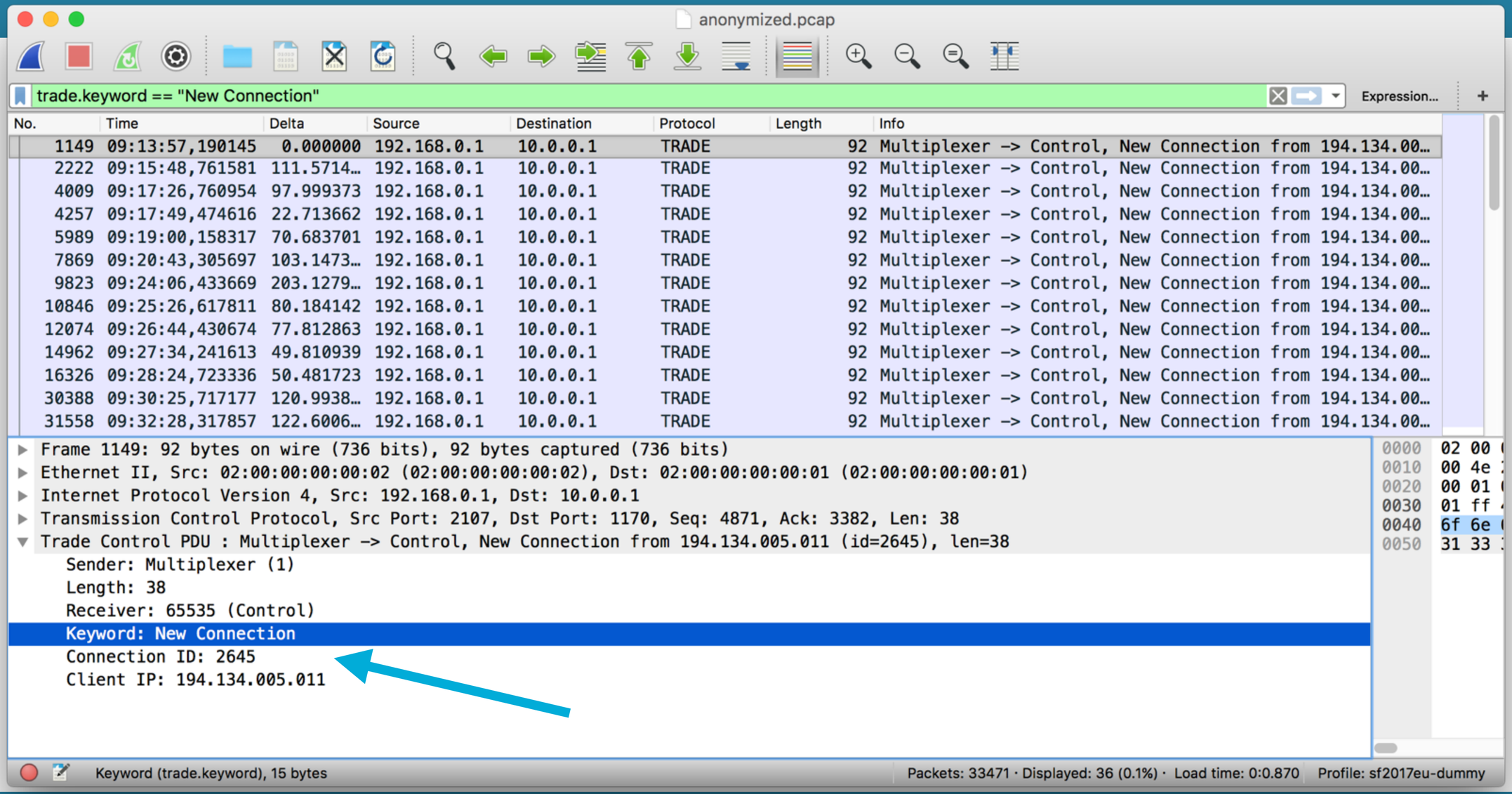
Result of 13.lua



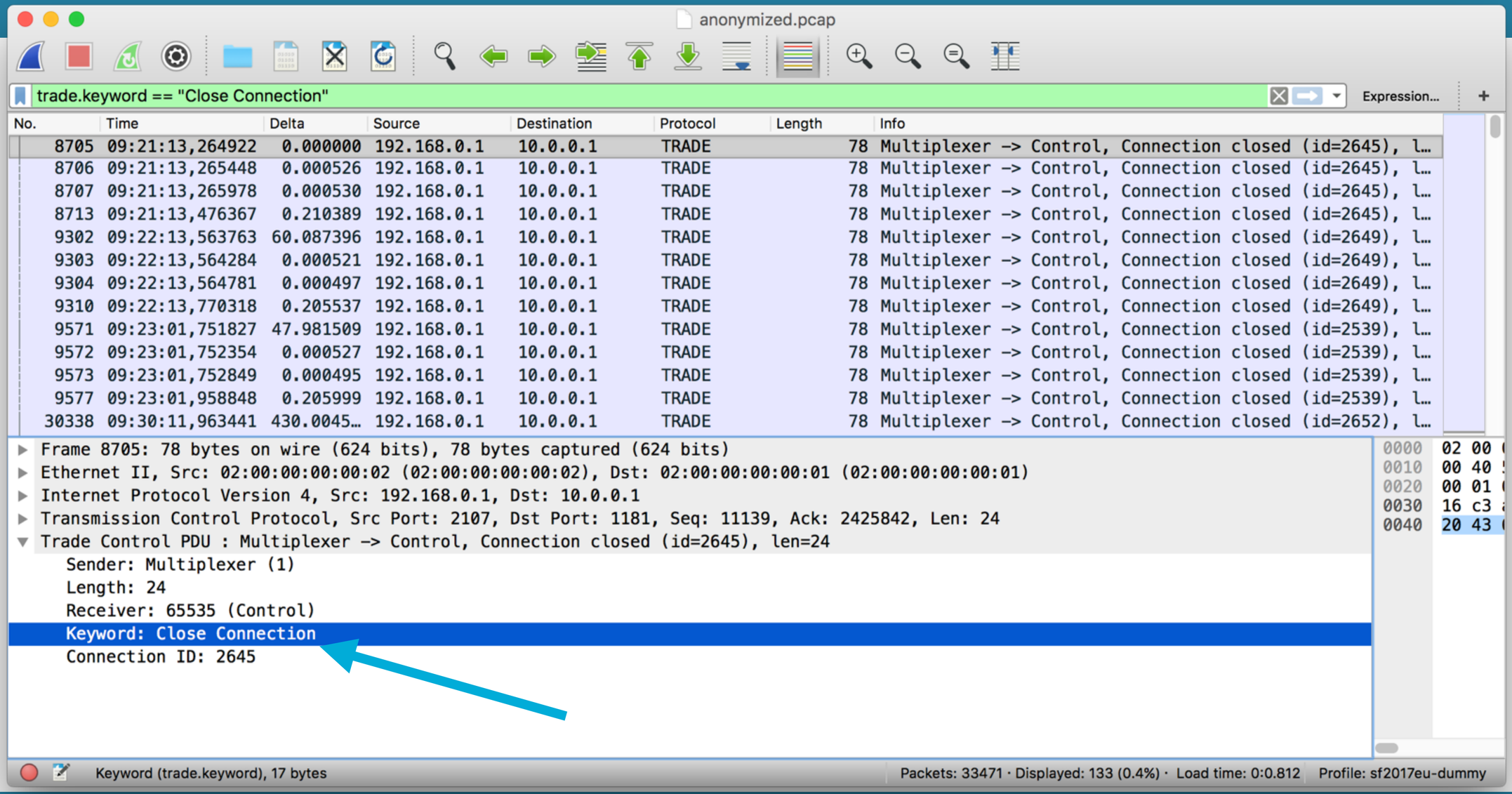
14.lua: Add more fields

```
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$ diff -b 13.lua 14.lua
29a30,31
>     f.conn_id    = ProtoField.uint16("trade.conn_id","Connection ID",base.DEC)
>     f.client_ip  = ProtoField.stringz("trade.client_ip","Client IP")
131a134,157
>                         elseif keyword == "New Connection" then
>                             -- 2 bytes connection ID
>                             -- string with IP address
>                             local conn_id = tvb(offset,2):uint()
>                             subtree:add(f.conn_id,tvb(offset,2))
>                             offset = offset + 2
>                             len = len - 2
>
>                             local ip = tvb(offset):stringz()
>                             subtree:add(f.client_ip,tvb(offset,ip:len()))
>                             offset = offset + ip:len() + 1
>                             len = len - ip:len() - 1
>
>                             info = info .. "New Connection from " .. ip .. " (id=" .. conn_id .. ")"
>
>                         elseif keyword == "Close Connection" then
>                             -- 2 bytes connection ID
>                             local conn_id = tvb(offset,2):uint()
>                             subtree:add(f.conn_id,tvb(offset,2))
>                             offset = offset + 2
>                             len = len - 2
>
>                             info = info .. "Connection closed (id=" .. conn_id .. ")"
>
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$
```

Result of 14.lua



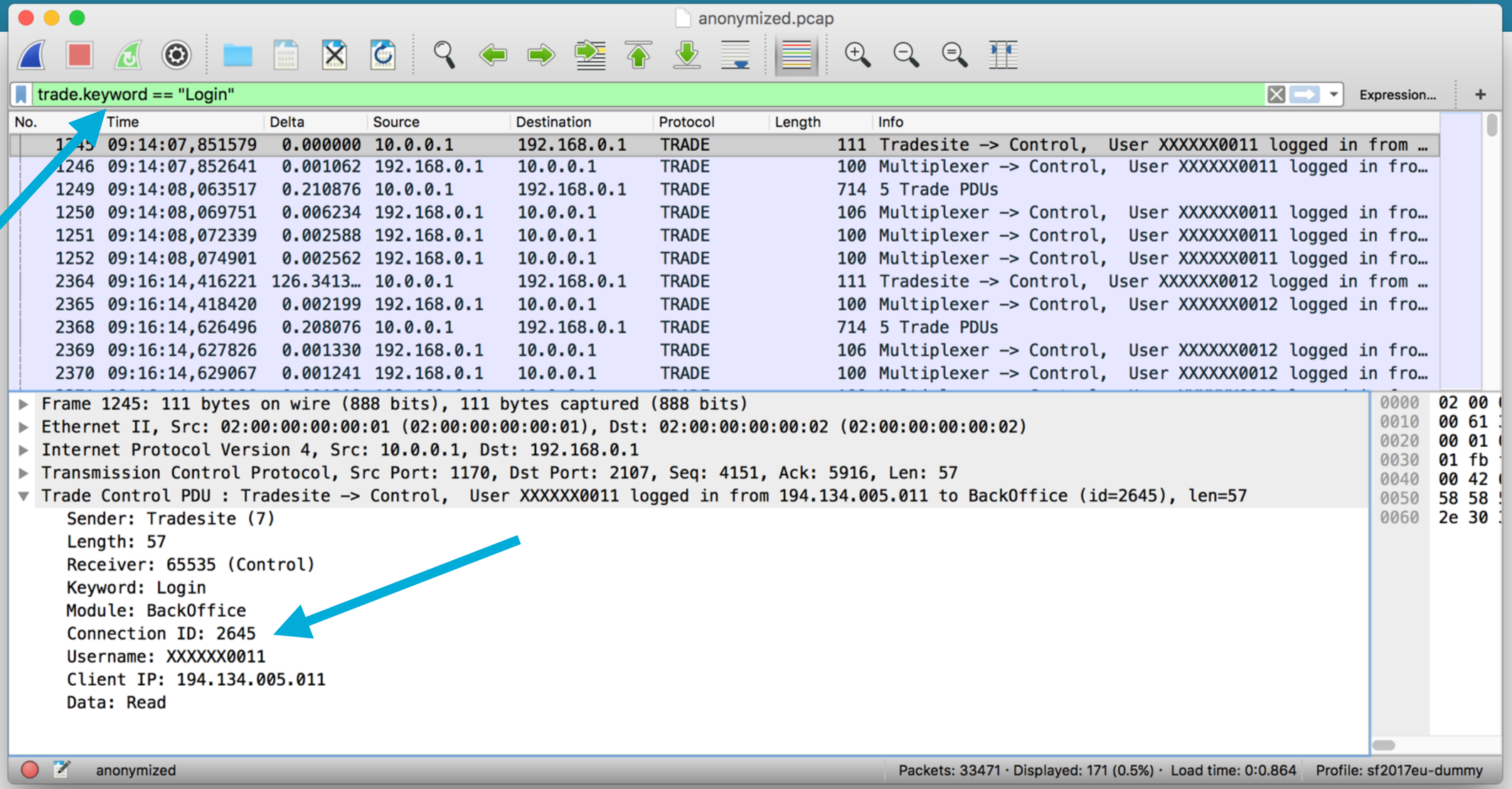
Result of 14.lua



15.lua: Add more fields

```
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$ diff -b 14.lua 15.lua
31a32
>     f.username    = ProtoField.stringz("trade.username","Username")
157a159,195
>                         elseif keyword == "Login" then
>                             -- string with module (by sender Sylvester (7))
>                             -- 2 bytes connection ID
>                             -- string with username
>                             -- string with IP address
>                             -- "read"
>                             -- 0x00
>                             local mod
>
>                             if sender == 7 then
>                                 mod = tvb(offset):stringz()
>                                 subtree:add(f.mod,tvb(offset,mod:len()))
>                                 offset = offset + mod:len() + 1
>                                 len = len - mod:len() - 1
>                             end
>
>                             local conn_id = tvb(offset,2):uint()
>                             subtree:add(f.conn_id,tvb(offset,2))
>                             offset = offset + 2
>                             len = len - 2
>
>                             local username = tvb(offset):stringz()
>                             subtree:add(f.username,tvb(offset,username:len()))
>                             offset = offset + username:len() + 1
>                             len = len - username:len() - 1
>
>                             local ip = tvb(offset):stringz()
>                             subtree:add(f.client_ip,tvb(offset,ip:len()))
>                             offset = offset + ip:len() + 1
>                             len = len - ip:len() - 1
>
>                             if sender == 7 then
>                                 info = info .. " User " .. username .. " logged in from " .. ip .. " to " .. mod .. " (id=" .. conn_id .. ")"
>                             else
>                                 info = info .. " User " .. username .. " logged in from " .. ip .. " (id=" .. conn_id .. ")"
>                             end
>
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$
```

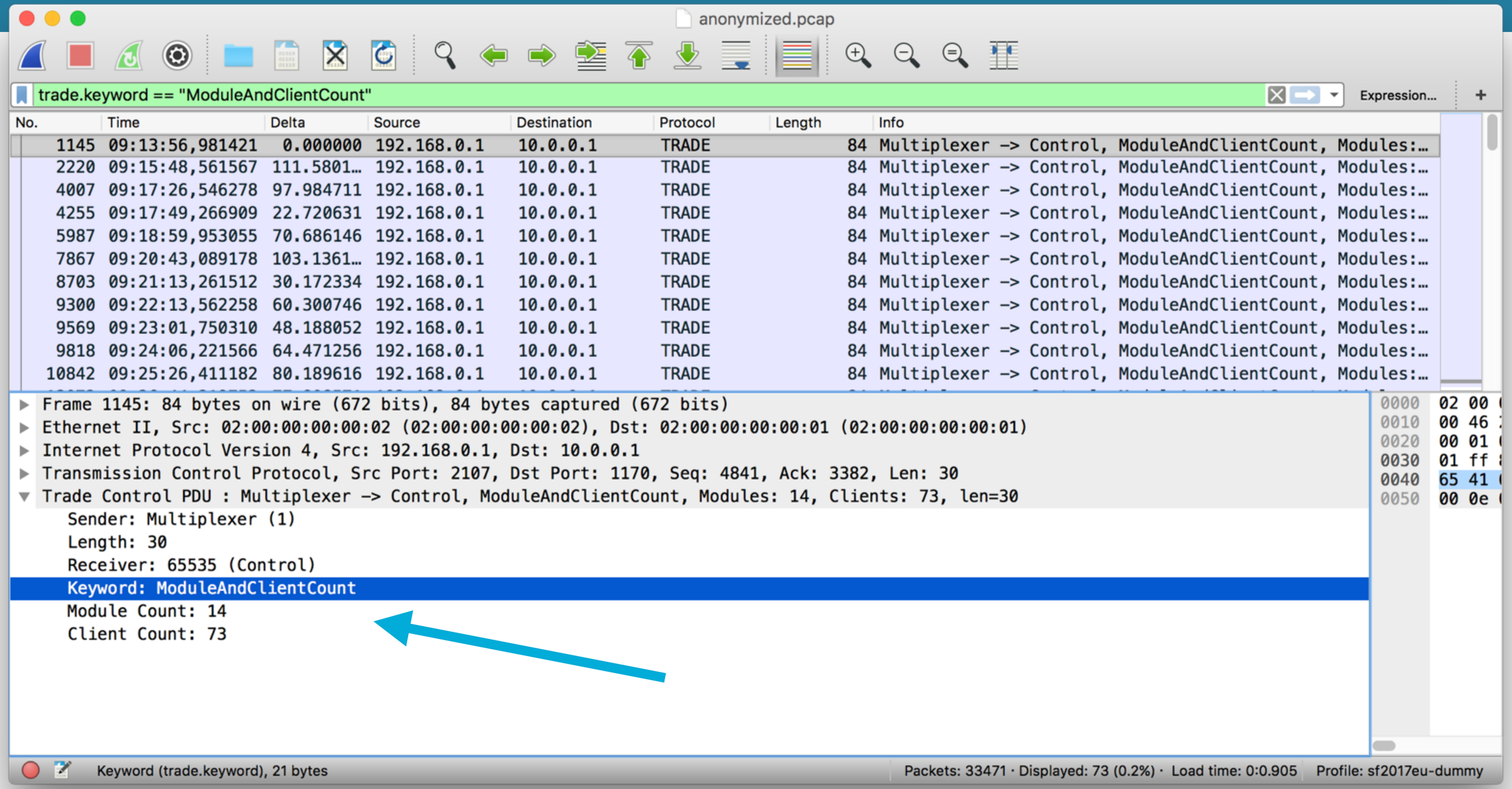
Result of 15.lua



16.lua: Add more fields

```
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$ diff -b 15.lua 16.lua
32a33,35
>     f.module_count  = ProtoField.uint16("trade.module_count","Module Count",base.DEC)
>     f.client_count  = ProtoField.uint16("trade.client_count","Client Count",base.DEC)
>     f.status        = ProtoField.stringz("trade.status","Status")
41c44
<         local offset = pinfo.desegment_offset or 0
---
>         local offset = 0
195a199,219
>                         elseif keyword == "ModuleAndClientCount" then
>                             local module_count = tvb(offset,2):uint()
>                             subtree:add(f.module_count,tvb(offset,2))
>                             offset = offset + 2
>                             len = len - 2
>
>                             local client_count = tvb(offset,2):uint()
>                             subtree:add(f.client_count,tvb(offset,2))
>                             offset = offset + 2
>                             len = len - 2
>
>                             info = info .. keyword .. ", Modules: " .. module_count .. ", Clients: " .. client_count
>
>                         elseif keyword == "Status" then
>                             local status = tvb(offset):stringz()
>                             subtree:add(f.status,tvb(offset,status:len()))
>                             offset = offset + status:len() + 1
>                             len = len - status:len() - 1
>
>                             info = info .. keyword .. ": " .. status
>
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$
```

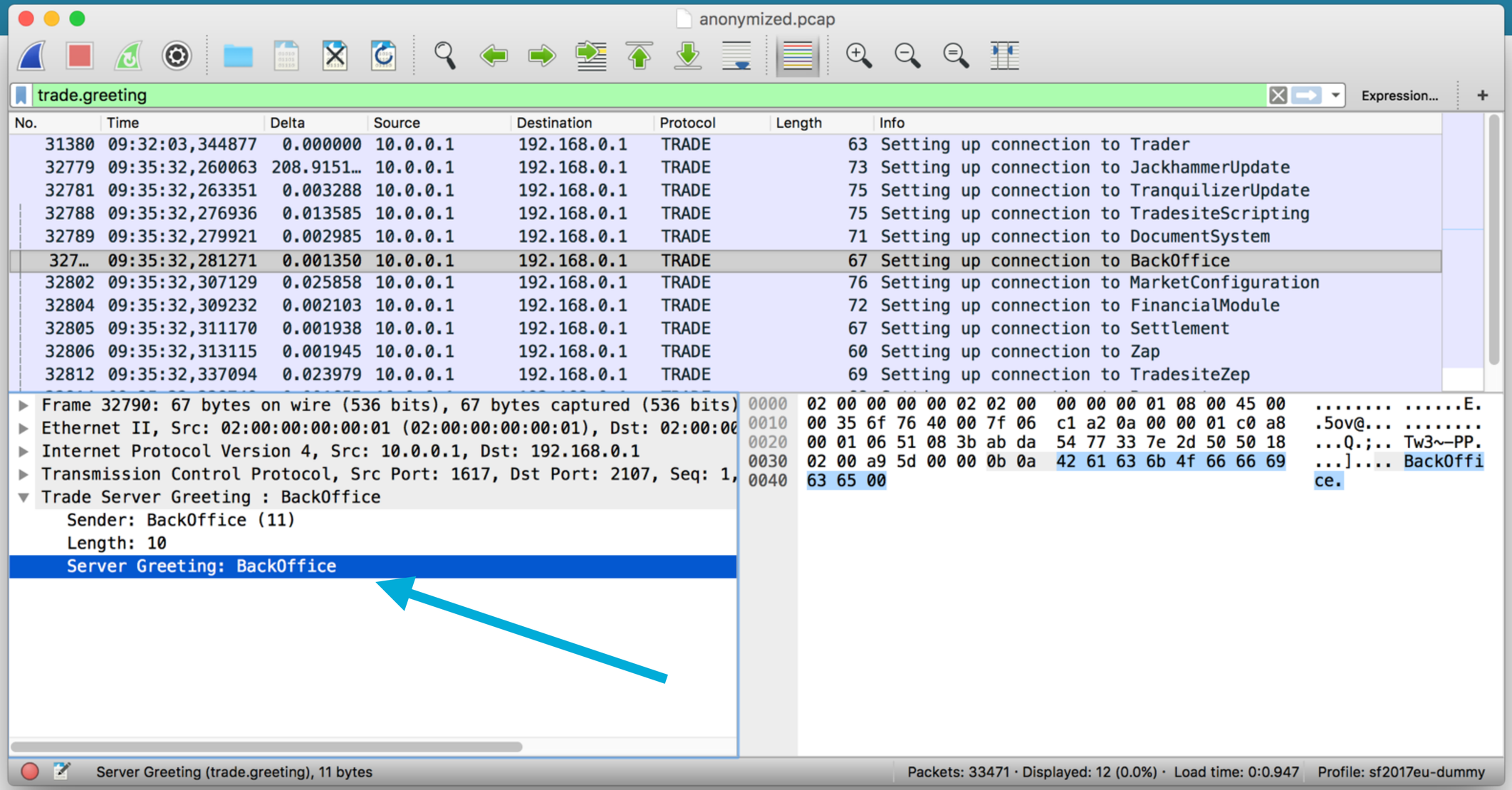
Result of 16.lua



17.lua: Use fields from other layers

```
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$ diff -b 16.lua 17.lua
4a5,6
>     local tcp_seq_f = Field.new("tcp.seq")
>
29a32
>     f.greeting      = ProtoField.stringz("trade.greeting","Server Greeting")
44c47
<         local offset = 0
---
>         local offset = pinfo.desegment_offset or 0
54a58,71
>             -- Handle the server greeting message
>             local tcp_seq = tostring(tcp_seq_f())
>             len = tvb(1,1):uint()
>             if tcp_seq == "1" and (tvb:len() == len + 3) then
>                 subtree = tree:add(trade,tvb(0,len),"Trade Server Greeting : ")
>                 subtree:add(f.sender,tvb(0,1))
>                 subtree:add(f.len,tvb(1,1))
>                 subtree:add(f.greeting,tvb(2,len - 2))
>                 local mod = tvb(2,len - 2):stringz()
>                 subtree:append_text(mod)
>                 pinfo.cols.info = "Setting up connection to " .. mod
>                 return len
>             end
>
sake@MacSake:~/Dropbox/sharkfest/2017eu/anonymized$
```

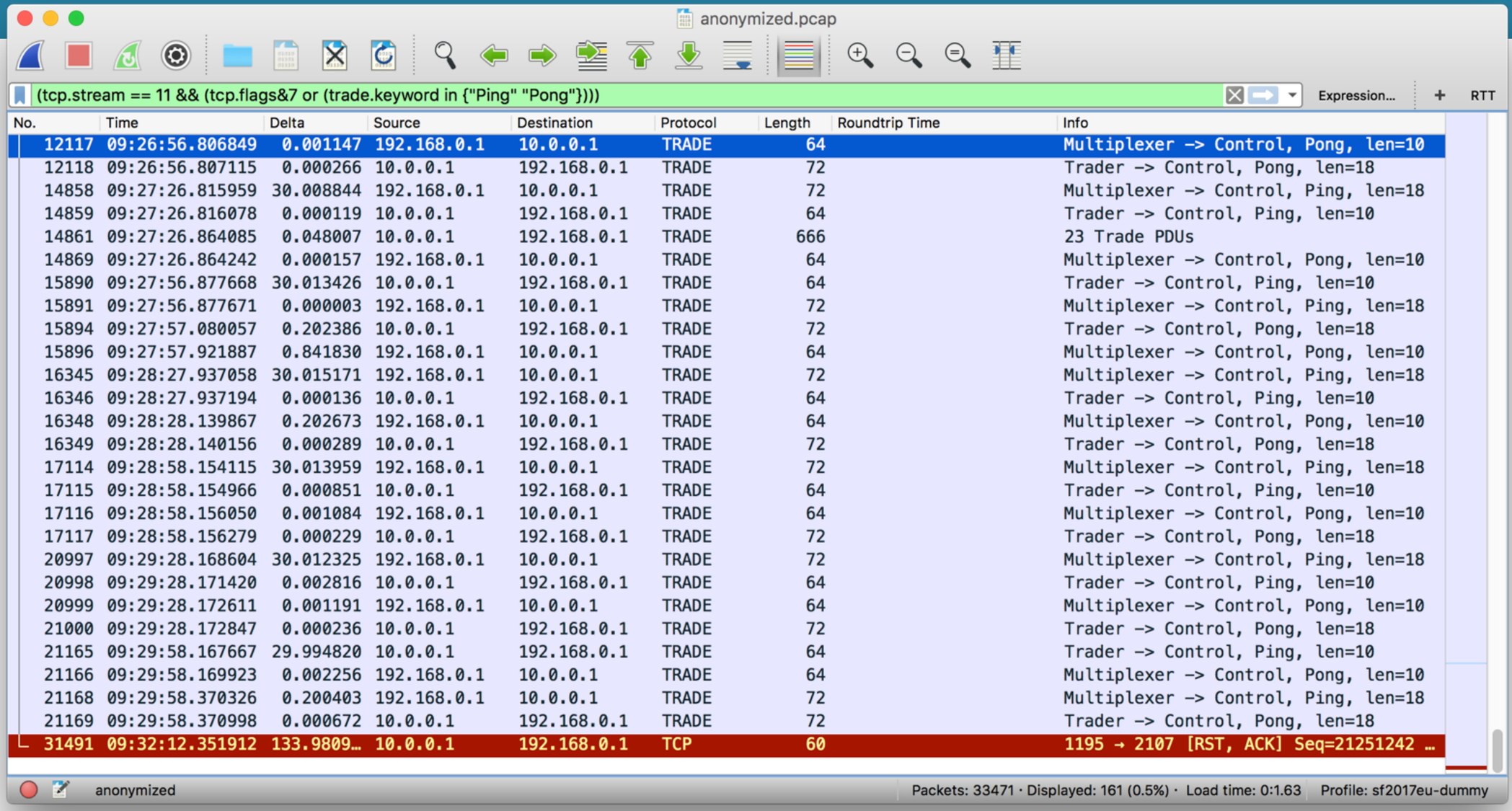
Result of 17.lua



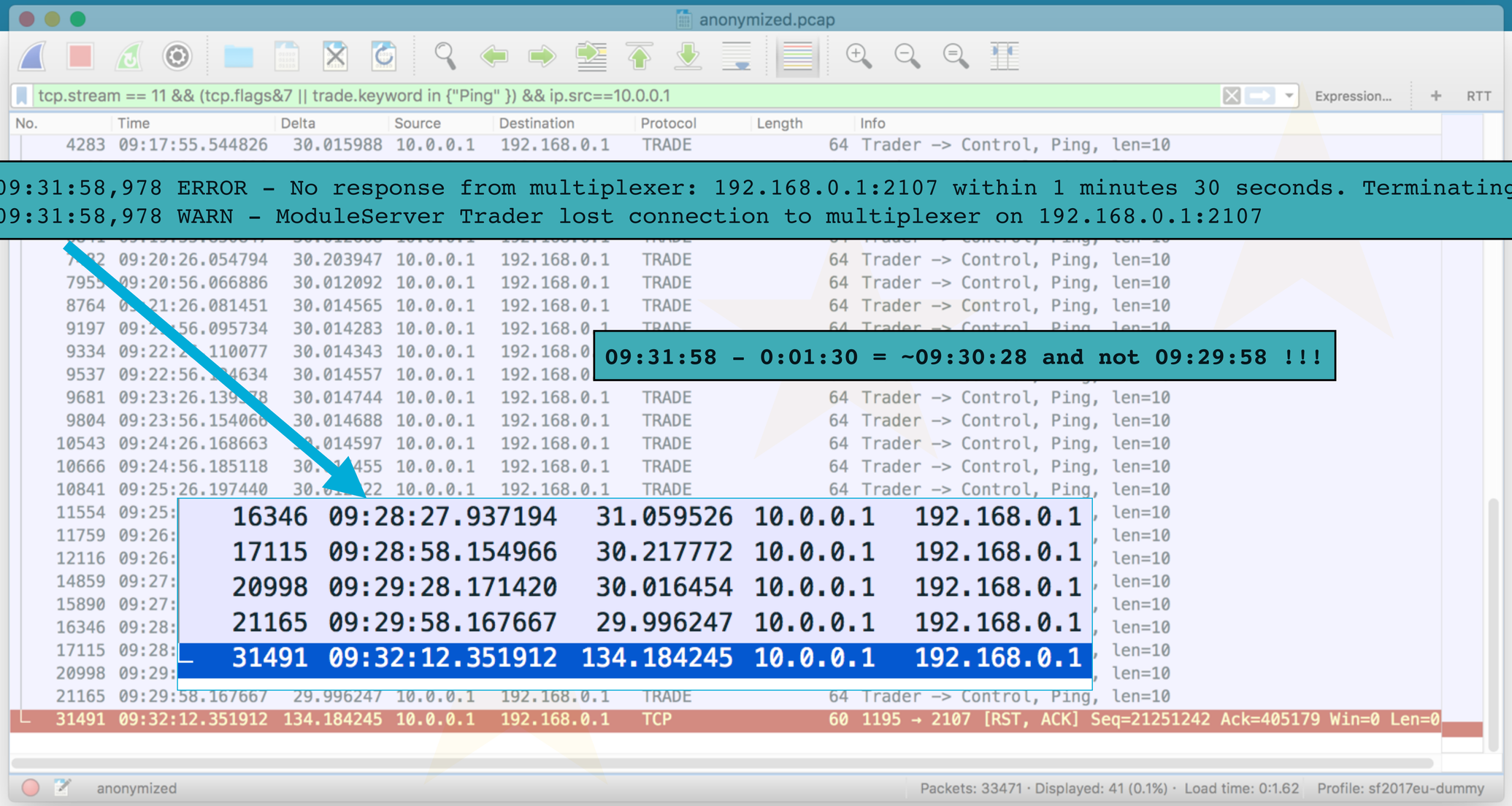
Road taken

- Create a LUA dissector skeleton
- Add fields, fill info column and subtree for the basic PDU header
- Add multi-PDU support
- Add reassembly
- Dissect more parts of the PDU's
- Now back to troubleshooting!!!

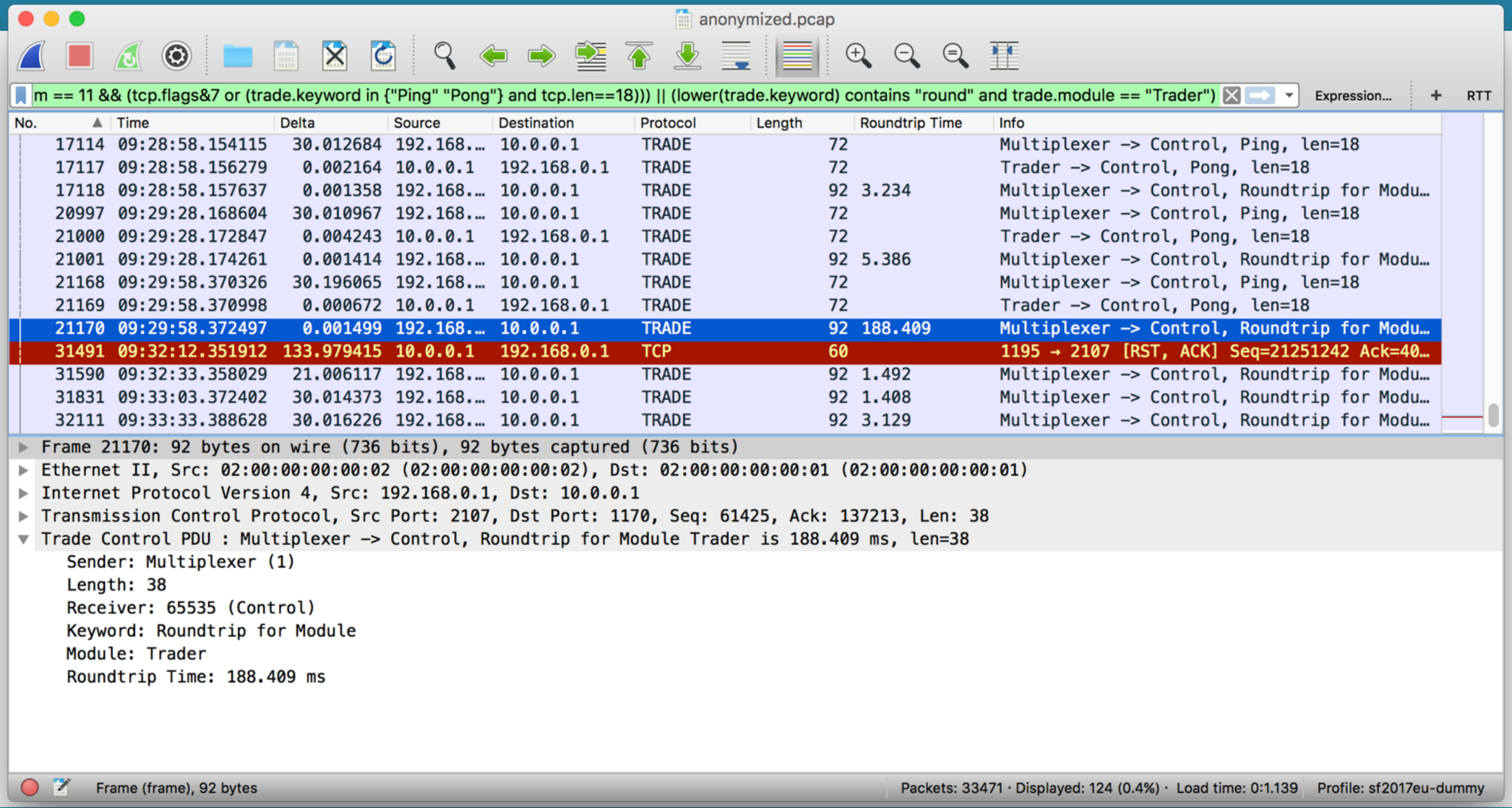
Ping-Pong?


anonymized.pcap
(tcp.stream == 11 && (tcp.flags&7 or (trade.keyword in {"Ping" "Pong"})))
Expression...
RTT
No. Time Delta Source Destination Protocol Length Roundtrip Time Info
12117 09:26:56.806849 0.001147 192.168.0.1 10.0.0.1 TRADE 64 Multiplexer -> Control, Pong, len=10
12118 09:26:56.807115 0.000266 10.0.0.1 192.168.0.1 TRADE 72 Trader -> Control, Pong, len=18
14858 09:27:26.815959 30.008844 192.168.0.1 10.0.0.1 TRADE 72 Multiplexer -> Control, Ping, len=18
14859 09:27:26.816078 0.000119 10.0.0.1 192.168.0.1 TRADE 64 Trader -> Control, Ping, len=10
14861 09:27:26.864085 0.048007 10.0.0.1 192.168.0.1 TRADE 666 23 Trade PDUs
14869 09:27:26.864242 0.000157 192.168.0.1 10.0.0.1 TRADE 64 Multiplexer -> Control, Pong, len=10
15890 09:27:56.877668 30.013426 10.0.0.1 192.168.0.1 TRADE 64 Trader -> Control, Ping, len=10
15891 09:27:56.877671 0.000003 192.168.0.1 10.0.0.1 TRADE 72 Multiplexer -> Control, Ping, len=18
15894 09:27:57.080057 0.202386 10.0.0.1 192.168.0.1 TRADE 72 Trader -> Control, Pong, len=18
15896 09:27:57.921887 0.841830 192.168.0.1 10.0.0.1 TRADE 64 Multiplexer -> Control, Pong, len=10
16345 09:28:27.937058 30.015171 192.168.0.1 10.0.0.1 TRADE 72 Multiplexer -> Control, Ping, len=18
16346 09:28:27.937194 0.000136 10.0.0.1 192.168.0.1 TRADE 64 Trader -> Control, Ping, len=10
16348 09:28:28.139867 0.202673 192.168.0.1 10.0.0.1 TRADE 64 Multiplexer -> Control, Pong, len=10
16349 09:28:28.140156 0.000289 10.0.0.1 192.168.0.1 TRADE 72 Trader -> Control, Pong, len=18
17114 09:28:58.154115 30.013959 192.168.0.1 10.0.0.1 TRADE 72 Multiplexer -> Control, Ping, len=18
17115 09:28:58.154966 0.000851 10.0.0.1 192.168.0.1 TRADE 64 Trader -> Control, Ping, len=10
17116 09:28:58.156050 0.001084 192.168.0.1 10.0.0.1 TRADE 64 Multiplexer -> Control, Pong, len=10
17117 09:28:58.156279 0.000229 10.0.0.1 192.168.0.1 TRADE 72 Trader -> Control, Pong, len=18
20997 09:29:28.168604 30.012325 192.168.0.1 10.0.0.1 TRADE 72 Multiplexer -> Control, Ping, len=18
20998 09:29:28.171420 0.002816 10.0.0.1 192.168.0.1 TRADE 64 Trader -> Control, Ping, len=10
20999 09:29:28.172611 0.001191 192.168.0.1 10.0.0.1 TRADE 64 Multiplexer -> Control, Pong, len=10
21000 09:29:28.172847 0.000236 10.0.0.1 192.168.0.1 TRADE 72 Trader -> Control, Pong, len=18
21165 09:29:58.167667 29.994820 10.0.0.1 192.168.0.1 TRADE 64 Trader -> Control, Ping, len=10
21166 09:29:58.169923 0.002256 192.168.0.1 10.0.0.1 TRADE 64 Multiplexer -> Control, Pong, len=10
21168 09:29:58.370326 0.200403 192.168.0.1 10.0.0.1 TRADE 72 Multiplexer -> Control, Ping, len=18
21169 09:29:58.370998 0.000672 10.0.0.1 192.168.0.1 TRADE 72 Trader -> Control, Pong, len=18
31491 09:32:12.351912 133.9809… 10.0.0.1 192.168.0.1 TCP 60 1195 → 2107 [RST, ACK] Seq=21251242 …
anonymized
Packets: 33471 · Displayed: 161 (0.5%) · Load time: 0:1.63
Profile: sf2017eu-dummy

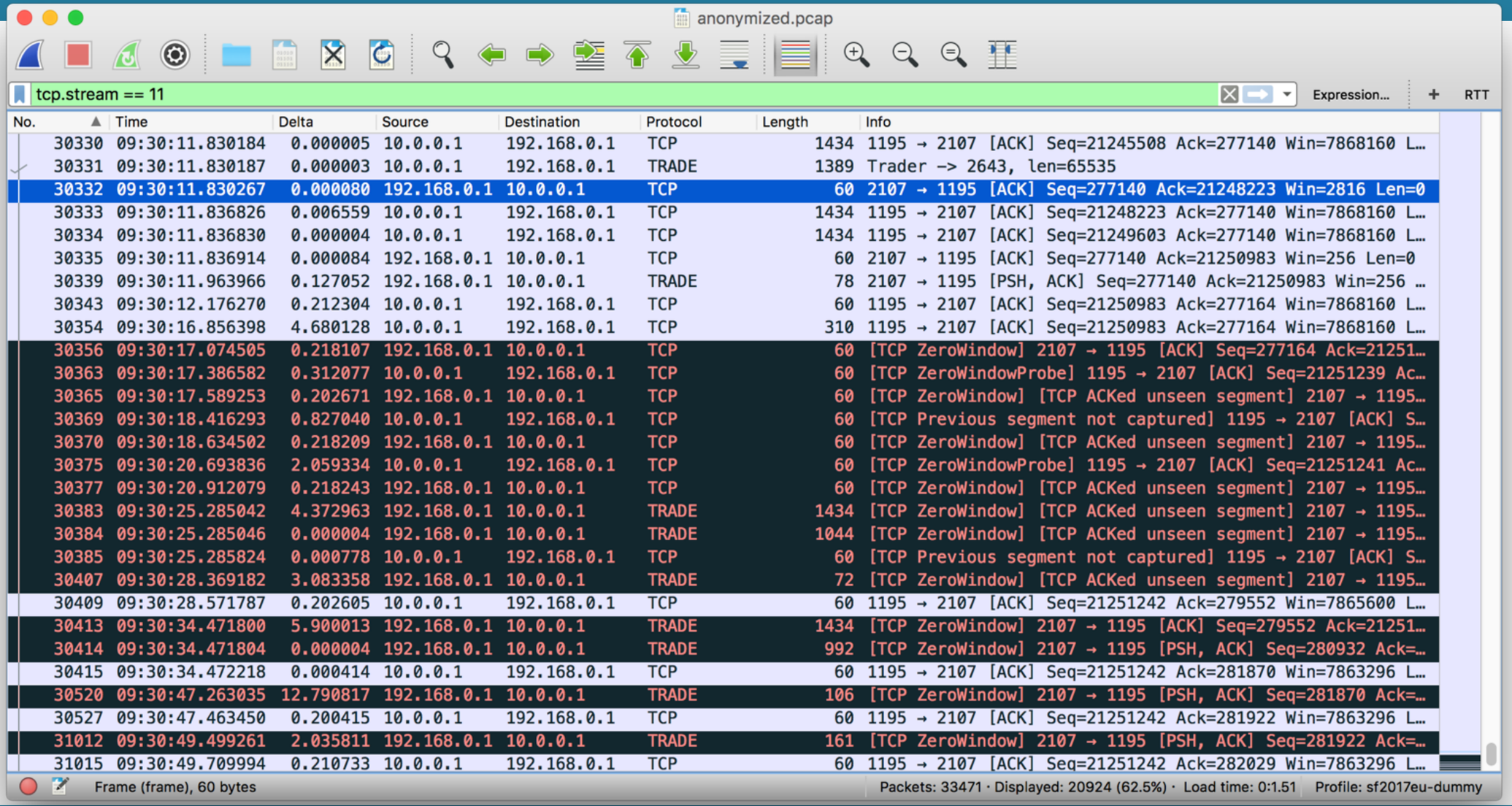
The case of the missing ping!



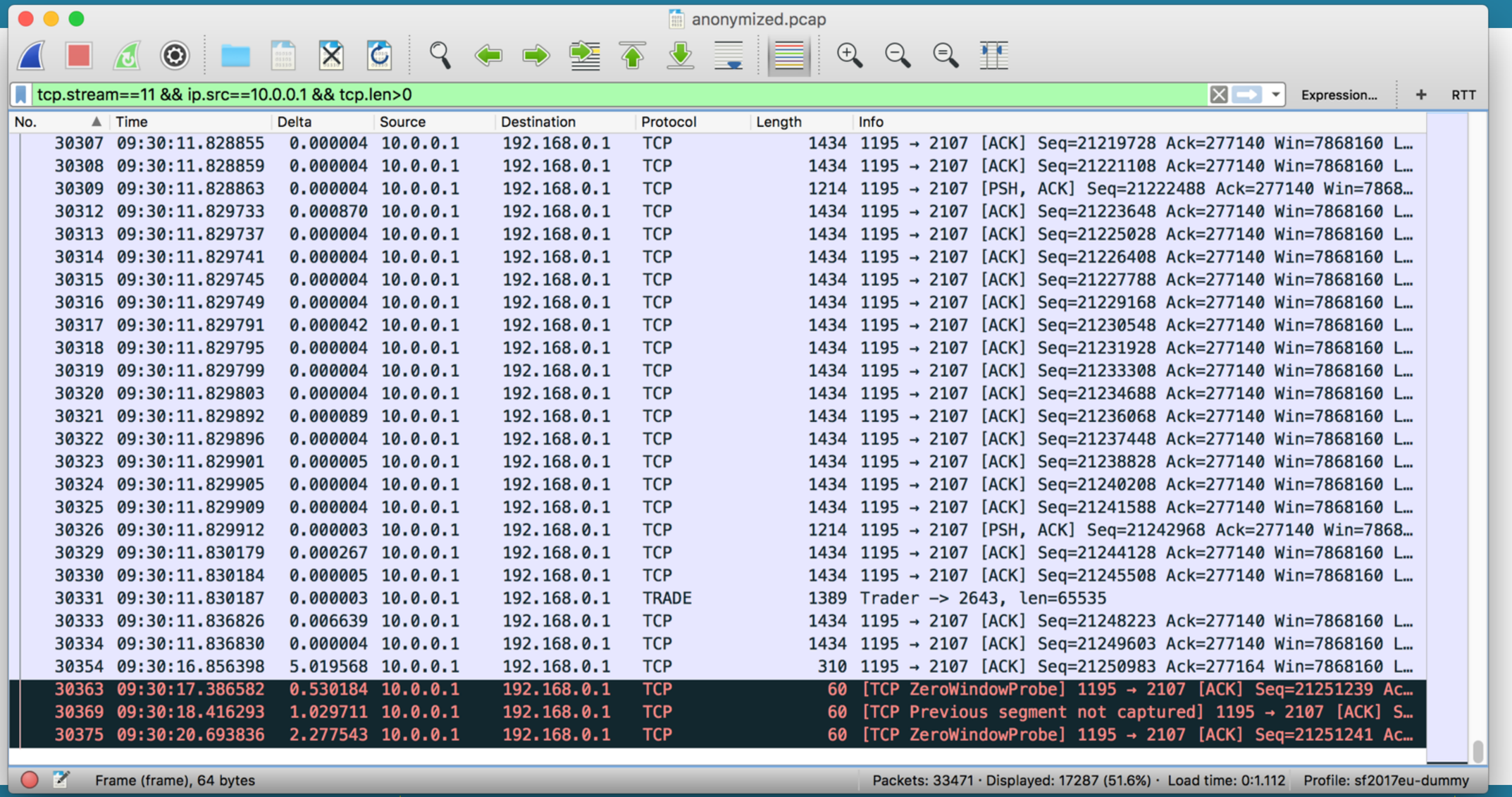
RTT=188.409 ms???



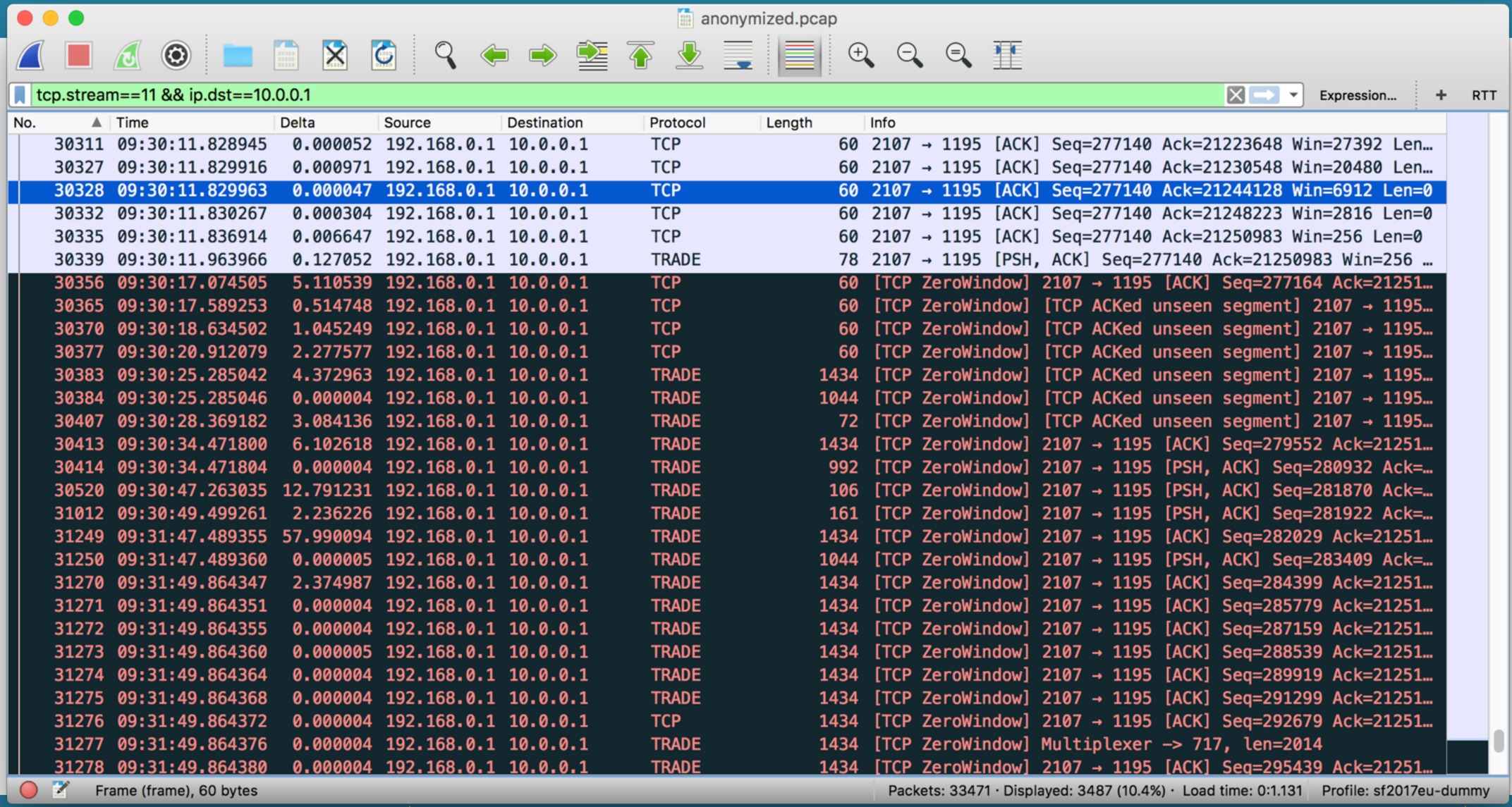
Stop talking… PLEASE!!!



But I have so much to say…



Sorry, busy!


anonymized.pcap
tcp.stream==11 && ip.dst==10.0.0.1
Expression...
RTT
No. Time Delta Source Destination Protocol Length Info
30311 09:30:11.828945 0.000052 192.168.0.1 10.0.0.1 TCP 60 2107 → 1195 [ACK] Seq=277140 Ack=21223648 Win=27392 Len…
30327 09:30:11.829916 0.000971 192.168.0.1 10.0.0.1 TCP 60 2107 → 1195 [ACK] Seq=277140 Ack=21230548 Win=20480 Len…
30328 09:30:11.829963 0.000047 192.168.0.1 10.0.0.1 TCP 60 2107 → 1195 [ACK] Seq=277140 Ack=21244128 Win=6912 Len=0
30332 09:30:11.830267 0.000304 192.168.0.1 10.0.0.1 TCP 60 2107 → 1195 [ACK] Seq=277140 Ack=21248223 Win=2816 Len=0
30335 09:30:11.836914 0.006647 192.168.0.1 10.0.0.1 TCP 60 2107 → 1195 [ACK] Seq=277140 Ack=21250983 Win=256 Len=0
30339 09:30:11.963966 0.127052 192.168.0.1 10.0.0.1 TRADE 78 2107 → 1195 [PSH, ACK] Seq=277140 Ack=21250983 Win=256 …
30356 09:30:17.074505 5.110539 192.168.0.1 10.0.0.1 TCP 60 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=277164 Ack=21251…
30365 09:30:17.589253 0.514748 192.168.0.1 10.0.0.1 TCP 60 [TCP ZeroWindow] [TCP ACKed unseen segment] 2107 → 1195…
30370 09:30:18.634502 1.045249 192.168.0.1 10.0.0.1 TCP 60 [TCP ZeroWindow] [TCP ACKed unseen segment] 2107 → 1195…
30377 09:30:20.912079 2.277577 192.168.0.1 10.0.0.1 TCP 60 [TCP ZeroWindow] [TCP ACKed unseen segment] 2107 → 1195…
30383 09:30:25.285042 4.372963 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] [TCP ACKed unseen segment] 2107 → 1195…
30384 09:30:25.285046 0.000004 192.168.0.1 10.0.0.1 TRADE 1044 [TCP ZeroWindow] [TCP ACKed unseen segment] 2107 → 1195…
30407 09:30:28.369182 3.084136 192.168.0.1 10.0.0.1 TRADE 72 [TCP ZeroWindow] [TCP ACKed unseen segment] 2107 → 1195…
30413 09:30:34.471800 6.102618 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=279552 Ack=21251…
30414 09:30:34.471804 0.000004 192.168.0.1 10.0.0.1 TRADE 992 [TCP ZeroWindow] 2107 → 1195 [PSH, ACK] Seq=280932 Ack=…
30520 09:30:47.263035 12.791231 192.168.0.1 10.0.0.1 TRADE 106 [TCP ZeroWindow] 2107 → 1195 [PSH, ACK] Seq=281870 Ack=…
31012 09:30:49.499261 2.236226 192.168.0.1 10.0.0.1 TRADE 161 [TCP ZeroWindow] 2107 → 1195 [PSH, ACK] Seq=281922 Ack=…
31249 09:31:47.489355 57.990094 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=282029 Ack=21251…
31250 09:31:47.489360 0.000005 192.168.0.1 10.0.0.1 TRADE 1044 [TCP ZeroWindow] 2107 → 1195 [PSH, ACK] Seq=283409 Ack=…
31270 09:31:49.864347 2.374987 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=284399 Ack=21251…
31271 09:31:49.864351 0.000004 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=285779 Ack=21251…
31272 09:31:49.864355 0.000004 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=287159 Ack=21251…
31273 09:31:49.864360 0.000005 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=288539 Ack=21251…
31274 09:31:49.864364 0.000004 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=289919 Ack=21251…
31275 09:31:49.864368 0.000004 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=291299 Ack=21251…
31276 09:31:49.864372 0.000004 192.168.0.1 10.0.0.1 TCP 1434 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=292679 Ack=21251…
31277 09:31:49.864376 0.000004 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] Multiplexer -> 717, len=2014
31278 09:31:49.864380 0.000004 192.168.0.1 10.0.0.1 TRADE 1434 [TCP ZeroWindow] 2107 → 1195 [ACK] Seq=295439 Ack=21251…
Frame (frame), 60 bytes
Packets: 33471 · Displayed: 3487 (10.4%) · Load time: 0:1.131
Profile: sf2017eu-dummy

Root Cause Analysis

- APPSERVER1 sends a lot of data and fills receive buffer of Multiplexer
- APPSERVER1 thinks it has sent a “Ping” packet, but packet never left the TCP buffer because of RWIN=0
- After 90 seconds without “Pong”, the APPSERVER1 closes the connection
- RCA: The processes on Multiplexer somehow don’t empty the receive buffer and block communication

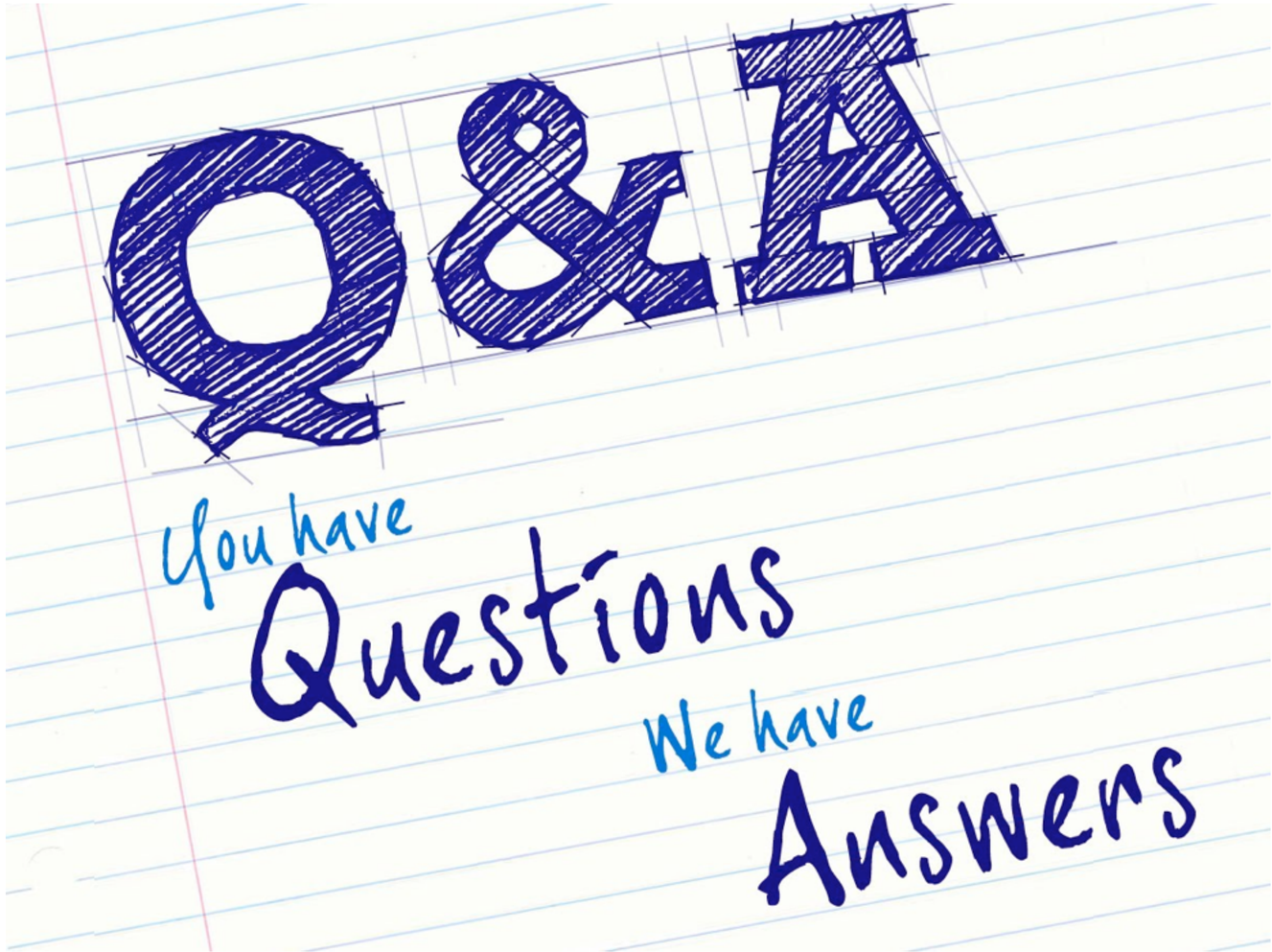
Did LUA-dissector help?

- Understand the Ping-Pong mechanism
- Understand the RTT measurement
- Made filtering easier (or even possible)
- Gives the customer insight in their traffic in future troubleshooting events

Summary…

- Proprietary protocols are hard to analyze from hex
- LUA dissector not extremely difficult to write
  - OK, after a lot of googling, so hopefully this presentation helps you from having to do the same ;-)
- Time to find Root Cause greatly reduced
  - Well, in this case, following up on the Zero-Window problem did point in the right direction. But knowing what actually happened helped in communication with the Software vendor so they could solve the issue more quickly

Q&A



Thank You!

sake.blok@SYN-bit.nl

SYN-bit
deep traffic analysis