第三讲
MQTT（III）
Lecture 3
MQTT (III)

明玉瑞 Yurui Ming

yrming@gmail.com

声明
Disclaimer

- 本讲义在准备过程中由于时间所限，所用材料来源并未规范标示引用来源。所引材料仅用于教学所用，作者无意侵犯原著者之知识版权，所引材料之知识版权均归原著者所有；若原著者介意之，请联系作者更正及删除。

  The time limit during the preparation of these slides incurs the situation that not all the sources of the used materials (texts or images) are properly referenced or clearly manifested. However, all materials in these slides are solely for teaching and the author is with no intention to infringe the copyright bestowed on the original authors or manufacturers. All credits go to corresponding IP holders. Please address the author for any concern for remedy including deletion.

抓包工具
Network Protocol Capturing Tools

- 抓包工具在和网络有关的情景中非常有用。在学习网络协议时，通过查看网络抓包工具抓取的在网络上传输的数据包，学习者可以对协议有直观的感受；在应用实践时，当应用出现问题时，工程师也可以利用抓包工具，通过对数据包的分析，定位问题。

  Network protocol capturing tools offer several benefits for scenarios concerning networking. To learn some networking protocol, learners can have the intuition by directly inspect the captured packets transmitted over he network. By analyzing the packets, the engineers can locate the problem when the deployed applications become faulty.

- Wireshark是常用的网络抓包工具或网络协议分析器。通过在链路层抓包，它可以让使用者在微观层面上看到网络上正在发生的事情。悠久的历史加强大的功能，Wireshark是许多行业进行网络相关开发和应用时的标配。

  Wireshark is the world's foremost network protocol capturing tool or analyzer. By capturing packets on the data-link layer, it lets the user see what's happening on the network at a microscopic level. The long history plus powerful functionalities entitles its de facto (and often de jure) standard across many industries when they come to networking development and deployment.

Wireshark

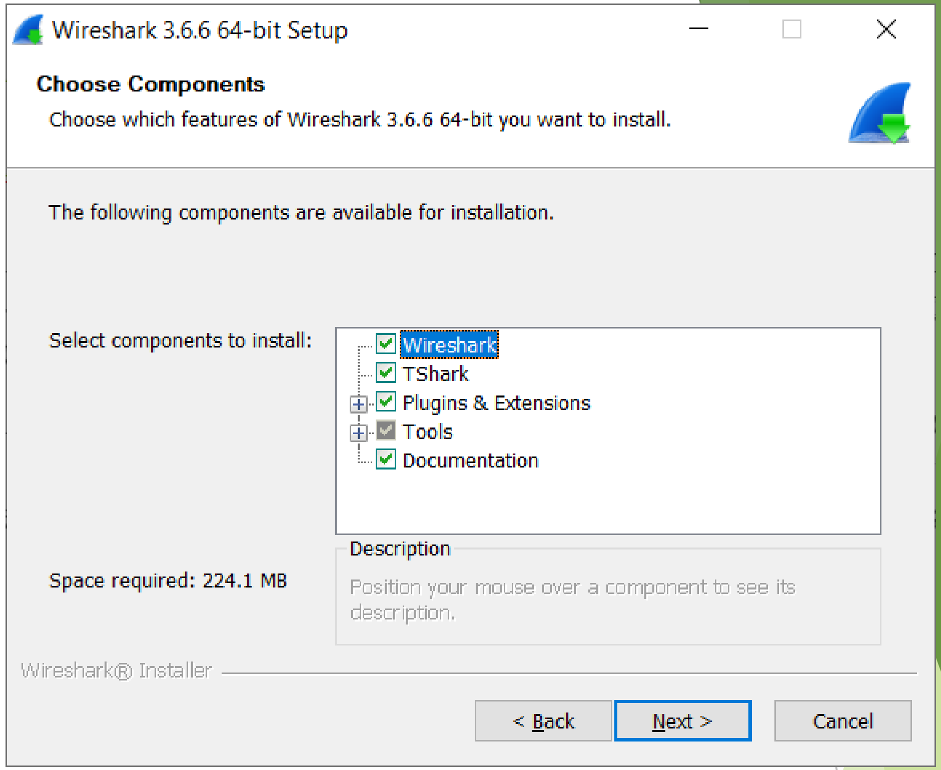
- Wireshark是Gerald Combs在1998 年开始的一个个人项目，随着软件的流行，越来越多的网络专家的加入，使得该项目在网络社区越来越风靡，从而使Wireshark包含越来越多的特性，包括但不限于：
  - 实时捕获和离线分析数百种协议，并不断添加更多协议
  - 标准三窗格数据包浏览器可以在多种平台上运行
  - 同时支持协议过滤器与强大的显示过滤器，有灵活的着色规则
  - 读/写许多不同的捕获文件格式，输出可以导出为多种格式文件.

- Wireshark started as a personal project by Gerald Combs in 1998. With the popularity of the software, network experts collaborate on the project to make it more prevailing in the network community. The features of Wireshark including but not limited to:
  - Live capture and offline analysis up to hundreds of protocols with more being added all the time
  - Standard three-pane packet browser runs on multiple operating systems
  - Comes with protocol filters and the most powerful display filters, with flexible Coloring rules can be applied to the packet list
  - Read/write many different capture file formats, and iutput can be exported to files of various formats

Wireshark
Wireshark

- Wireshark 可以从其官网下载安装：https://www.wireshark.org/#download。一般情况下，选择默认安装就好。安装过程中，可能会询问是否安装第三方驱动，通常选择信任，全部安装就好。

  Wireshark could be downloaded from its official website: https://www.wireshark.org. Generally, the default installation can satisfy the daily usage. During the installation process, the installer might ask for the permission to install third-party drivers. The usual choice is to trust these third-party modules and install all of them.


Wireshark 3.6.6 64-bit Setup

Choose Components

Choose which features of Wireshark 3.6.6 64-bit you want to install.

The following components are available for installation.

Select components to install:

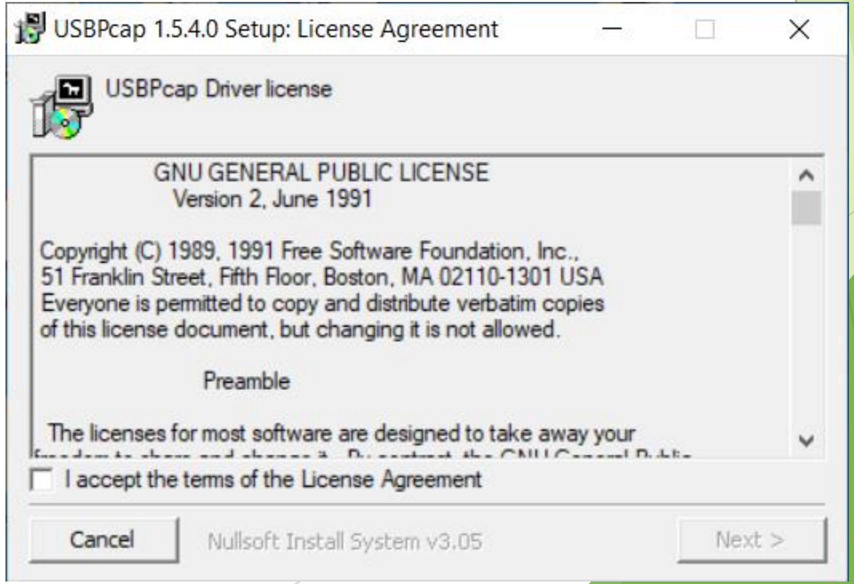
- [✓] Wireshark
- [✓] TShark
- [✓] Plugins & Extensions
- [✓] Tools
- [✓] Documentation

Space required: 224.1 MB

Description

Position your mouse over a component to see its description.

Wireshark® Installer


USBPcap 1.5.4.0 Setup: License Agreement

USBPcap Driver license

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your

[ ] I accept the terms of the License Agreement

Nullsoft Install System v3.05

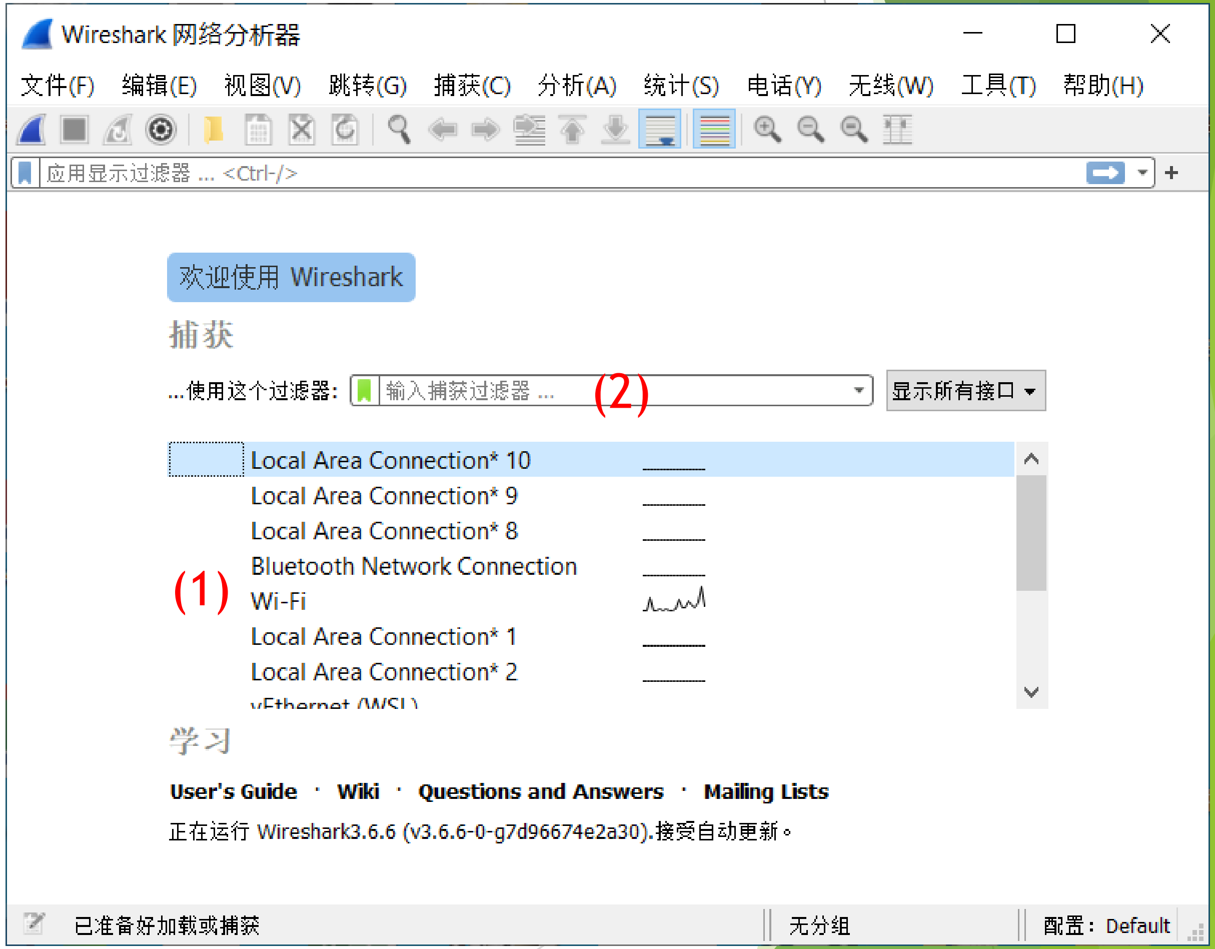
Wireshark使用
Using Wireshark

- Wireshark启动后，首先需要选择在哪个网络接口抓包，通常根据应用经由哪个接口通信，选择即可。

  After launching the Wireshark, the first step is to choose which network interface the Wireshark resides. Choose the interface used by the application for communication.

- 同时，可以选择协议过滤器以减少不感兴趣的包的数量。由于协议过滤器遵循Berkeley Packet Filter（BPF）语法，会被编译为伪汇编代码，因此有很高的执行效率。

  At the same time, users can specify the protocol filter. Since protocol filters conforms with the syntax of Berkeley Packet Filter (BPF), which will be compiled into pseudo assembly code, resulting in high efficiency execution.


Wireshark 网络分析器
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>
欢迎使用 Wireshark
捕获
…使用这个过滤器: 输入捕获过滤器 …
(2)
显示所有接口
(1)
学习
User's Guide · Wiki · Questions and Answers · Mailing Lists
已准备好加载或捕获
无分组
配置: Default

Local Area Connection* 10
Local Area Connection* 9
Local Area Connection* 8
Bluetooth Network Connection
Wi-Fi
Local Area Connection* 1
Local Area Connection* 2
vEthernet (WSL)

正在运行 Wireshark3.6.6 (v3.6.6-0-g7d96674e2a30).接受自动更新。

Wireshark使用
Using Wireshark

- 当启动抓包并捕获一定时间后，当估计需要捕获的包涵盖后，可以停止捕获，准备分析。

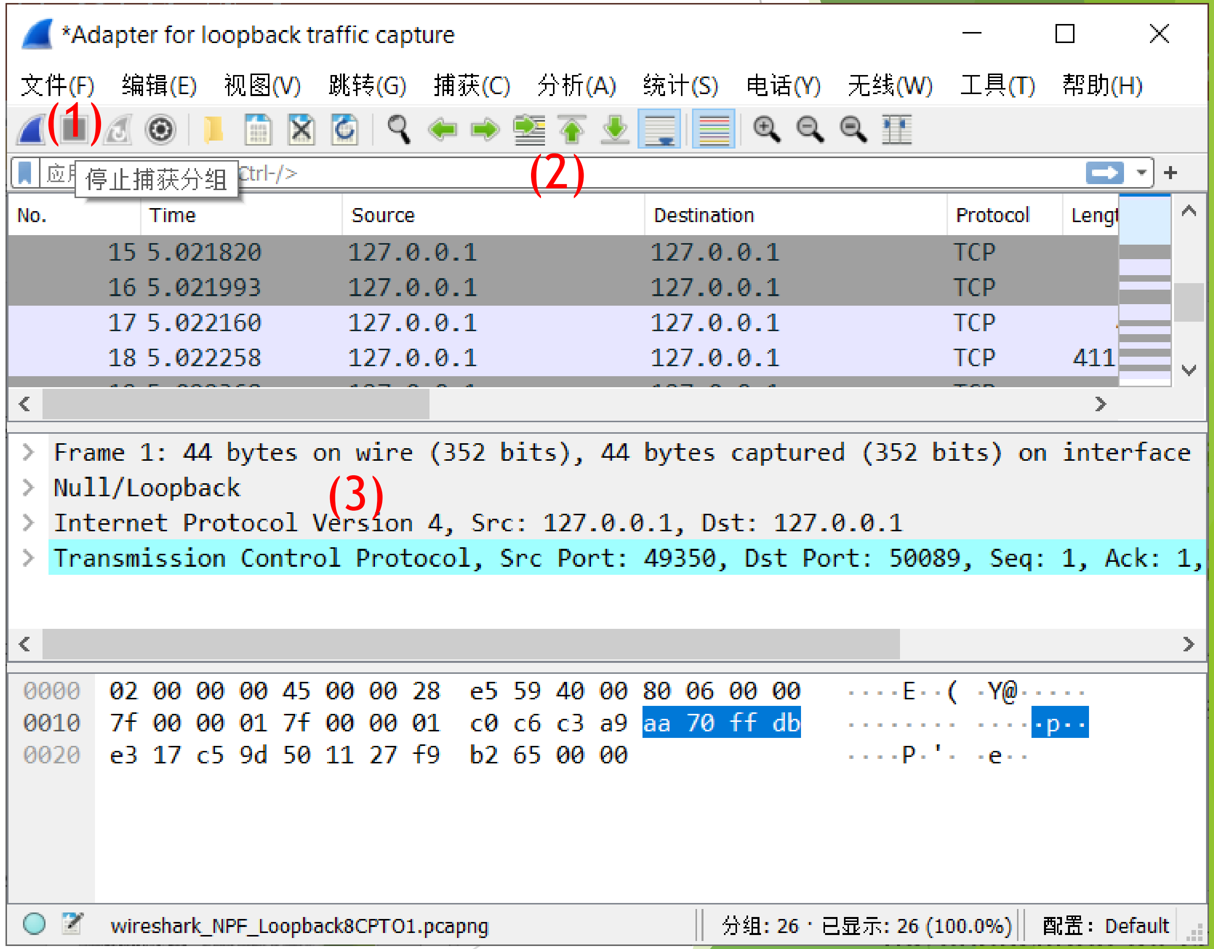
  The user can stop capturing to initiate analysis after the tool runs for some while, especially has the packets interested covered.

- 可以设置显示过滤器以便定位到特定的流或会话。

  The user can set display filters to pin the specific stream or session out.

- 结合对协议的理解，用户可以在以堆栈式显示的协议数据上进行研判，学习或解决问题。

  Based on the understanding to a specific protocol, the user can inspect the protocol data displayed as protocol stack, to learn or solve the problem.


*Adapter for loopback traffic capture
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
(1)
停止捕获分组
应用 Ctrl-/>
(2)
(3)

No.	Time	Source	Destination	Protocol	Lengt
15	5.021820	127.0.0.1	127.0.0.1	TCP	
16	5.021993	127.0.0.1	127.0.0.1	TCP	
17	5.022160	127.0.0.1	127.0.0.1	TCP	
18	5.022258	127.0.0.1	127.0.0.1	TCP	411

```
> Frame 1: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 49350, Dst Port: 50089, Seq: 1, Ack: 1,
```

```
0000  02 00 00 00 45 00 00 28  e5 59 40 00 80 06 00 00   ····E··( ·Y@·····
0010  7f 00 00 01 7f 00 00 01  c0 c6 c3 a9 aa 70 ff db   ········ ····p··
0020  e3 17 c5 9d 50 11 27 f9  b2 65 00 00               ····P·'· ·e··
```

wireshark_NPF_Loopback8CPTO1.pcapng　分组: 26 · 已显示: 26 (100.0%)　配置: Default

MQTT代理/服务器
MQTT Broker/Server

- 在MQTT相关文献中，代理与服务器似乎是同义语，但讲代理时，似乎偏重消息的中继，而讲服务器时，似乎偏重全方位的功能，例如，一些IoT设备的控制功能，需要服务器对收集的感测数据进行分析，从而做出相应的动作，这一般超出了代理的功能范围。

  In the literature of MQTT, broker and server seem to be synonyms. However, when we talk about relay, we emphasize its message relay functionality; when we talk about server, we indicate a full-fledged server. For example, to instruct some IoT device do some controlling work, the server needs to compute based on data from edge sensors. This is usually out of the capability of a simple broker.

- 代理另外一个用处是桥接。由于代理偏重连接，因此代理可以实现一些高级连接特性，比如安全连接。考虑近端与远端各一个代理的情景。近端设备可以通过普通连接连到近端代理，而近端代理可以通过安全连接连到远端代理。而远端代理可以通过普通连接与后端服务器相连。

  Another role for broker to play is bridging. Due to broker’s emphasis on connection, so broker can implement some advanced connection features, such as secure connection. Consider the scenario of two brokers within the local and remote premise.

Mosquitto

- Mosquitto是MQTT协议发布/订阅模型的一个开源实现，作为消息代理，它支持MQTT协议版本从3.1到最新的5.0。Mosquitto相对轻量，适用于从低功耗单板计算机到完整服务器的所有设备，这使其物联网应用中适用较广，从而被Eclipse基金会接纳成为旗下iot.eclipse.org项目的一部分。同时，Mosquitto项目还提供了一个用于实现MQTT客户端的C库，以及非常流行的mosquitto_pub和mosquitto_sub命令行工具。

  Eclipse Mosquitto is an open source implementation of MQTT's publish/subscribe model. As a message broker, it supports MQTT protocol versions from 3.1 up to the newest 5.0. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers, which gains its popularity in the applications of Internet of Things, and funded by the Eclipse Foundation as a part of the iot.eclipse.org project. The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular mosquitto_pub and mosquitto_sub command line MQTT clients.

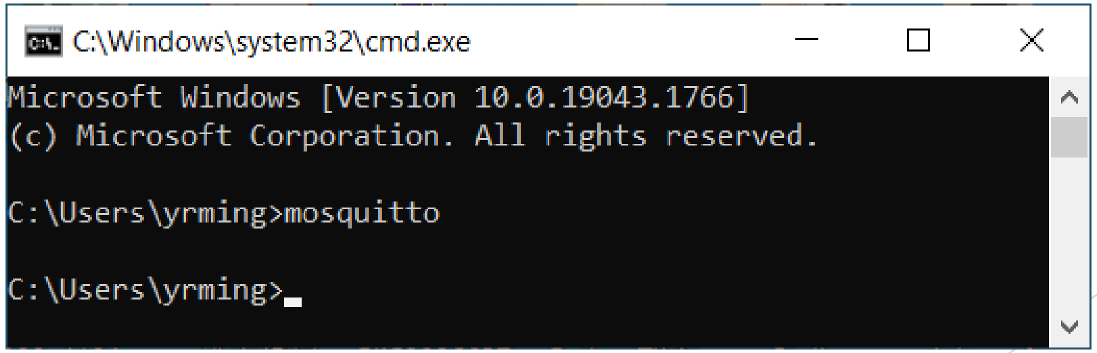
Mosquitto安装与运行
Installing and Launching Mosquitto

- Mosquitto可以从https://mosquitto.org/download/网站下载，通常选择默认安装方式即可。Mosquitto通常不会自动将可执行文件所在目录添加到环境变量，因此需要手动将相关路径添加到PATH变量。

  Mosquitto could be downloaded from https://mosquitto.org/download/, and the default installation option is sufficient. Usually Mosquitto will not automatically add the executable file path into environment varibles, it requires to add it into the PATH variable manually.

- 在将Mosquitto可执行文件所在路径添加进PATH变量之后，即可启动命令行执行。

  After appending the Mosquitto executable into the PATH variable, it can be launched from the command line.



```
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\yrming>mosquitto

C:\Users\yrming>_
```

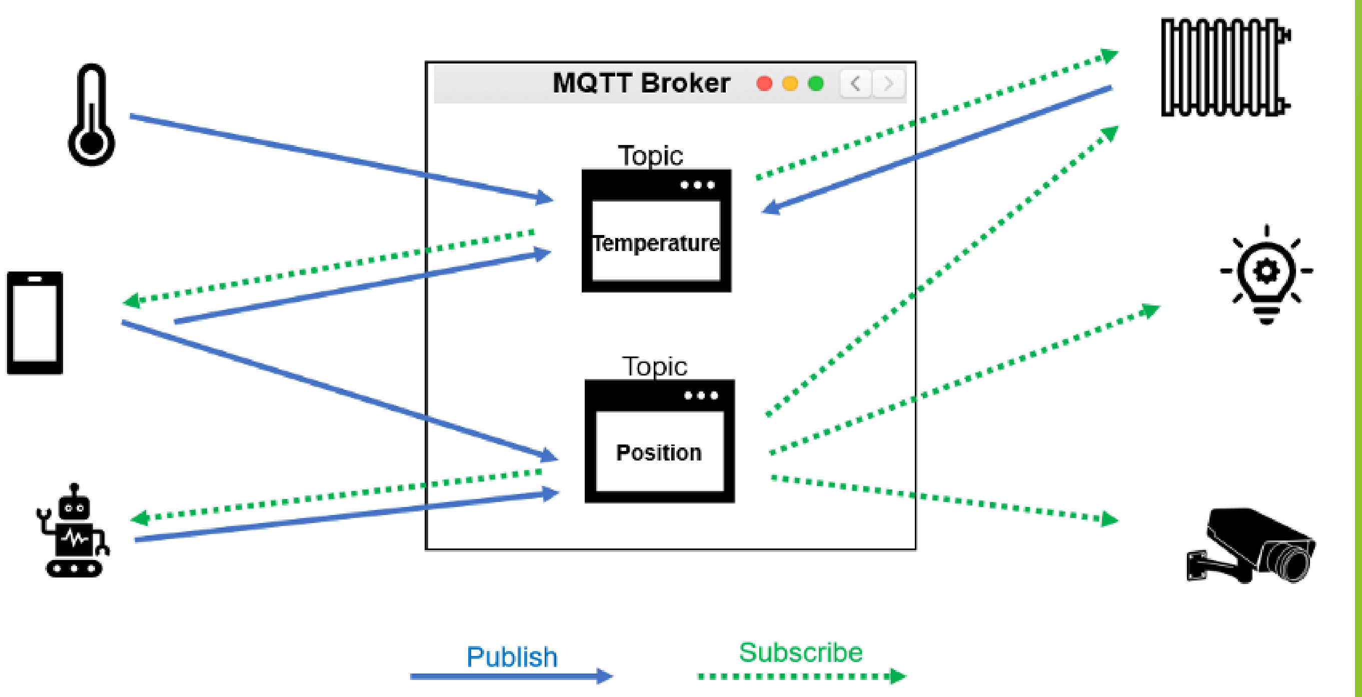
Mosquitto运行
Launching Mosquitto

- 为验证Mosquitto能按默认方式工作，可以通过mosquitto_sub与mosquitto_pub两个命令行程序验证。 mosquitto_sub与mosquitto_pub相当于以软件的方式实现了MQTT客户端，通过与Mosquitto交互完成发布与订阅的功能。

  To verify that Mosquitto works at least in the default mode, command line tools mosquitto_sub and mosquitto_pub can be use to verify it. mosquitto_sub and mosquitto_pub are MQTT clients implemented purely in software way, by communicating with Mosquitto to accomplish publishing and subscription.

- 当直接执行Mosquitto时，应用会使用默认操作。可以通过配置文件的方式，调整Mosquitto的行为，使其更能满足具体应用需求。

  Directly launching Mosquitto will have default configuration applied. By specifying specific values in the configuration file, behaviours of Mosquitto can be adjusted to cater to the needs of applications.

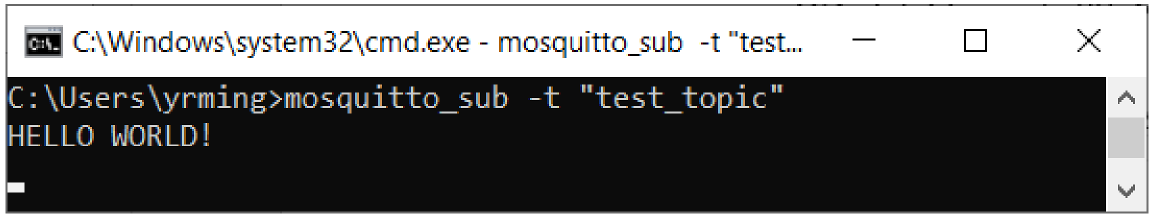


Mosquitto运行
Launching Mosquitto

- 具体验证步骤如下：
  - 打开一个命令行窗口W1，执行命令：mosquitto_sub -t “test_topic”;
  - 打开另一个命令行窗口W2，执行命令：mosquitto_pub -t "test_topic" -m "HELLO WORLD!“
  - 若在W1中可以看到HELLO WORLD!输出，则说明Mosquitto正常工作。

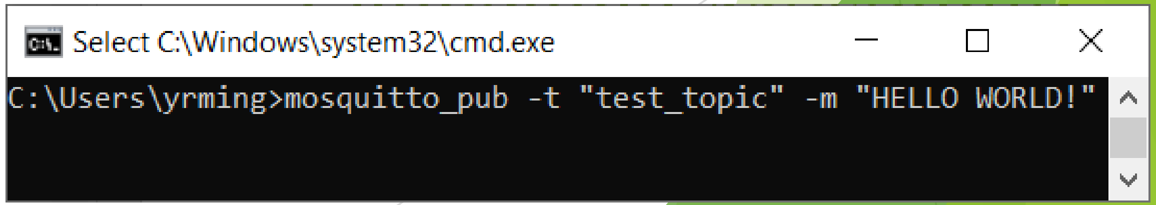
The steps to verify it are as follows:

  - Open one terminal window W1 and execute: mosquitto_sub -t “test_topic”;
  - Open one terminal window W2 and execute: mosquitto_pub -t "test_topic" -m "HELLO WORLD!“
  - If you can see output of the string “HELLO WORLD!” in W1, it indicates Mosquitto works properly.



C:\Windows\system32\cmd.exe - mosquitto_sub -t "test...

```
C:\Users\yrming>mosquitto_sub -t "test_topic"
HELLO WORLD!
```



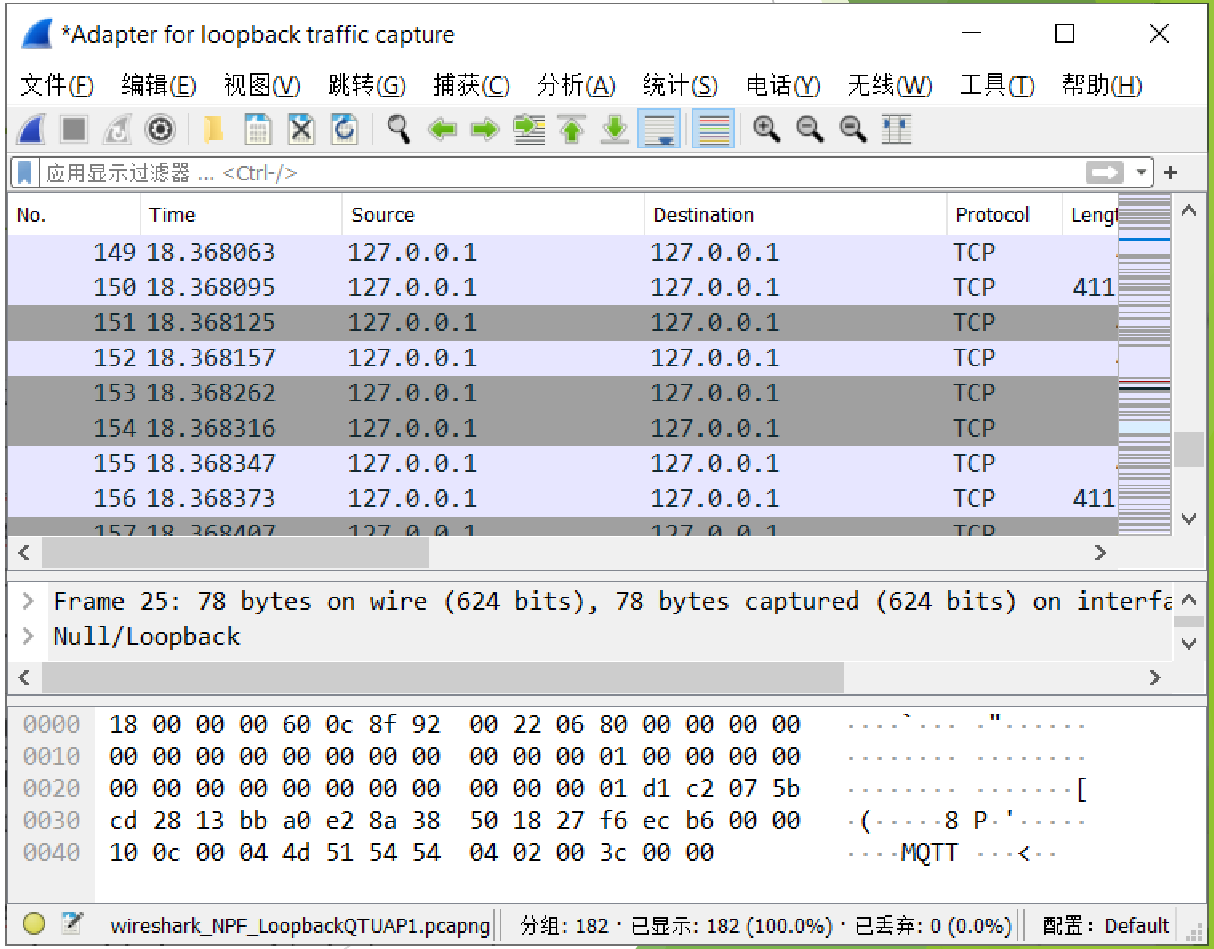
Select C:\Windows\system32\cmd.exe

```
C:\Users\yrming>mosquitto_pub -t "test_topic" -m "HELLO WORLD!"
```

MQTT抓包分析
MQTT Analysis By Capturing

- 在MQTT的开发或应用中，通过抓包，对学习或排错都是比较有效的手段。例如，我们首先在loop端口上启动Wireshark开始抓包，然后启动Mosquitto，通过mosquitto_sub与mosquitto_pub模拟订阅与发布交互过程，最后停止抓包进行分析。

  Packet capturing is an efficient method for learning and debugging during the developing or deploying MQTT applications. For example, we can launch Wireshark on the loopback interface to start capturing packets, then we start mosquito, meanwhile execute mosquitto_sub and mosquitto_pub to simulate the process of subscribing and publishing. At last we stop capturing and proceed to analyze these packets.


*Adapter for loopback traffic capture
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>

```
No.    Time        Source      Destination   Protocol  Lengt
   149 18.368063   127.0.0.1   127.0.0.1     TCP
   150 18.368095   127.0.0.1   127.0.0.1     TCP       411
   151 18.368125   127.0.0.1   127.0.0.1     TCP
   152 18.368157   127.0.0.1   127.0.0.1     TCP
   153 18.368262   127.0.0.1   127.0.0.1     TCP
   154 18.368316   127.0.0.1   127.0.0.1     TCP
   155 18.368347   127.0.0.1   127.0.0.1     TCP
   156 18.368373   127.0.0.1   127.0.0.1     TCP       411
```

```
> Frame 25: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interfa
> Null/Loopback
```

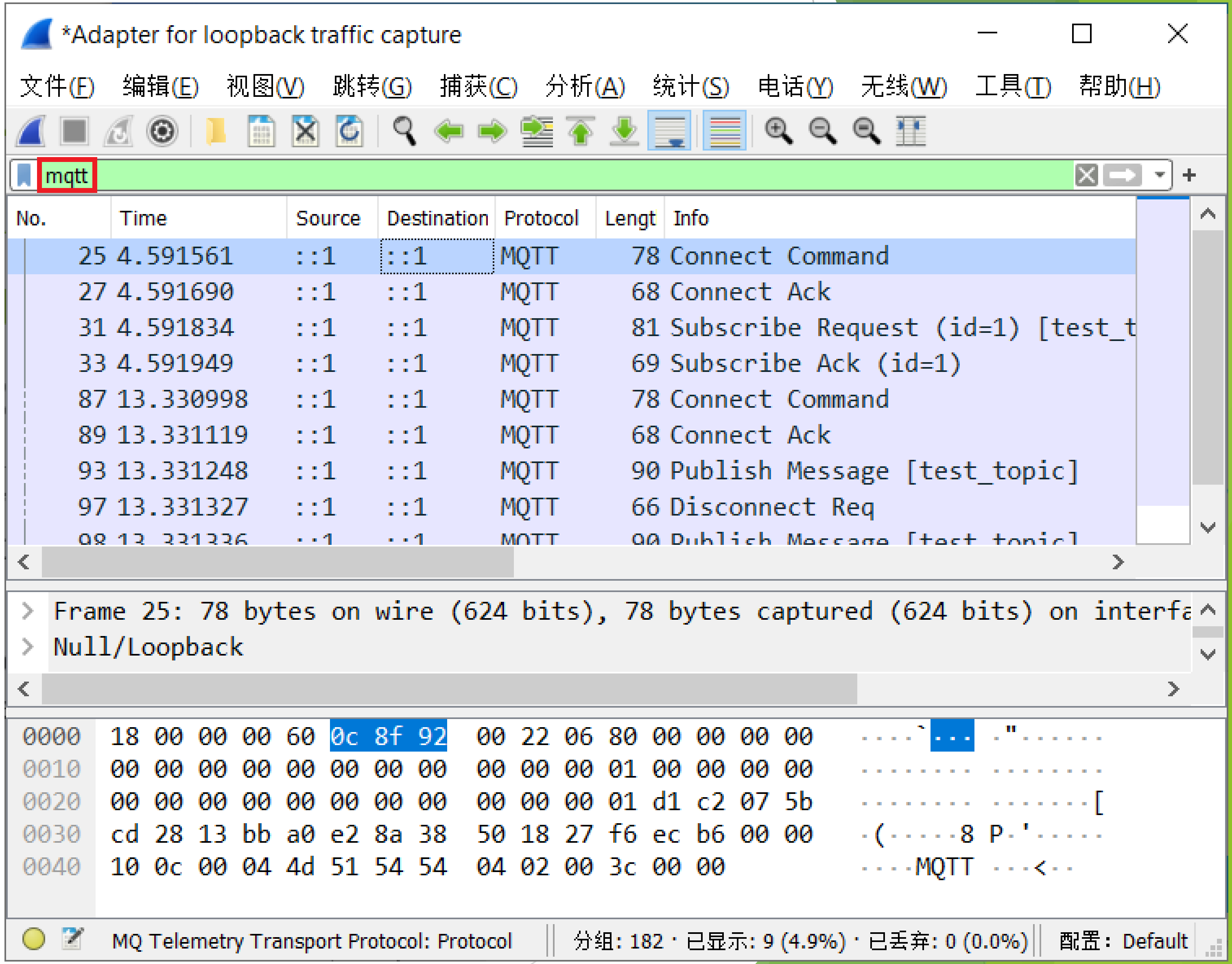
```
0000  18 00 00 00 60 0c 8f 92  00 22 06 80 00 00 00 00   ....`... ."......
0010  00 00 00 00 00 00 00 00  00 00 00 01 00 00 00 00   ........ ........
0020  00 00 00 00 00 00 00 00  00 00 00 01 d1 c2 07 5b   ........ .......[
0030  cd 28 13 bb a0 e2 8a 38  50 18 27 f6 ec b6 00 00   .(.....8 P.'.....
0040  10 0c 00 04 4d 51 54 54  04 02 00 3c 00 00         ....MQTT ...<..
```

wireshark_NPF_LoopbackQTUAP1.pcapng | 分组: 182 · 已显示: 182 (100.0%) · 已丢弃: 0 (0.0%) | 配置: Default

MQTT抓包分析
MQTT Analysis By Capturing

- 分析的第一步，是将MQTT相关包找出来，此时用mqtt作为显示过滤器即可。

  As the first step for analysis, one can just applies “mqtt” as display filter to list all MQTT packets.



*Adapter for loopback traffic capture
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

mqtt

```
No.   Time          Source  Destination  Protocol  Lengt  Info
   25 4.591561      ::1     ::1          MQTT         78  Connect Command
   27 4.591690      ::1     ::1          MQTT         68  Connect Ack
   31 4.591834      ::1     ::1          MQTT         81  Subscribe Request (id=1) [test_t
   33 4.591949      ::1     ::1          MQTT         69  Subscribe Ack (id=1)
   87 13.330998     ::1     ::1          MQTT         78  Connect Command
   89 13.331119     ::1     ::1          MQTT         68  Connect Ack
   93 13.331248     ::1     ::1          MQTT         90  Publish Message [test_topic]
   97 13.331327     ::1     ::1          MQTT         66  Disconnect Req
   98 13.331336     ::1     ::1          MQTT         90  Publish Message [test_topic]
```

```
> Frame 25: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interfa
> Null/Loopback
```

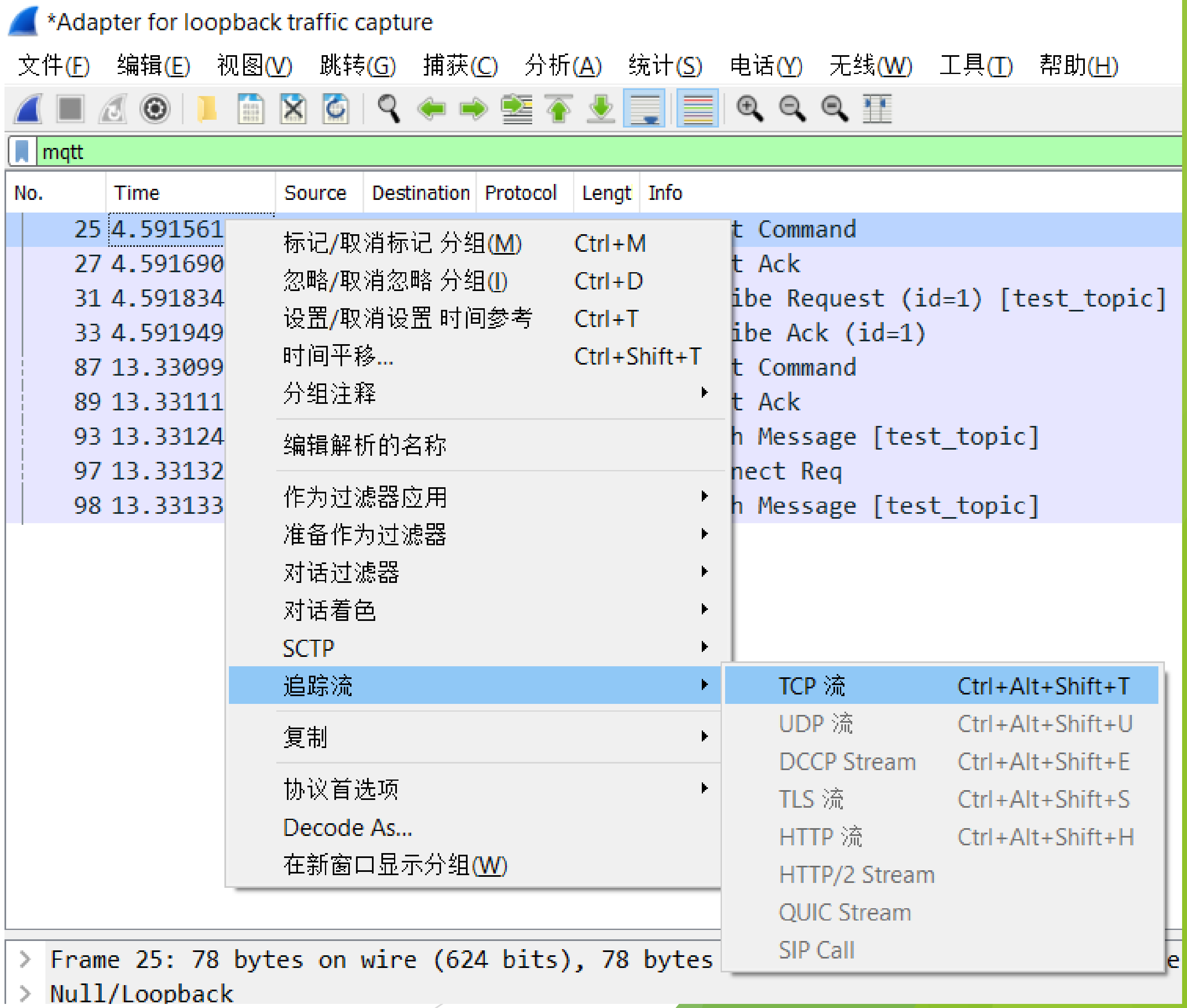
```
0000  18 00 00 00 60 0c 8f 92  00 22 06 80 00 00 00 00   ....`... ."......
0010  00 00 00 00 00 00 00 00  00 00 00 01 00 00 00 00   ........ ........
0020  00 00 00 00 00 00 00 00  00 00 00 01 d1 c2 07 5b   ........ .......[
0030  cd 28 13 bb a0 e2 8a 38  50 18 27 f6 ec b6 00 00   .(.....8 P.'.....
0040  10 0c 00 04 4d 51 54 54  04 02 00 3c 00 00         ....MQTT ...<..
```

MQ Telemetry Transport Protocol: Protocol | 分组: 182 · 已显示: 9 (4.9%) · 已丢弃: 0 (0.0%) | 配置: Default

MQTT抓包分析
MQTT Analysis By Capturing

- 由于可能抓到的包里，包括多个客户端与代理交互的数据，MQTT显示过滤器会将所有的这样的包显示出来，因此作为分析的第二步，可以通过追踪TCP流显示特定的客户端与代理的数据。

  The captured packets highlighted by the MQTT display filter might includes session data between multiple clients and the broker. To emphasize the data between specific client and the broker, as the second step for analysis, the user can trace the TCP stream to narrow down to the connection between MQTT client and the broker.


*Adapter for loopback traffic capture

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

mqtt

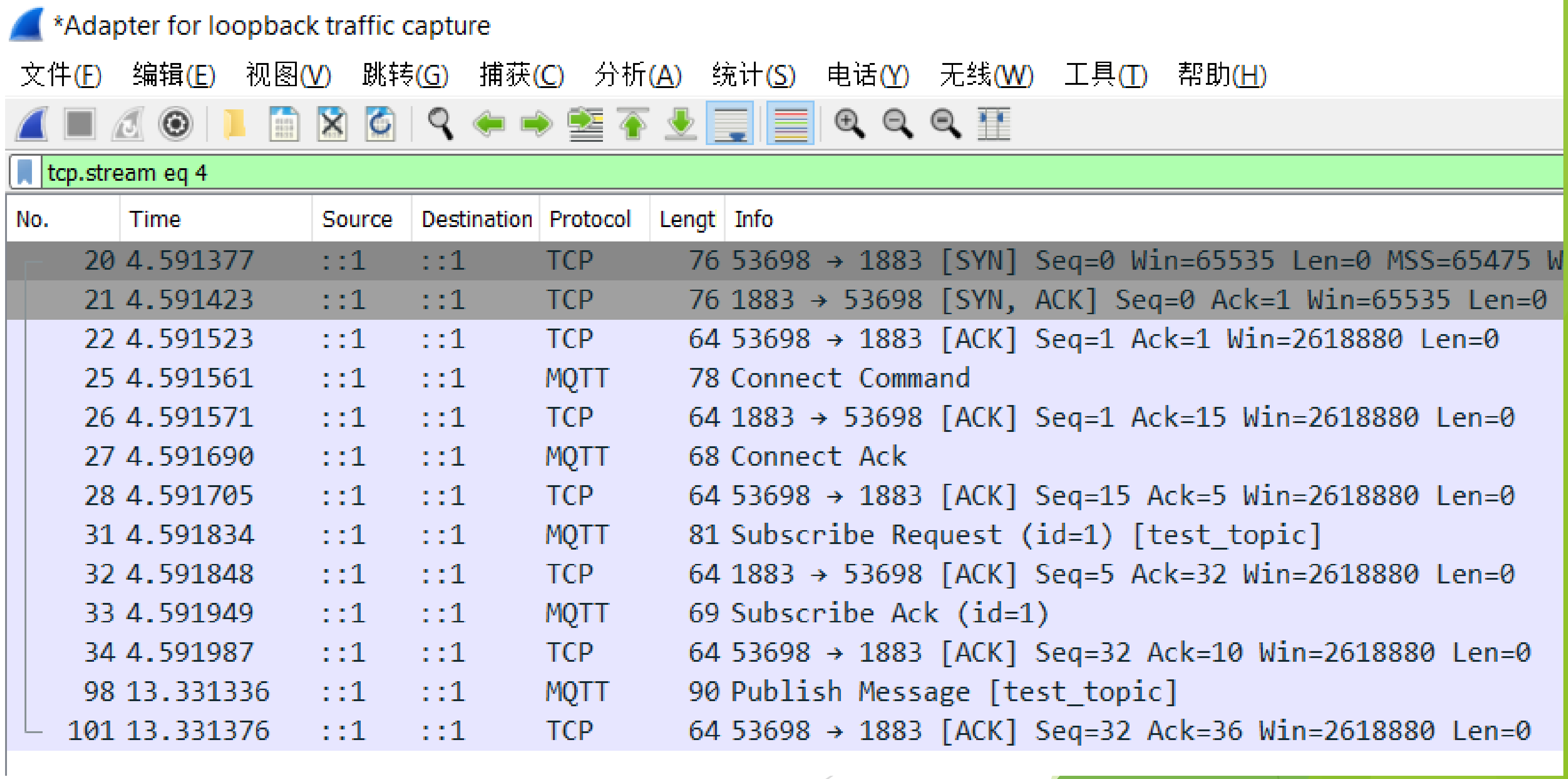
No.	Time	Source	Destination	Protocol	Lengt	Info
25	4.591561					t Command
27	4.591690					t Ack
31	4.591834					ibe Request (id=1) [test_topic]
33	4.591949					ibe Ack (id=1)
87	13.33099					t Command
89	13.33111					t Ack
93	13.33124					h Message [test_topic]
97	13.33132					nect Req
98	13.33133					h Message [test_topic]

```
> Frame 25: 78 bytes on wire (624 bits), 78 bytes
> Null/Loopback
```

MQTT抓包分析
MQTT Analysis By Capturing

- 此时，便可以对特定客户端与代理的交互进行分析。如图，其显示了mosquitto_sub 与 mosquitto交互的若干消息类型，如连接、订阅、发布等等。

  Now we can analyze the connection between the client and the proxy. As in the figure, it shows several message types, such as CONNECT, SUBSCRIBE, PUBLISH, etc., between the client and the proxy.


*Adapter for loopback traffic capture
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

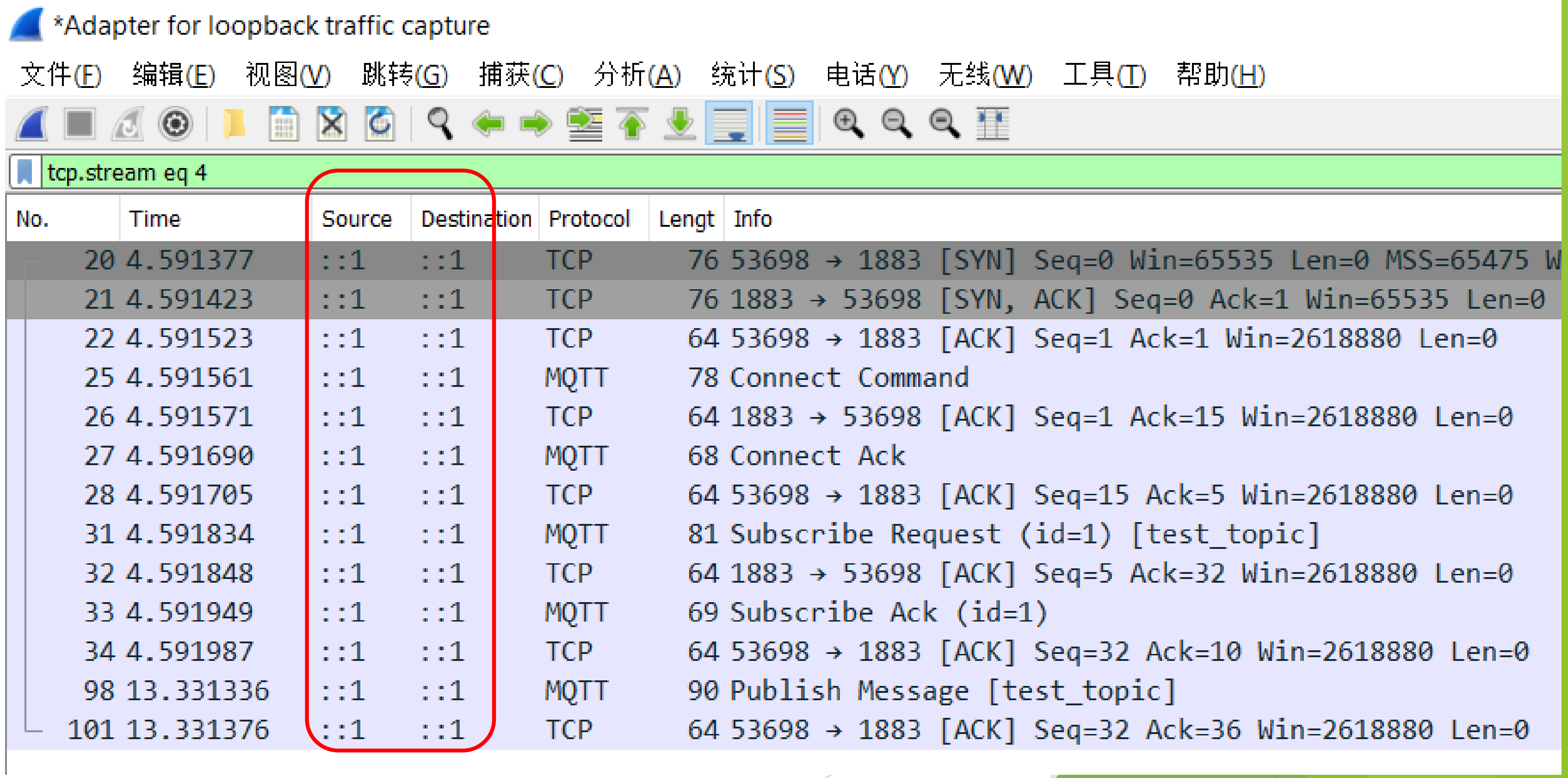
tcp.stream eq 4

```
No.  Time       Source  Destination Protocol Lengt Info
  20 4.591377   ::1     ::1         TCP      76 53698 → 1883 [SYN] Seq=0 Win=65535 Len=0 MSS=65475 W
  21 4.591423   ::1     ::1         TCP      76 1883 → 53698 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
  22 4.591523   ::1     ::1         TCP      64 53698 → 1883 [ACK] Seq=1 Ack=1 Win=2618880 Len=0
  25 4.591561   ::1     ::1         MQTT     78 Connect Command
  26 4.591571   ::1     ::1         TCP      64 1883 → 53698 [ACK] Seq=1 Ack=15 Win=2618880 Len=0
  27 4.591690   ::1     ::1         MQTT     68 Connect Ack
  28 4.591705   ::1     ::1         TCP      64 53698 → 1883 [ACK] Seq=15 Ack=5 Win=2618880 Len=0
  31 4.591834   ::1     ::1         MQTT     81 Subscribe Request (id=1) [test_topic]
  32 4.591848   ::1     ::1         TCP      64 1883 → 53698 [ACK] Seq=5 Ack=32 Win=2618880 Len=0
  33 4.591949   ::1     ::1         MQTT     69 Subscribe Ack (id=1)
  34 4.591987   ::1     ::1         TCP      64 53698 → 1883 [ACK] Seq=32 Ack=10 Win=2618880 Len=0
  98 13.331336  ::1     ::1         MQTT     90 Publish Message [test_topic]
 101 13.331376  ::1     ::1         TCP      64 53698 → 1883 [ACK] Seq=32 Ack=36 Win=2618880 Len=0
```

MQTT抓包分析
MQTT Analysis By Capturing

- 但由于我们在同一个机器上运行客户端与代理，会显示源与目的是一样的，可以进一步通过流量图，以端口区分客户端与代理。

  Note that we are running the client and the proxy on the same local machine, which causes the source and destination are indistinguishable. We can differentiate them using ports from the flow graph.


*Adapter for loopback traffic capture
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

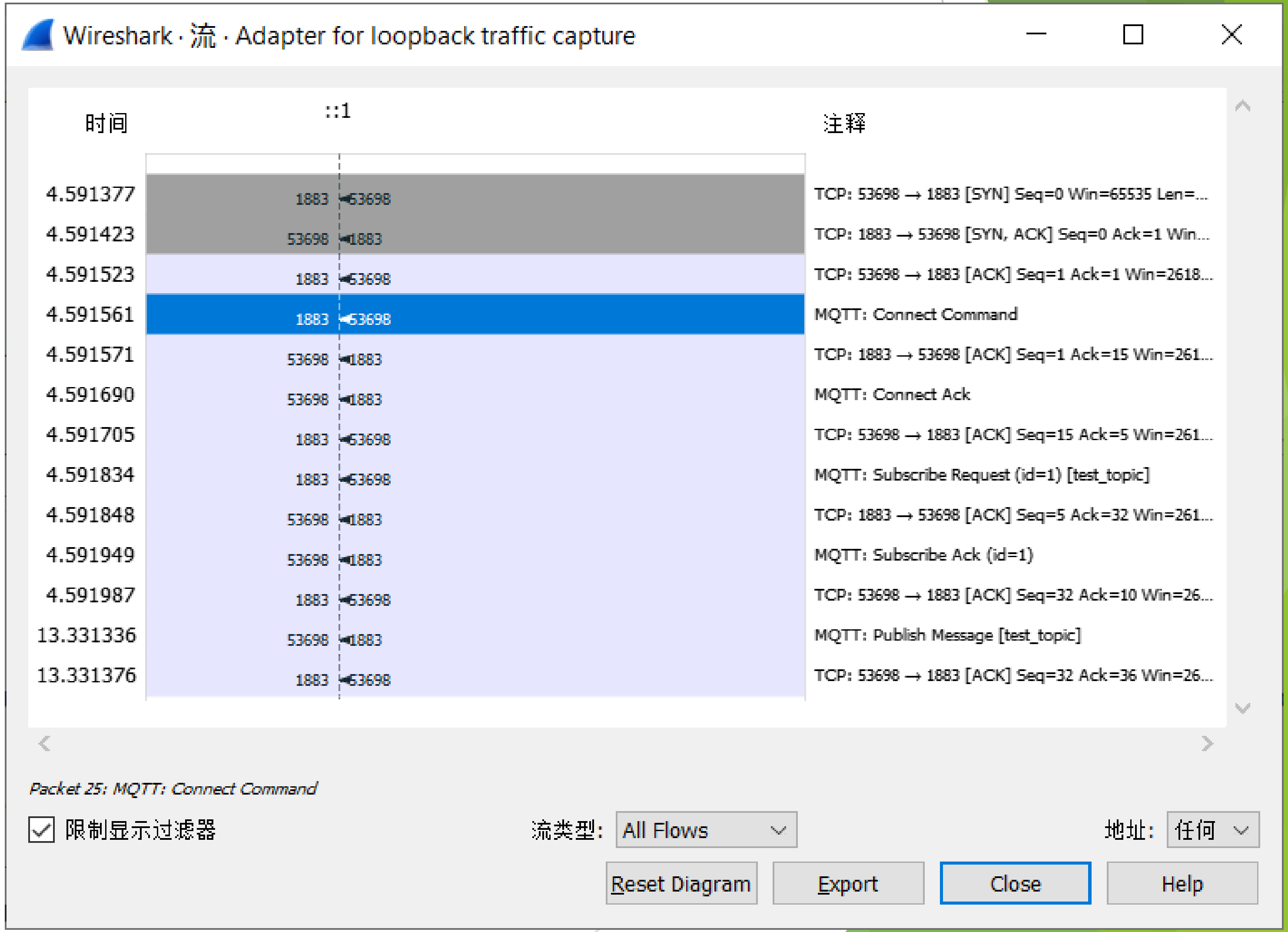
tcp.stream eq 4

No.	Time	Source	Destination	Protocol	Lengt	Info
20	4.591377	::1	::1	TCP	76	53698 → 1883 [SYN] Seq=0 Win=65535 Len=0 MSS=65475 W
21	4.591423	::1	::1	TCP	76	1883 → 53698 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
22	4.591523	::1	::1	TCP	64	53698 → 1883 [ACK] Seq=1 Ack=1 Win=2618880 Len=0
25	4.591561	::1	::1	MQTT	78	Connect Command
26	4.591571	::1	::1	TCP	64	1883 → 53698 [ACK] Seq=1 Ack=15 Win=2618880 Len=0
27	4.591690	::1	::1	MQTT	68	Connect Ack
28	4.591705	::1	::1	TCP	64	53698 → 1883 [ACK] Seq=15 Ack=5 Win=2618880 Len=0
31	4.591834	::1	::1	MQTT	81	Subscribe Request (id=1) [test_topic]
32	4.591848	::1	::1	TCP	64	1883 → 53698 [ACK] Seq=5 Ack=32 Win=2618880 Len=0
33	4.591949	::1	::1	MQTT	69	Subscribe Ack (id=1)
34	4.591987	::1	::1	TCP	64	53698 → 1883 [ACK] Seq=32 Ack=10 Win=2618880 Len=0
98	13.331336	::1	::1	MQTT	90	Publish Message [test_topic]
101	13.331376	::1	::1	TCP	64	53698 → 1883 [ACK] Seq=32 Ack=36 Win=2618880 Len=0

MQTT抓包分析 MQTT Analysis By Capturing

- 从“统计”菜单选择“流量图”，同时限制所展示的内容为显示过滤器，即可看到具体的交互。流量图中的黑三角，表示了源与目的，这里以端口加以区分。

  From the “Statistics” menu, we choose the “flow graph” item, and the same time we apply the “Limit to display filter” constraint, then we have the diagram of interactions.



时间	::1	注释
4.591377	1883 ◄ 53698	TCP: 53698 → 1883 [SYN] Seq=0 Win=65535 Len=...
4.591423	53698 ◄ 1883	TCP: 1883 → 53698 [SYN, ACK] Seq=0 Ack=1 Win...
4.591523	1883 ◄ 53698	TCP: 53698 → 1883 [ACK] Seq=1 Ack=1 Win=2618...
4.591561	1883 ◄ 53698	MQTT: Connect Command
4.591571	53698 ◄ 1883	TCP: 1883 → 53698 [ACK] Seq=1 Ack=15 Win=261...
4.591690	53698 ◄ 1883	MQTT: Connect Ack
4.591705	1883 ◄ 53698	TCP: 53698 → 1883 [ACK] Seq=15 Ack=5 Win=261...
4.591834	1883 ◄ 53698	MQTT: Subscribe Request (id=1) [test_topic]
4.591848	53698 ◄ 1883	TCP: 1883 → 53698 [ACK] Seq=5 Ack=32 Win=261...
4.591949	53698 ◄ 1883	MQTT: Subscribe Ack (id=1)
4.591987	1883 ◄ 53698	TCP: 53698 → 1883 [ACK] Seq=32 Ack=10 Win=26...
13.331336	53698 ◄ 1883	MQTT: Publish Message [test_topic]
13.331376	1883 ◄ 53698	TCP: 53698 → 1883 [ACK] Seq=32 Ack=36 Win=26...

Packet 25: MQTT: Connect Command

☑ 限制显示过滤器　　流类型: All Flows　　地址: 任何

MQTT客户端
MQTT Client

- 在物联网应用设计中，由于代理或服务器功能较为复杂，通常采用中心化与标准的实现方式，用户更多的是使用方面的工作。而客户端由于形态各异、功能多样，通常是设计与开发的重点。在端设备的机电构件满足需求后，通常通过集成MQTT库，将设备连上物联网。Paho是比较常用的MQTT库，其作为Eclipse开源物联网项目的一部分，其主要侧重于客户端，以各种编程语言提供了MQTT和MQTT-SN的开源实现。

  For the design of IoT applications, due to the complexity of brokers or servers, they are usually implemented via centralized or canonical way, and users focus on the clients instead. The clients are usually presented in different form and with versatile functionalities, they are the keys to design and implement successful IoT applications. Usually, when the electro-mechanical components are ready in accordance with the requirement, MQTT library is incorporated to connect the endpoint device to the network. As part of the Eclipse IoT project, paho is the most commonly-used MQTT library, which provides open source, mainly client side, implementations of MQTT and MQTT-SN in a variety of programming languages.

MQTT客户端
MQTT Client

- 对于以Python为编程语言使用paho的方式，可以通过如下方式进行安装：

  For the way of using paho via Python, the library can be installed as follows:

  ```
  pip install paho-mqtt
  conda install -c conda-forge paho-mqtt
  ```

- 对于偏感知类的应用，基本就是以发布方式进行数据上报，示例程序如下：

  For applications mainly used for perception, they tend to simply upload data via publishing, the sample code is in below:

  ```
  import paho.mqtt.client as mqtt
  mqttBroker ="localhost"
  client = mqtt.Client("Test")
  client.connect(mqttBroker)
  client.publish("test_topic", “Hello, world")
  ```

MQTT客户端
MQTT Client

- 对于控制类的应用，有的控制命令可能来自感测器根据对环境的感知，直接发布的命令；有些可能是服务器综合各种数据后，发布的命令，这些命令在实现时往往关联到特定主题，通过订阅的方式到达端点，相应示例程序如下：

  For control applications, some command can originate from sensors which trigger the command by environmental perception. Some commands are from the server which issue the command by analyzing all sorts of data. These commands are usually associated with some topics, and the endpoint device retrieve them via subscription. The sample code is as follows:

```
import paho.mqtt.client as mqtt
import time, signal
def on_message_test(client, userdata, msg):
    print('Received a topic data ', msg.payload.decode('utf-8'))
client = mqtt.Client("Test")
client.message_callback_add("test_topic", on_message_test)
client.connect("localhost", 1883)
client.subscribe("test_topic")
client.loop_start()
def handler(signum, frame):
    client.loop_stop()
    exit(0)
signal.signal(signal.SIGINT, handler)
while True:
    time.sleep(5) # do something you like
```

MQTT 安全连接
MQTT Secure Connection

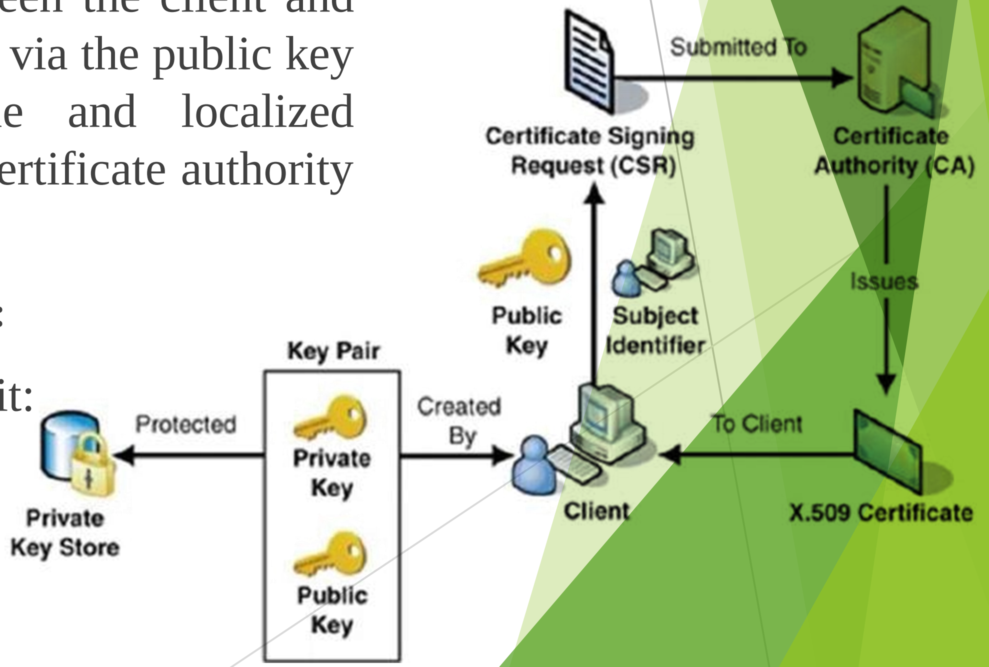
- 在物联网应用中，有时客户端与代理之间的连接出于某种目的，需要用到安全连接。MQTT支持的安全连接是通过公共秘钥体系来完成的。由于物联网应用的多样化与局部化，可以利用私有证书进行安全连接的部署。

  In IoT applications, it might require secure connection between the client and broker due to some reason. MQTT supports secure connection via the public key infrastructure (PKI). However, considering the versatile and localized deployment of IoT applications, it is usually utilized private certificate authority (CA) for deployment.

- 在创建私有证书的过程中，常利用OpenSSL工具进行完成：

  During the creation of private CA, we can use OpenSSL to do it:

  https://slproweb.com/products/Win32OpenSSL.html



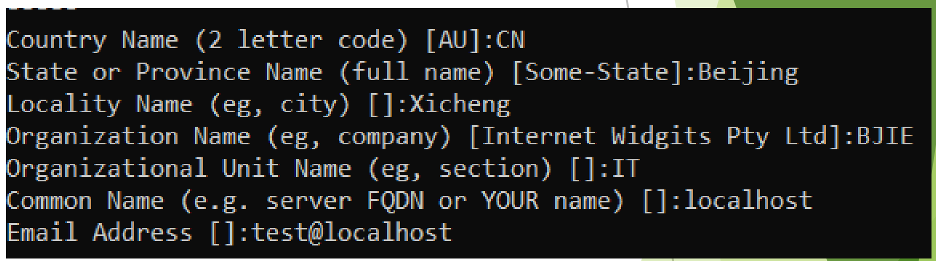
MQTT 安全连接
MQTT Secure Connection

- 可以按照如下步骤证书颁发机构证书、服务器端证书、客户端证书：

  The root CA, CA for Server and CA for client can be generated via the procedures below:

- 机构证书（CA Certificate）：

```
openssl req -new -x509 -extensions v3_ca -keyout ca.key -out ca.crt
```



```
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:Beijing
Locality Name (eg, city) []:Xicheng
Organization Name (eg, company) [Internet Widgits Pty Ltd]:BJIE
Organizational Unit Name (eg, section) []:IT
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:test@localhost
```

- 服务器证书（Server Certificate）：

```
openssl genrsa -des3 -out server.key 2048
openssl req -out server.csr -key server.key -new
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt
```

- 客户端证书（Client Certificate）：

```
openssl genrsa -des3 -out client.key 2048
openssl req -out client.csr -key client.key -new
openssl x509 -req -in client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out client.crt
```

注意，对于机构证书的Common Name，须与服务器证书的Common Name须不一样，不然会产生认证问题。

MQTT 安全连接
MQTT Secure Connection

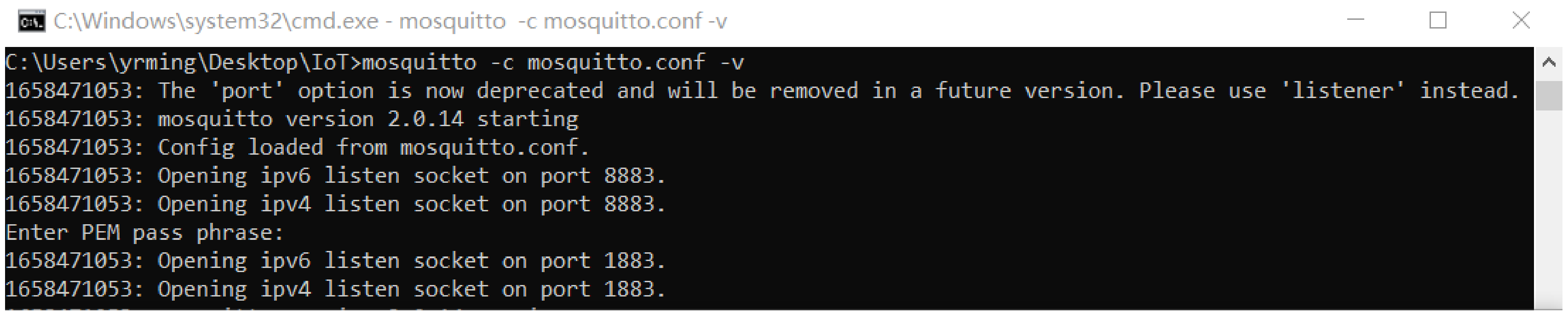
- 在所有证书具备后，即可使用这些证书进行配置安全连接。这里以mosquitto为例，下面为mosquitto的配置文件，额外指定了建立安全连接的相关参数：

  When the certificates are ready, we can configure secure connection using them. Exemplified by mosquito, the following shows an example configuration file specifying extra parameters:

```
port 1883
log_type error
log_type notice
log_type information
#allow_anonymous false
#password_file /etc/mosquitto/pass.txt
#Extra Listeners
listener 8883 localhost
```

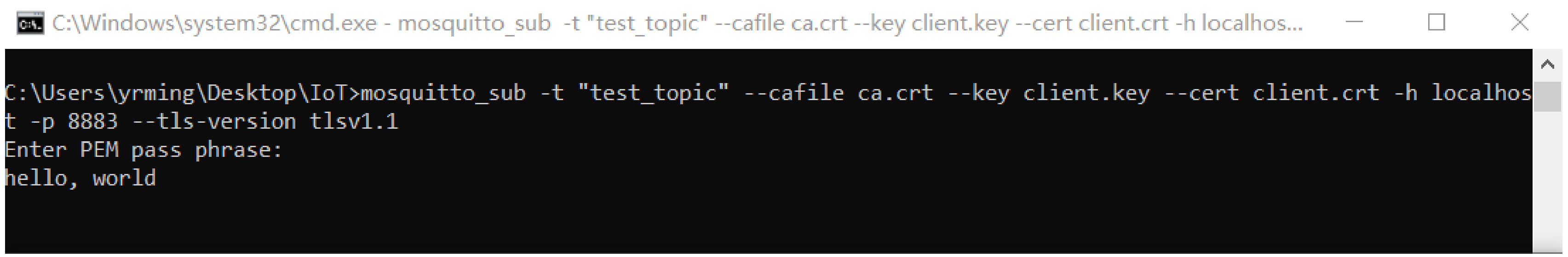
```
#ssl settings
cafile C:\Users\yrming\Desktop\IoT\ca.crt
keyfile C:\Users\yrming\Desktop\IoT\server.key
certfile C:\Users\yrming\Desktop\IoT\server.crt
# tls_version tlsv1.2
#client certifcate settings
require_certificate true
use_identity_as_username true
```

MQTT 安全连接
MQTT Secure Connection



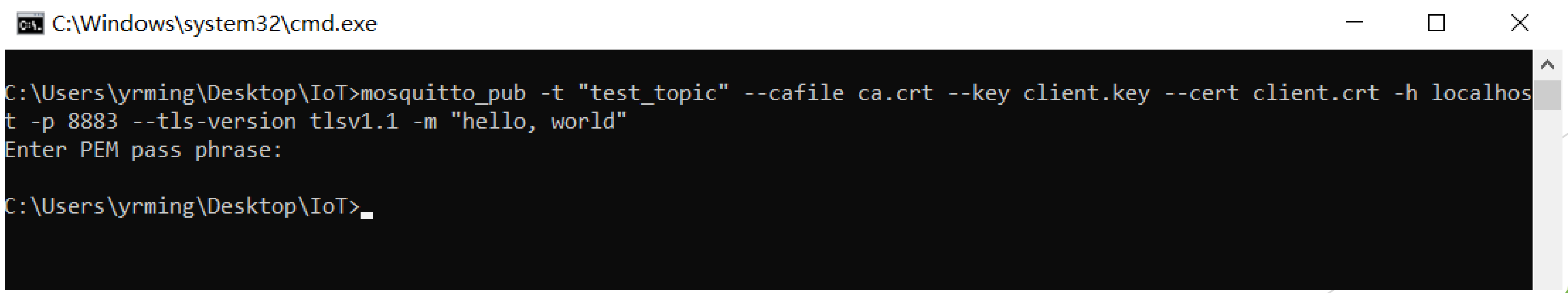
C:\Windows\system32\cmd.exe - mosquitto -c mosquitto.conf -v

```
C:\Users\yrming\Desktop\IoT>mosquitto -c mosquitto.conf -v
1658471053: The 'port' option is now deprecated and will be removed in a future version. Please use 'listener' instead.
1658471053: mosquitto version 2.0.14 starting
1658471053: Config loaded from mosquitto.conf.
1658471053: Opening ipv6 listen socket on port 8883.
1658471053: Opening ipv4 listen socket on port 8883.
Enter PEM pass phrase:
1658471053: Opening ipv6 listen socket on port 1883.
1658471053: Opening ipv4 listen socket on port 1883.
```



C:\Windows\system32\cmd.exe - mosquitto_sub -t "test_topic" --cafile ca.crt --key client.key --cert client.crt -h localhos...

```
C:\Users\yrming\Desktop\IoT>mosquitto_sub -t "test_topic" --cafile ca.crt --key client.key --cert client.crt -h localhos
t -p 8883 --tls-version tlsv1.1
Enter PEM pass phrase:
hello, world
```



C:\Windows\system32\cmd.exe

```
C:\Users\yrming\Desktop\IoT>mosquitto_pub -t "test_topic" --cafile ca.crt --key client.key --cert client.crt -h localhos
t -p 8883 --tls-version tlsv1.1 -m "hello, world"
Enter PEM pass phrase:

C:\Users\yrming\Desktop\IoT>_
```