January 5th, 2020

UNIVERSITEIT GENT

ALLTHINGSTALK

Internet of Things

2019/2020 - Master of Science in Computer Science Engineering

2019/2020 - Master of Science in Electrical Engineering

Group 6:

Afonso Pinto Afonso.Pinto@ugent.be

Andreia Seabra Andreia.Seabra@ugent.be

Tomás Oliveira Tomas.Oliveira@ugent.be

Index

1. Introduction

AllThingsTalk is a start-up company that offers an IoT Application Enablement platform for prototyping and full scale product deployment. The present report results from a quick assessment of that platform.

AllThingsTalkMaker[1], the AllThingsTalk Cloud Instance for Prototyping, is free to use for developers and innovators and was used to discover the accessible functionality of the AllThingsTalk Cloud and to experiment on how to put the Internet of Things to use in our line of business. Using the platform and one of the available client SDKs[2], we setted up a prototype that enables interactions to/from the Cloud and between the client and the Cloud platform. Real data (e.g. data generated by our PC such as CPU temperature and battery) was used to test the platform functionalities such as automation.

All the relevant code can be accessed in *https://github.ugent.be/afpinto/iot_all_things_talk*.

[1] https://maker.allthingstalk.com/
[2] http://docs.allthingstalk.com

2. Setup

To connect your device to the platform you need to do the following:

1. *git clone https://github.ugent.be/afpinto/iot_all_things_talk.git*
2. *cd iot_all_things_talk*
3. *pip install -r requirements.txt* (We recommend that you perform this step inside a Python Virtual Environment)
4. Create a new device at https://maker.allthingstalk.com
   a. Register / Sign in
   b. Select Devices, New Device, Your Own Device
   c. Choose a catchy name

5. Edit an .env file with your Device Token and Device ID
   a. Select your device at https://maker.allthingstalk.com
   b. Choose Settings, Authentication
   c. Your .env file should have a DEVICE_TOKEN and a DEVICE_ID
6. *python3 main.py*
7. Add logic rules:
   a. Select Rules, New Rules
   b. Add temperature and battery rules as shown below:

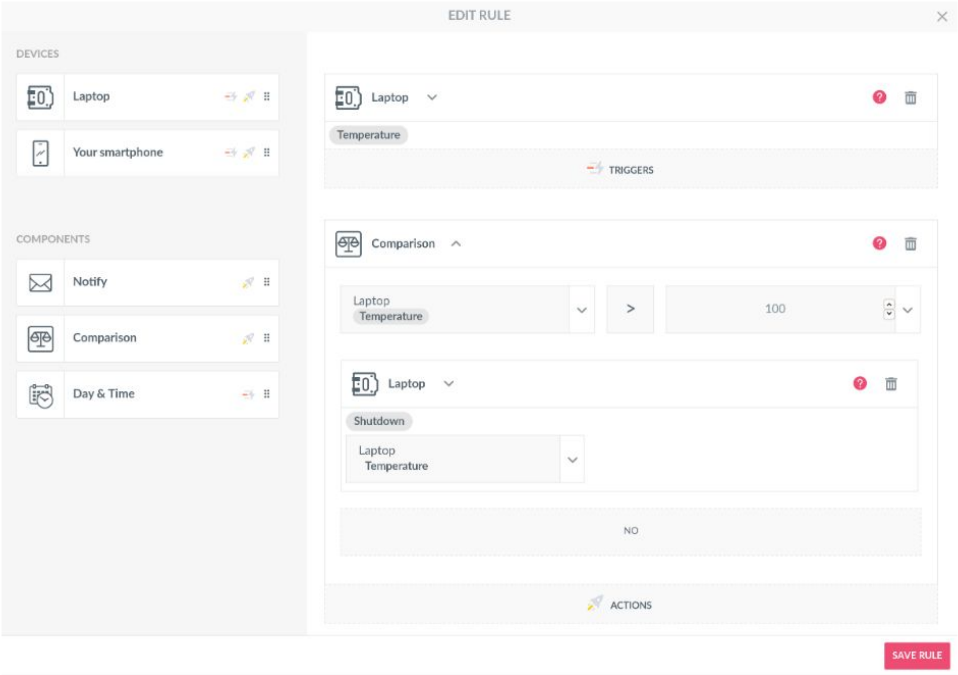


Figure 1: Temperature Rule

3. Functionalities

3.1 Application

The AllThingsTalk cloud application has support for multiple features, as shown in the following image.

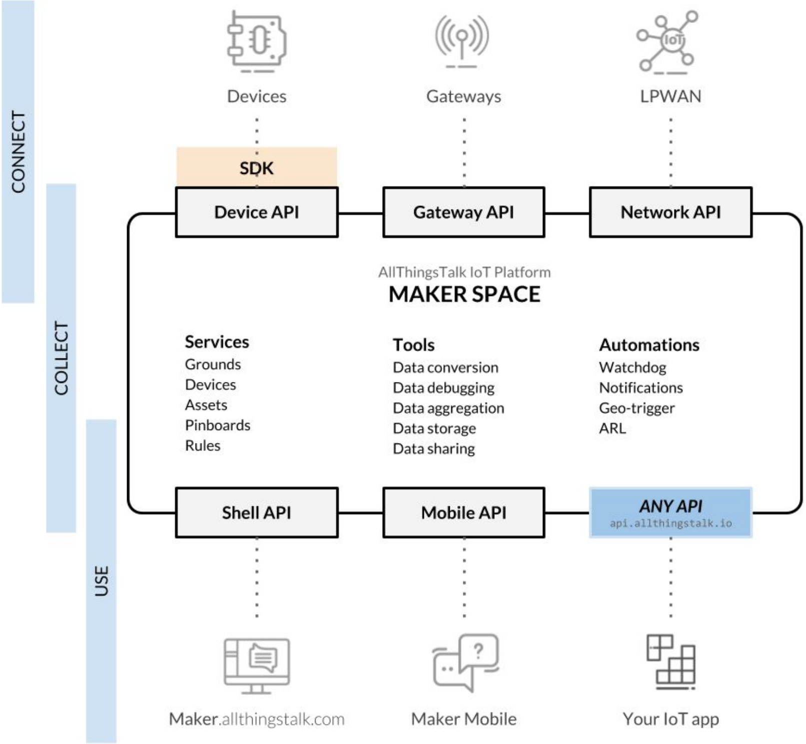


Figure 2: AllThingsTalk Functionalities

3.2 Prototype

Our application created with AllThingsTalk Maker has the following functionalities:

1. Management:
    a. Shows to the user the battery of the PC.
    b. Shows to the user the temperature of the core0 of the PC.
2. Automation:
    a. If the battery is less than 10%, the PC is suspended.
    b. If the temperature is bigger than 100ºC, the PC is suspended.

4. Security

The access to the devices is made by the use of tokens, there are two different types of token that grant different levels of access. Each token has information about the expire date, if it has one, which can be useful as a protection measure.

The device token allows the user to:

a. Add and configure assets;
b. Publish asset state;
c. Read asset feed;
d. Receive asset commands;

The ground token will give you extra permissions to also publish asset commands.

If you want to send information through the web browser or mobile app the authentication can be done with OAuth2 authorization protocol.

By default, only the owner has access to the rules, yet it is possible to give other users the access to it (and any other ground information).

In what refers to the communication, by default it is done with MQTT or HTTP, without SSL. However it is possible to provide an SSL certificate and use MQTTS or HTTPS which will create a secure connection between the server and the client. One of the main advantages of this secure connection is to prevent unauthorized third parties from reading the data even if they can intercept the packets increasing the customer's trust.

5. Platform

5.1 Data formats

The AllThingsTalk platform supports many formats to do the communication between the client and the server. Some of them are already well defined and well-known like JSON and CBOR, but it comes with the advantage of allowing the customer to create its own custom format by using ABCL.

The choice of which format to use must be done taking into account the desired application. For example, if we plan to use HTTP or MQTT protocols, JSON may be the best choice due to its lightweight data-interchange format, making it easy to read and write for both, machines and humans. Hence it being widely adopted on the web and therefore there is plenty of information available online making it easy to work, with just few knowledge.

The second possibility is CBOR, which is based on the JSON data model, however it allows the data transmission in a more concise way by giving the possibility to have extremely small code size and therefore a smaller message size making it and it also comes with the advantage of being able to grow without the need for version negotiation. Thus, it is the most common choice when dealing with constrained networks such as LoRa, Sigfox or NB-IOT.

Lastly, there is the possibility of using the AllThingsTalk Binary Conversion Language (ABCL) when none of the options described above fulfil our task, which is a JSON based, domain specific language that support any custom binary data format and consequently allows the user to create a custom format.

5.2 Communication

For the communication the Cloud allows the clients to send states,upplink, which contains an asset value at a specific time. It is normally used to send the sensor information to the cloud for further processing. For the downlink communication, the cloud is able to send commands that can trigger an actuation function or an actuator.

5.3 Example

As a simple example, to describe the communication process and possibilities described above, we associated a smartphone device to the AllThingsTalkMaker and using the quick demo feature we sent a few data related to device, one of which is the battery. Afterwards, we wrote a simple python script, smartphone.py, to retrieve the information sent to the cloud and traced the packet using wireshark (attachment A) which resulted in the following sequence diagram.

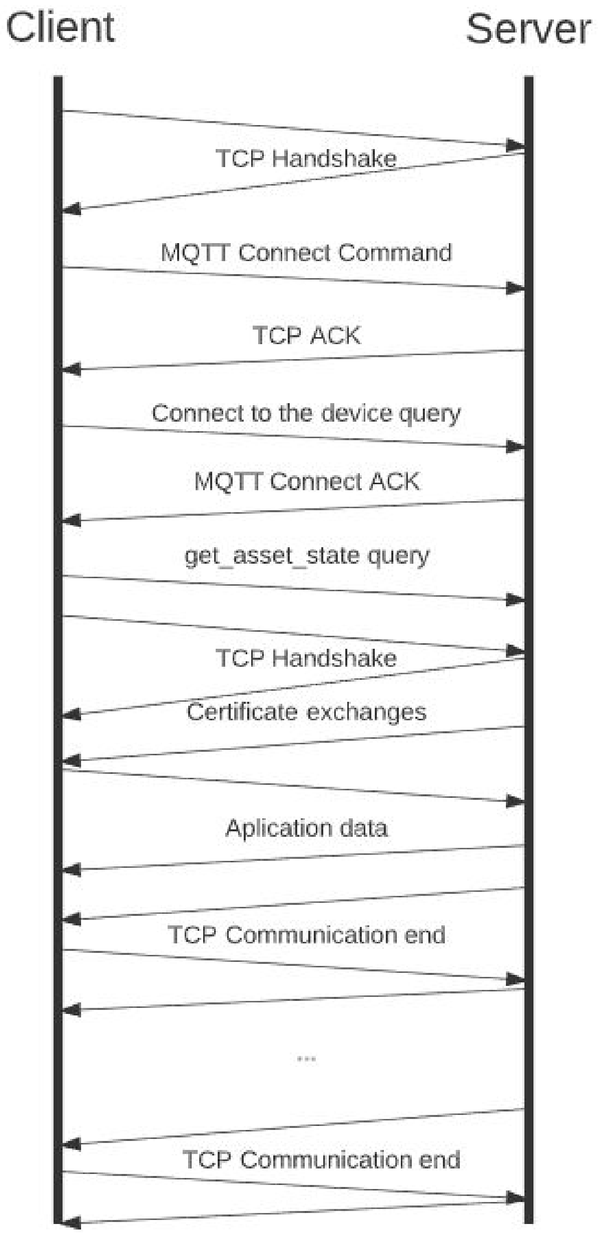


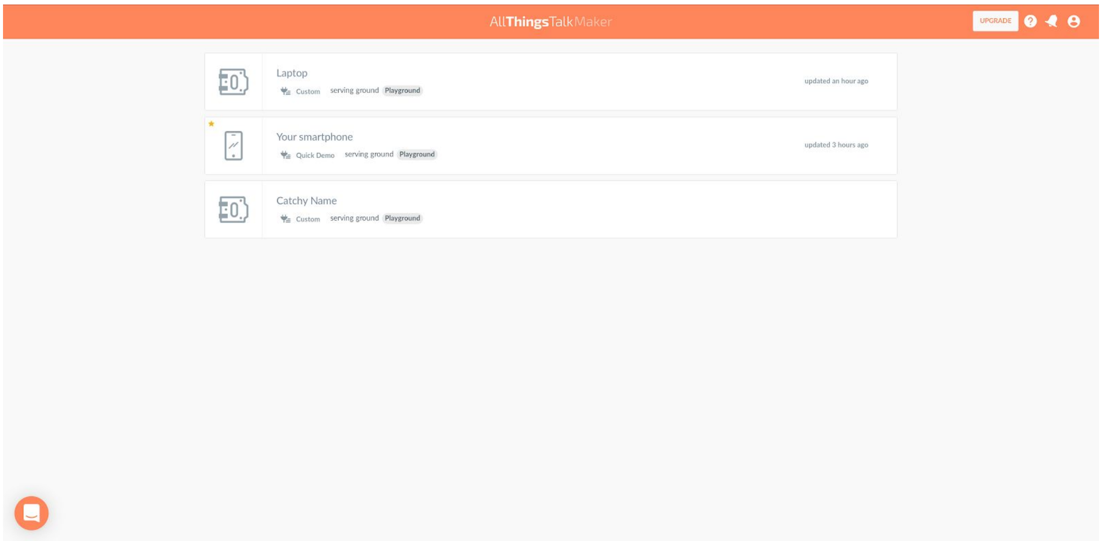
Figure 3: Sequence diagram using Lucidchard

As one can see, because the python client that we used, makes used of the JSON data format the communication will be used through MQTT and TCP, with one open TCP communication for each query.

6. Device Management

As an user you can have many different devices where you share business/app information. AllThingsTalk device management enables you to protect and secure that that data. As a ground owner, you can allow people in your ground to co-manage devices together with you.

Each device can be uniquely identified by a Device ID, besides that it also contains information like device title and description, assets associated with it and authentication details, between other settings. It’s a platform with ease device setup, efficient integrations, easy management of multiple device types and team friendly.



Laptop
Custom serving ground Playground
updated an hour ago

Your smartphone
Quick Demo serving ground Playground
updated 3 hours ago

Catchy Name
Custom serving ground Playground

Figure 4: Devices in the platform

7. IPR and Extensibility

Any user may use the information and documents offered in AllThingsTalk Cloud only for the purposes for which the information and documents were provided. Any other use of the information and/or documentations, especially any type of replication, alteration, or integration in any publishing or advertising of any kind is only permitted with prior, explicit and written consent of AllThingsTalk or the particular owner of the information. The information can be subject to various intellectual property rights and these property rights shall remain unaffected.

Except to the extent permitted by law, the user may not modify, distribute, prepare derivative works of, reverse engineer, reverse assemble, disassemble, decompile or otherwise attempt to decipher any code in connection with the AllThingsTalk Cloud and/or any other aspect of AllThingsTalk technology.

8. Alternative Platforms

8.1 Google Cloud Platform

Google Cloud provides a multi-layered secure infrastructure. It helps in improving operational efficiency. It provides predictive maintenance for equipment, solutions for smart cities & buildings, and real-time asset tracking. Some of their best features are the following:

- Machine learning capabilities for any IoT need.
- Real-time business insights for globally dispersed devices.
- AI capabilities.
- Provides support for a wide range of embedded operating systems.
- Location intelligence.

8.2 Zetta

Zetta is API based IoT platform based on Node.js. It is considered as a complete toolkit to make HTTP APIs for devices. Zetta combines REST APIs, WebSockets to make data-intensive and real-time applications. The following are some notable features:

- It can run on the cloud, or a PC, or even modest development boards.
- Easy interface and necessary programming to control sensors, actuators, and controllers.
- Allows developers to assemble smartphone apps, device apps, and cloud apps.
- It is developed for data-intensive and real-time applications.
- Turns any machine into an API.

8.3 ThingsBoard

ThingsBoard is for data collection, processing, visualization, and device management. It upholds all standard IoT protocols like CoAP, MQTT, and HTTP as quickly as cloud and on-premise deployments. It builds workflows based on design life cycle events, REST API events, RPC requests. Let’s take a look at the following ThingsBoard features:

- A stable platform that is combining scalability, production, and fault-tolerance.
- Easy control of all connected devices in an exceptionally secure system
- Transforms and normalizes device inputs and facilitates alarms for generating alerts on all telemetry events, restores, and inactivity.
- Enables use-state specific features using customizable rule groups.
- Handles millions of devices at the same time.

9. Conclusions

AllThingsTalk proposes a common approach to facilitate the evaluation and monitoring of multiple IOT devices. The Maker framework is great for fast prototyping and has an appealing user-interface.

It has many SDK available, from which we used the Python one; overall we found it easy to use and well documented.

On the not so bright side we noticed some user experience issues with the Maker UI, namely on the drag and drop of the rules interface;
the conditionals of that same rules doesn't work as we expected upon first value read:

Condition: Temperature > 60 -> Shutdown command

Program starts with Temperature at 61 and the condition is not triggered.

and the last issue we found is regarding the platform's support which seemed to us that were not used to work with the software and therefore they weren't much of a help.

In terms of our prototype we believe we met all the requirements.

10. References

I. “AllThingsTalk” https://maker.allthingstalk.com/.

II. “AllThingsTalk Python SDK.” AllThingsTalk Python SDK - AllThingsTalk Python SDK 0.1.0 documentation. https://allthingstalk.github.io/python-sdk/.

III. “Google Cloud IoT - Fully Managed IoT Services | Google Cloud.” Google. Google. https://cloud.google.com/solutions/iot/.

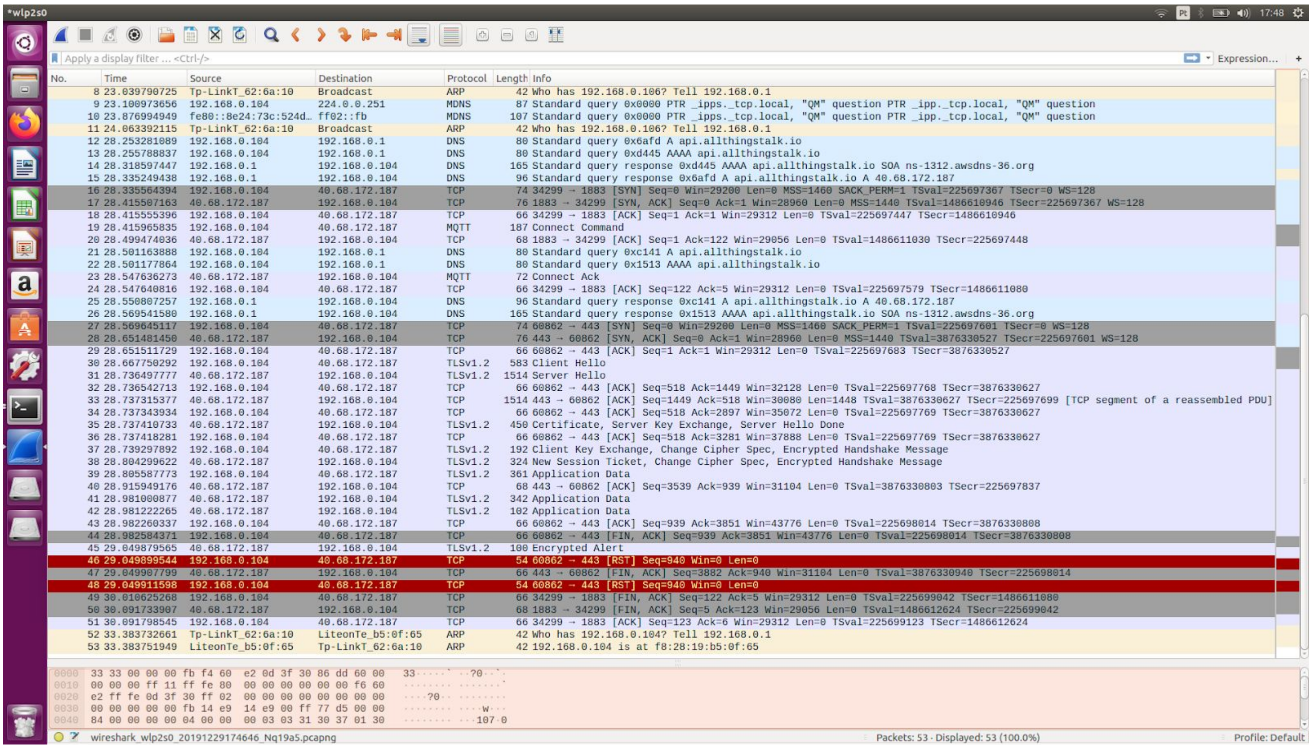
IV. “Online Diagram Software & Visual Solution.” Lucidchart. http://www.lucidchart.com/.

V. “Thingsboard. “Open-Source IoT Platform.” ThingsBoard. https://thingsboard.io/.

VI. “An API-First Internet of Things Platform.” Zetta. http://www.zettajs.org/.

11. Attachments

A. Wireshark capture



```
No.   Time         Source               Destination        Protocol Length Info
    8 23.039790725 Tp-LinkT_62:6a:10    Broadcast          ARP        42 Who has 192.168.0.106? Tell 192.168.0.1
    9 23.100973656 192.168.0.104        224.0.0.251        MDNS       87 Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question
   10 23.876994949 fe80::8e24:73c:524d… ff02::fb           MDNS      107 Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question
   11 24.063392115 Tp-LinkT_62:6a:10    Broadcast          ARP        42 Who has 192.168.0.106? Tell 192.168.0.1
   12 28.253281089 192.168.0.104        192.168.0.1        DNS        80 Standard query 0x6afd A api.allthingstalk.io
   13 28.255788837 192.168.0.104        192.168.0.1        DNS        80 Standard query 0xd445 AAAA api.allthingstalk.io
   14 28.318597447 192.168.0.1          192.168.0.104      DNS       165 Standard query response 0xd445 AAAA api.allthingstalk.io SOA ns-1312.awsdns-36.org
   15 28.335249438 192.168.0.1          192.168.0.104      DNS        96 Standard query response 0x6afd A api.allthingstalk.io A 40.68.172.187
   16 28.335564394 192.168.0.104        40.68.172.187      TCP        74 34299 → 1883 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=225697367 TSecr=0 WS=128
   17 28.415507163 40.68.172.187        192.168.0.104      TCP        76 1883 → 34299 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1440 TSval=1486610946 TSecr=225697367 WS=128
   18 28.415555396 192.168.0.104        40.68.172.187      TCP        66 34299 → 1883 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=225697447 TSecr=1486610946
   19 28.415965835 192.168.0.104        40.68.172.187      MQTT      187 Connect Command
   20 28.499474036 40.68.172.187        192.168.0.104      TCP        68 1883 → 34299 [ACK] Seq=1 Ack=122 Win=29056 Len=0 TSval=1486611030 TSecr=225697448
   21 28.501163888 192.168.0.104        192.168.0.1        DNS        80 Standard query 0xc141 A api.allthingstalk.io
   22 28.501177864 192.168.0.104        192.168.0.1        DNS        80 Standard query 0x1513 AAAA api.allthingstalk.io
   23 28.547636273 40.68.172.187        192.168.0.104      MQTT       72 Connect Ack
   24 28.547640816 192.168.0.104        40.68.172.187      TCP        66 34299 → 1883 [ACK] Seq=122 Ack=5 Win=29312 Len=0 TSval=225697579 TSecr=1486611080
   25 28.550807257 192.168.0.1          192.168.0.104      DNS        96 Standard query response 0xc141 A api.allthingstalk.io A 40.68.172.187
   26 28.569541580 192.168.0.1          192.168.0.104      DNS       165 Standard query response 0x1513 AAAA api.allthingstalk.io SOA ns-1312.awsdns-36.org
   27 28.569645117 192.168.0.104        40.68.172.187      TCP        74 60862 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=225697601 TSecr=0 WS=128
   28 28.651481450 40.68.172.187        192.168.0.104      TCP        76 443 → 60862 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1440 TSval=3876330527 TSecr=225697601 WS=128
   29 28.651511729 192.168.0.104        40.68.172.187      TCP        66 60862 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=225697683 TSecr=3876330527
   30 28.667750292 192.168.0.104        40.68.172.187      TLSv1.2   583 Client Hello
   31 28.736497777 40.68.172.187        192.168.0.104      TLSv1.2  1514 Server Hello
   32 28.736542713 192.168.0.104        40.68.172.187      TCP        66 60862 → 443 [ACK] Seq=518 Ack=1449 Win=32128 Len=0 TSval=225697768 TSecr=3876330627
   33 28.737315377 40.68.172.187        192.168.0.104      TCP      1514 443 → 60862 [ACK] Seq=1449 Ack=518 Win=30080 Len=1448 TSval=3876330627 TSecr=225697699 [TCP segment of a reassembled PDU]
   34 28.737343934 192.168.0.104        40.68.172.187      TCP        66 60862 → 443 [ACK] Seq=518 Ack=2897 Win=35072 Len=0 TSval=225697769 TSecr=3876330627
   35 28.737410733 40.68.172.187        192.168.0.104      TLSv1.2   450 Certificate, Server Key Exchange, Server Hello Done
   36 28.737418281 192.168.0.104        40.68.172.187      TCP        66 60862 → 443 [ACK] Seq=518 Ack=3281 Win=37888 Len=0 TSval=225697769 TSecr=3876330627
   37 28.739297892 192.168.0.104        40.68.172.187      TLSv1.2   192 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
   38 28.804299622 40.68.172.187        192.168.0.104      TLSv1.2   324 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
   39 28.805587773 192.168.0.104        40.68.172.187      TLSv1.2   361 Application Data
   40 28.915949176 40.68.172.187        192.168.0.104      TCP        68 443 → 60862 [ACK] Seq=3539 Ack=939 Win=31104 Len=0 TSval=3876330803 TSecr=225697837
   41 28.981000877 40.68.172.187        192.168.0.104      TLSv1.2   342 Application Data
   42 28.981222265 40.68.172.187        192.168.0.104      TLSv1.2   102 Application Data
   43 28.982260337 192.168.0.104        40.68.172.187      TCP        66 60862 → 443 [ACK] Seq=939 Ack=3851 Win=43776 Len=0 TSval=225698014 TSecr=3876330808
   44 28.982584371 192.168.0.104        40.68.172.187      TCP        66 60862 → 443 [FIN, ACK] Seq=939 Ack=3851 Win=43776 Len=0 TSval=225698014 TSecr=3876330808
   45 29.049879565 40.68.172.187        192.168.0.104      TLSv1.2   100 Encrypted Alert
   46 29.049899544 192.168.0.104        40.68.172.187      TCP        54 60862 → 443 [RST] Seq=940 Win=0 Len=0
   47 29.049907799 40.68.172.187        192.168.0.104      TCP        66 443 → 60862 [FIN, ACK] Seq=3882 Ack=940 Win=31104 Len=0 TSval=3876330940 TSecr=225698014
   48 29.049911598 192.168.0.104        40.68.172.187      TCP        54 60862 → 443 [RST] Seq=940 Win=0 Len=0
   49 30.010625268 192.168.0.104        40.68.172.187      TCP        66 34299 → 1883 [FIN, ACK] Seq=122 Ack=5 Win=29312 Len=0 TSval=225699042 TSecr=1486611080
   50 30.091733907 40.68.172.187        192.168.0.104      TCP        68 1883 → 34299 [FIN, ACK] Seq=5 Ack=123 Win=29056 Len=0 TSval=1486612624 TSecr=225699042
   51 30.091798545 192.168.0.104        40.68.172.187      TCP        66 34299 → 1883 [ACK] Seq=123 Ack=6 Win=29312 Len=0 TSval=225699123 TSecr=1486612624
   52 33.383732661 Tp-LinkT_62:6a:10    LiteonTe_b5:0f:65  ARP        42 Who has 192.168.0.104? Tell 192.168.0.1
   53 33.383751949 LiteonTe_b5:0f:65    Tp-LinkT_62:6a:10  ARP        42 192.168.0.104 is at f8:28:19:b5:0f:65
```

```
0000  33 33 00 00 00 fb f4 60  e2 0d 3f 30 86 dd 60 00   33·····` ··?0··`·
0010  00 00 00 ff 11 ff fe 80  00 00 00 00 00 00 f6 60   ········ ·······`
0020  e2 ff fe 0d 3f 30 ff 02  00 00 00 00 00 00 00 00   ····?0·· ········
0030  00 00 00 00 00 fb 14 e9  14 e9 00 ff 77 d5 00 00   ········ ····w···
0040  84 00 00 00 00 04 00 00  00 03 03 31 30 37 01 30   ········ ···107·0
```

wireshark_wlp2s0_20191229174646_Nq19a5.pcapng   Packets: 53 · Displayed: 53 (100.0%)   Profile: Default