Development of IoTcloudServe@TEIN Smart-Energy@Chula Service Gateway : Case Study of Secured On-Demand Building Energy Management System Data Platform Using NETPIE

Ririnda Thirasupa

Meechai Homchan

Siravit Kwankajornkeat

Chaodit Aswakul

Department of Electrical Engineering, Faculty of Engineering,
Chulalongkorn University, Bangkok, Thailand

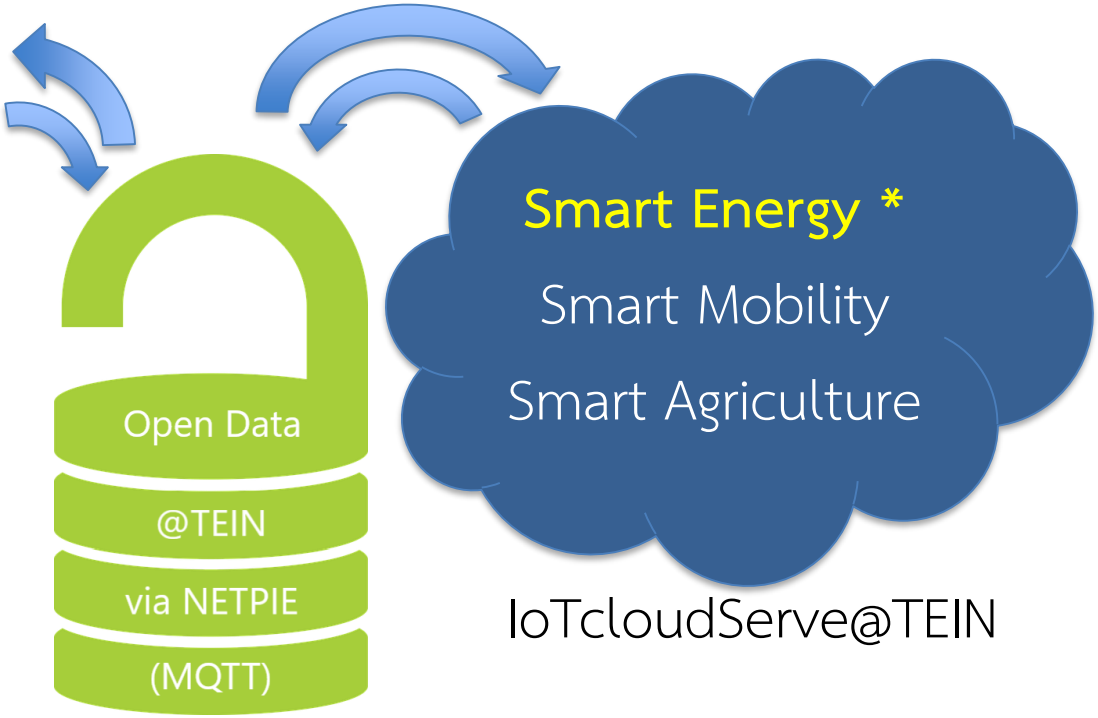
Challenges

- There may be many IoTs with different protocols in the future.
  - At first, assume that there is nothing used to translate protocols
- How to request for secured on-demand data?
  - Use NETPIE to secure data, but there is nothing used to request data

Overview of this Research

100th Anniversary of Chula Engineering 2013

CU-BEMS : Building Energy Management in EECU (IEEE 1888)

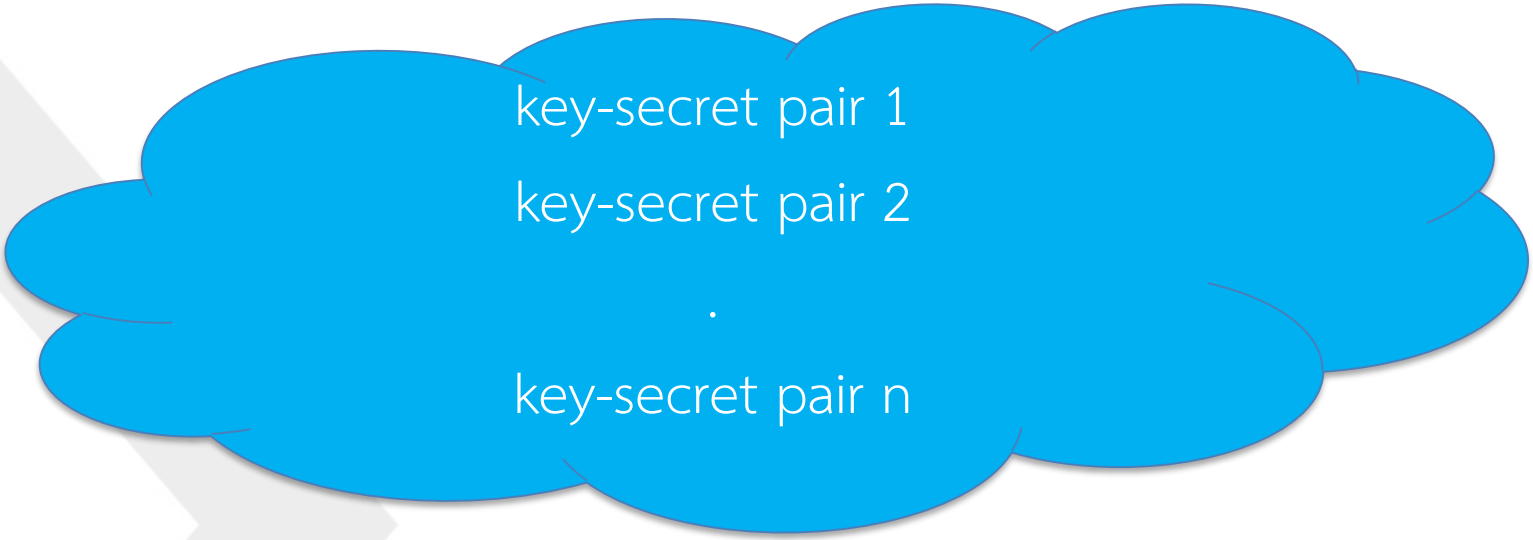


IoTcloudServe@TEIN

Join CU-BEMS to IoTcloudServe@TEIN via NETPIE

Ref & Pictures Credit : https://apan.net/meetings/apan46/files/11/11-02-04-01.pdf & https://pixabay.com/

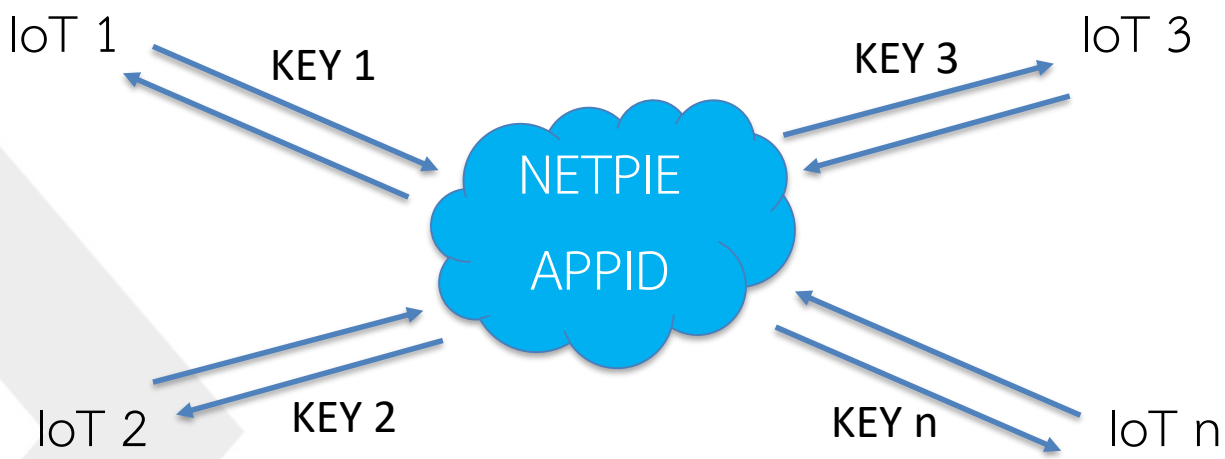
NETPIE Security Mechanisms



NETPIE APPID

There are key-secret pairs created in each NETPIE APPID

NETPIE Security Mechanisms (cont.)



Only the IoTs which uses key-secret pairs created in the same APPID can exchange data to each other

Ref : https://netpie.io/

Objectives

- Create a **service gateway** used to transmit data between IoTs with different protocols (IEEE 1888 & MQTT)

- Also create a html webpage connected to NETPIE to exchange data with the **gateway**

- Use *NETPIE security mechanisms* to check users (html webpage) authentication and authorization before access to the **gateway**

Scopes & Design

- Focus on sensors and actuators in *CU-BEMS (Smart-Energy@Chula)* only.
- Create **node.js service gateway** to transmit the sensors' data.
- Create only html webpage to request data from *CU-BEMS*.
- Create Topics in *NETPIE* to be data exchange points between html webpage and **node.js service gateway**.

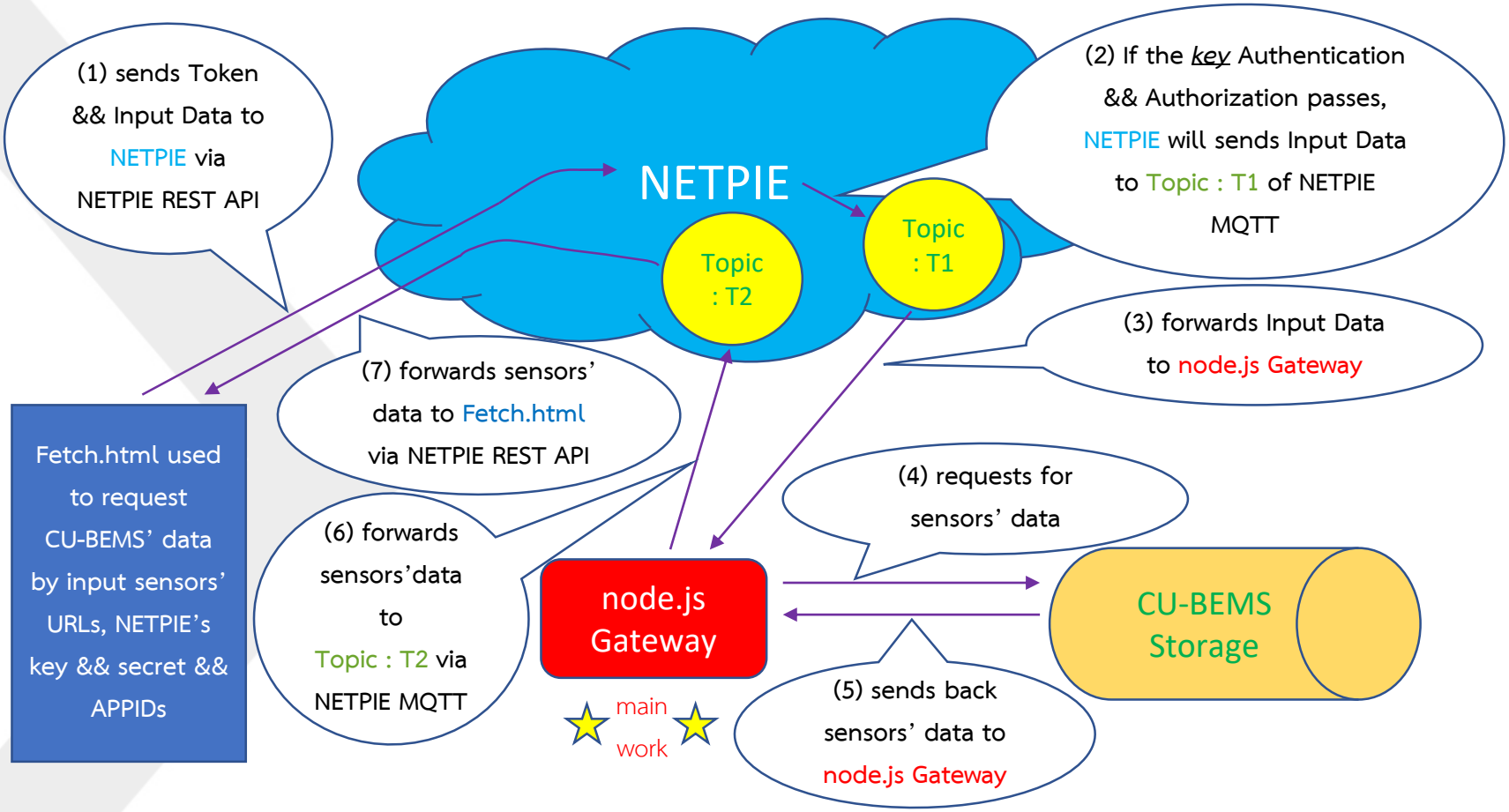
Initial Set-Up before Fetching

- Admin creates APPID, key-secret pairs in the APPID for the **service Gateway** and Fetch html (html webpage).

- Admin sets Topics in the **Gateway** and Fetch html for data exchanging between them.

- Admin runs the **Gateway** on a server ⟶ the **Gateway** connect to *NETPIE*.

Fetching Procedures



Fetching Result

Point ID #1 :

www.dr100.com/north/cmi/cmi2/meter/1/monitor/power_all_1m

Point ID #2 :

www.dr100.com/northeast/nma/nma9/meter/1/monitor/power_all_1m

APP ID :

CUBEMS

KEY :

GCMw2epbF8xdP3k

SECRET :

sKdeoP00gmbxc9eRHe3GoGcwq

Start FETCH | Stop FETCH

Fetched result:

xmlns:ns2="http://soap.fiap.org/"><transport xmlns="http://gutp.jp/fiap/2009/11/"><header><OK />
<query id="12ed9de4-1c48-4b08-a41d-dac067fc1c0d" type="storage" acceptableSize="1000"><key
id="www.dr100.com/north/cmi/cmi2/meter/1/monitor/power_all_1m" attrName="time" select="maximum" />
<key id="www.dr100.com/northeast/nma/nma9/meter/1/monitor/power_all_1m" attrName="time"
select="maximum" /></query></header><body><point
id="www.dr100.com/north/cmi/cmi2/meter/1/monitor/power_all_1m"><value time="2017-11-
22T01:10:08.000+07:00">319.2</value></point><point
id="www.dr100.com/northeast/nma/nma9/meter/1/monitor/power_all_1m"><value time="2017-03-
27T10:59:14.000+07:00">496.2</value></point></body></transport></ns2:queryRS></soapenv:Body>
</soapenv:Envelope>

sensos' values

Gateway Evaluation

Test **node.js service gateway** performance

- Time delay : time duration since the gateway sends request data to CU-BEMS until it receives CU-BEMS reply data
- CPU usage : percentage of CPU usage of server which runs the **gateway** while fetching data
- Server specifications : Windows 10 Pro 64 bits, 16 GB RAM, and Core i7-8850U, 1.8 GHz, 8 Cores CPU
- 30 minutes continuously test via 20 Mbps internet package

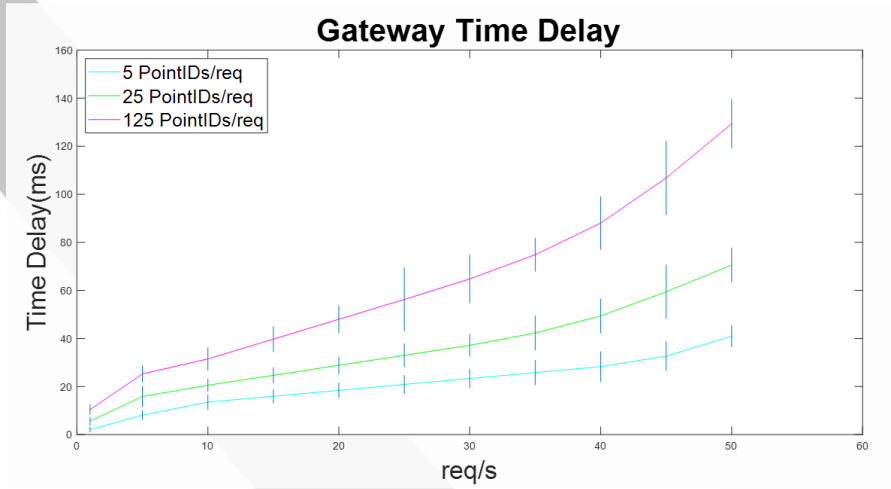
Gateway Evaluation (cont.)

⟶ 3 main cases : fetching sensors' data at 5, 25, and 125 URLs per gateway request

⟶ 11 subcases : fetching the data at rate 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50 requests per second (req/s) for each main cases

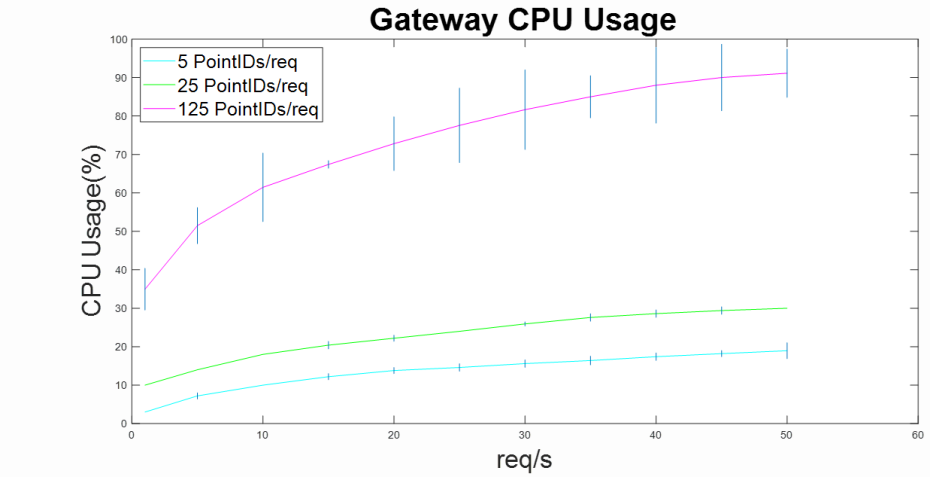
Each case was repeated for 25 times using 95% confidence interval

Gateway Performance Test Results



Time delay depends on CPU usage which depends on a number of requested sensors per second.

A vertical blue line represents mean range of each case which depends on network state between the server and CU-BEMS at a time.



Conclusion

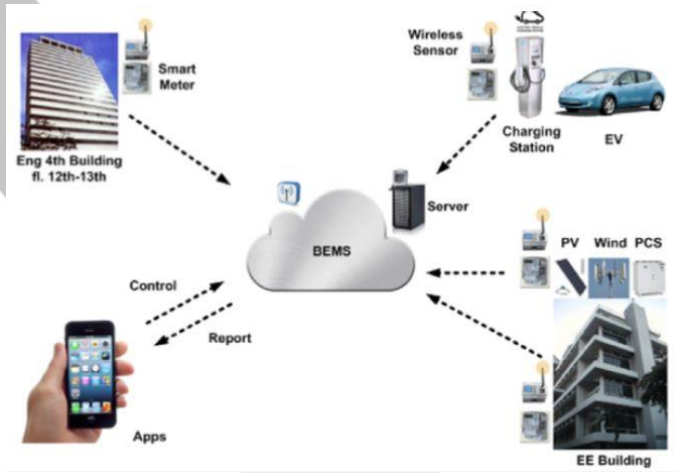
- The development of the **Gateway** in this research enhances the building energy management to be compliant with different communication protocols, which could be useful for researchers and manufacturers in the future.

- Secured on-demand data can be requested by using the **Gateway** to fetch CU-BEMS's sensor values from the html webpage with NETPIE Security Mechanisms

Thank You

A closer look of the existing systems

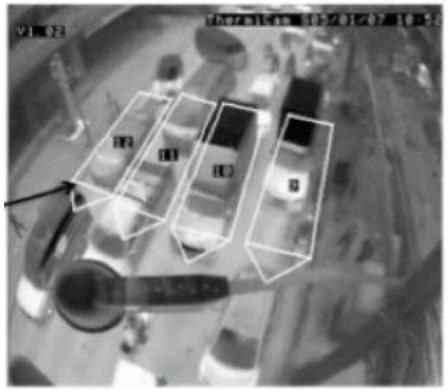
IoTcloudServe@TEIN



Smart Energy

(My Work)

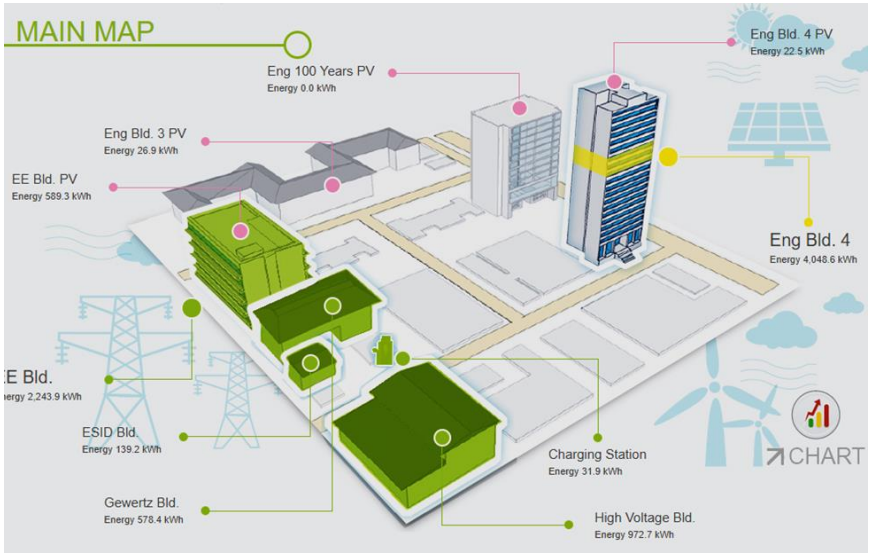
Smart Agriculture



Smart Mobility

Ref & Pictures Credit : https://apan.net/meetings/apan46/files/11/11-02-04-01.pdf

Smart-Energy@Chula (CU-BEMS)



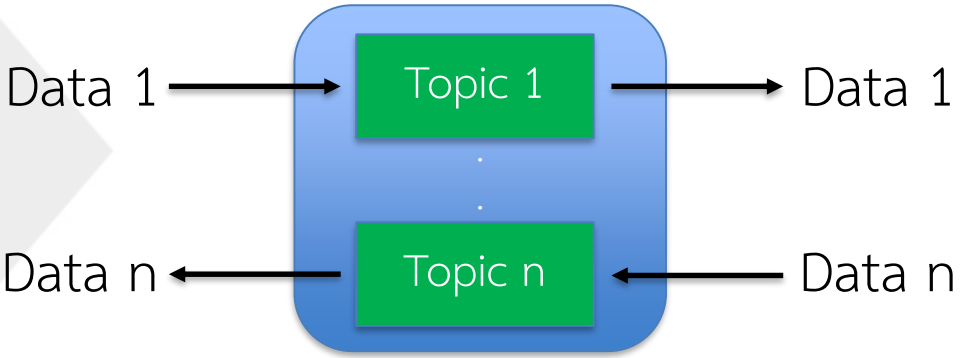
- IEEE 1888 protocol server with sensors and actuators in EECU
- More than 250 energy-related sensors and smart meters send the real-time energy and room ambient readings to CU-BEMS storage

Ref : https://apan.net/meetings/apan46/files/11/11-02-04-01.pdf & http://www.bems.ee.eng.chula.ac.th:9061/bems.web/

IEEE 1888 & MQTT Protocols

- IEEE 1888 : Simple Object Access Protocol (SOAP) ⟶ support only data in XML format

- MQTT : exchange data via created Topics

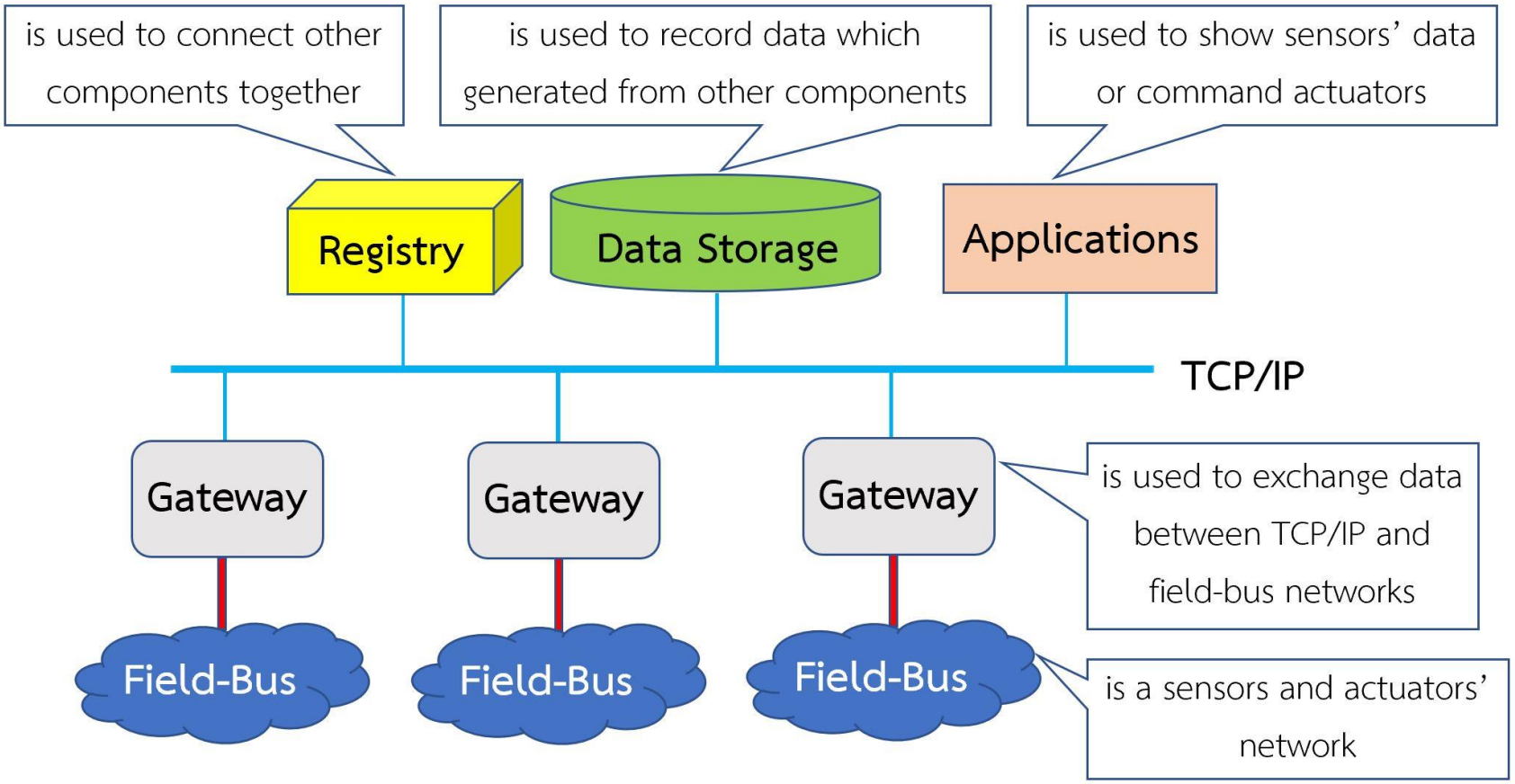


MQTT Broker

Ref : https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7098298 & https://netpie.io/

A closer look of IEEE 1888

IEEE 1888 Protocol



Ref : https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7098298

IEEE 1888 Communication Protocols

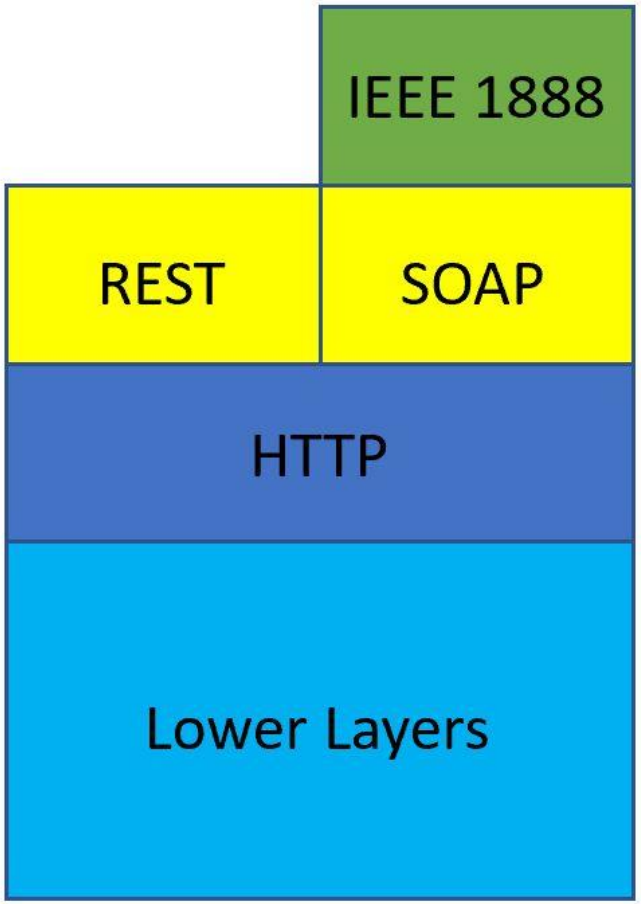
- FETCH is used to request *data* from one component to another.
- TRAP is used to request *warning according to set conditions* from one component to another.
- WRITE is used to *write* data from one component to another.

IEEE 1888 Message Protocol

Simple object access protocol, SOAP supports only XML data

Ref : https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7098298

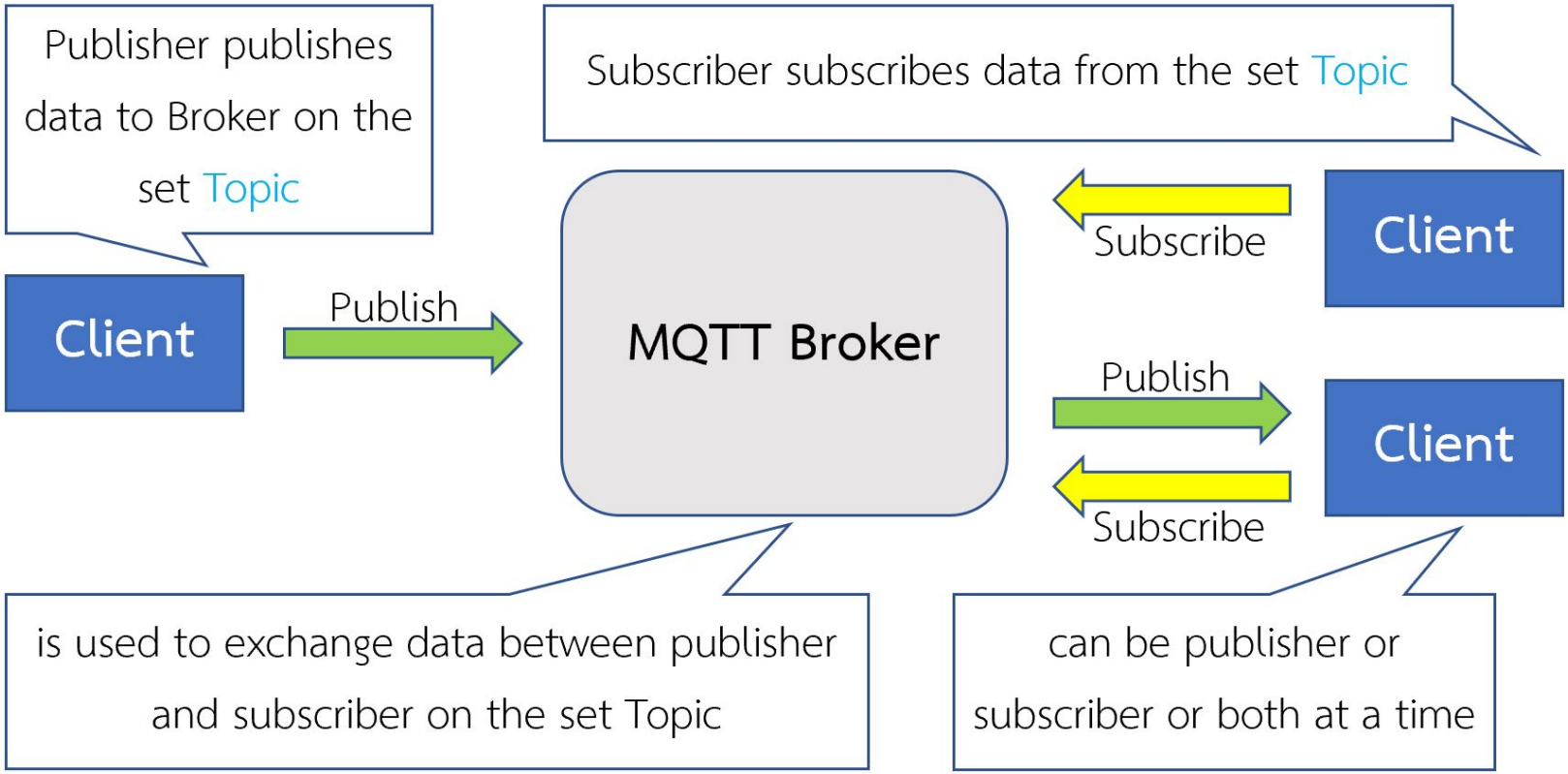
Architecture of each Hardware



Ref : https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7098298

A closer look of MQTT & Microgear

MQTT Protocol



Operates on TCP/IP network

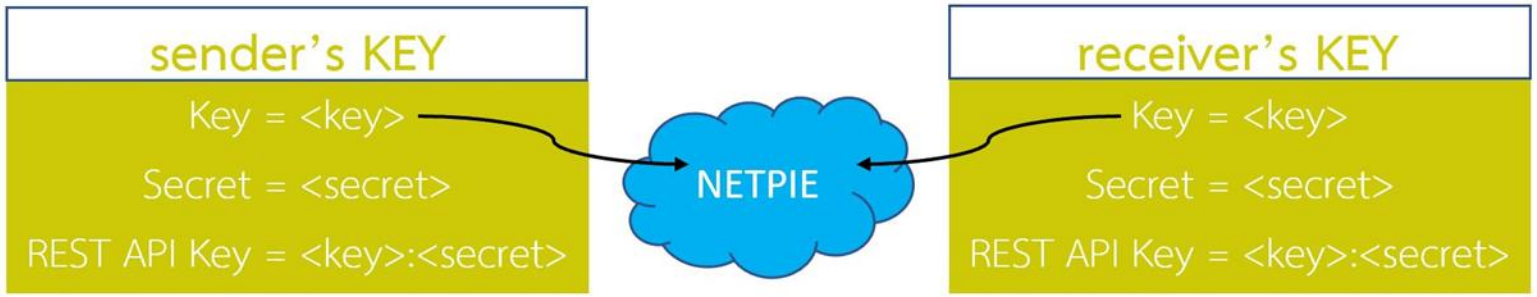
Ref : https://netpie.io/

NETPIE Microgear

- Be installed on devices (except HTML 5 Microgear) that want to connect to NETPIE by using MQTT protocol.

- Uses to authenticate & authorize devices that want to join NETPIE by using AppID, Key in KEY, and Token.

NETPIE Microgear (cont.)



Each user sends Key to NETPIE as public. Sender always encrypts data by using receiver's Key before sends it to receiver. Receiver decrypts the data by using receiver's Secret. For each Key, there is only its pair Secret to decrypts data which encrypted by the Key. So, there is only Key's owner who knows its pair Secret.

Ref : https://netpie.io/

NETPIE Microgear (cont.)

	Pre-approved KEY	Third-party KEY
Device KEY		
Session KEY		

Device KEY is permanent, so, the IoTs which use this KEY receive permanent Token.

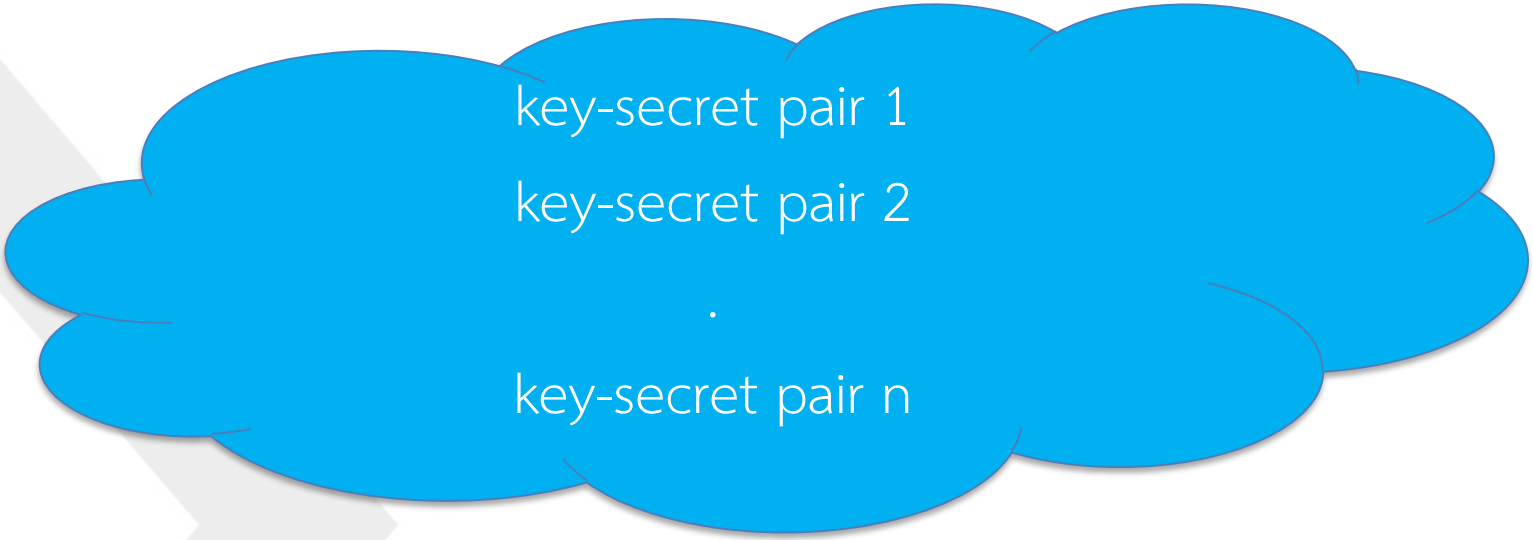
Session KEY is temporarily (Token will be cleared after the device is offline), so, the IoTs which use this KEY must request Token every time to enter NETPIE except Pre-approved Session KEY.

Pre-approved KEY, which is a KEY in owner's AppID, is auto-approved, so, the IoTs which use this KEY will receive Tokens immediately after connect to NETPIE.

Third-party KEY is an outsider KEY allowed to join owner's data.

Ref : https://netpie.io/

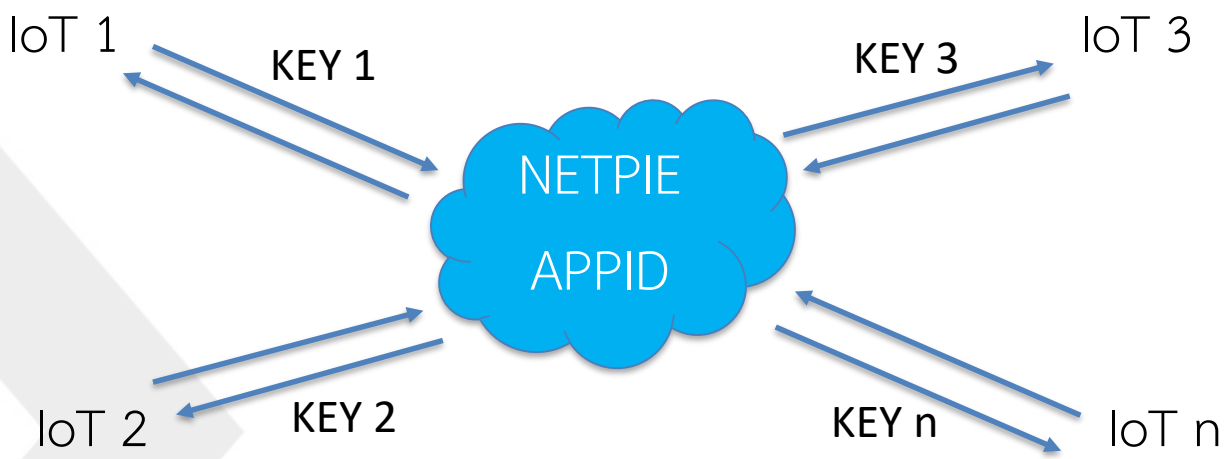
NETPIE Security Mechanisms



NETPIE APPID

There are key-secret pairs created in each NETPIE APPID

NETPIE Security Mechanisms (cont.)



only the IoTs which uses key-secret pairs created in the same APPID can exchange data to each other

Ref : https://netpie.io/