Intro to NETPIE

NETPIE R&D TEAM

NECTEC

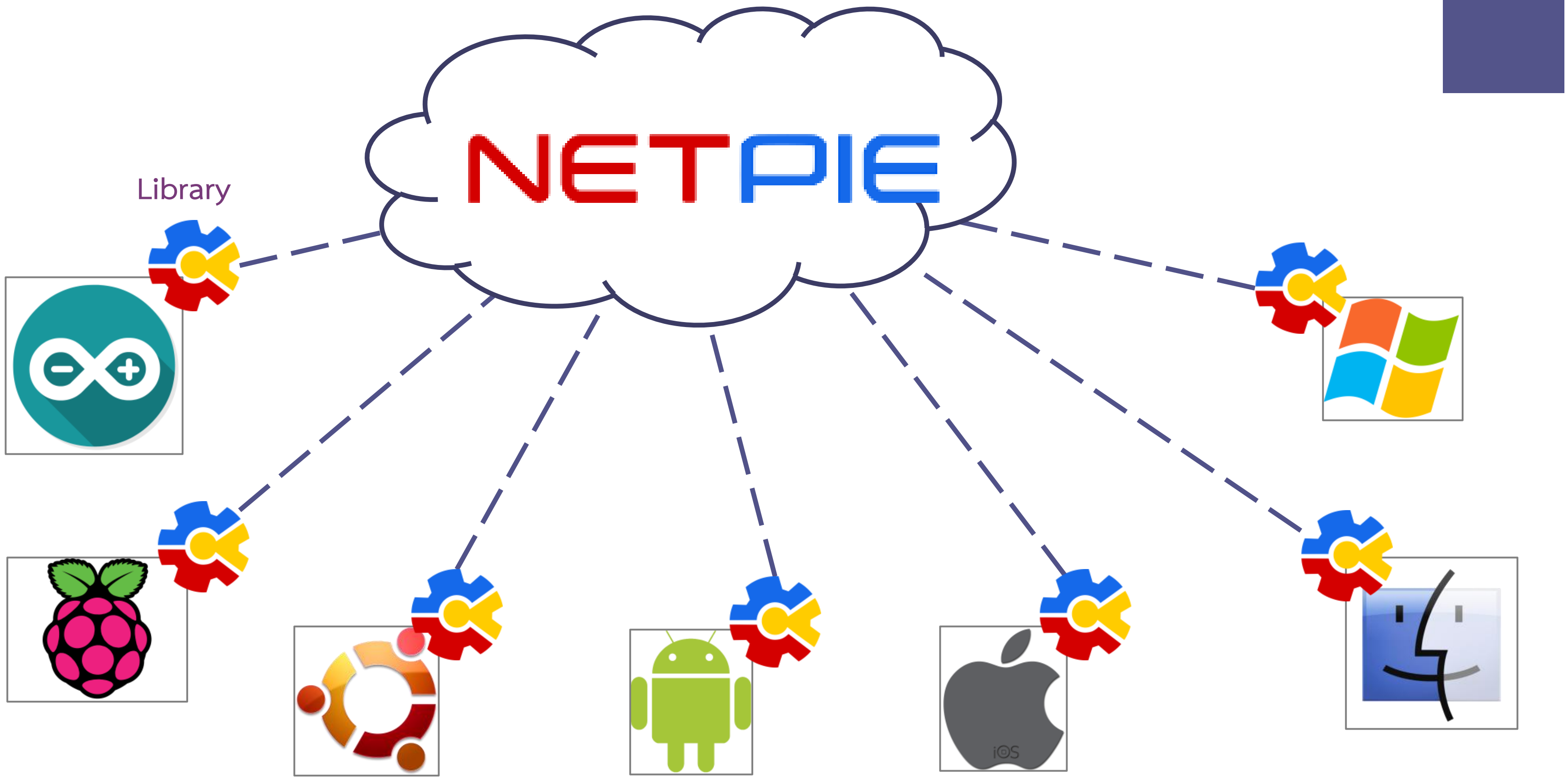
NETPIE

Facts about NETPIE

- An IoT platform focusing on real-time communication
- Provide services to things rather than to human
- “Social network of things”

In other words, NETPIE is :

a Cloud Platform + Libraries



What NETPIE Offers

Instant-Message Communication

Reliable Backend

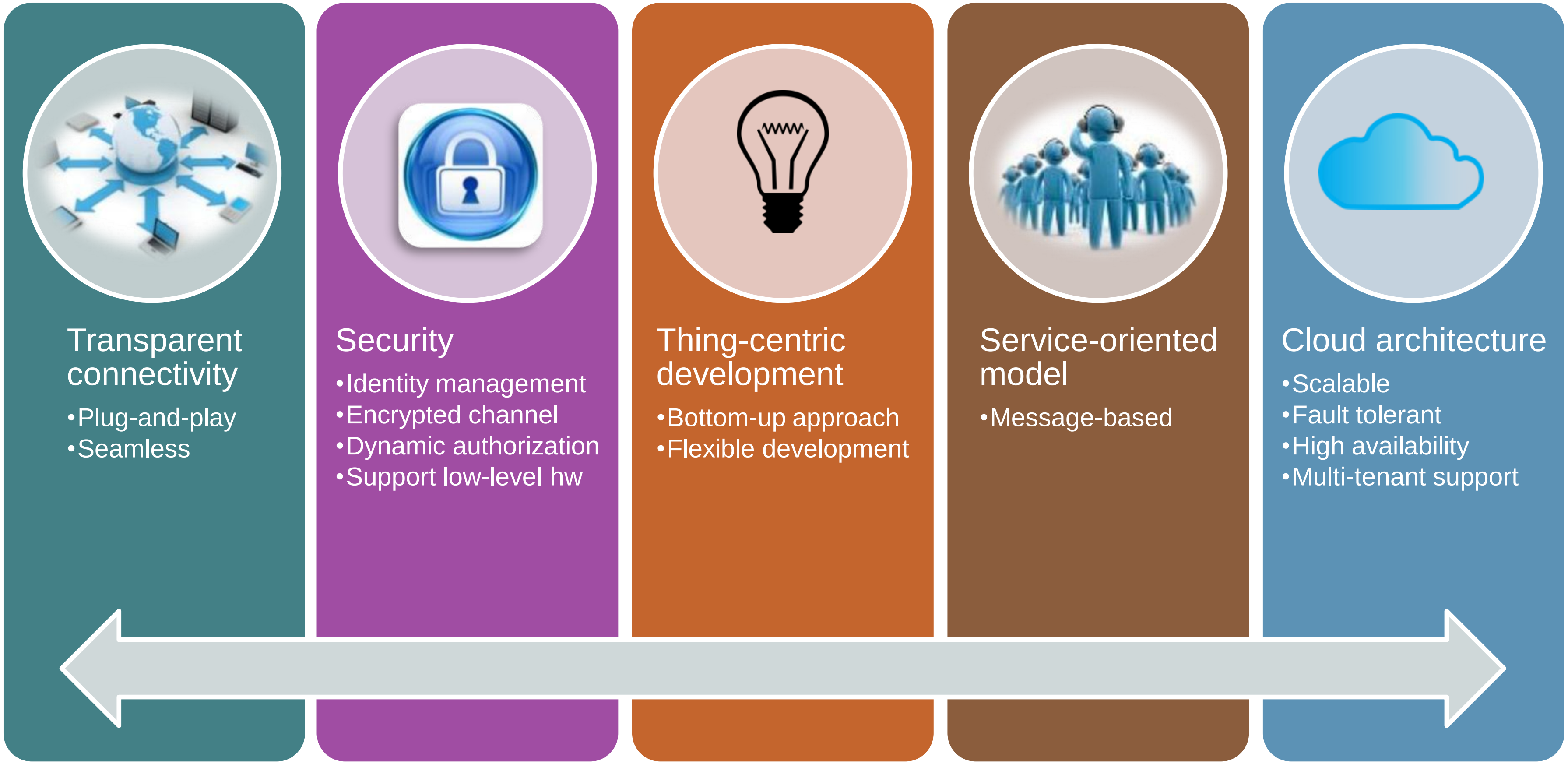
Access Control

Local Data Storage

Connection Security

Domestic Communication

Features



Transparent connectivity

- Plug-and-play
- Seamless

Security

- Identity management
- Encrypted channel
- Dynamic authorization
- Support low-level hw

Thing-centric development

- Bottom-up approach
- Flexible development

Service-oriented model

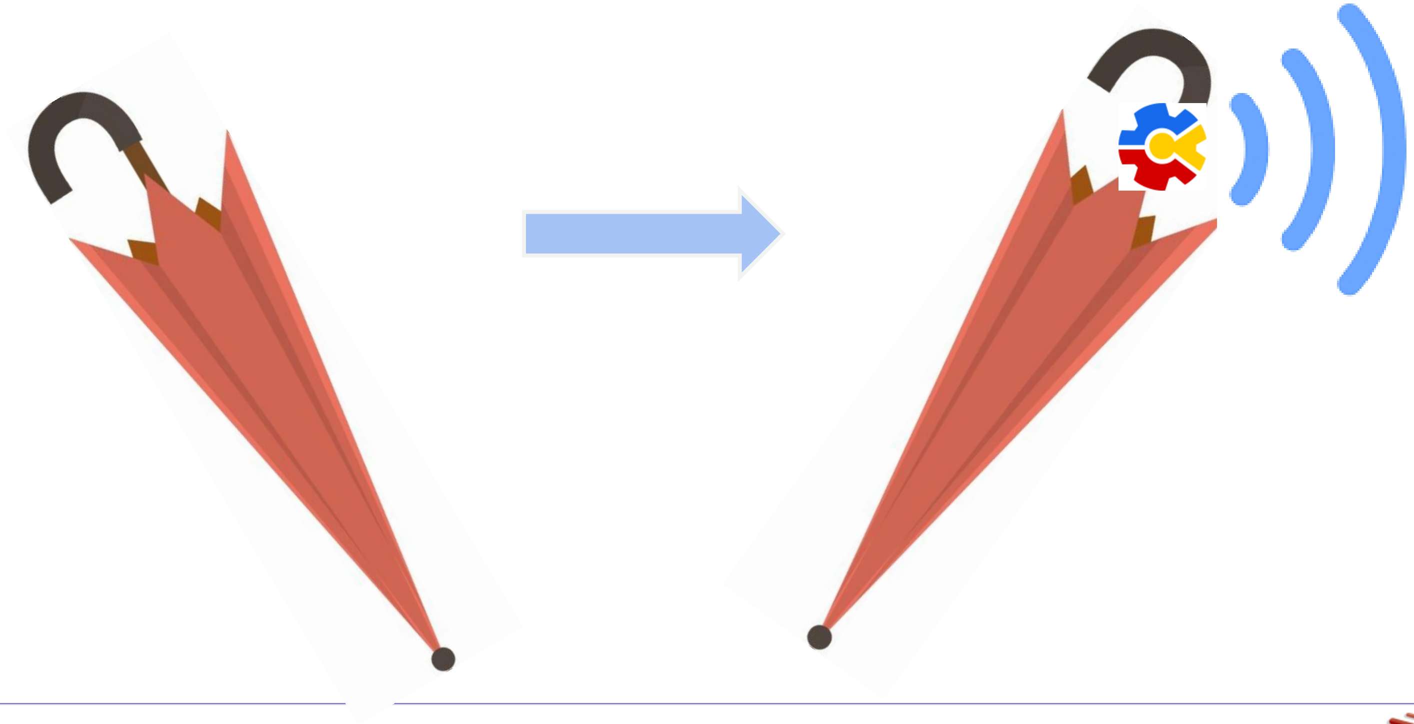
- Message-based

Cloud architecture

- Scalable
- Fault tolerant
- High availability
- Multi-tenant support

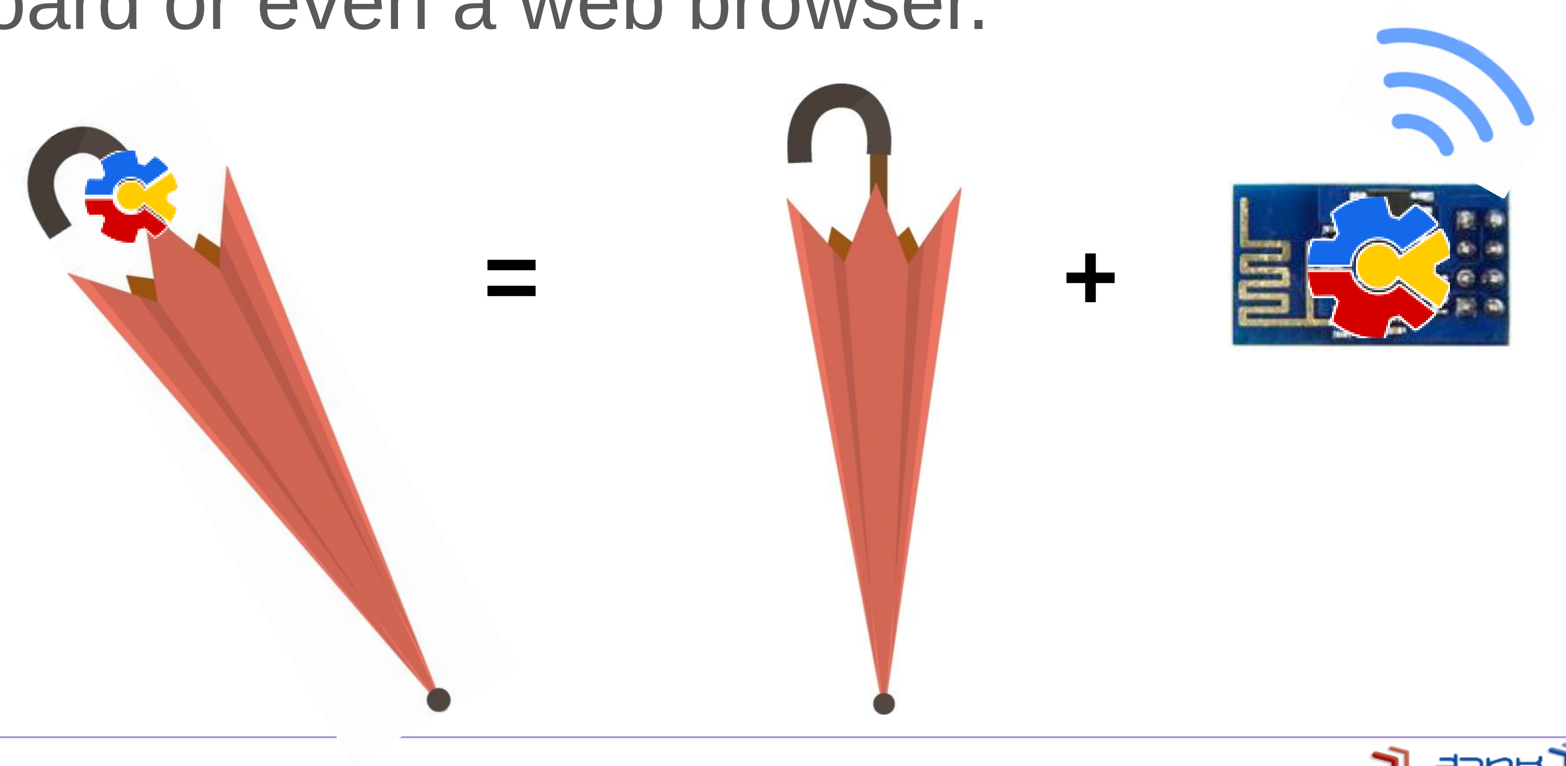
Microgear

Microgear transforms hardwares into NETPIE connected devices



Microgear

A Hardware that microgear resides can be your computer, your smartphone, microcontroller board or even a web browser.

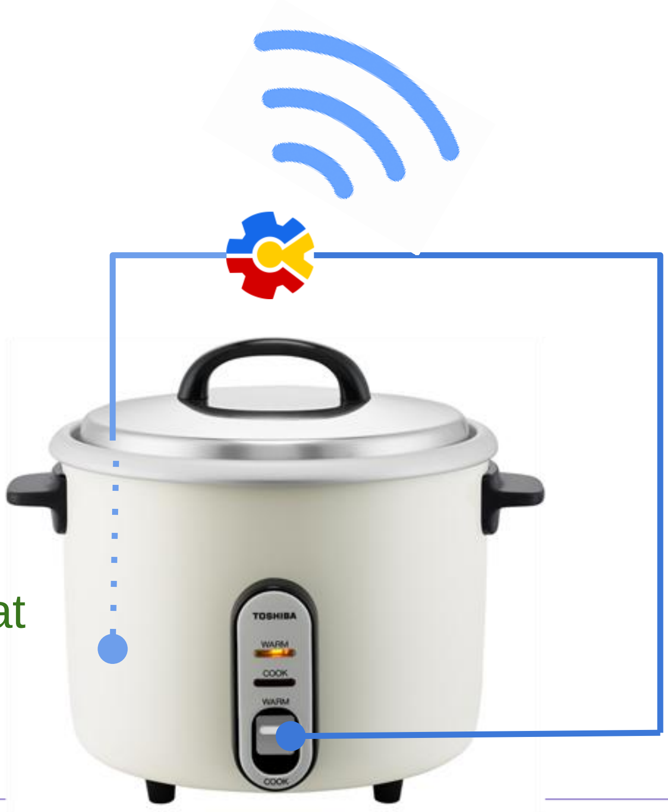


Microgear

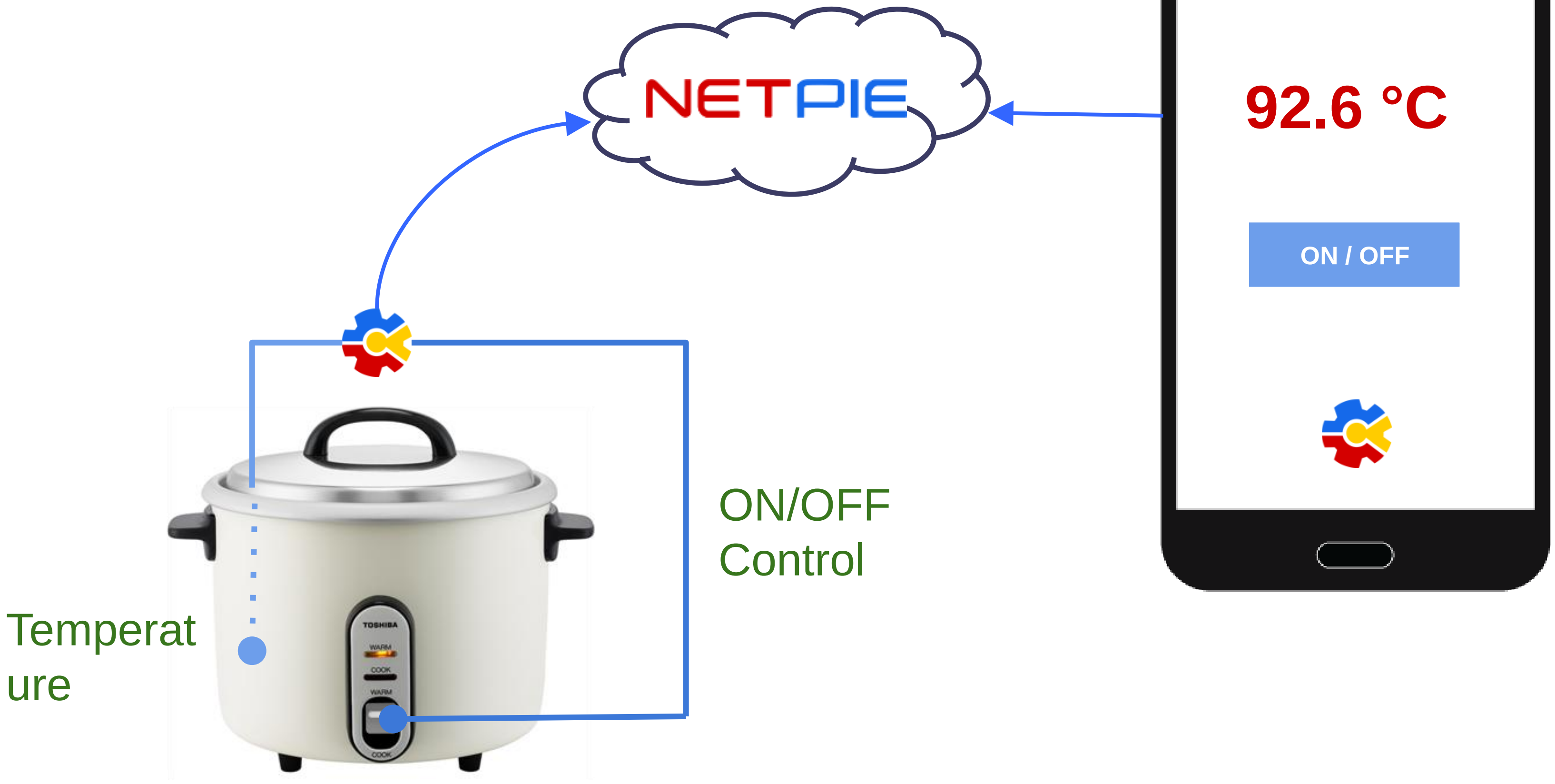
Using a cheap microcontroller board and a microgear library, everything can become a connected device.

Microgear

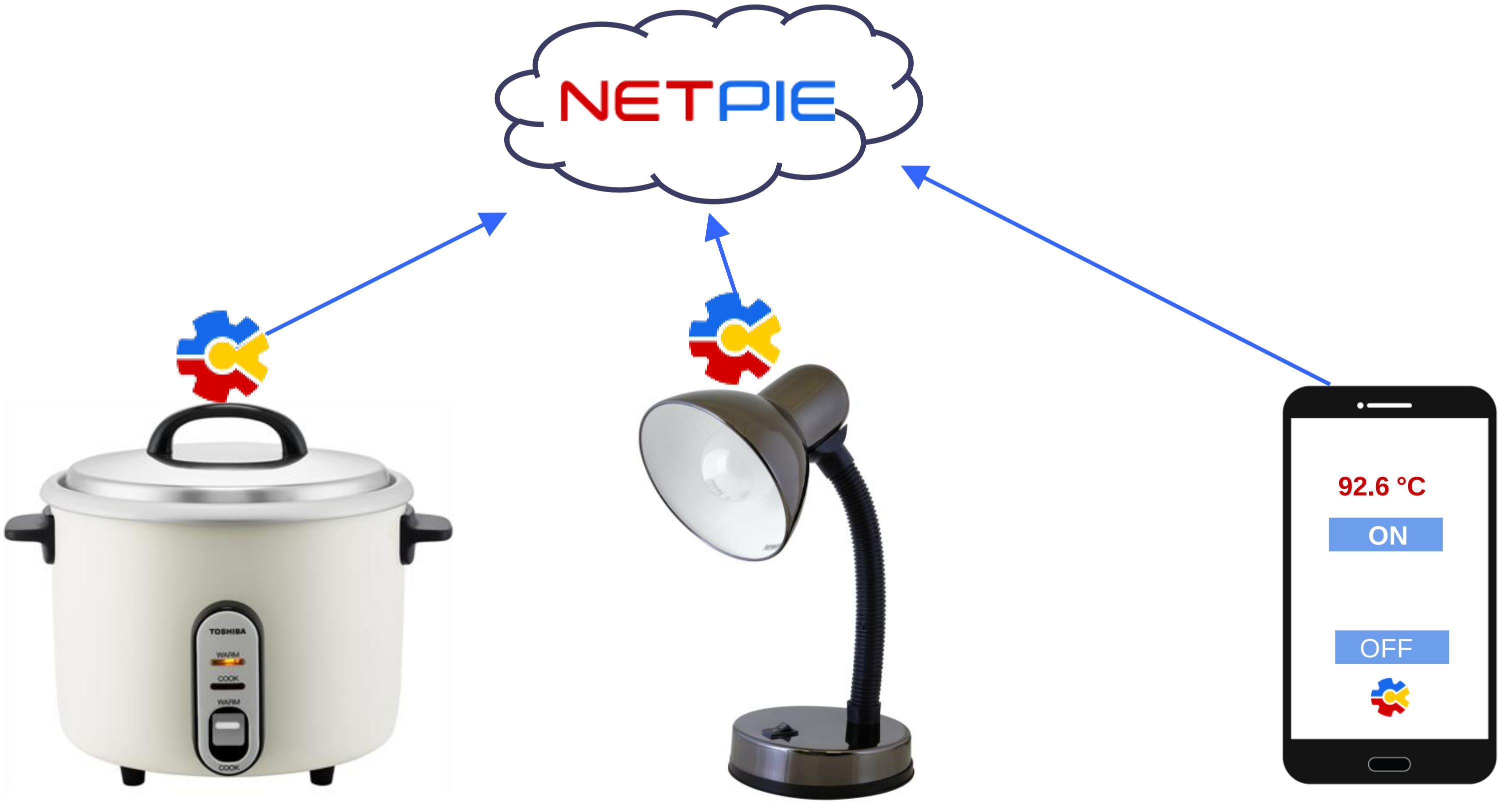
Once connected, NETPIE turns local primitive global.



Microgear

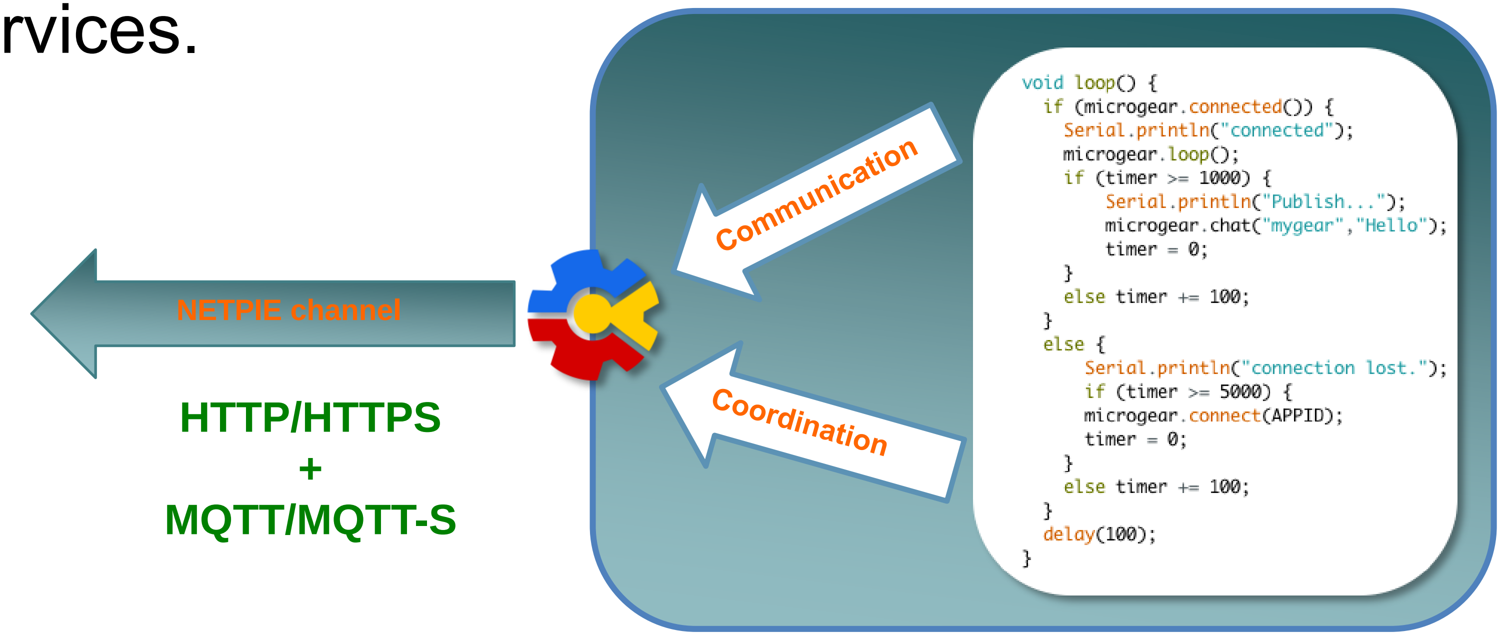


Microgear



Microgear

NETPIE's client library is called a microgear library. It can be thought of as a bridge between the application codes and the cloud, providing authorization, authentication, communication and coordination services.



```
void loop() {
  if (microgear.connected()) {
    Serial.println("connected");
    microgear.loop();
    if (timer >= 1000) {
        Serial.println("Publish...");
        microgear.chat("mygear","Hello");
        timer = 0;
    }
    else timer += 100;
  }
  else {
      Serial.println("connection lost.");
      if (timer >= 5000) {
      microgear.connect(APPID);
      timer = 0;
    }
    else timer += 100;
  }
  delay(100);
}
```

HTTP/HTTPS
+
MQTT/MQTT-S

IoT Communication Protocols

MQTT

AMQP

HTTP

CoAP

NETPIE Communication Protocols

Microgears

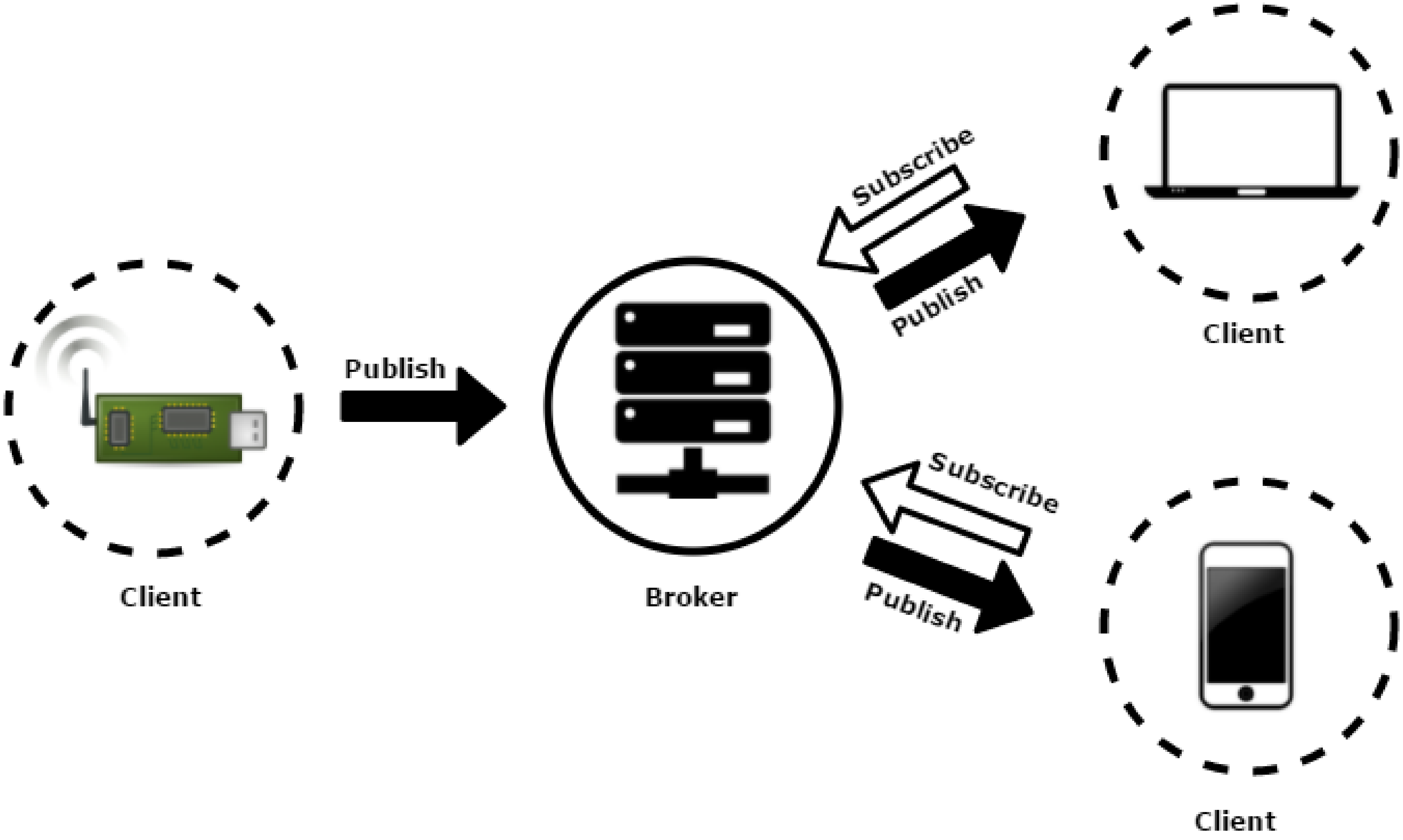
- MQTT port 1883
- MQTT-S port 8883
- MQTT over websocket port 8083
- MQTT over secure websocket port 8084

REST API

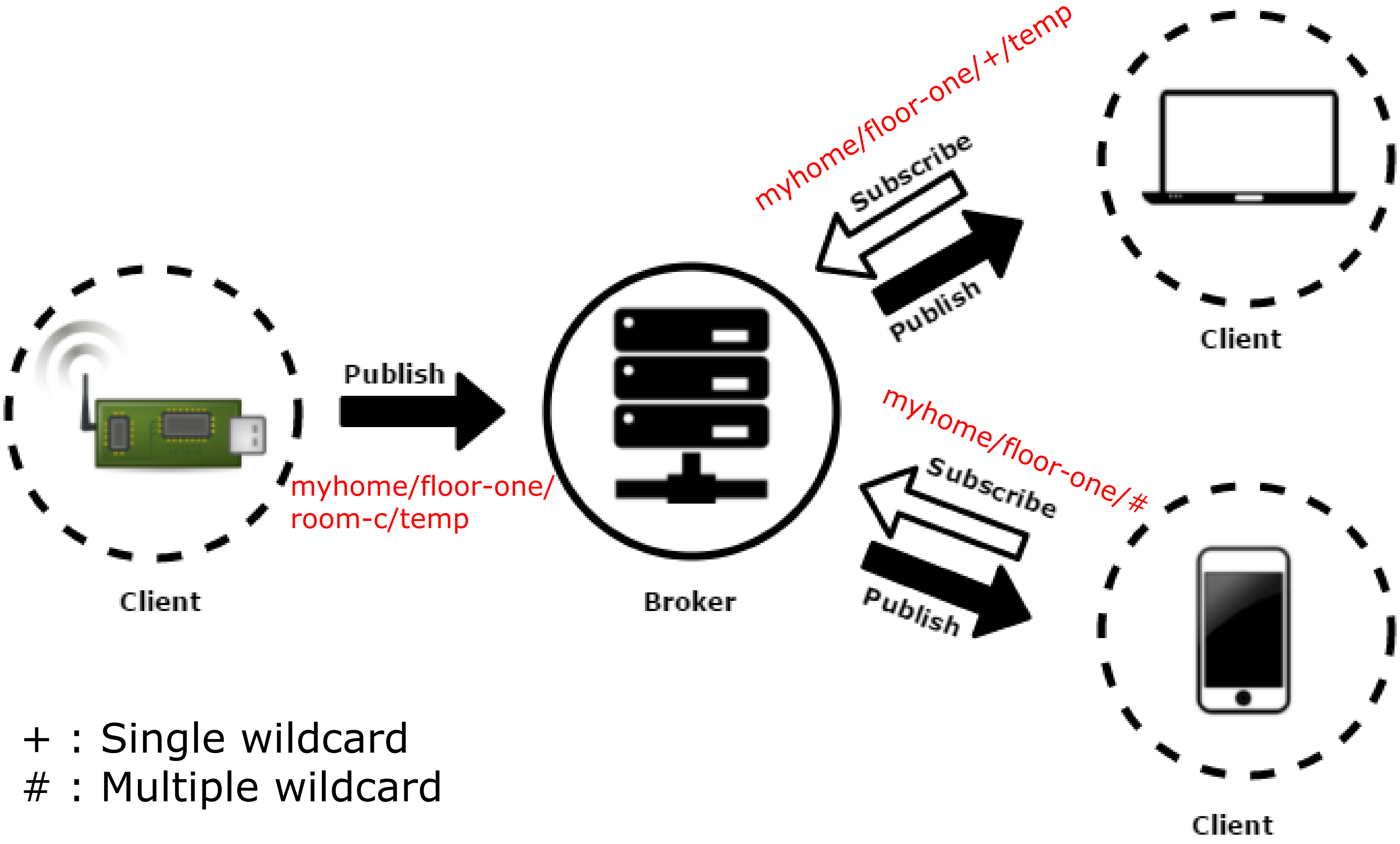
- HTTP port 80
- HTTPS port 443

MQTT

MQTT

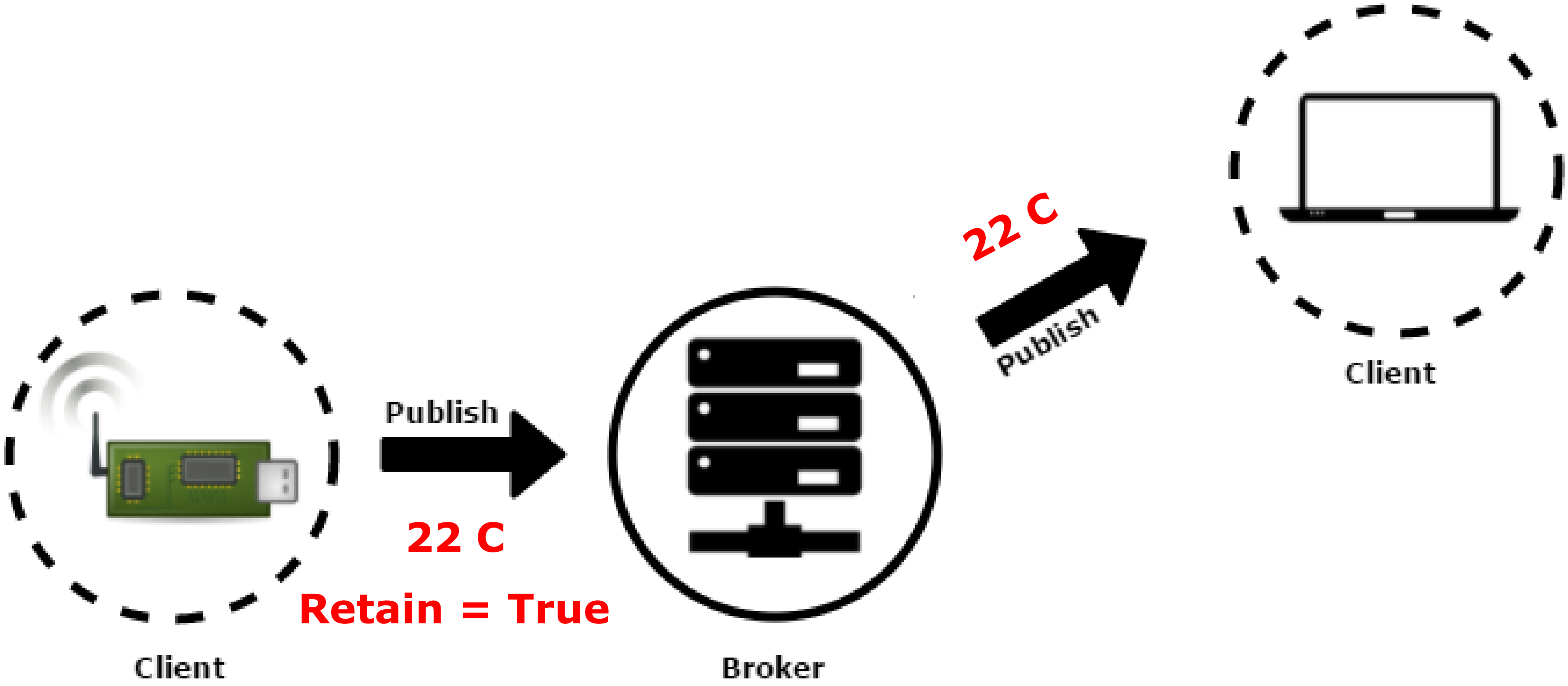

Client
Publish
Broker
Subscribe
Publish
Client
Subscribe
Publish
Client

MQTT Routing – Topic based



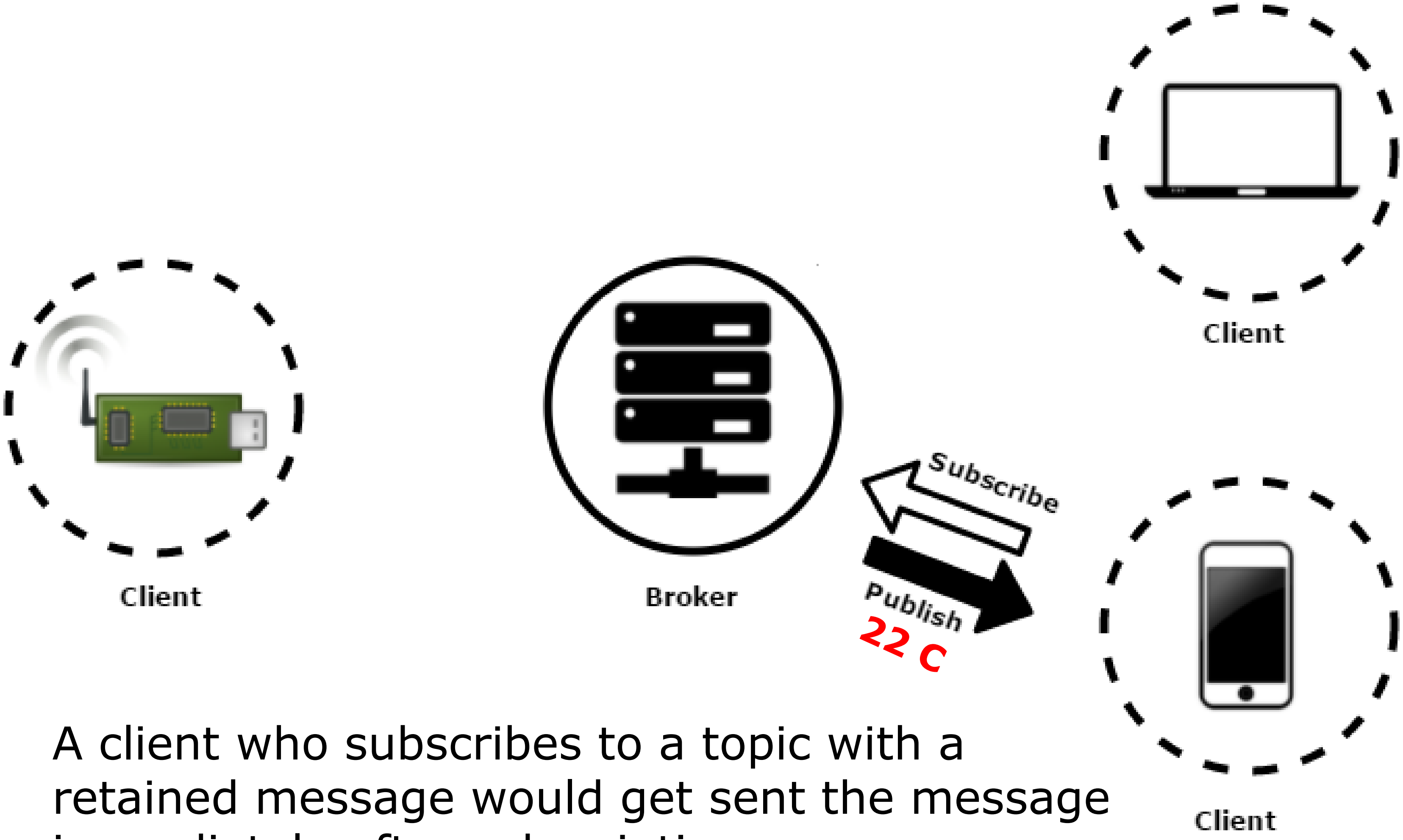
+ : Single wildcard
# : Multiple wildcard

MQTT – Retained Messages



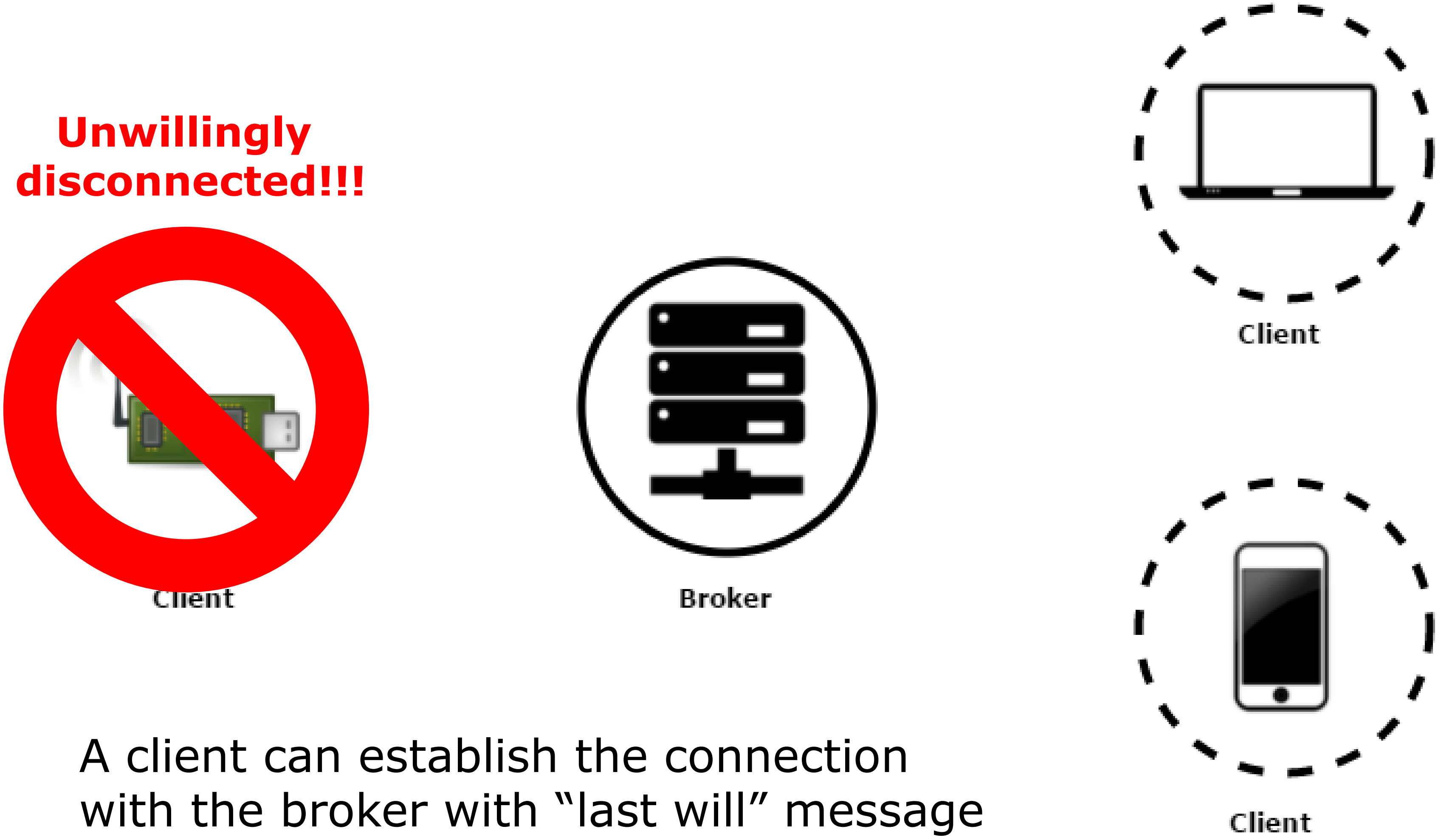
A MQTT client can publish a retained message by setting Retain Flag to True in the PUBLISH packet.

MQTT – Retained Messages



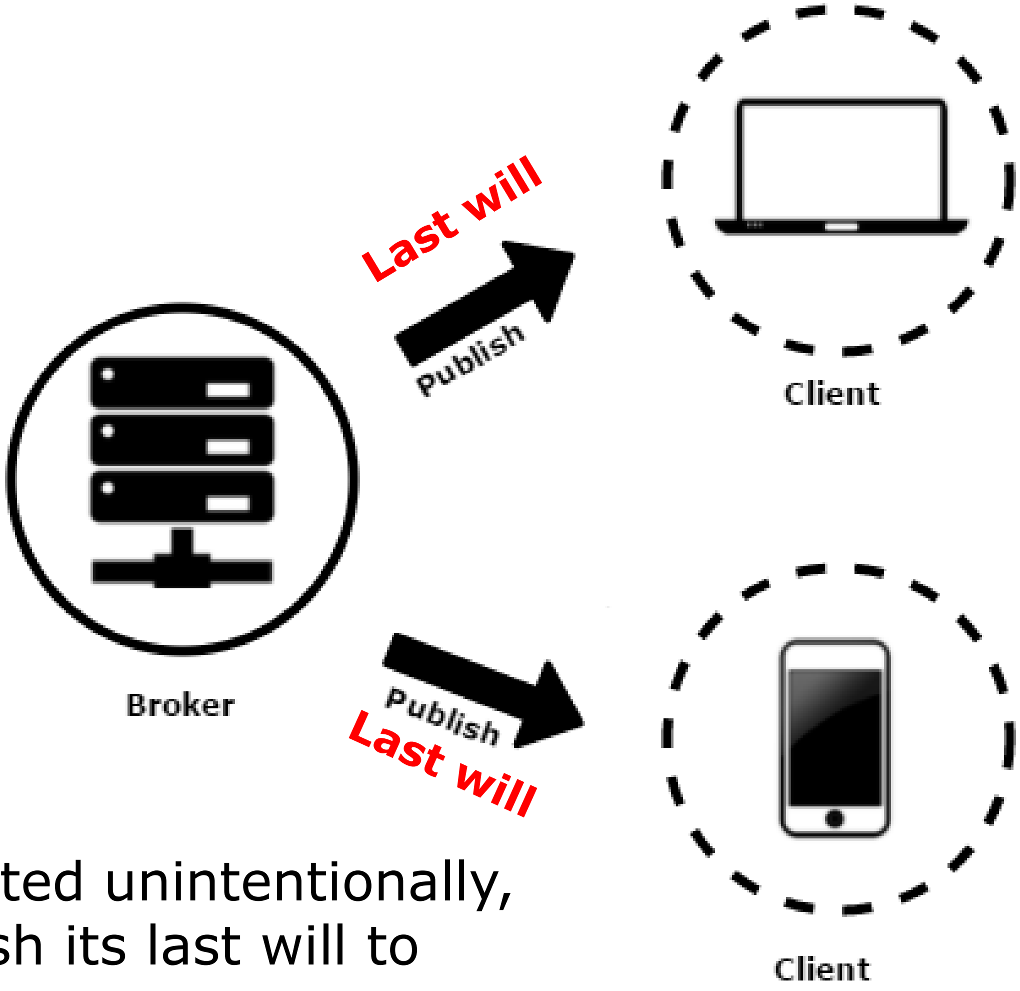
A client who subscribes to a topic with a retained message would get sent the message immediately after subscription.

MQTT – Last Will



A client can establish the connection with the broker with "last will" message attached to its CONNECT packet.

MQTT – Last Will



When it is disconnected unintentionally, the broker will publish its last will to the other clients.

MQTT - QoS

- QoS 0 – At most once
  - Least reliable
  - Fastest
  - No acknowledgment
- QoS 1 – At least once
  - One loop of acknowledgment
- QoS 2 – Exactly once
  - Most reliable
  - Slowest with the largest headers
  - Two loops of acknowlegment

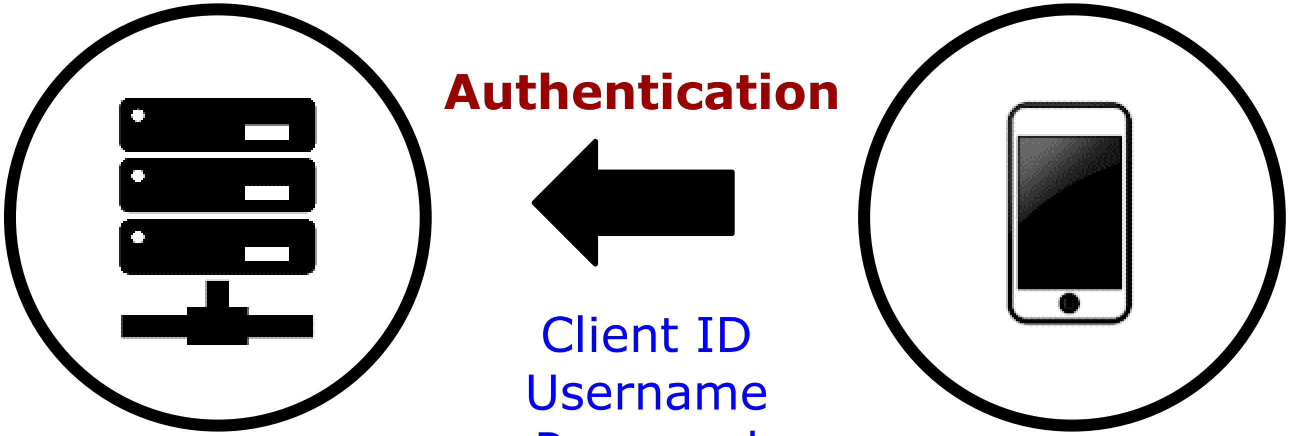
Ownership & Authorization

- Scope of communication
  - AppID
- Thing Authentication
  - Key (previously called AppKey)
    - Device Key
    - Session Key
  - Secret (previously called AppSecret)
- NETPIE Access Control
  - Key + Key Secret
  - Token + Token Secret

Key & Secret ควร

เก็บเป็นความลับ!

MQTT – Authentication & Authorization



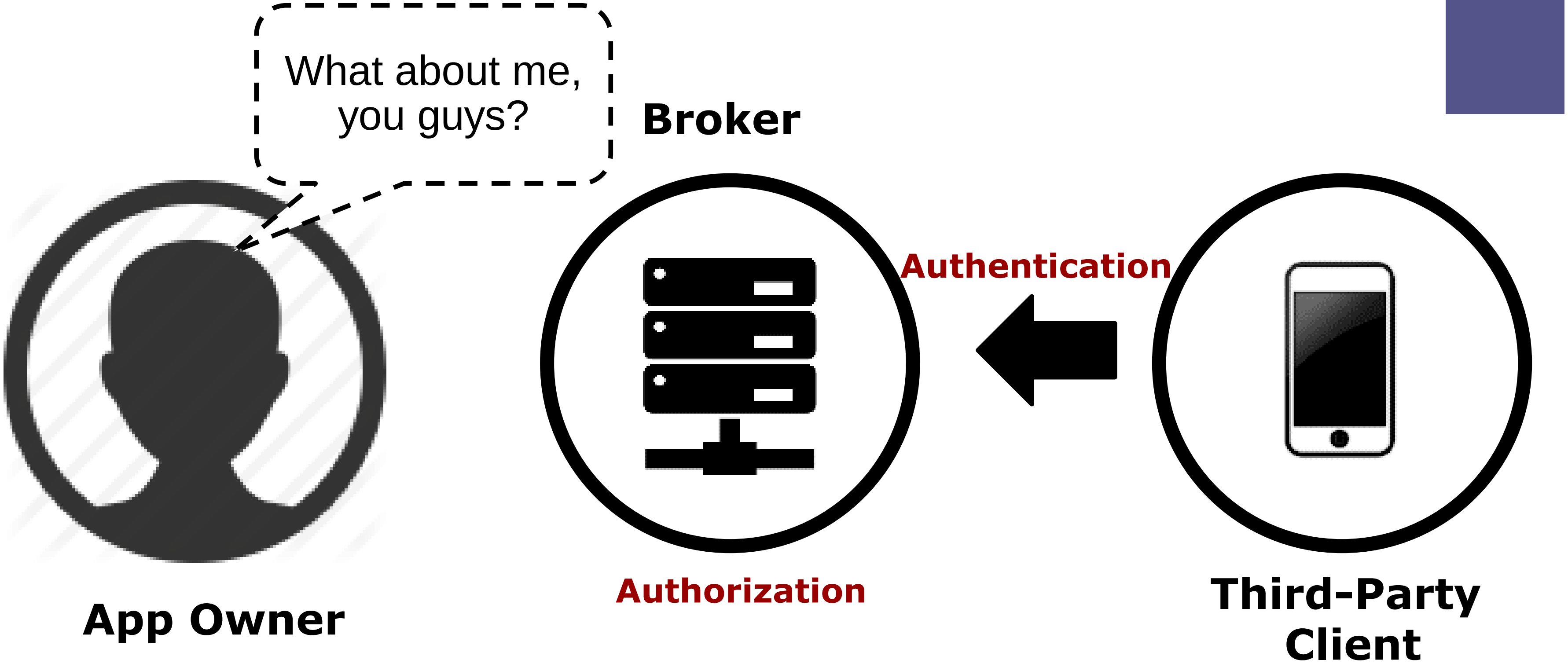
Authentication

Client ID
Username
Password

Authorization

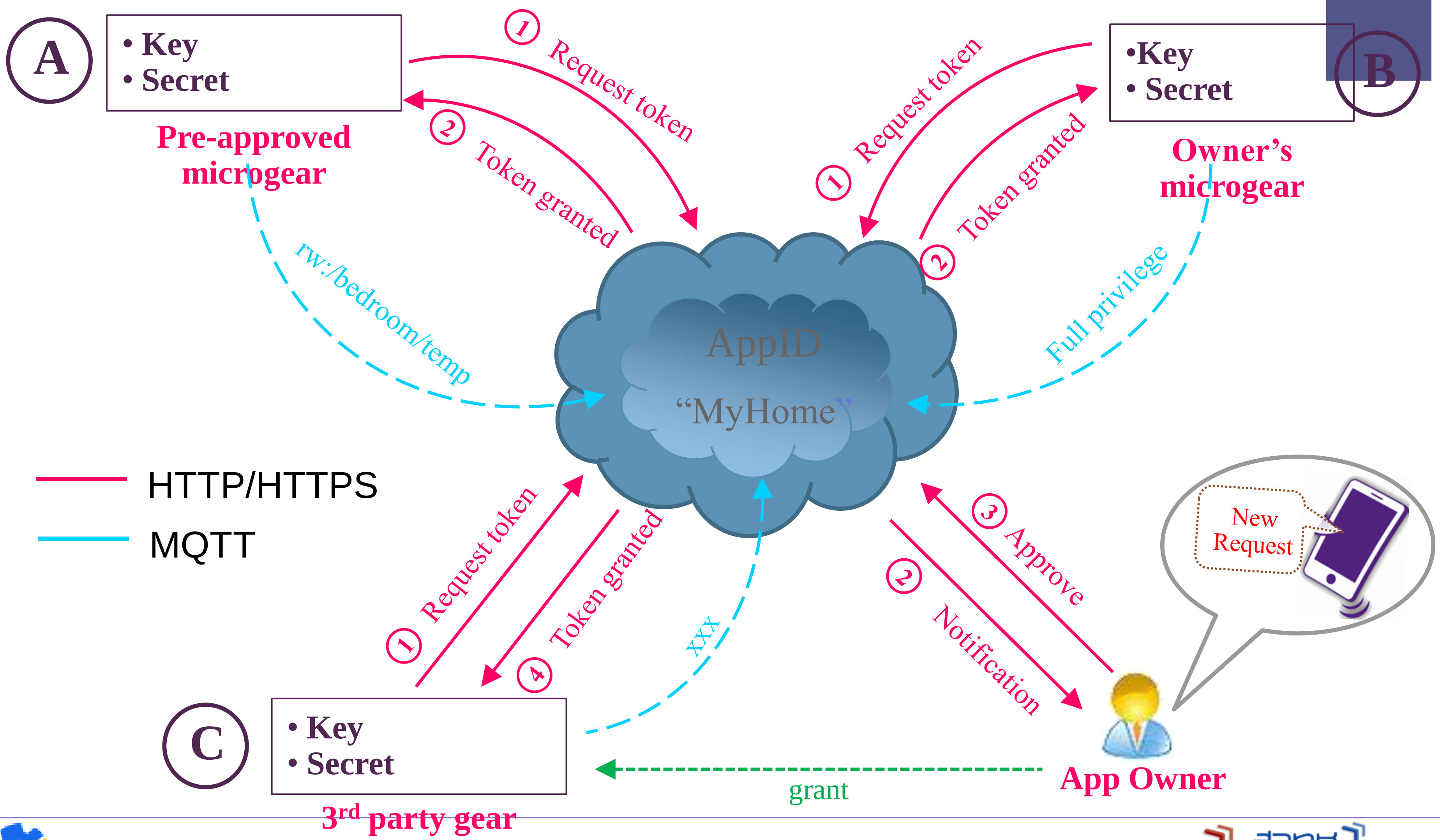
Broker-side Implementation thru plugin
Allowed Topic
Allowed Operation
Allowed QoS

MQTT Security for IoT Ecosystem



In order to use MQTT as IoT communication protocol, we need more dynamic authorization method that gets app owner involved.

Ownership & Authorization



A
- Key
- Secret

Pre-approved microgear

B
- Key
- Secret

Owner's microgear

C
- Key
- Secret

3rd party gear

AppID "MyHome"

1 Request token

2 Token granted

rw:/bedroom/temp

Full privilege

4 Token granted

xxx

2 Notification

3 Approve

New Request

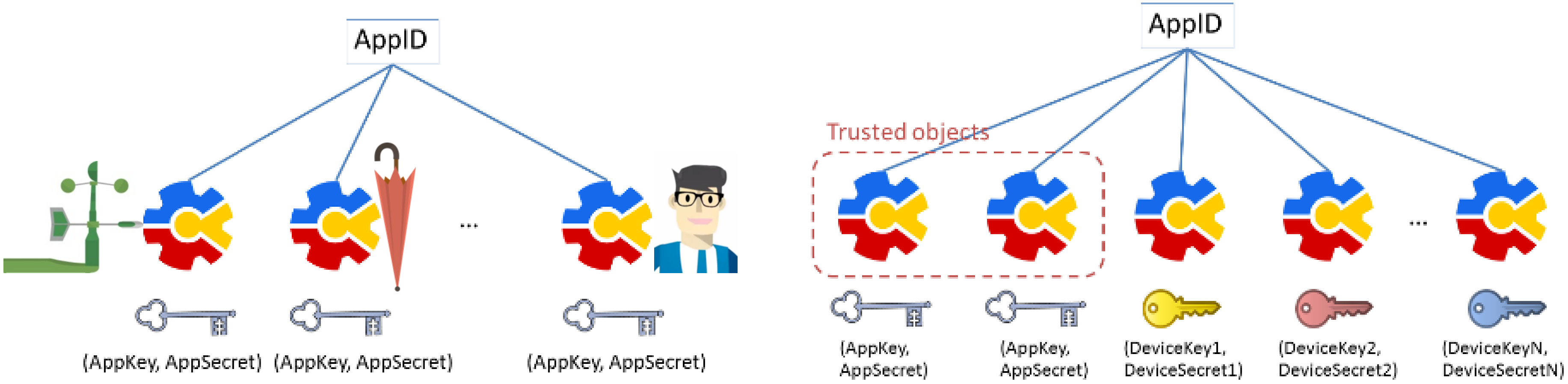
App Owner

grant

HTTP/HTTPS

MQTT

Dynamic authorization



Trusted System:
Same App Key and App Secret for all devices

AppID

...

(AppKey, AppSecret) (AppKey, AppSecret) (AppKey, AppSecret)

Hybrid System:
Use both App Keys and Device Keys

AppID

Trusted objects

...

(AppKey, AppSecret) (AppKey, AppSecret) (DeviceKey1, DeviceSecret1) (DeviceKey2, DeviceSecret2) (DeviceKeyN, DeviceSecretN)

NETPIE Authentication

Credentials	MQTT Auth Purpose	NETPIE
Client ID	Maintaining a user’s commun. Session	Token
Username	User ID	Key&Time_Stamp

Password	User Password	HMAC-SHA1(Token&Username);
		Hkey=Token_Secret&Key_Secret

Key Types

- Device Key
  - Device key generates permanent token(s).
  - Permanent token is valid for life unless it is revoked.
  - Recommended for physical devices where identities are persistent.
- Session Key
  - Session key generates one-time token(s).
  - Recommended for logical devices e.g. web browser
    Once disconnected from NETPIE, one-time token will be revoked

Microgear Functions

- connect(*appid*)
- setAlias(*alias*)
- chat(*alias,message*)
- publish(*topic,message*)
- subscribe(*topic*)
- unsubscribe(*topic*)
- useTLS(*use*)

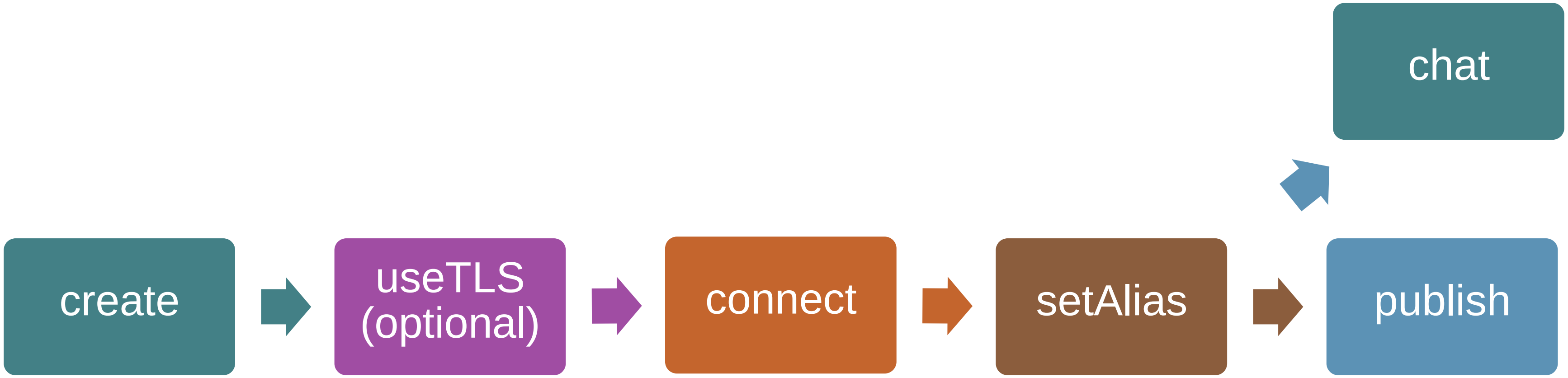
setName(name) is deprecated.
Do not use!

Default is TRUE for HTML5, Node.js
Default is FALSE for ESP866

Microgear Event

- CONNECTED
- DISCONNECTED
- ERROR
- MESSAGE
- ABSENT
- PRESENT

Typical Microgear Initialization Steps



Available Microgears

Production

- ESP8266
- Arduino-Ethernet

Under Testing

- Java
- Android

- Node.js
- HTML5
- Python
- Freeboard

Supported platforms