Distributed Device Management for IoT

Speakers:
- BERLEMONT Samuel
- MICHE Arnaud

In action with Eclipse Leshan, Eclipse Wakaama and OMA-LWM2M

Orange Labs
IoT Research Domain

19/02/2019 Eclipse IoT Days, Grenoble

Outline

1. Device Management?
2. Standards and best practices
3. DM for IoT, a new paradigm
4. The Future of DM: a multi-server architecture
5. Next Steps

Device Management… ?

4 categories of operations

- Remote service activation and configuration

- Error and log collection for analysis

- Configuration visualization
- Diagnostics
- Remote actions (reboot…)

- Firmware update

Standards and Best Practices

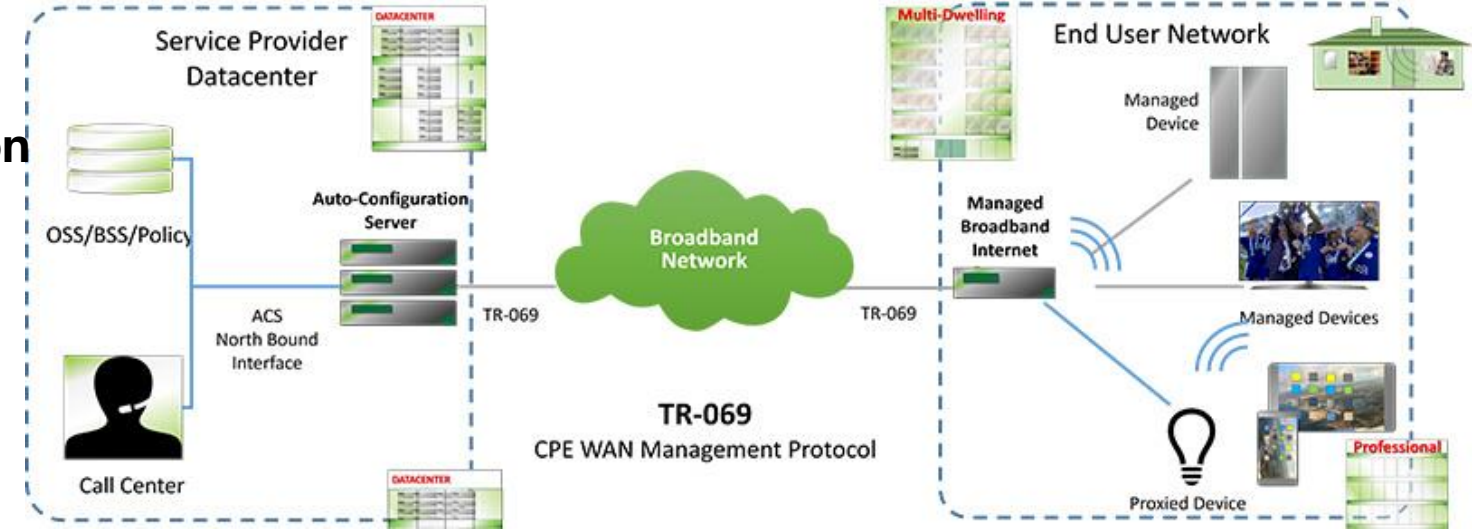
CPE WAN Management Protocol v1.4, Issue: 1 amendement 6

Centralized
Auto-Configuration Server

Data models

RPC Methods
device / server

(HTTP/SOAP)



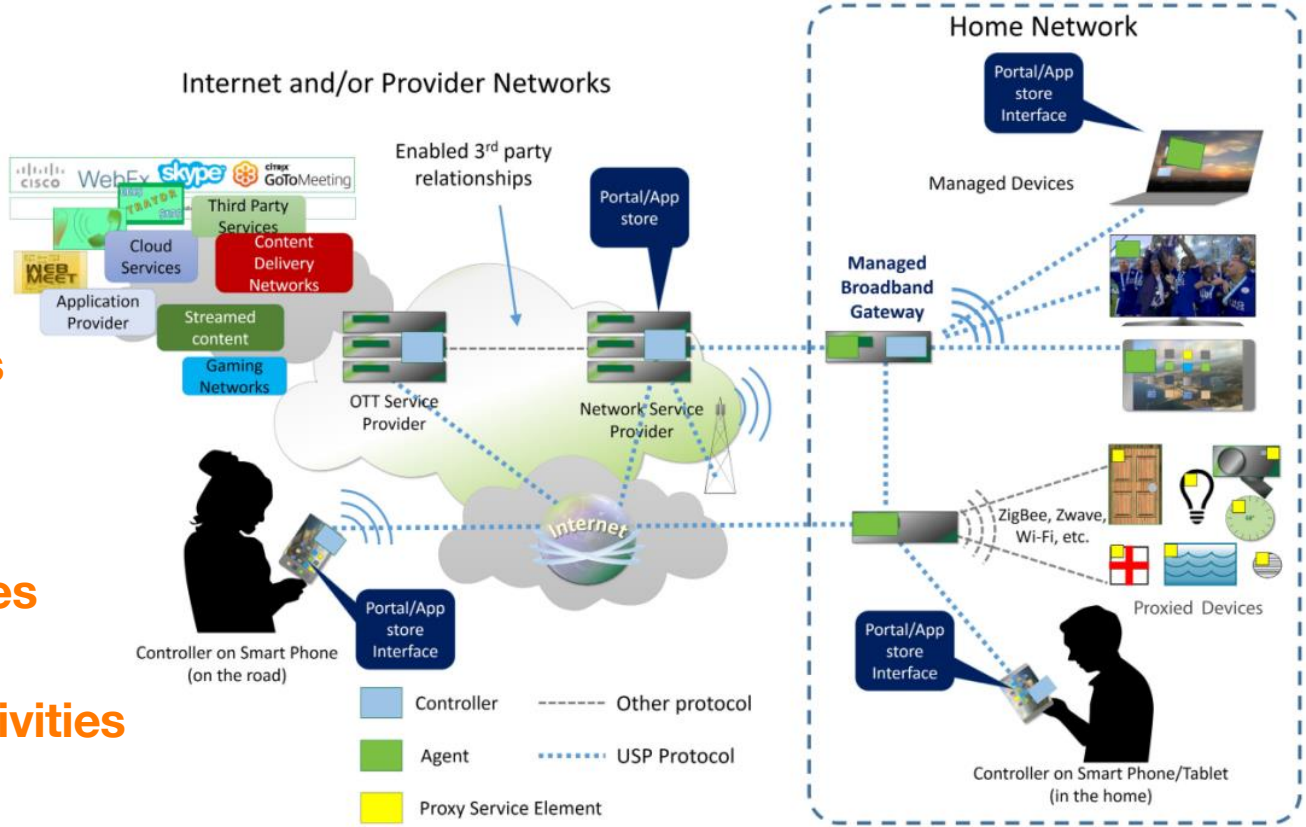
User Services Platform V1.0

Multi-controller
architecture

Data models w/ methods
RESTful
Full discovery

Service elements / Proxies

Start of Open Source activities



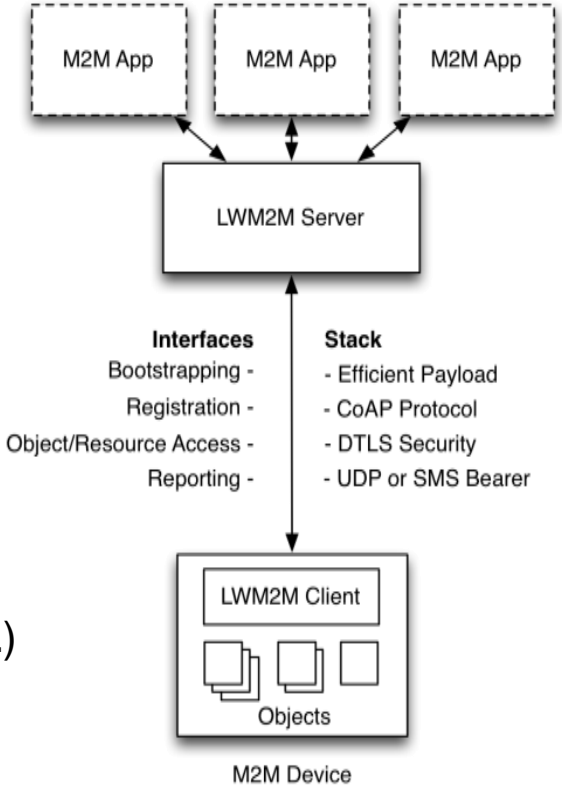
OMA Lightweight M2M

Multi-server
architecture

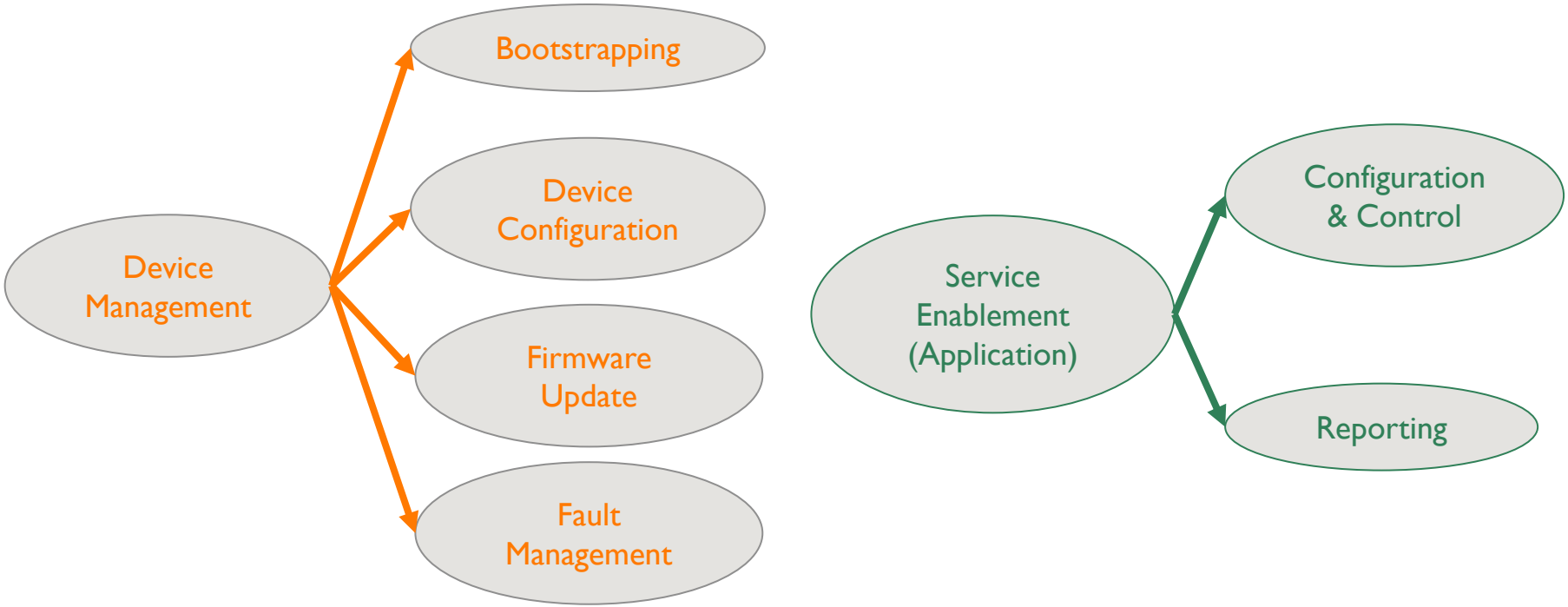
Data models w/ methods
RESTful
Shared models

Service objects (OMNA, IPSO) / Proxies

Dynamic ecosystem
- **Open Source** implementations (Wakaama, Leshan, …)



OMA Lightweight M2M :
Device Management & Service Enablement Standard for IoT



DM for IoT, a new paradigm

IoT DM challenges

- **Heterogeneity**
  - ➔ life-cycle profiles
  - ➔ DM features
- **New architectures**
  - ➔ DM/service convergence
  - ➔ Multi-protocol management
  - ➔ Softwarization : NFV, SDN
- **Security**
- **Scalability**

The impact on DM solutions

Device profiles

Requirements

Corresponding DM/SM protocol

Multiple relevant Standards & Proprietary protocols

The impact on DM solutions

Multi-protocol DM solution

Integration of additional DM servers

Integration of non-DM-enabled devices

Distributed Device Management for IoT

In action with Eclipse Leshan, Eclipse Wakaama and OMA-LWM2M

Imagined by the research team working for
Orange Labs IoT Research Domain

« Under the hood »

Who am I ?

Arnaud MICHÉ

Software developer at OBS SA (subsidiary of Orange SA)

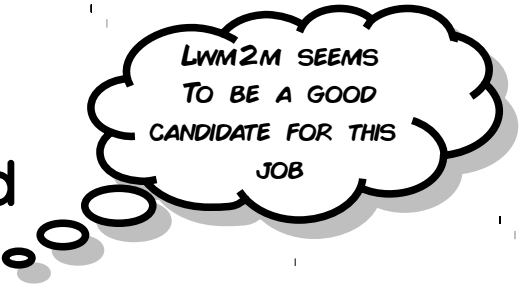
My job on this project :
Implement prototypes in order to evaluate the ideas and theories envisionned by the research team.

Contents

- Why Leshan ?
- How we used it
- Proxies
  - How a proxy work
  - Changes in leshan-client-demo
- Servers
    - How a server works
    - Changes in leshan-server-cluster
- Wakaama on ESP32
- How it is integrated in the demonstration
- Near future work

Why LESHAN ?

1. Major goals → Decentralized, Multi server



2. Need of an implementation for our experiments

3. Leshan is mature, in active development and provides good examples ...

Contents

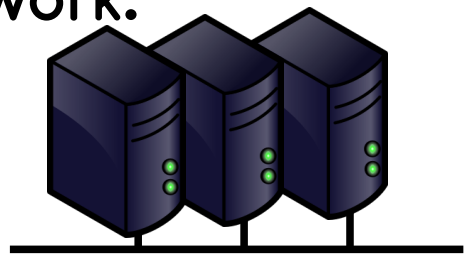
- Why Leshan ?
- How we used it
- Proxies
- Servers
- Wakaama on ESP32
- How it is integrated in the demonstration
- Near future work

How we use LESHAN

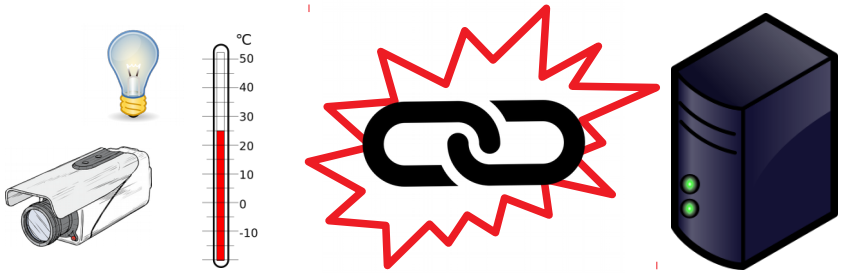
Disclaimer:

The code base used for our prototype has been cloned in september 2017 and did not follow the changes of upstream developments since this date.

1. As a server of Device Management (DM) ready to be clustered in our network.

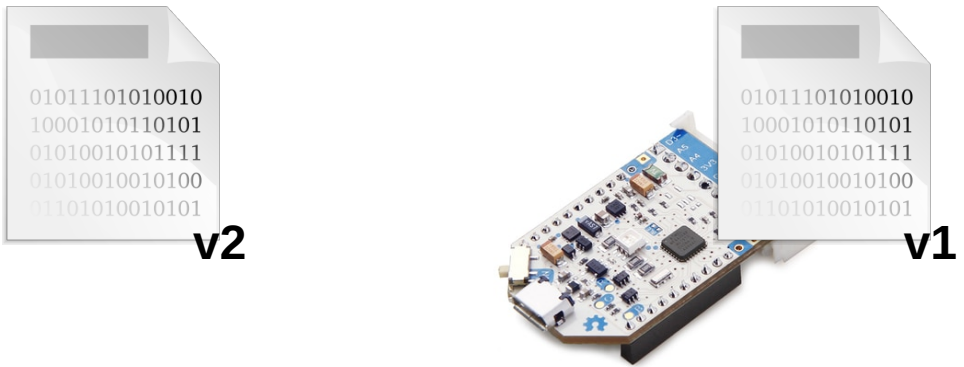


2. As a proxy for connecting constrained devices to our DM network.



How we use LESHAN

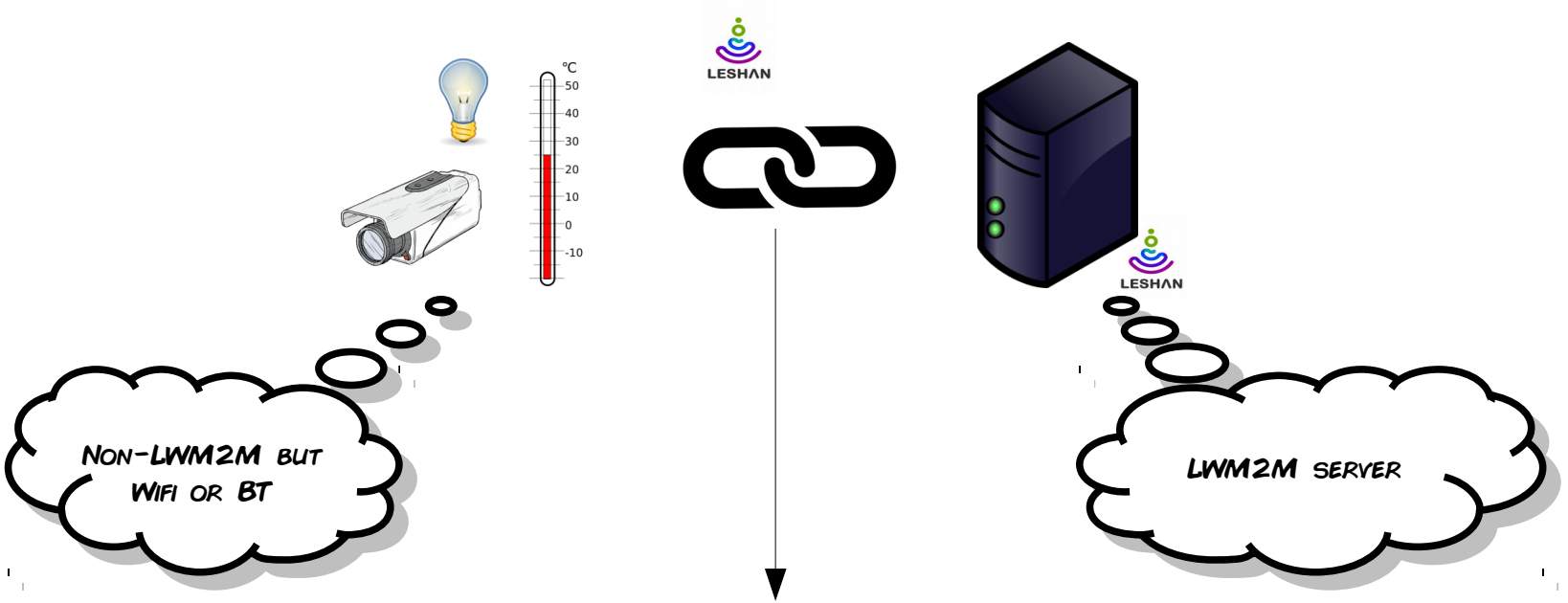
3. For now, only Firmware Upgrade is implemented in the Proof-of-Concept.



Contents

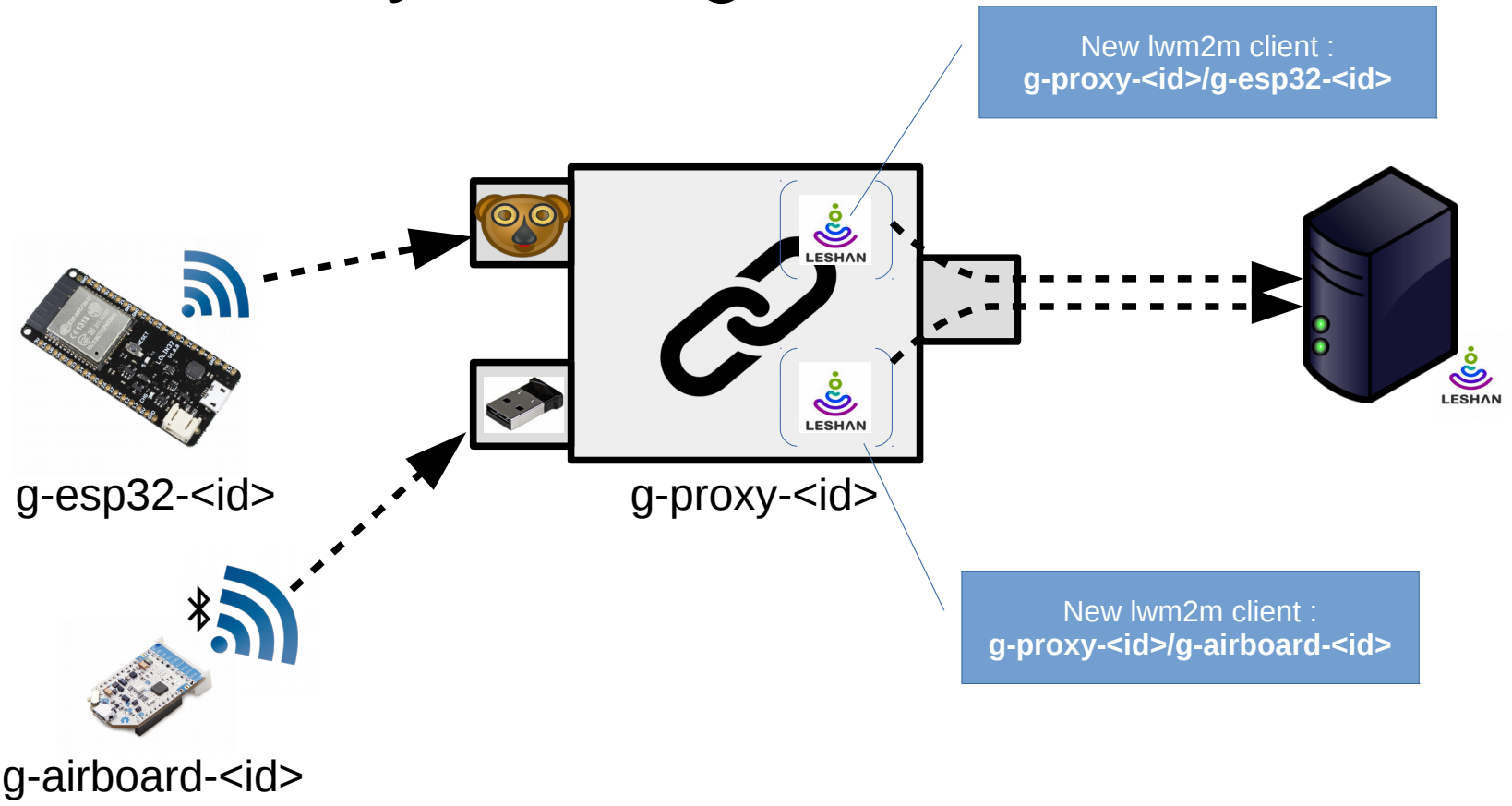
- Why Leshan ?
- How we used it
- Proxies
- Servers
- Wakaama on ESP32
- How it is integrated in the demonstration
- Near future work

Proxies

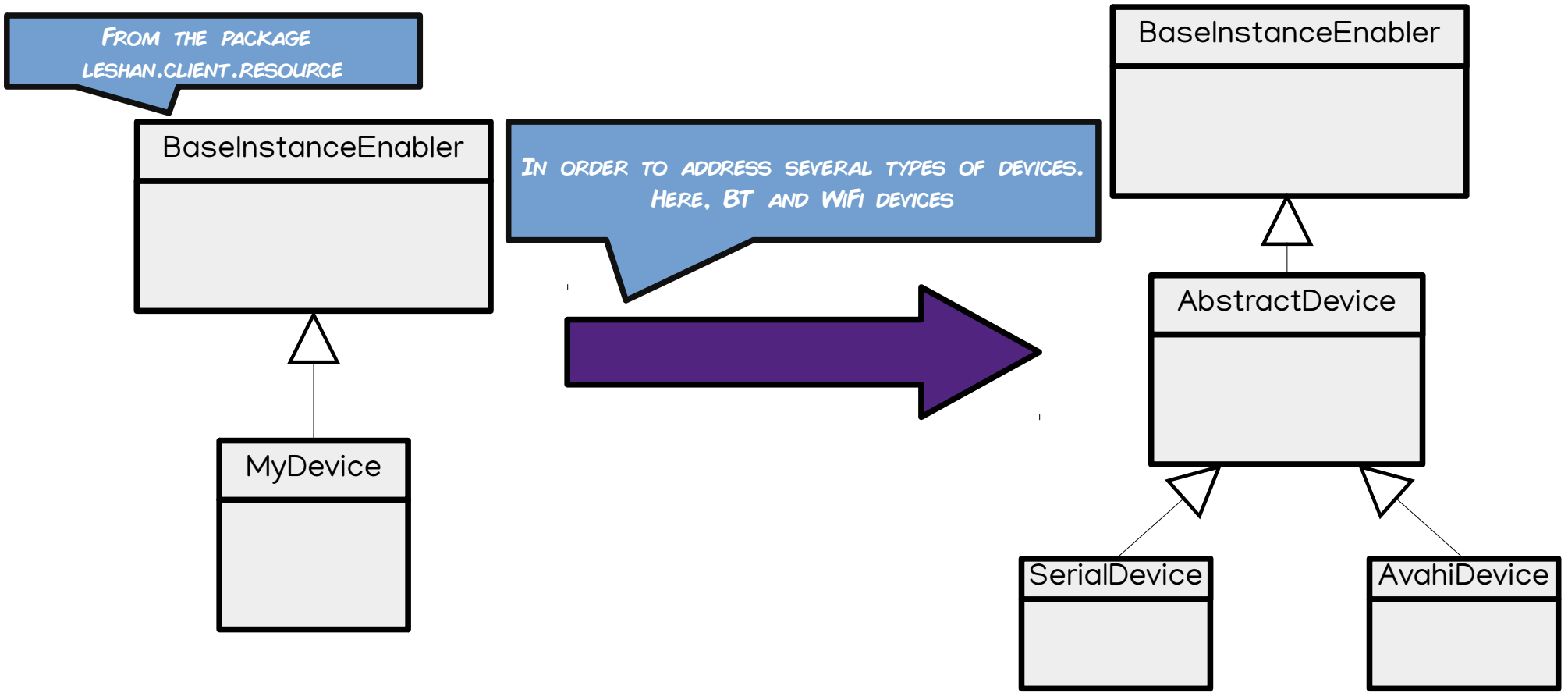


- Based on **leshan-client-demo** package provided with Leshan sources
- Addition of an **Avahi** service for device over TCP (here, via WiFi)
- Addition of a service polling serial connections for Bluetooth device connected via an USB dongle
- Process wrapper for launching tools for flashing device

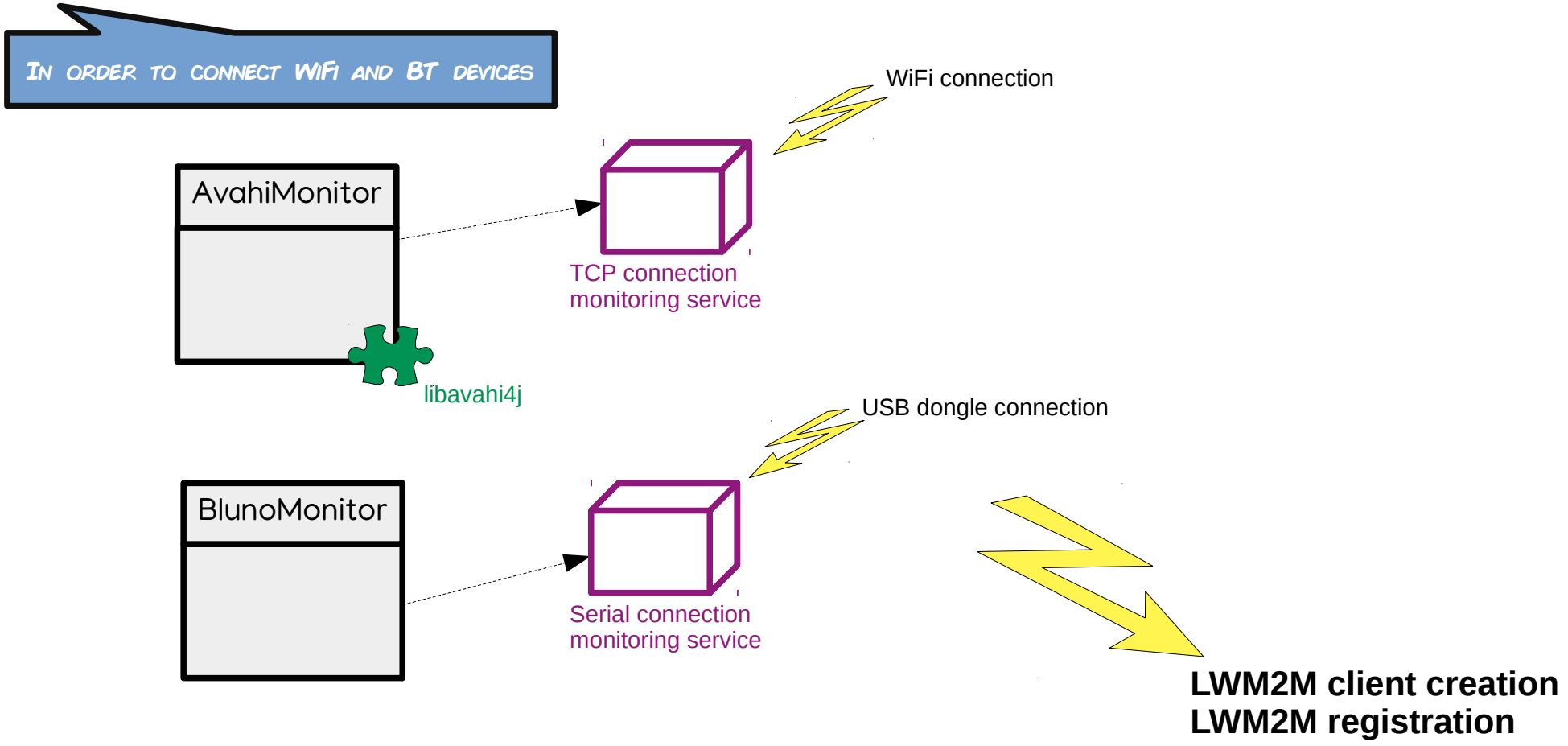
How a proxy manager works



Changes on leshan-client-demo

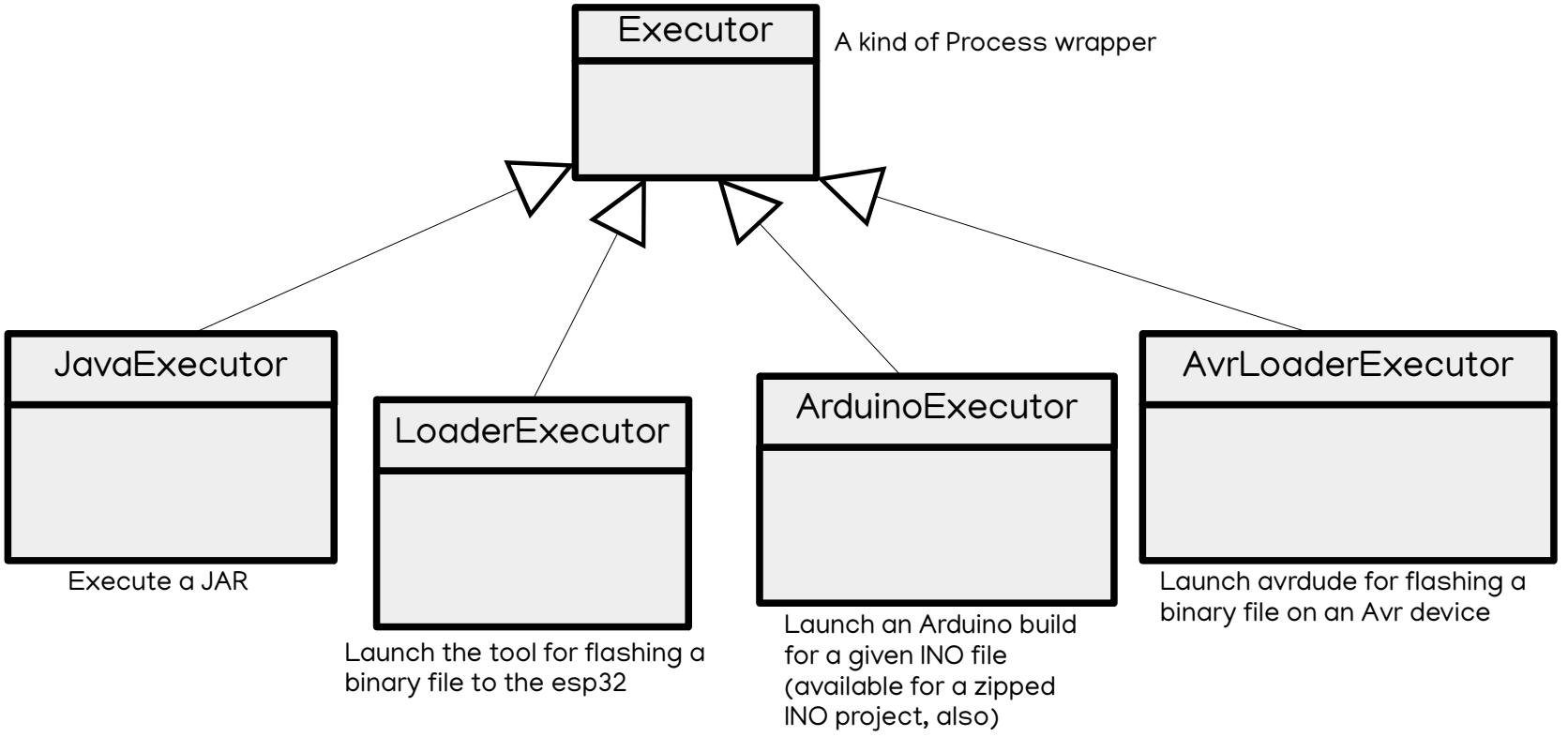


Changes on leshan-client-demo



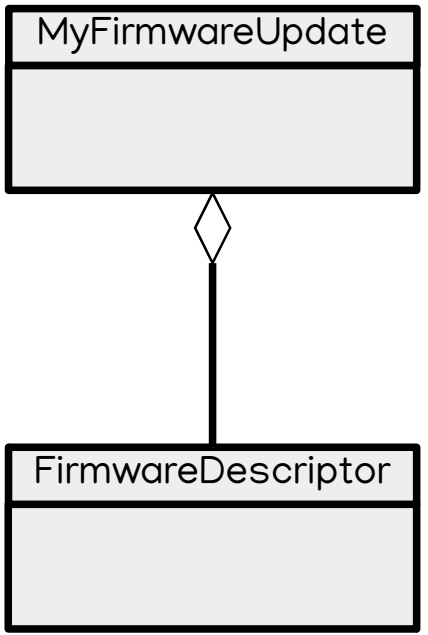
Changes on leshan-client-demo

In order to perform the firmware update (1/2)



Changes on leshan-client-demo

In order to perform the firmware update (2/2)



- The LWM2M object « Firmware »
- It is also in charge of downloading the firmware and launch the execution of the flash tool (by intermediate of the Executors)

- It holds informations necessary to the firmware update :
  - Version of software
  - url where to download the file
  - format of the file and the type of the board enabling the choice of the right executor for flashing the device.

Changes on leshan-client-demo

IN ORDER TO ALLOW MULTISERVER (1/2)

Declaration of two server URLs inside **LeshanClientDemo.java :**

1.

```
private static String serverURI_1 = "coap://192.168.0.100:5683";
private static String serverURI_2 = "coap://192.168.0.101:5683";
```

2.

```
Security sec1 = Security.noSec(serverURI_1, 123);
Security sec2 = Security.noSec(serverURI_2, 234);

Server serv1 = new Server(123, 30, BindingMode.U, false);
Server serv2 = new Server(234, 30, BindingMode.U, false);

initializer.setInstancesForObject(SECURITY, sec1, sec2);
initializer.setInstancesForObject(SERVER, serv1, serv2);
```

Changes on leshan-client-demo

IN ORDER TO ALLOW MULTISERVER (2/2)

Some changes in LWM2M Registration related classes of Leshan core :

1. Inside **leshan-master_bluno/leshan-client-core/src/main/java/org/eclipse/leshan/client/servers/RegistrationEngine.java**

Changed data structure which holds the registration ID for one client **from a variable storing the reg_id to a hash map which holds several couples (server_uri, reg_id)** as a client can be registered to more than one server.

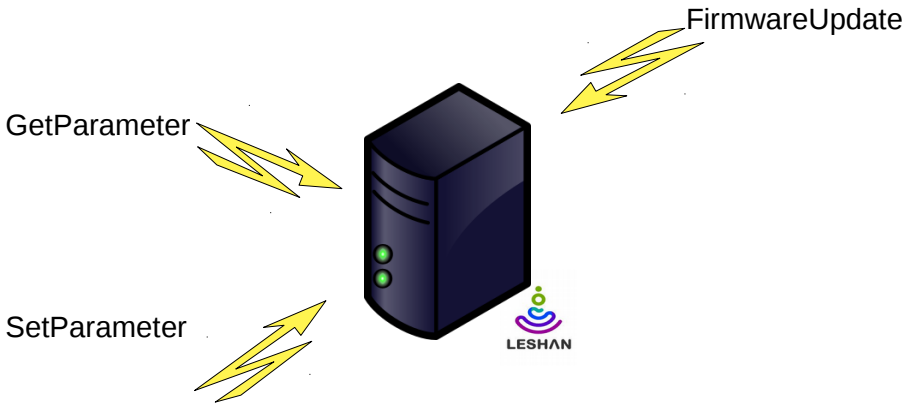
2. Inside **leshan-client-cf/src/main/java/org/eclipse/leshan/client/californium/LeshanClient.java**

Added a function getRegistrationId which **calls the getRegistrationId of the RegistrationEngine** with the server informations. Server informations passed in parameters enable to retrieve the registration Id of a device giving its server Id.

Contents

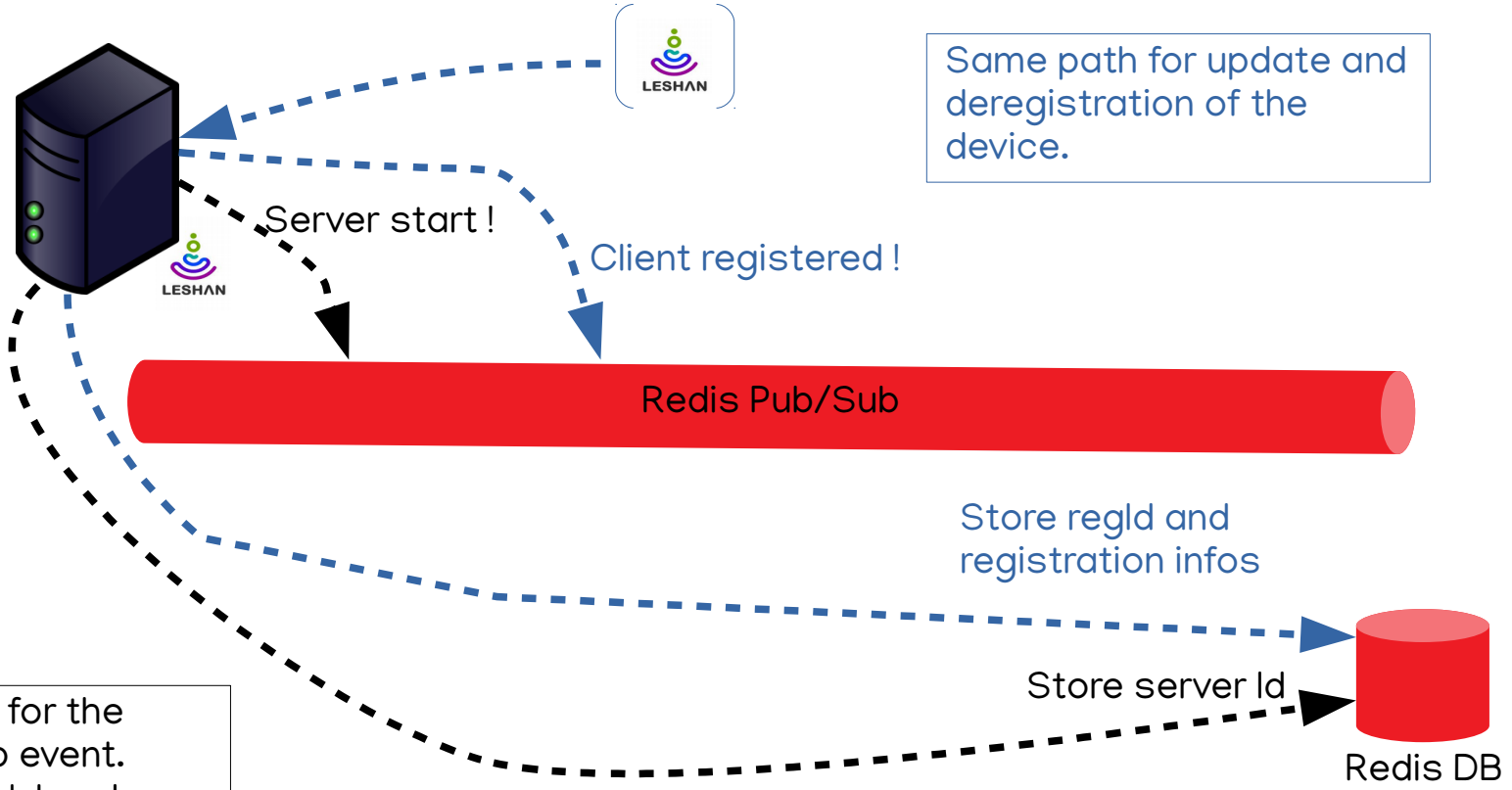
- Why Leshan ?
- How we used it
- Proxies
- Servers
- Wakaama on ESP32
- How it is integrated in the demonstration
- Near future work

Servers

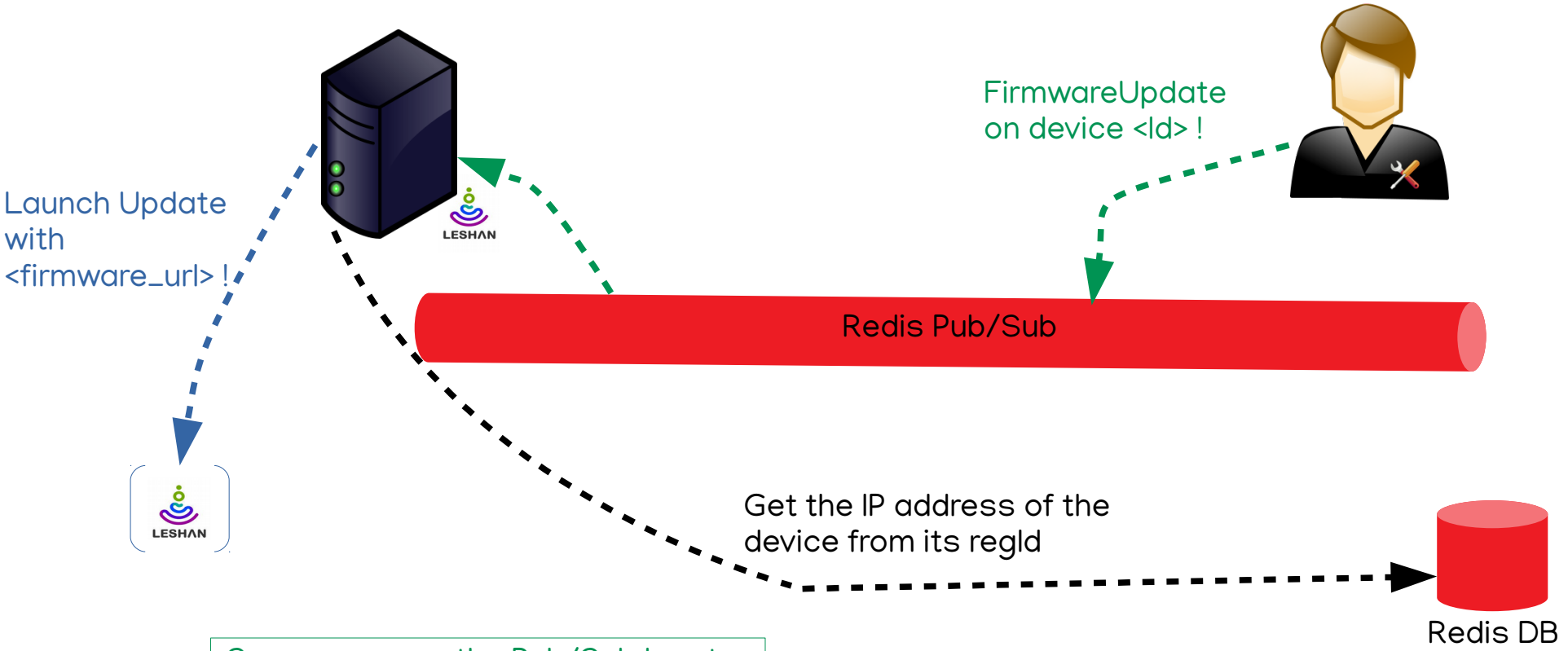


- Based on **leshan-server-cluster** package provided with Leshan sources
- Manage registered devices
- Relying on Redis PubSub
- Reg IDs stored in Redis Key/Value data base

How a server works



How a server works



Server can use the Pub/Sub bus to notify the progress of operations

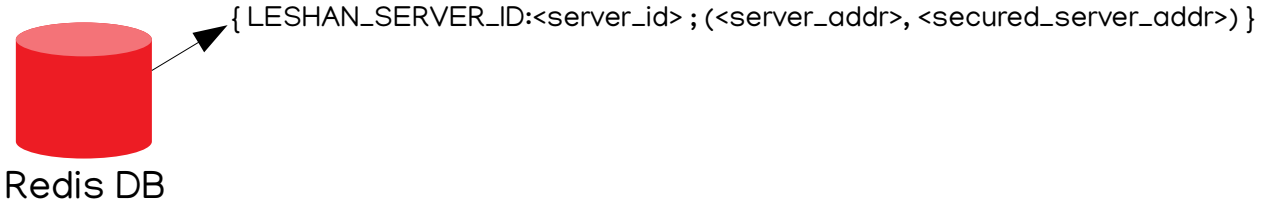
Changes on leshan-server-cluster

In order to allow the detection of arrivals and exits of servers in the cluster

Creation of the class **leshan-server-cluster/src/main/java/org/eclipse/leshan/server/cluster/RedisServerEventPublisher.java :**

- Publish start/stop event of a server
- Store Server ID inside database

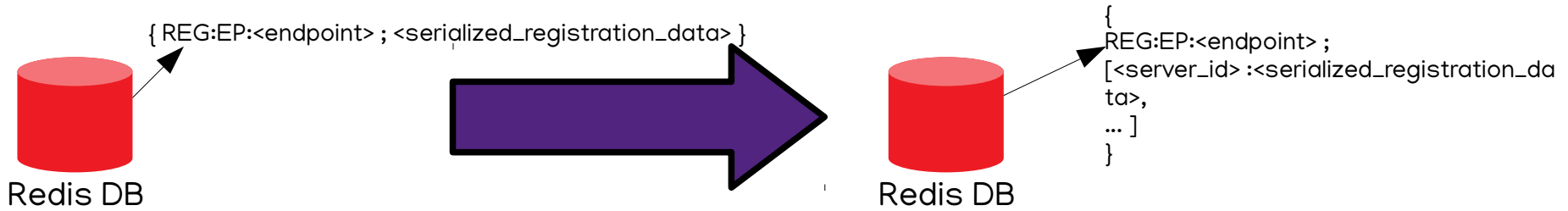
Creation of a new key/value in the store :



Changes on leshan-server-cluster

In order to allow multiserver

Modification of the data structure stored in Redis Database :



And token handlers hold now, the regID in addition of endpoint name :
Before token handlers were **EP#UID#endpoint** and now it is **EP#UID#regId#endpoint**

Finally following classes have been impacted :

- **leshan-server-cluster/src/main/java/org/eclipse/leshan/server/cluster/LeshanClusterServer.java**
- **leshan-server-cluster/src/main/java/org/eclipse/leshan/server/cluster/RedisRegistrationStore.java**
- **leshan-server-cluster/src/main/java/org/eclipse/leshan/server/cluster/RedisRequestResponseHandler.java**
- **leshan-server-cluster/src/main/java/org/eclipse/leshan/server/cluster/RedisTokenHandler.java**

Contents

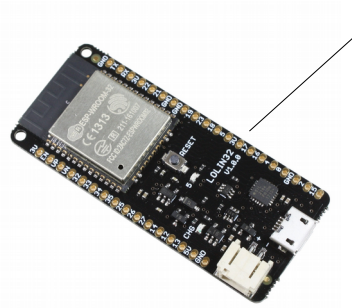
- Why Leshan ?
- How we used it
- Proxies
- Servers
- Wakaama on ESP32
- How it is integrated in the demonstration
- Near future work

Wakaama on ESP32

In order to allow firmware update on ESP32 without the help of the proxy

Based on wakaama client example distributed with Wakaama sources

Using the ESP32 SDK (with FreeRTOS)

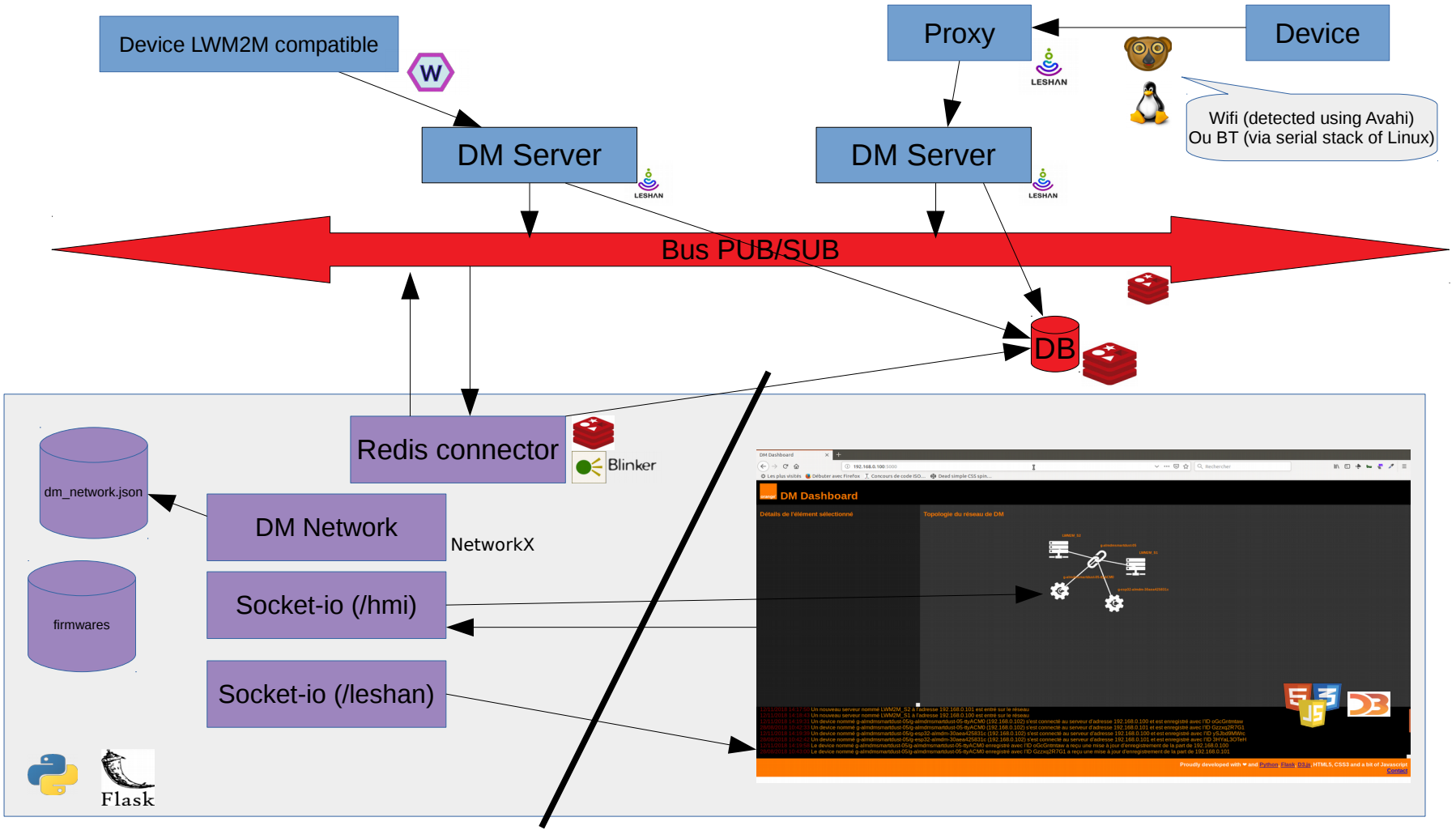


RGB LED controller	To light on/off an RGB LED
HTTP Downloader	To download the firmware on the board
OTA	To process the firmware update

Contents

- Why Leshan ?
- How we used it
- Proxies
- Servers
- Wakaama on ESP32
- How it is integrated in the demonstration
- Near future work

How it is integrated in the demonstration



Contents

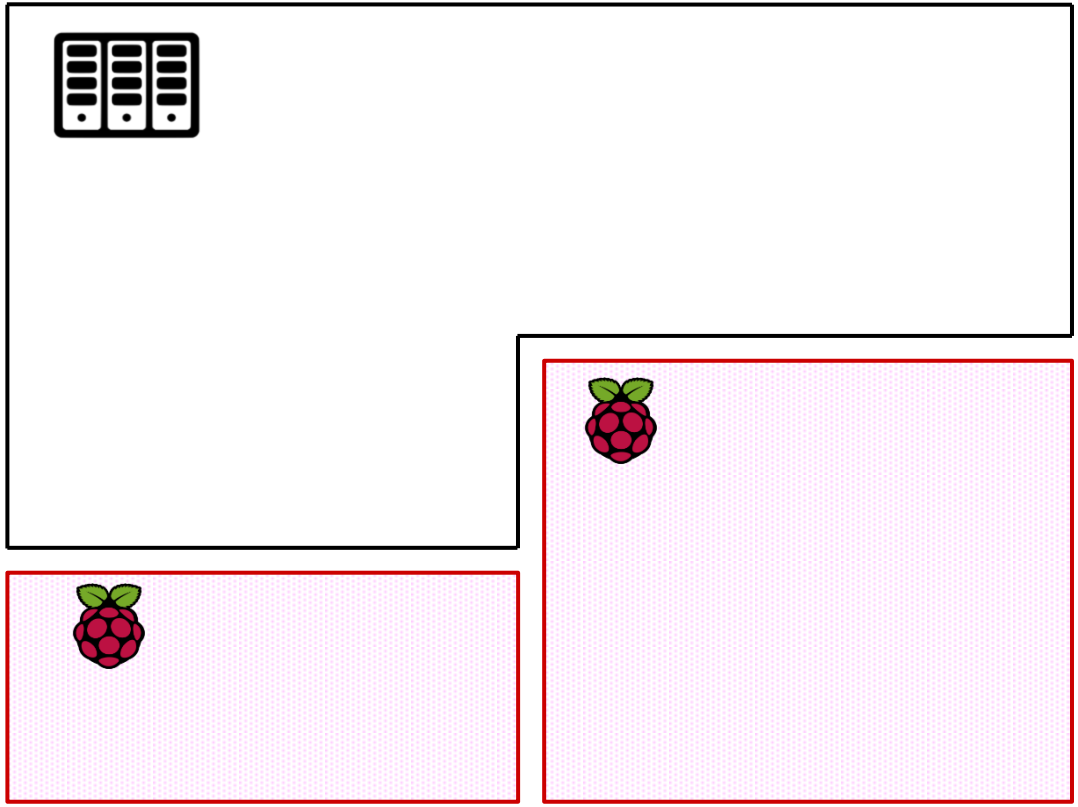
- Why Leshan ?
- How we used it
- Proxies
  - How a proxy work
  - Changes in leshan-client-demo
- Servers
    - How a server works
    - Changes in leshan-server-cluster
- Wakaama on ESP32
- How it is integrated in the demonstration
- Near future work

Near future work

- Using the acquired experience to clean the architecture (micro-services?)
- Implement a generic layer to address other protocols (not only LWM2M)
- Synchronize with upstream Leshan code.

Thanks

Multi-server demo architecture



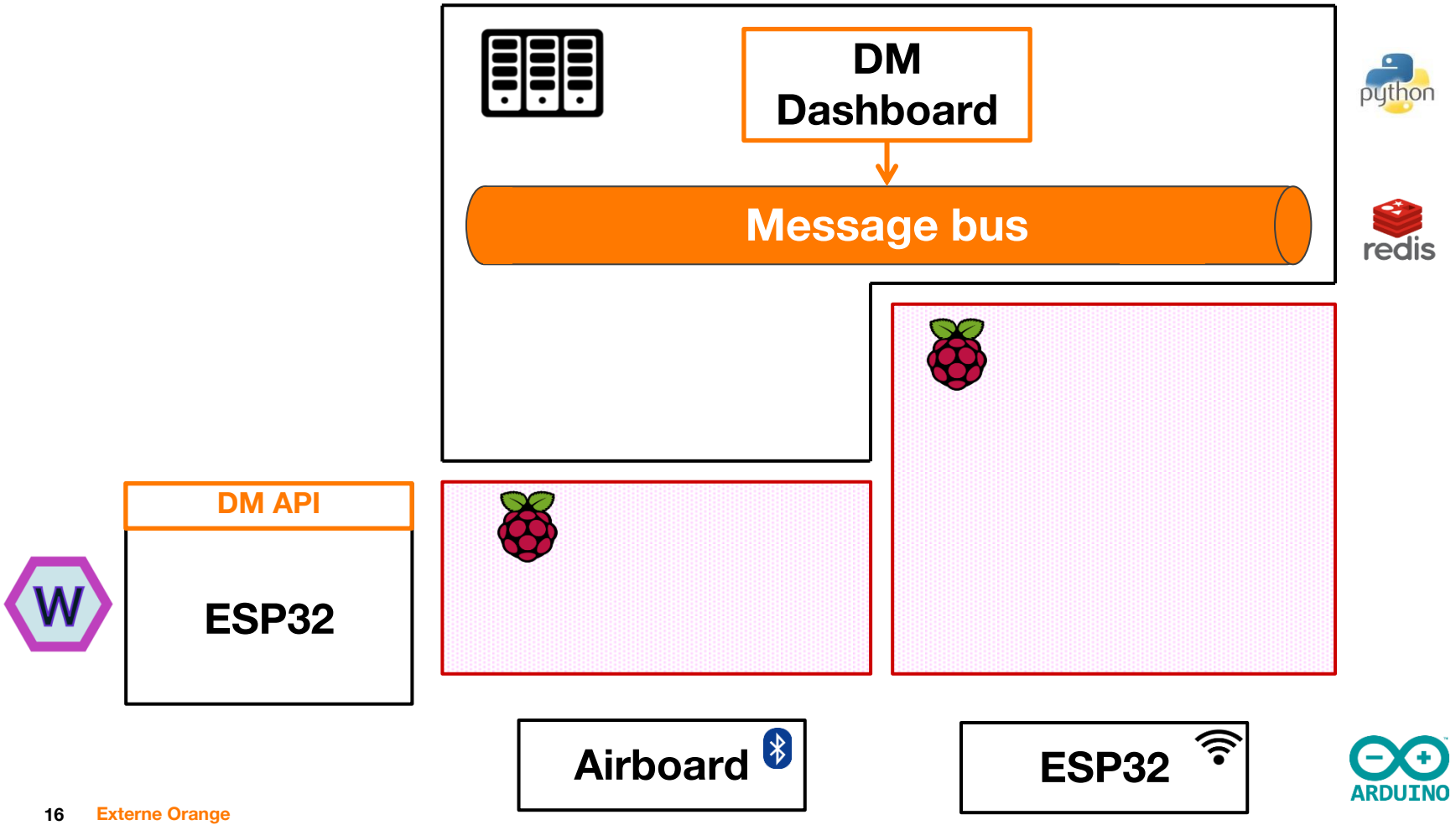
DM API

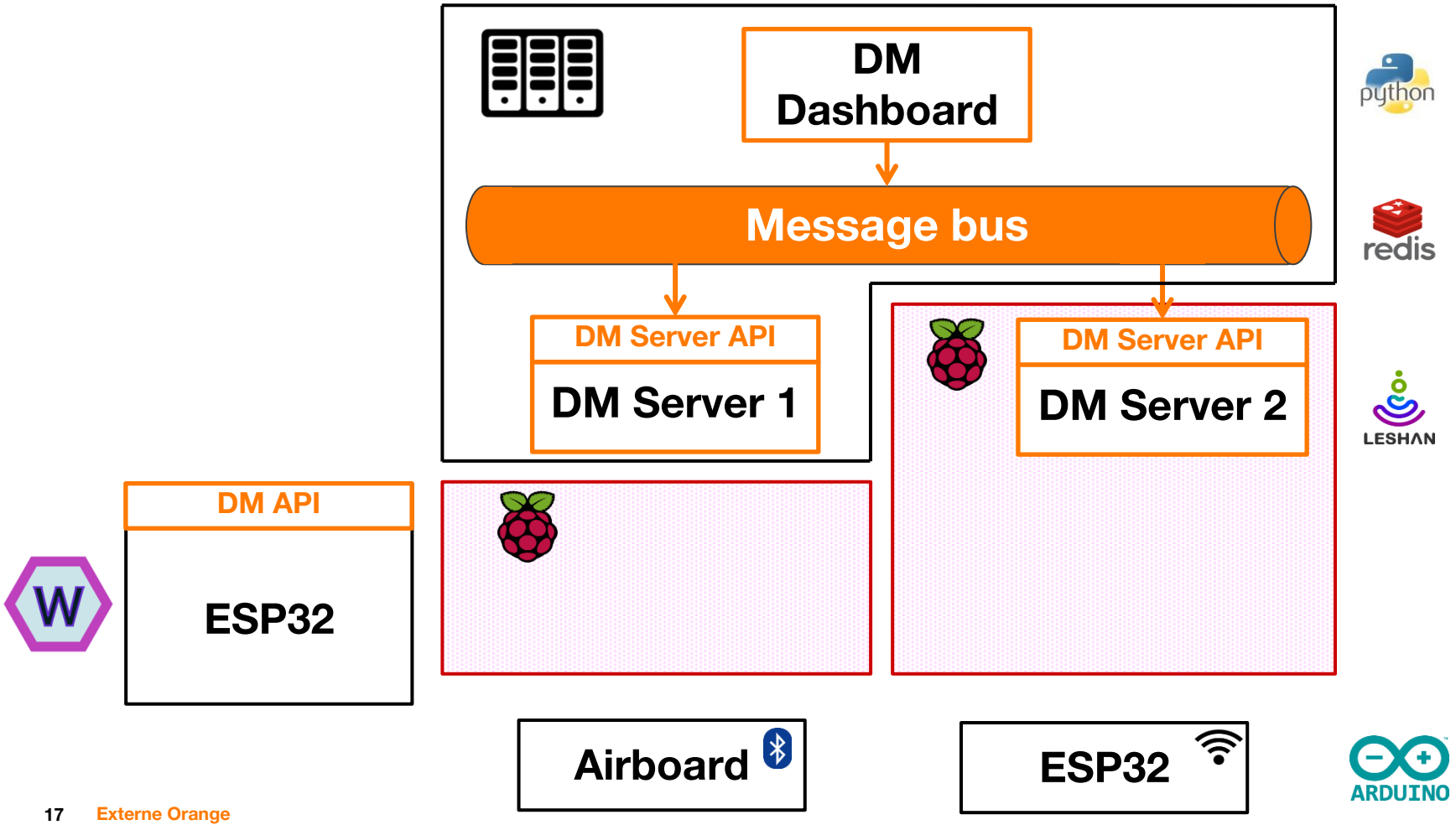
ESP32

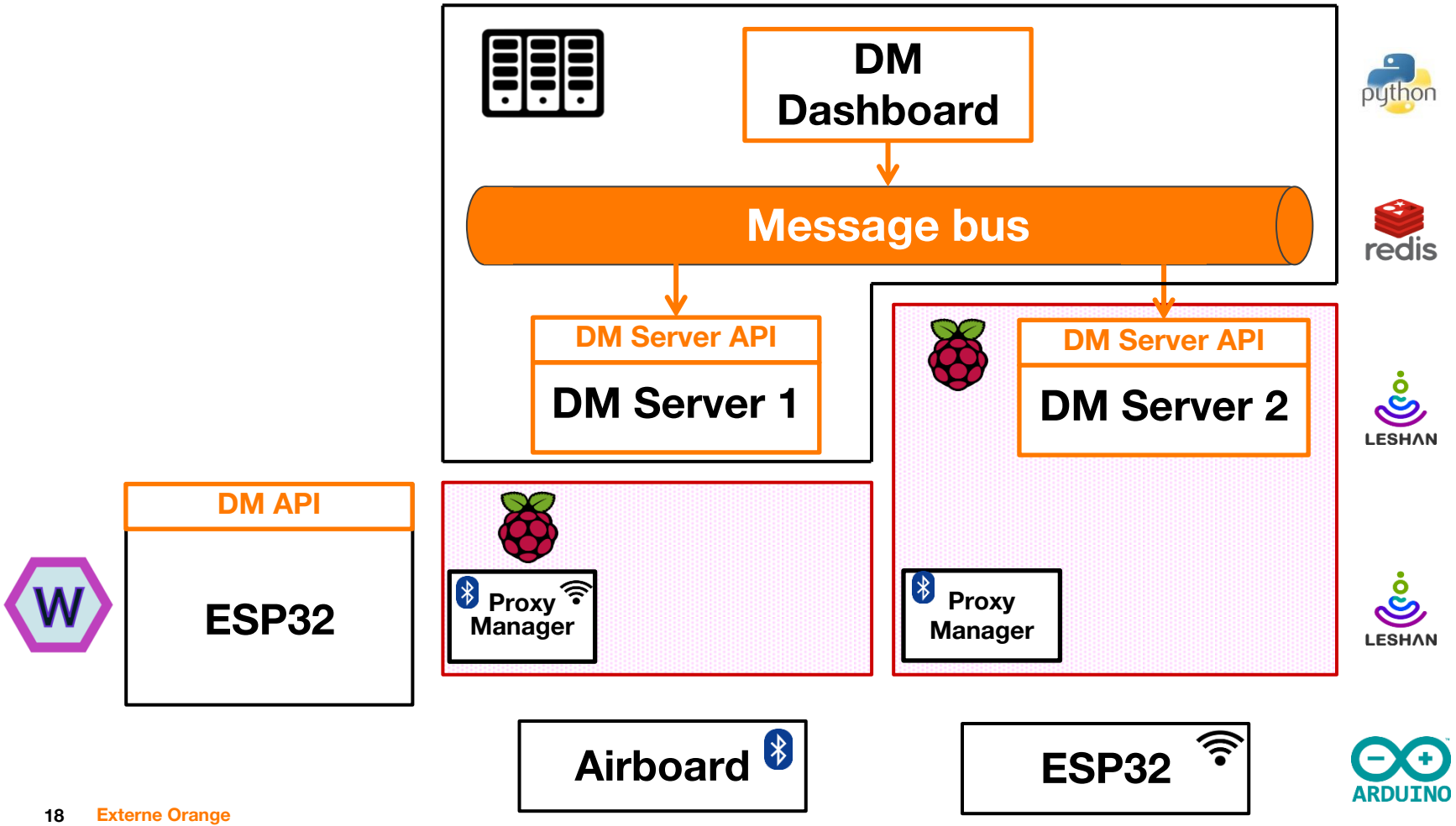
Airboard

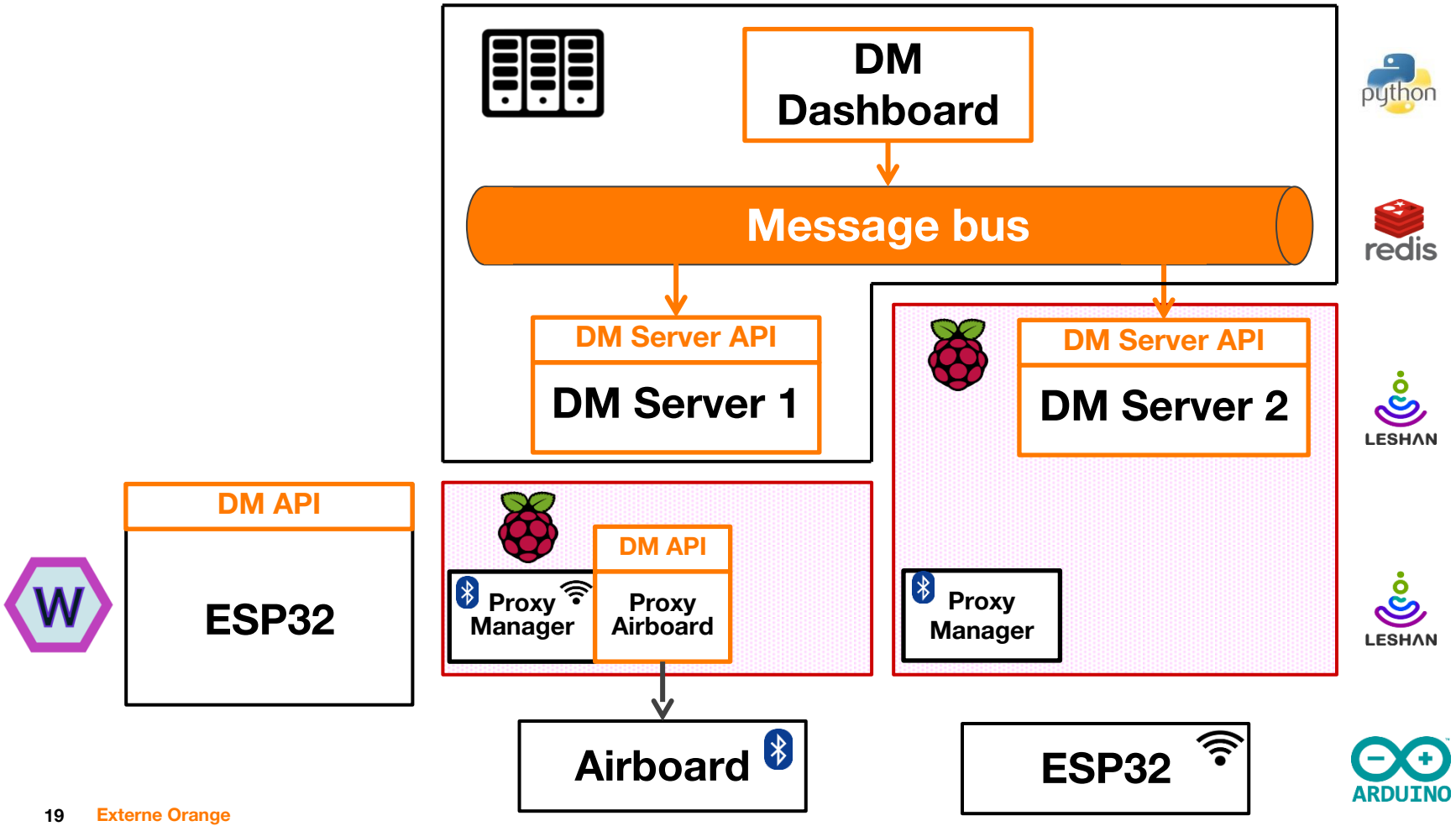
ESP32

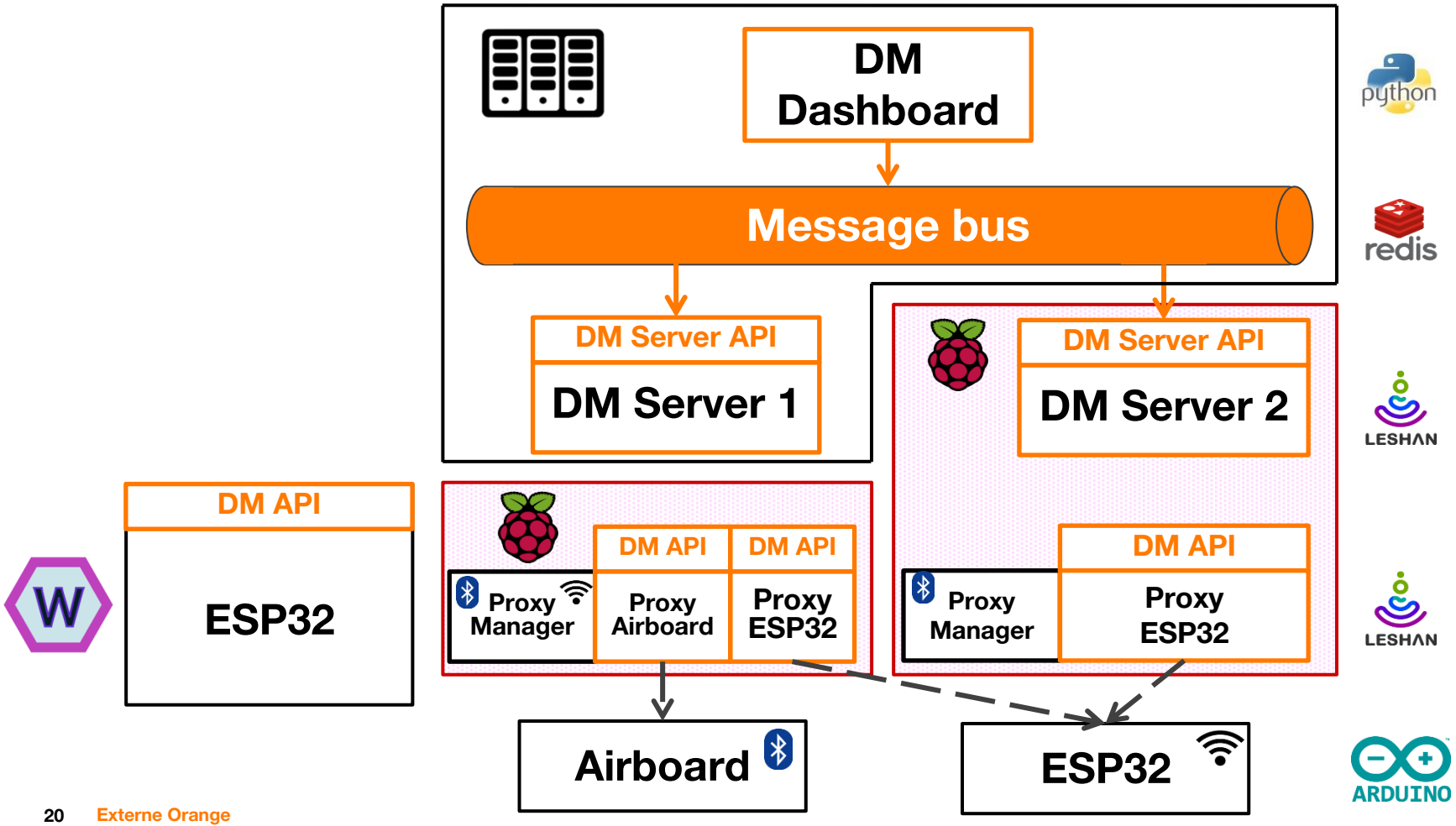
ARDUINO

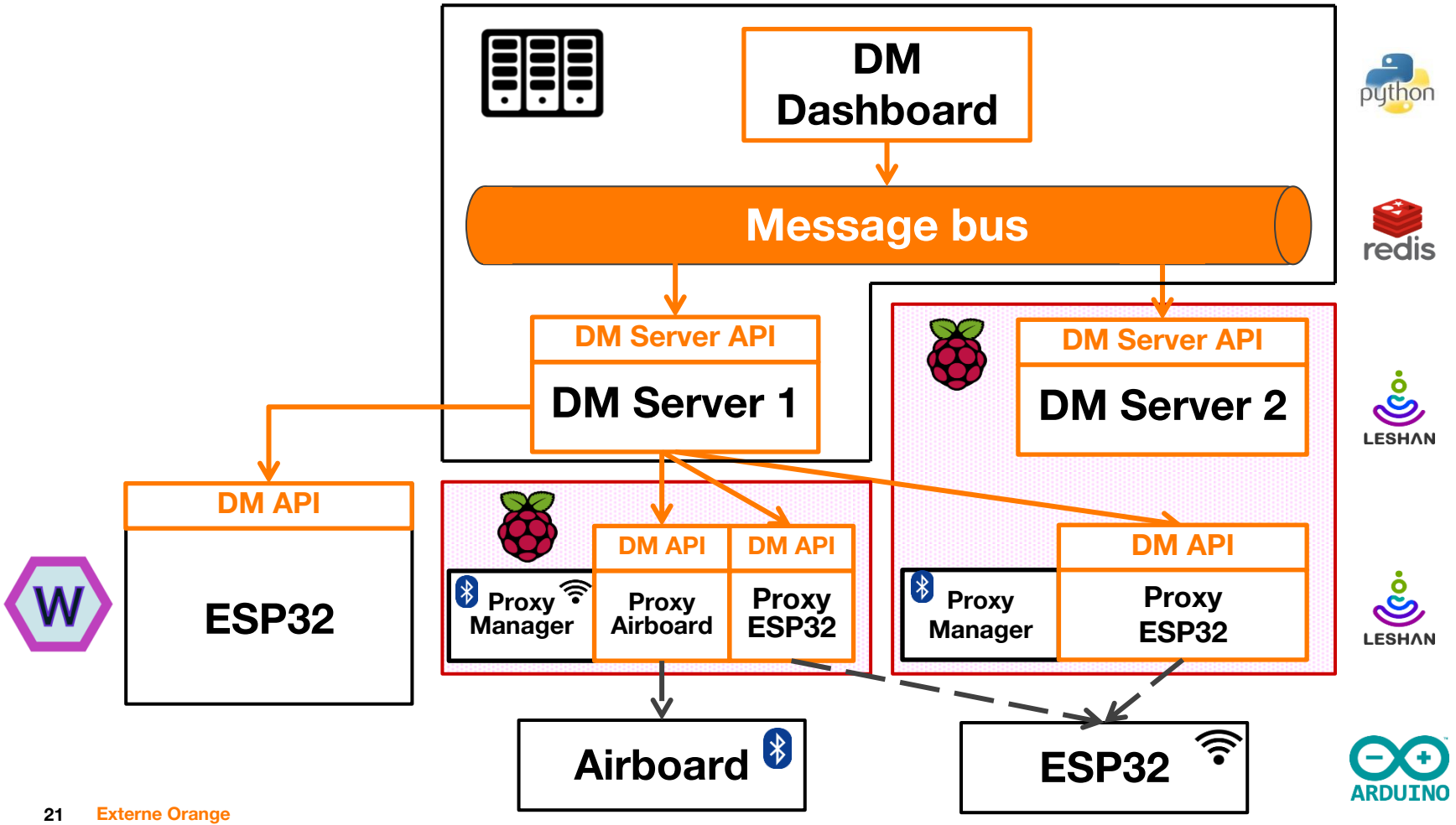

DM
Dashboard
python
Message bus
redis
DM API
ESP32
Airboard
ESP32
ARDUINO

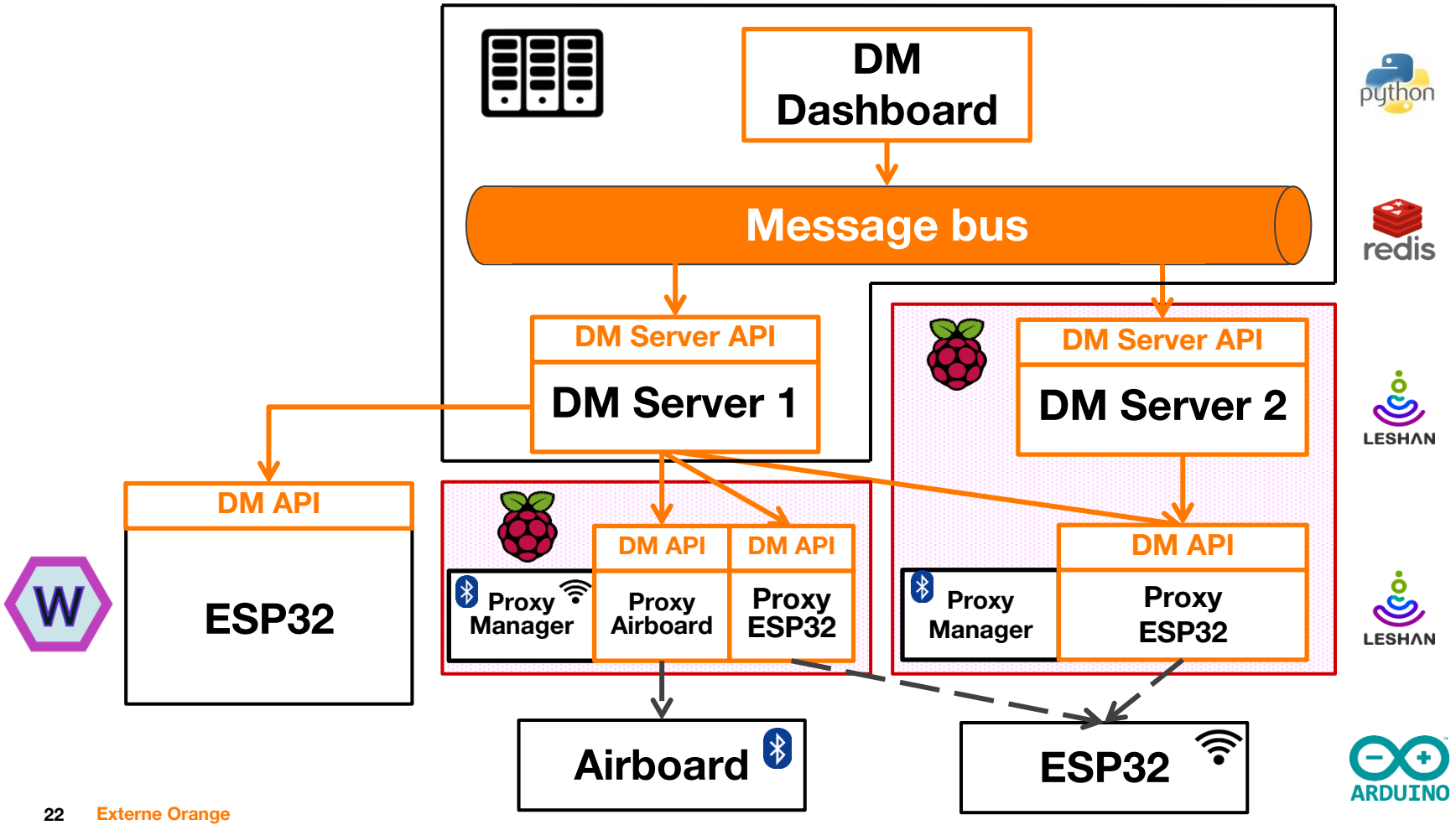

DM
Dashboard
Message bus
DM Server API
DM Server 1
DM Server API
DM Server 2
DM API
ESP32
Airboard
ESP32
python
redis
LESHAN
ARDUINO


DM
Dashboard
Message bus
DM Server API
DM Server 1
DM Server API
DM Server 2
DM API
ESP32
Proxy
Manager
Proxy
Manager
Airboard
ESP32
python
redis
LESHAN
LESHAN
ARDUINO


DM
Dashboard
Message bus
DM Server API
DM Server 1
DM Server API
DM Server 2
DM API
ESP32
DM API
Proxy
Manager
Proxy
Airboard
Proxy
Manager
Airboard
ESP32
python
redis
LESHAN
LESHAN
ARDUINO


DM
Dashboard
Message bus
DM Server API
DM Server 1
DM Server API
DM Server 2
DM API
ESP32
DM API
DM API
Proxy
Manager
Proxy
Airboard
Proxy
ESP32
DM API
Proxy
Manager
Proxy
ESP32
Airboard
ESP32
python
redis
LESHAN
LESHAN
ARDUINO


DM
Dashboard
Message bus
DM Server API
DM Server 1
DM Server API
DM Server 2
DM API
ESP32
DM API
DM API
Proxy
Manager
Proxy
Airboard
Proxy
ESP32
DM API
Proxy
Manager
Proxy
ESP32
Airboard
ESP32
python
redis
LESHAN
LESHAN
ARDUINO


DM
Dashboard
Message bus
DM Server API
DM Server 1
DM Server API
DM Server 2
DM API
ESP32
DM API
DM API
Proxy
Manager
Proxy
Airboard
Proxy
ESP32
DM API
Proxy
Manager
Proxy
ESP32
Airboard
ESP32
python
redis
LESHAN
LESHAN
ARDUINO

Time for the demo!

Conclusion

New solutions

- **Multi-server, multi-protocol architecture**
- **Integration of multiple and new DM servers**
- **A need for abstraction**

Next Steps with Eclipse IoT

- **Contributions**
  - Leshan
  - Wakaama + ESP32
- **Study of integration**
  - Hono
  - hawkBit

Thank you