Python for IoT

A return of experience

Alexandre Abadie, Inria

Outline

What IoT are we talking about ?

The usual protocols for IoT

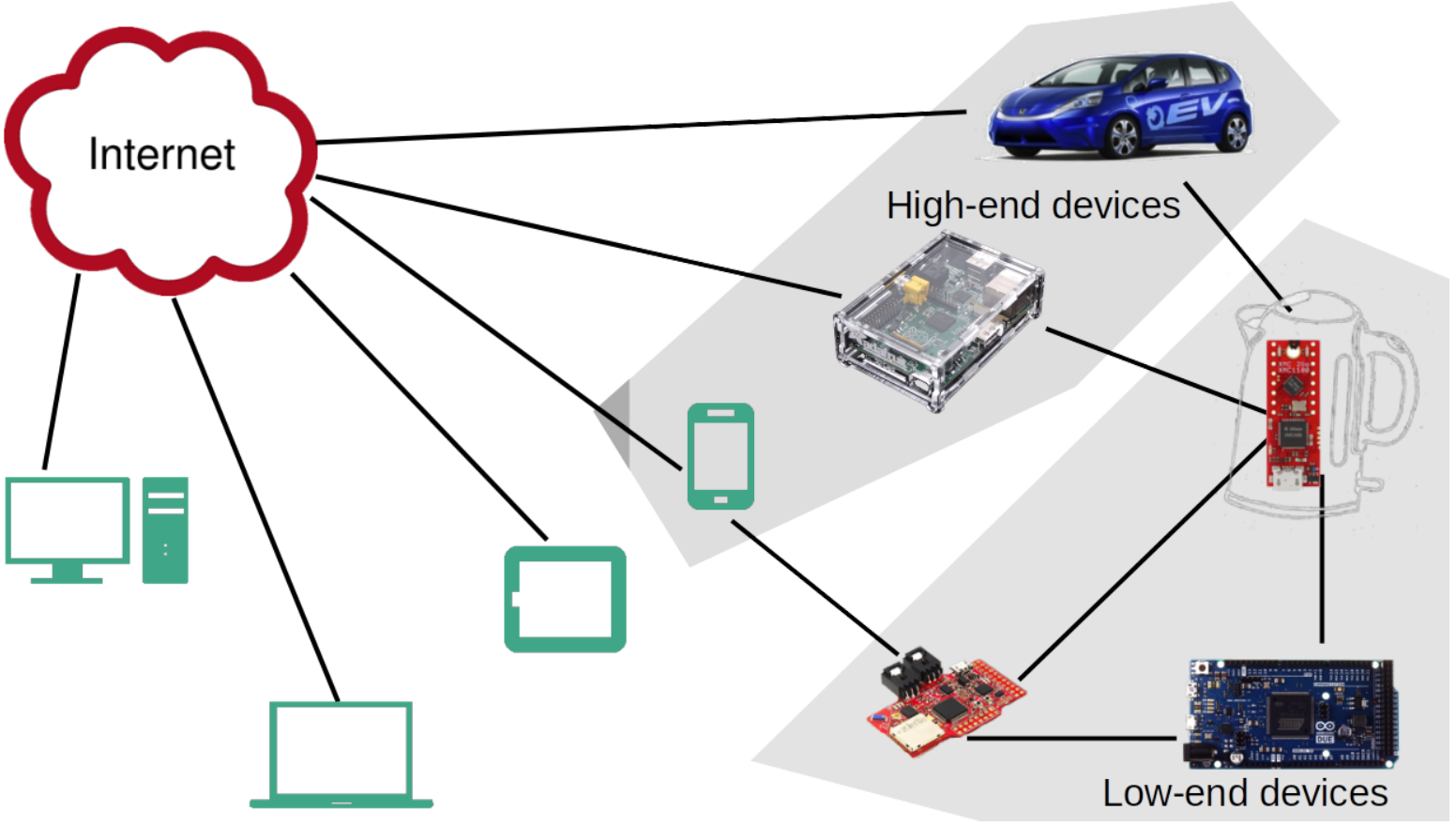
Pyaiot, connecting objects to the web

How we built Pyaiot

Lessons learned

Conclusion

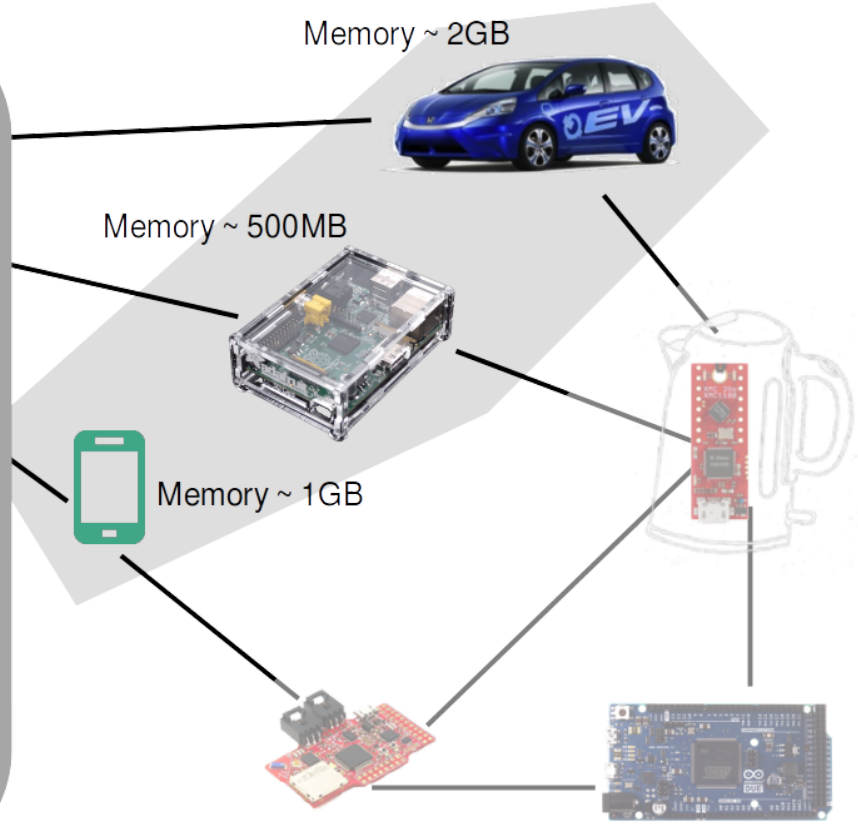
What IoT are we talking about ?



The Internet of Things today

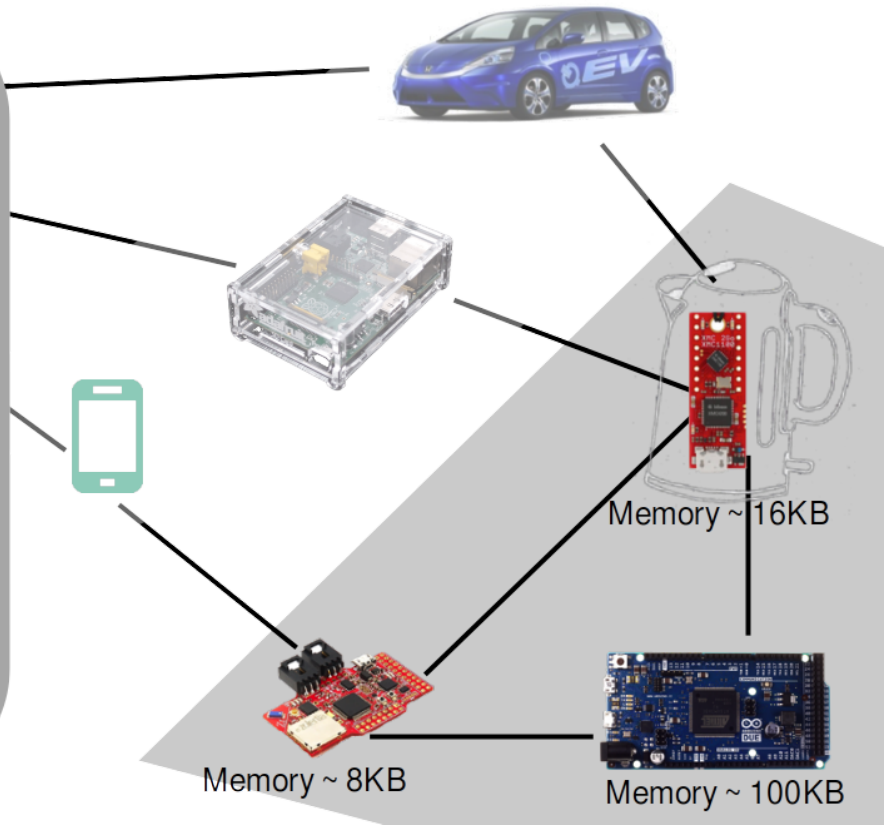
High-end devices

- Single board computers
  → Raspberry Pi, connected cars, smartphones...
- Resources similar to average internet devices
  → memory, CPU, network, etc
- Cannot run on battery on a long period of time
- **Can use TCP/IP based protocols**
- **Can support usual OS such as Linux**



Low-end devices

- Smaller & cheaper
  → Smart objects
- Low power Microcontroller & Radios
  → Can run on battery on a long period of time
- **Cannot use TCP/IP based protocols**
- **Cannot support usual OS such as Linux**



⇒ adapted protocols are required

Outline

What IoT are we talking about ?

⇒ **The usual protocols for IoT**

Pyaiot, connecting objects to the web

How we built Pyaiot

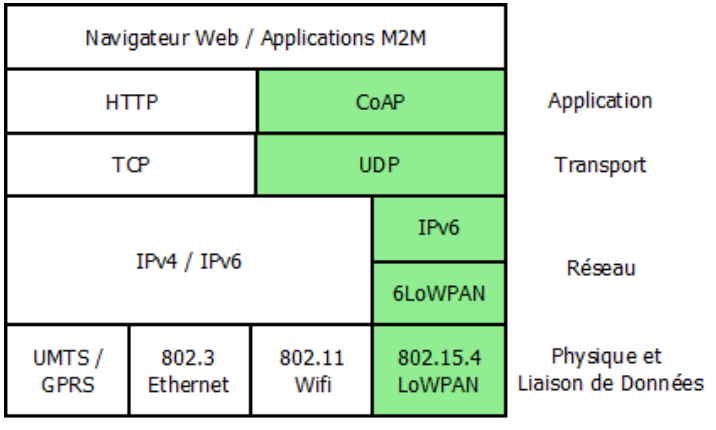
Lessons learned

Conclusion

Usual protocols for IoT: CoAP

- Core WG at IETF specifications (2010)
- RFC 7252
- Similar to HTTP REST:

  GET/PUT/POST/DELETE + OBSERVE
- Works on UDP with small payload overhead



source: https://fr.wikipedia.org/wiki/CoAP

More information at http://coap.technology/

CoAP: available implementations in Python

3 available implementations:

- TxThings: Twisted based, Python 2 & 3

  https://github.com/mwasilak/txThings

- Aiocoap: asyncio based, Python 3 only

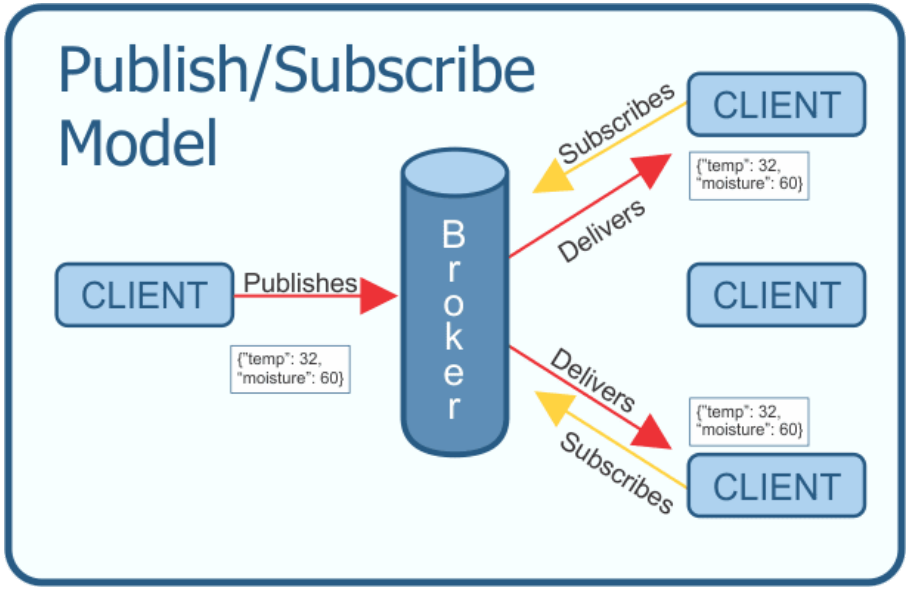
  https://github.com/chrysn/aiocoap

- CoaPthon: Threading based, Python 2 & 3

  https://github.com/Tanganelli/CoAPthon

Implementations exist for other languages: **http://coap.technology/impls.html**

Usual protocols for IoT: MQTT



source: https://dev.to/kenwalger/overview-of-the-mqtt-protocol

- Based on publication/subscriptions to topics pattern
- Topics have a path form: `this/is/a/topic`
- MQTT v3.1.1 is an OASIS standard
- **MQTT Sensor Network (MQTT-SN)**: adapted for constrained devices

MQTT: available implementations in Python

2 available implementations:

- Paho-mqtt: threading based, considered as the reference implementation

  https://pypi.python.org/pypi/paho-mqtt

- HBMQTT: asyncio based

  https://github.com/beerfactory/hbmqtt

Outline

What IoT are we talking about ?

The usual protocols for IoT

⇒ **Pyaiot, connecting objects to the web**

How we built Pyaiot

Lessons learned

Conclusion

Why Pyaiot?

- Need for a web application able to communicate with contrained devices

    ⇒ but constrained devices **cannot use usual web protocols**

Why Pyaiot?

- Need for a web application able to communicate with contrained devices

    ⇒ but constrained devices **cannot use usual web protocols**

- Need for multi-site support

    ⇒ but constrained devices **cannot be exposed directly to the web**

Why Pyaiot?

- Need for a web application able to communicate with contrained devices

  ⇒ but constrained devices **cannot use usual web protocols**

- Need for multi-site support

  ⇒ but constrained devices **cannot be exposed directly to the web**

- Heterogeneous protocol support

  ⇒ **various IoT protocols exist**

General guidelines

- Open-Source and simple design⇒ **can be deployed by anyone**

https://github.com/pyaiot/pyaiot

General guidelines

- Open-Source and simple design⇒ **can be deployed by anyone**

  https://github.com/pyaiot/pyaiot

- Multiprotocol: CoAP, MQTT, etc ⇒ **interoperability**

General guidelines

- Open-Source and simple design⇒ **can be deployed by anyone**

https://github.com/pyaiot/pyaiot

- Multiprotocol: CoAP, MQTT, etc ⇒ **interoperability**
- Modular ⇒ **extensible**

General guidelines

- Open-Source and simple design⇒ **can be deployed by anyone**

https://github.com/pyaiot/pyaiot

- Multiprotocol: CoAP, MQTT, etc ⇒ **interoperability**
- Modular ⇒ **extensible**
- Bi-directionnal and real time access to nodes ⇒ **reactive**

General guidelines

- Open-Source and simple design⇒ **can be deployed by anyone**

  https://github.com/pyaiot/pyaiot

- Multiprotocol: CoAP, MQTT, etc ⇒ **interoperability**
- Modular ⇒ **extensible**
- Bi-directionnal and real time access to nodes ⇒ **reactive**
- No constraint regarding the backend language ⇒ **let's choose Python!**

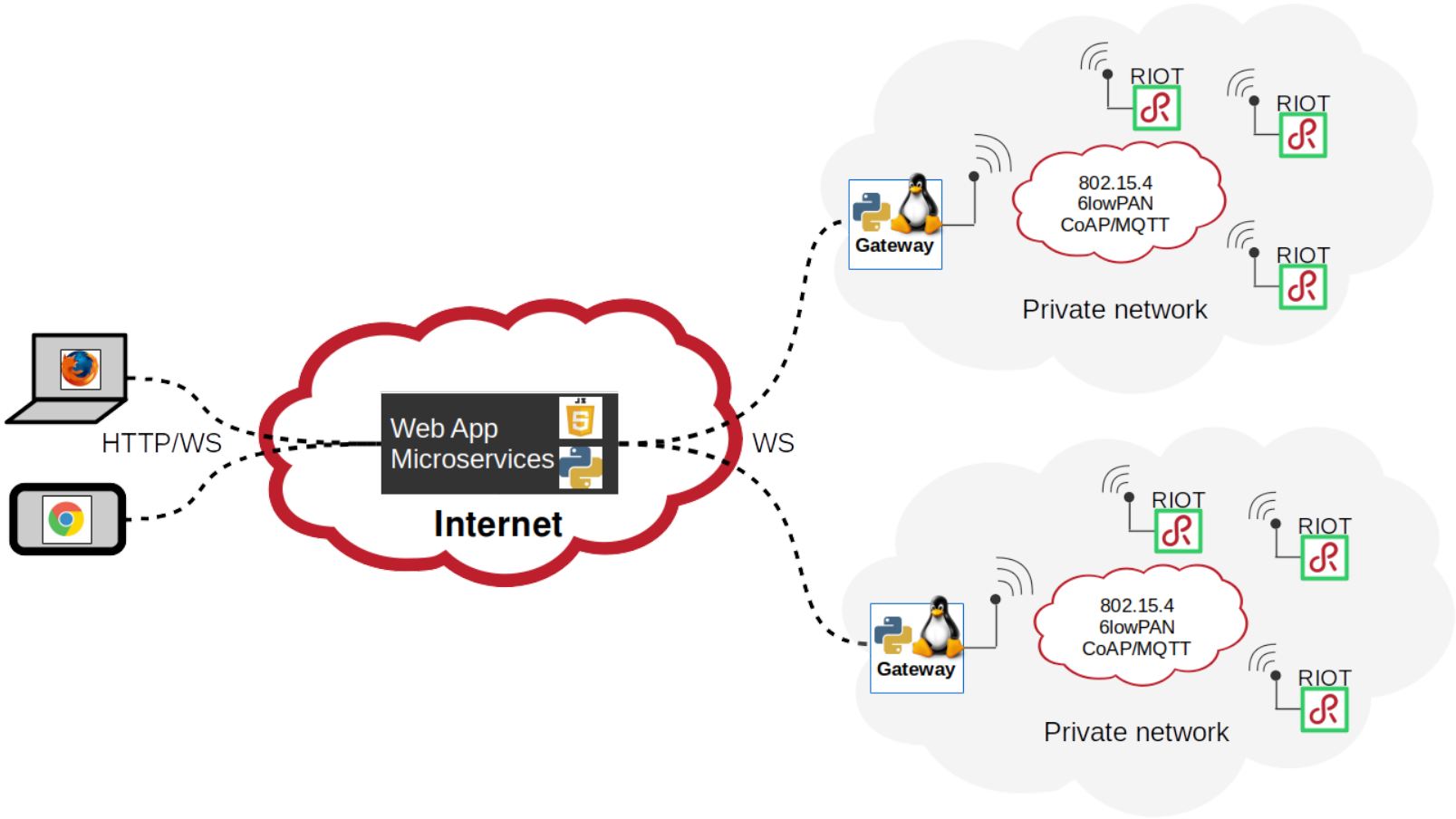
General guidelines

- Open-Source and simple design⇒ **can be deployed by anyone**

https://github.com/pyaiot/pyaiot

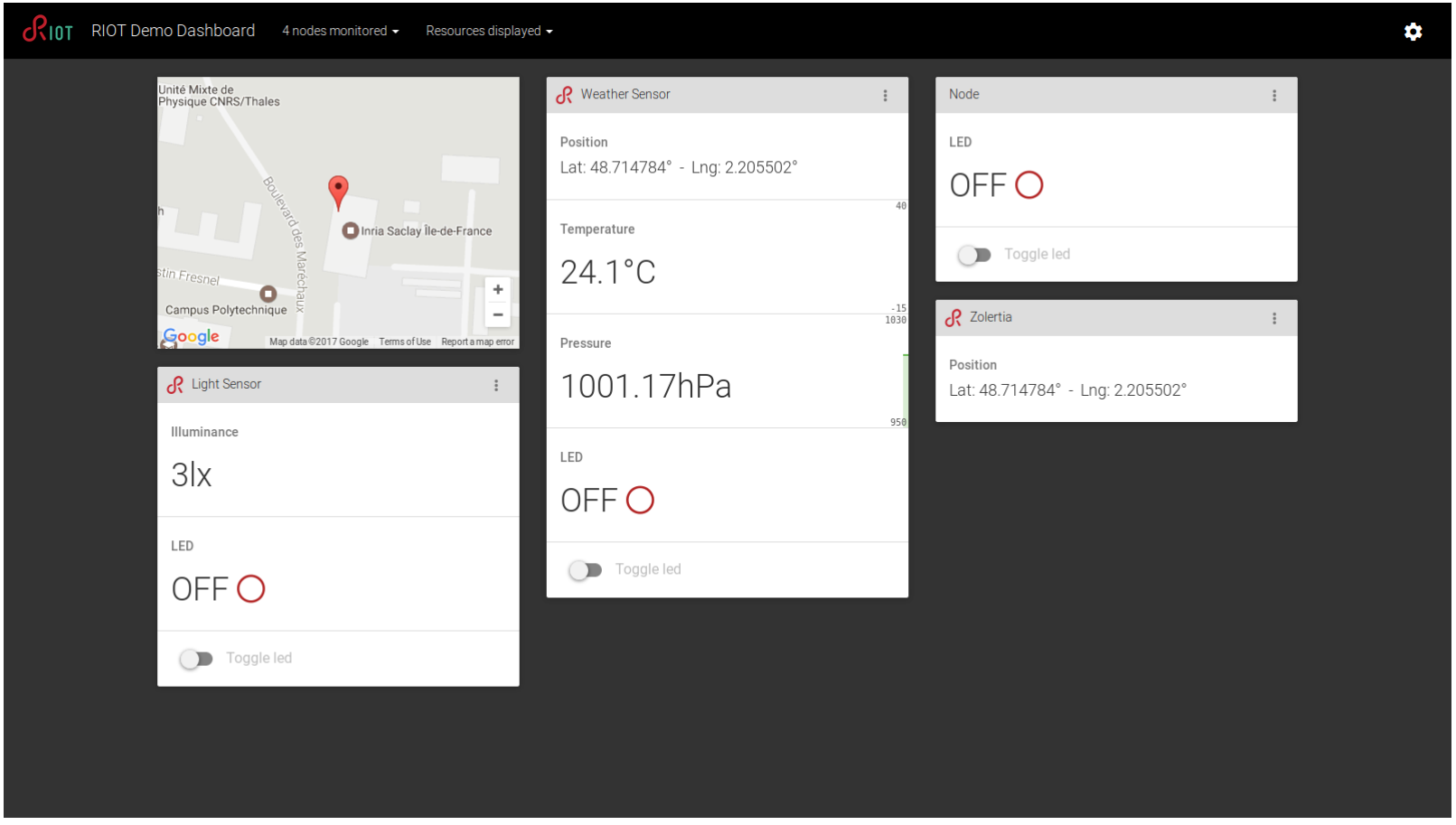
- Multiprotocol: CoAP, MQTT, etc ⇒ **interoperability**
- Modular ⇒ **extensible**
- Bi-directionnal and real time access to nodes ⇒ **reactive**
- No constraint regarding the backend language ⇒ **let's choose Python!**
- Pyaiot targets contrained nodes running RIOT: https://riot-os.org

Pyaiot overview



Permanent web showcase for RIOT available at
http://riot-demo.inria.fr

Pyaiot: The web dashboard



Outline

What IoT are we talking about ?

The usual protocols for IoT

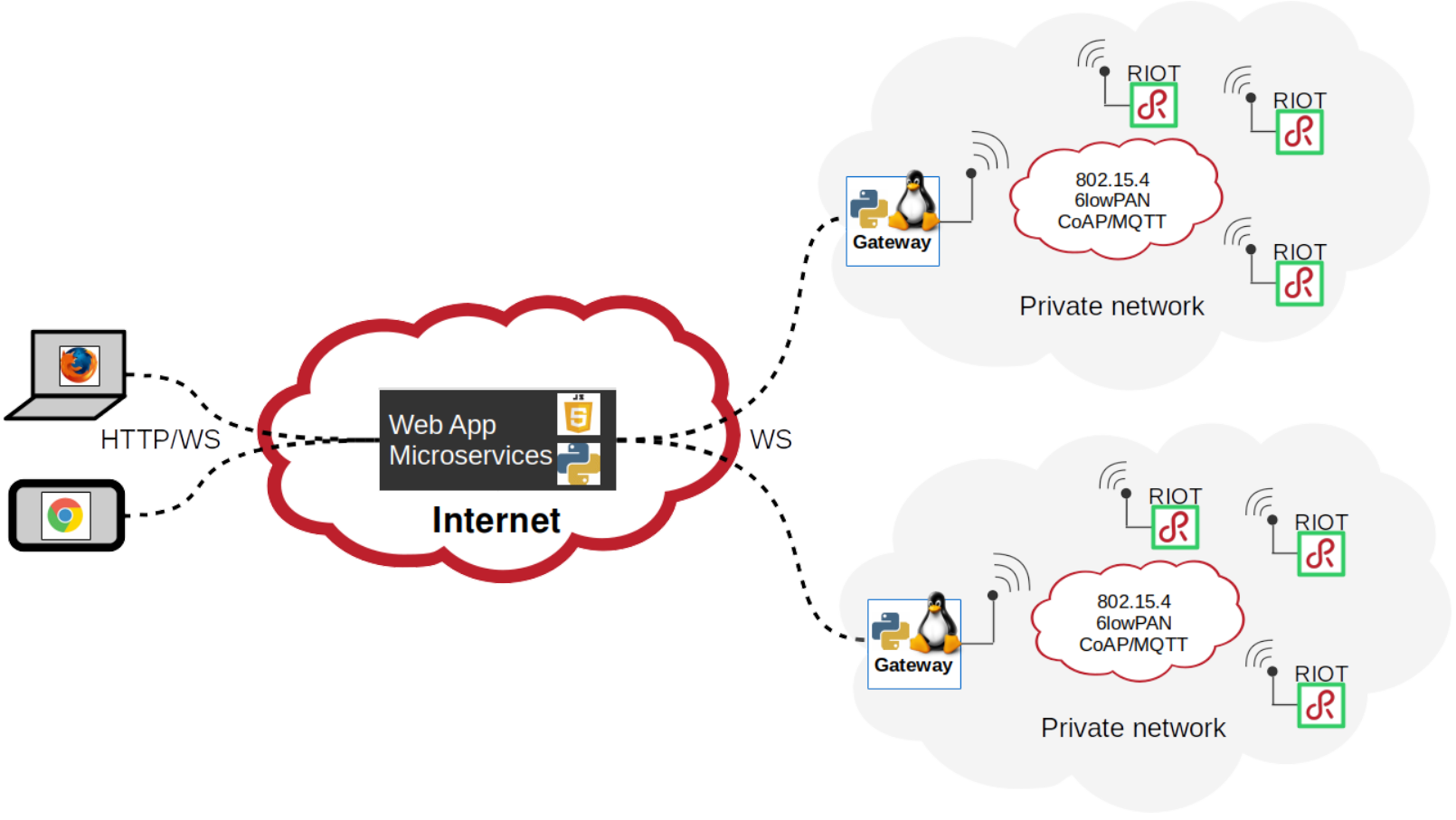
Pyaiot, connecting objects to the web

⇒ **How we built Pyaiot**

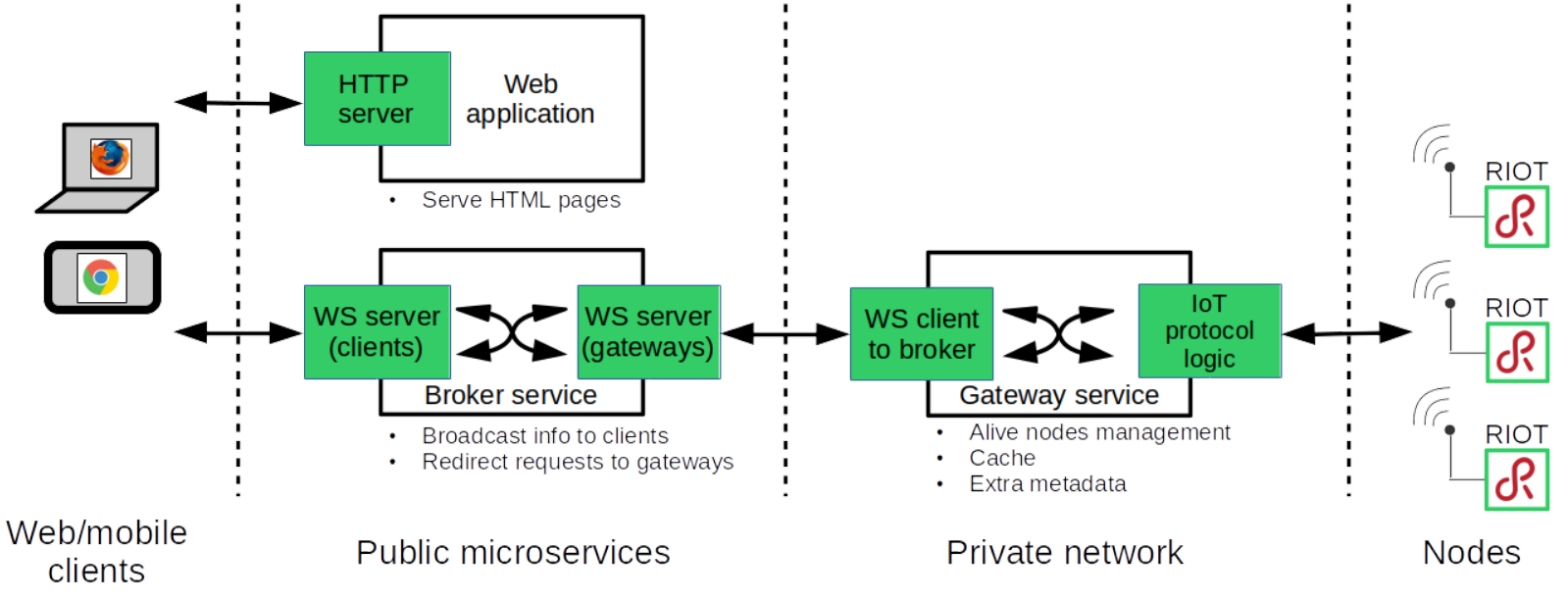
Lessons learned

Conclusion

Pyaiot overview

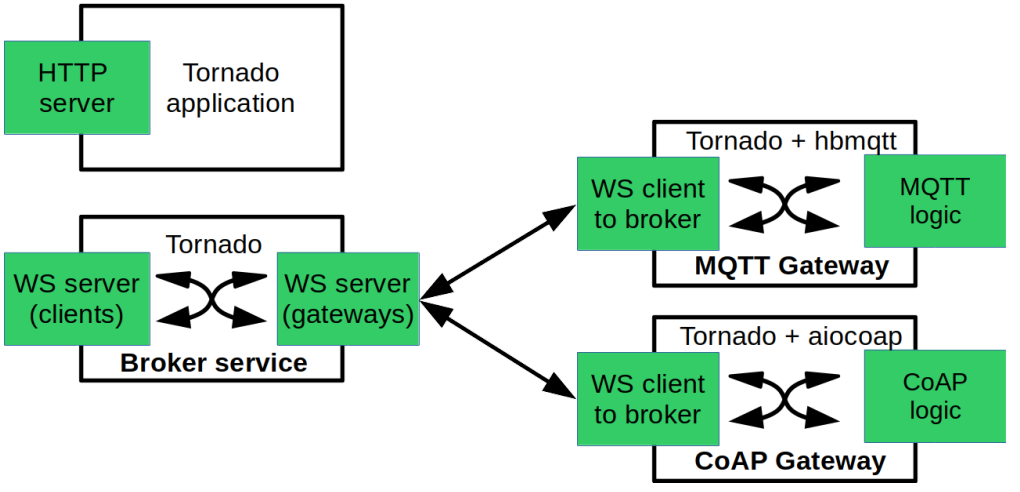


Pyaiot services



- Gateways are clients running in private networks
- Nodes are kept isolated from Internet
- Messages exchanged in JSON format
- Works with low-end devices (RIOT) and high-end devices (Python)

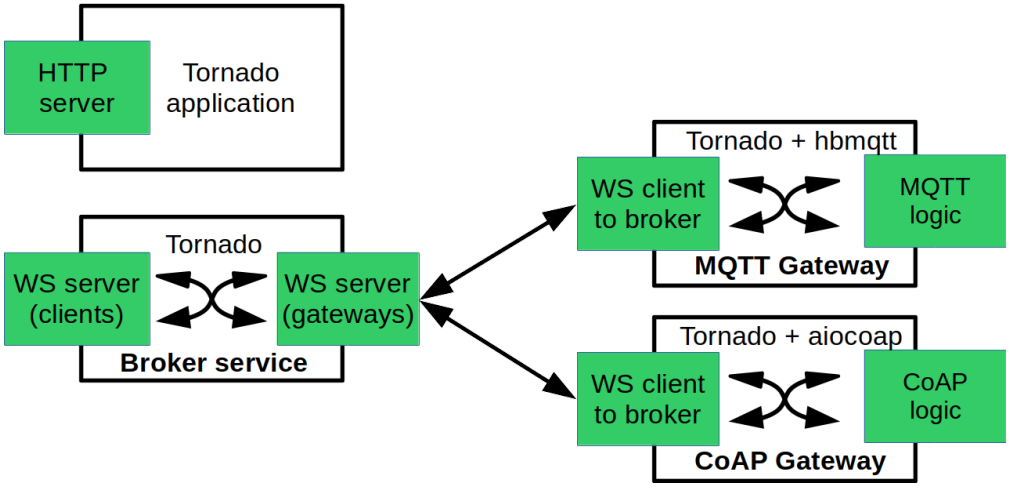
Technical choices



- Web dashboard developed with **Vue.js** http://vuejs.org
- Service applications based on **Tornado** framework with:
  - HTTP server
  - Websocket server and client
- **Aiocoap** for CoAP protocol support
- **HBMQTT** for MQTT protocol support

Tornado

Technical choices



- Web dashboard developed with **Vue.js** http://vuejs.org
- Service applications based on **Tornado** framework with:
  - HTTP server
  - Websocket server and client
- **Aiocoap** for CoAP protocol support
- **HBMQTT** for MQTT protocol support

⇒ All python packages are **asyncio** based/compatible ⇒ **simplify integration**

The MQTT gateway in detail

- MQTT-SN is required for low-end device

  ⇒ a MQTT to MQTT-SN gateway/broker is required

- No implementation in Python

  ⇒ let's go for mosquitto.rsmb

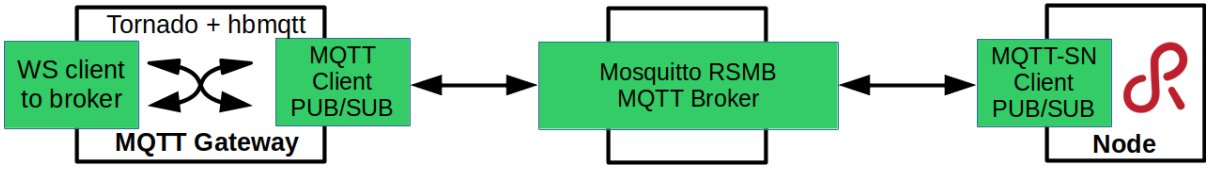
The MQTT gateway in detail

- MQTT-SN is required for low-end device

  ⇒ a MQTT to MQTT-SN gateway/broker is required

- No implementation in Python

  ⇒ let's go for [mosquitto.rsmb]



Node/Gateway

subscribe/publish to topic

- *gateway//discover*

publish/subscribe to topics

- *node/check*
- *node//resources*
- *node//*

Outline

What IoT are we talking about ?

The usual protocols for IoT

Pyaiot, connecting objects to the web

How we built Pyaiot

⇒ **Lessons learned**

Conclusion

Using asyncio

- Easy to read asynchronous programming language

Using asyncio

- Easy to read asynchronous programming language
- Asyncio new syntax available with Python >= 3.5

Using asyncio

- Easy to read asynchronous programming language
- Asyncio new syntax available with Python >= 3.5

  ... but Python 3.4.2 available on Raspbian

```
@asyncio.coroutine
def my_coroutine():
    my_long_call()

yield from my_coroutine()  # wait until done
asyncio.get_event_loop().create_task(my_coroutine)  # scheduled in ioloop
asyncio.ensure_future(my_coroutine)  # scheduled in ioloop, requires python 3.4.4
```

Using asyncio

- Easy to read asynchronous programming language
- Asyncio new syntax available with Python >= 3.5

  ... but Python 3.4.2 available on Raspbian

```
@asyncio.coroutine
def my_coroutine():
    my_long_call()

yield from my_coroutine()  # wait until done
asyncio.get_event_loop().create_task(my_coroutine)  # scheduled in ioloop
asyncio.ensure_future(my_coroutine)  # scheduled in ioloop, requires python 3.4.4
```

with python 3.5 new syntax:

```
async def my_coroutine():
    my_long_call()

await my_coroutine()  # wait until done
asyncio.ensure_future(my_coroutine) # scheduled in ioloop
```

The benefits of Python

- Develop fast, even with complex things

The benefits of Python

- Develop fast, even with complex things
- Can run on any high-end device : from a Raspberry PI to a Cloud server

The benefits of Python

- Develop fast, even with complex things
- Can run on any high-end device : from a Raspberry PI to a Cloud server
- Off-the-shelf packages for IoT available: Aiocoap, HBMQTT

The benefits of Python

- Develop fast, even with complex things
- Can run on any high-end device : from a Raspberry PI to a Cloud server
- Off-the-shelf packages for IoT available: Aiocoap, HBMQTT

⇒ Python is adapted to IoT

Conclusion

- Widely used protocol in IoT is MQTT
- Adapted protocols are required for constrained devices (microcontrollers)

  ⇒ CoAP, MQTT-SN

Conclusion

- Widely used protocol in IoT is MQTT
- Adapted protocols are required for constrained devices (microcontrollers)

  ⇒ CoAP, MQTT-SN
- We easily built an application following the initial requirements

  ⇒ Pyaiot: https://github.com/pyaiot/pyaiot

Conclusion

- Widely used protocol in IoT is MQTT
- Adapted protocols are required for constrained devices (microcontrollers)

  ⇒ CoAP, MQTT-SN
- We easily built an application following the initial requirements

  ⇒ Pyaiot: https://github.com/pyaiot/pyaiot
- Asyncio made things simpler... after some headaches

Conclusion

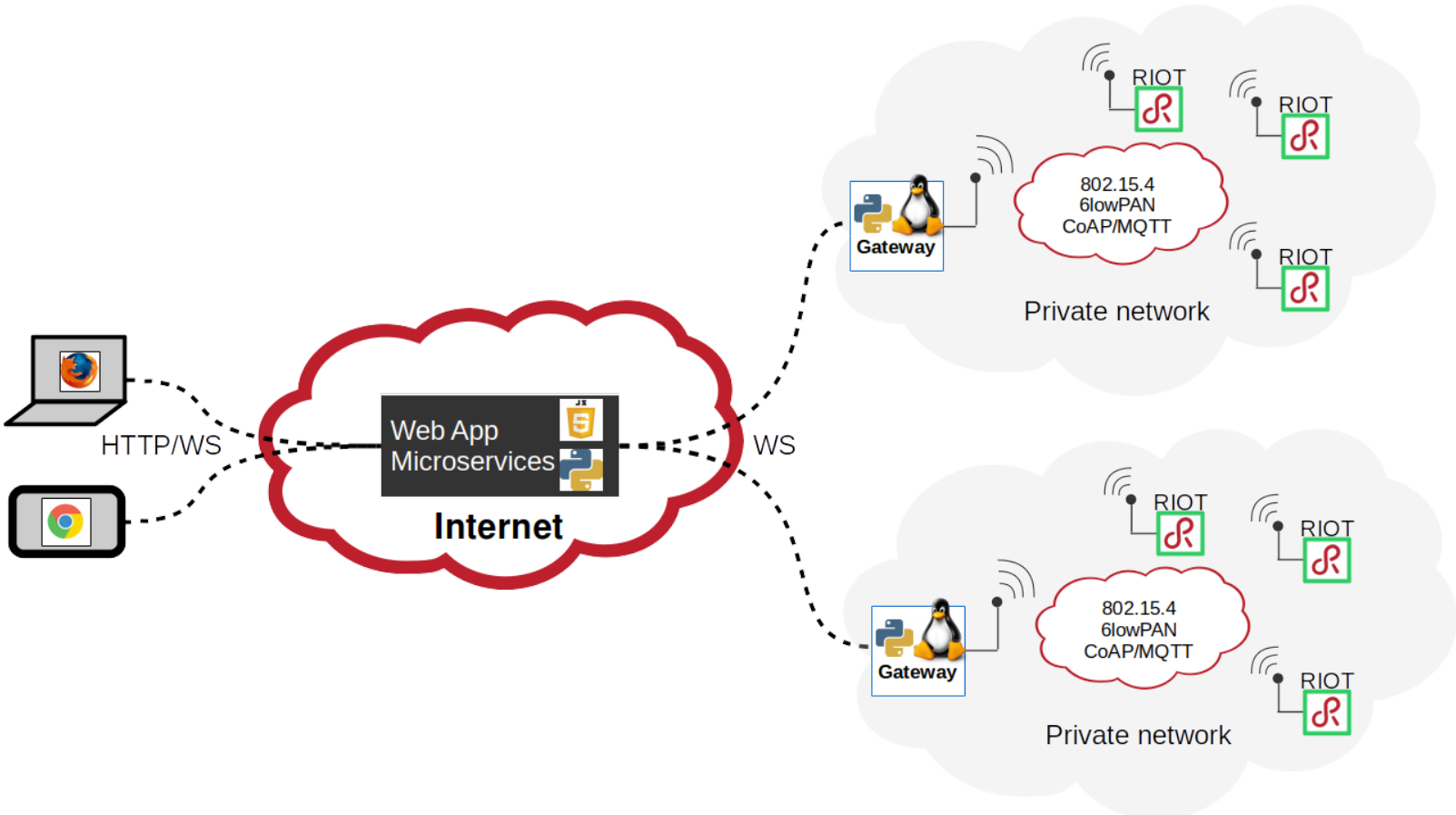
- Widely used protocol in IoT is MQTT
- Adapted protocols are required for constrained devices (microcontrollers)

  ⇒ CoAP, MQTT-SN
- We easily built an application following the initial requirements

  ⇒ Pyaiot: https://github.com/pyaiot/pyaiot
- Asyncio made things simpler... after some headaches
- Pyaiot is still work in progress... even if it works pretty well

Demo!

http://riot-demo.inria.fr



Thanks!