POLITECNICO DI TORINO

IoT concepts

Andrea Acquaviva – EDA group

Politecnico di Torino, Italy

Outline

- Introduction to the concept of IoT: paradigm, functionalities and requirements
- IoT devices features: sensing, processing, communication
- Main protocols and technologies: middleware, web services, communication paradigms (e.g. publish/subscribe)
- IoT programming: currently available methodologies, tools and platoforms (e.g. IBM bluemix)
- Current and future applications for IoT

IoT concept

- US National Intelligence Council (NIC) consider Internet of Things as one of the 6 ‘‘Disruptive Civil Technologies”

  (April 2008)

- IEEE ranks IoT #1 in the list of “Top Trends for 2013” (Winter 2012):
  - “…The IoT promises to be the *most disruptive technological revolution since the advent of the World Wide Web*. Projections indicate that up to *100 billion uniquely identifiable objects will be connected to the Internet by 2020*…”

Tentative definition

- IoT is…

"Things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts"

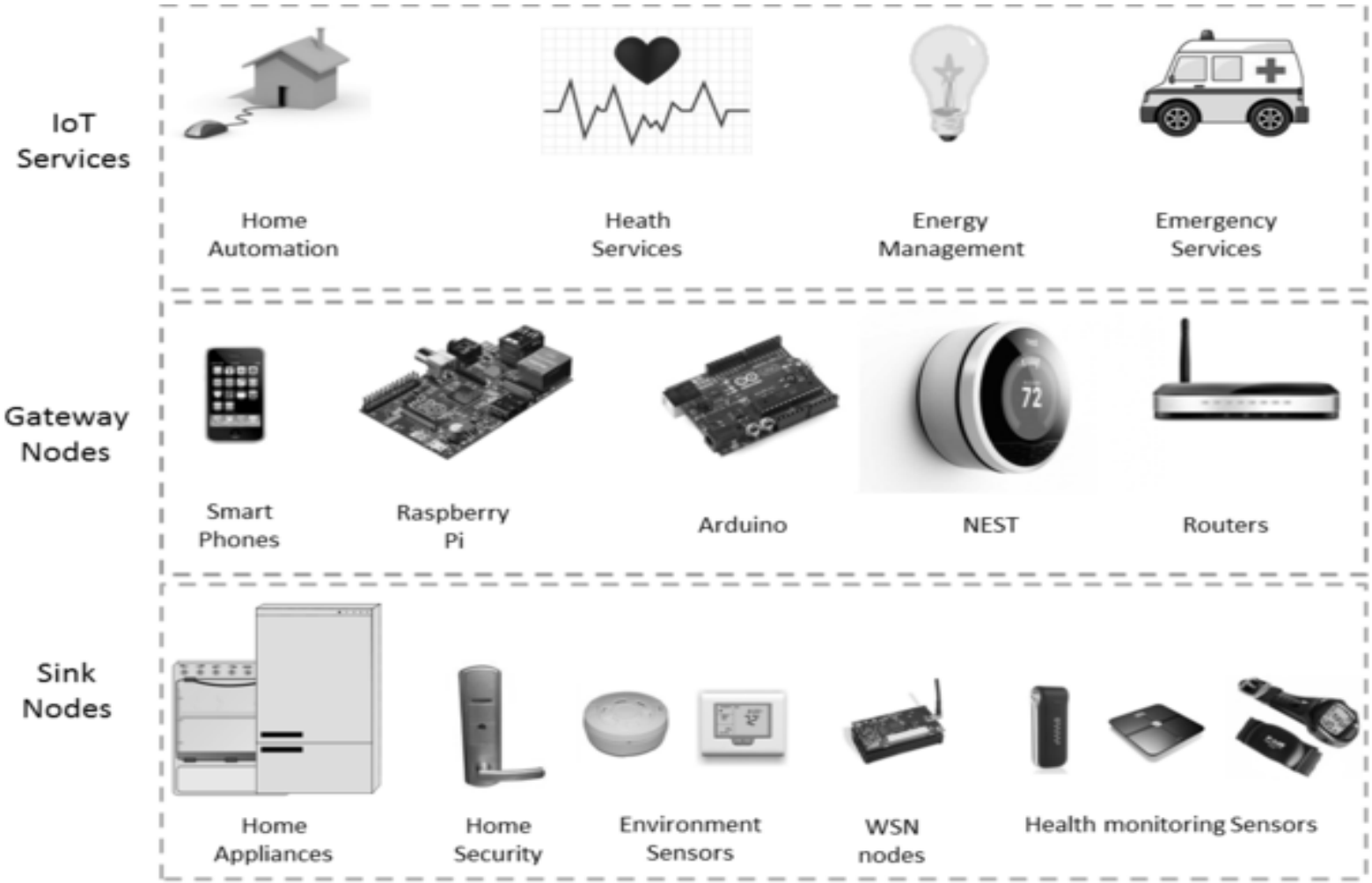
A more practical one

- According to Wikipedia, IoT refers to the interconnection of uniquely identifiable embedded computing-like devices within the existing Internet infrastructure. Typically, IoT is expected to offer advanced connectivity of devices, systems, and services that goes beyond machine-to-machine communications (M2M) and covers a variety of protocols, domains, and applications. The interconnection of these embedded devices (including smart objects), is expected to usher in automation in nearly all fields, while also enabling advanced applications like a Smart Grid.

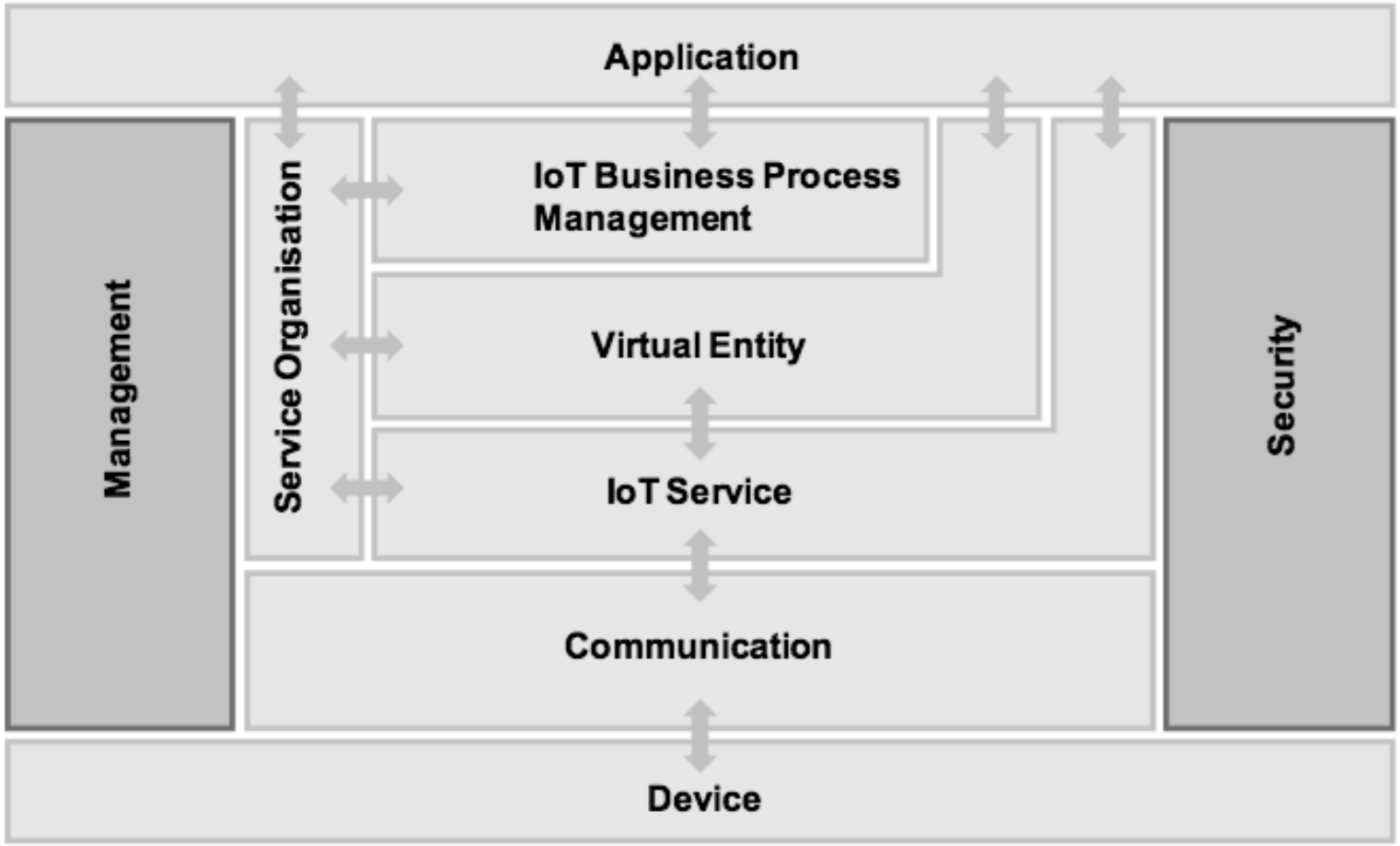
Things?

- Things, in the IoT, can refer to a wide variety of devices such as heart monitoring implants, bio-chip transponders on farm animals, automobiles with built-in sensors, or field operation devices that assist fire-fighters in search and rescue. Current market examples include smart thermostat systems and washer/dryers that utilize WiFi for remote monitoring.

IoT World


IoT Services
Home Automation
Heath Services
Energy Management
Emergency Services
Gateway Nodes
Smart Phones
Raspberry Pi
Arduino
NEST
Routers
Sink Nodes
Home Appliances
Home Security
Environment Sensors
WSN nodes
Health monitoring Sensors

IoT Architecture



Enabling Technologies

1. Sensors/Actuators
2. Communication protocols (REST, CoAP, MQTT)
3. Microservices and Middleware
4. Data Analytics Engines
5. Apps (iOS, Android, Web)

Web services & IoT

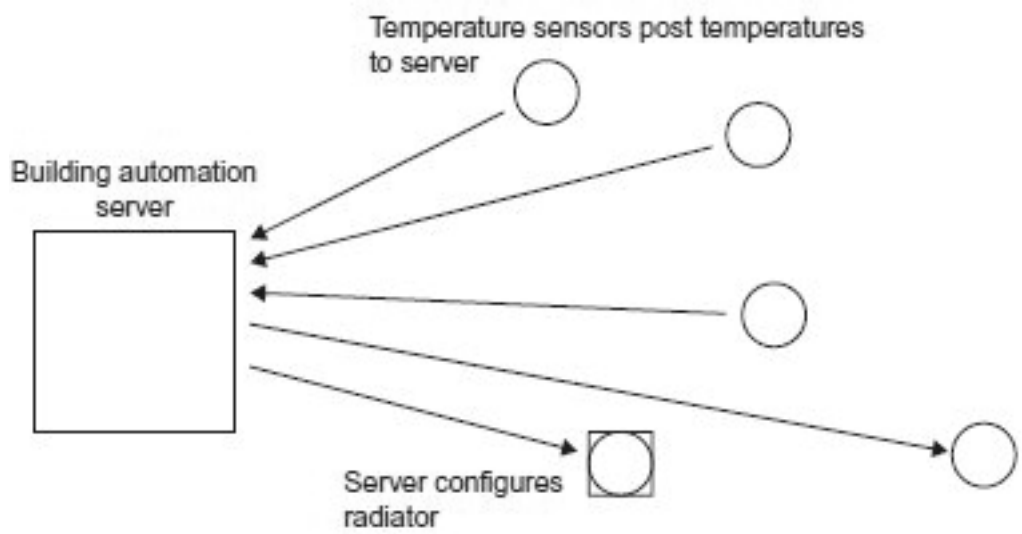
- A smart object system implemented with web services.
- Temperature sensors post temperature data to a building automation server.
- The building automation server configures a radiator based on the temperature data.

XML

```
<xml>
 <sensors>
  <sensor>
    <name>Temperature</name>
    <value>27.1</value>
  </sensor>
 </sensors>
</xml>
```

JSON

```
{"sensors":
 [{"name": "Temperature",
"value": 26.1}]
}
```



Web Services: RESTful API

- A **Web service** communication between two electronic devices over a network.It is a software function provided at a network address
- W3C defines a Web service generally as a software system designed to support interoperable machine-to-machine interaction over a network.
- A widespread class of webservices for IoT is REST-compliant web services
- RESTful systems typically, but not always, communicate over the HTTP using the same HTTP verbsa (GET, POST, PUT, DELETE) which web browsers use to retrieve web pages and send data to remote servers

MQTT

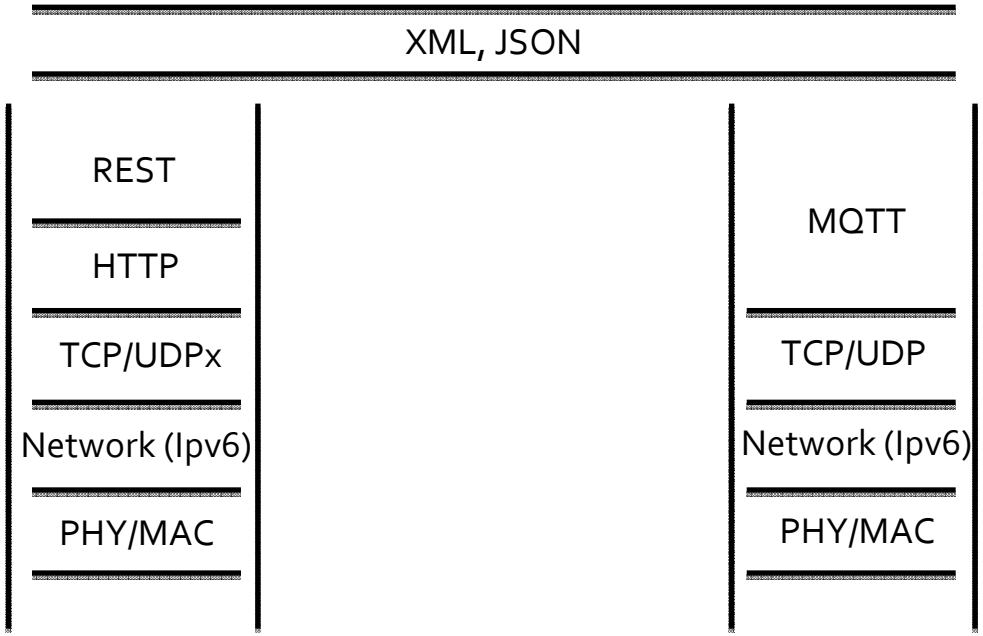
MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium.

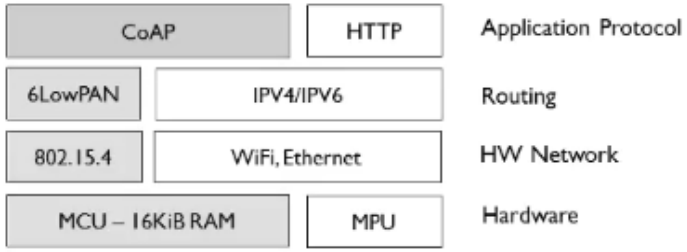
For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios.

It is also ideal for mobile applications because of its small size, low power usage, minimised data packets, and efficient distribution of information to one or many receivers

http://mqtt.org

MQTT, REST/HTTP, CoAP





CoAP is a lightweight version of HTTP for IoT

Why MQTT?

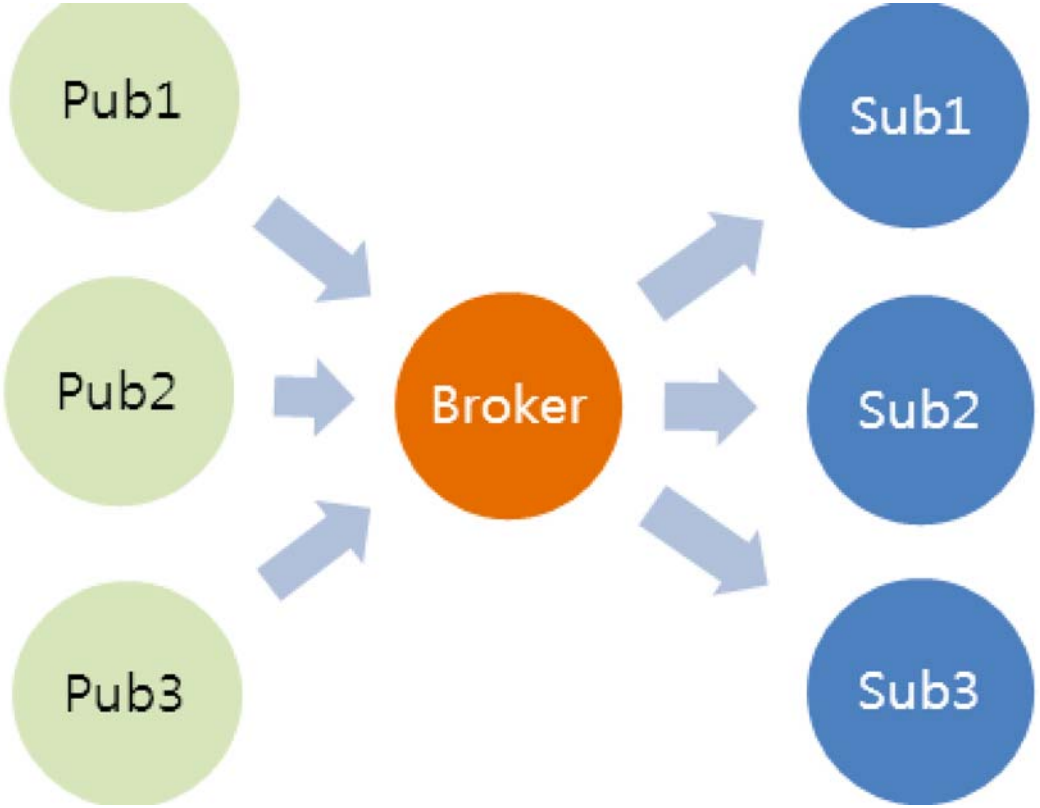
- Provides reliable messaging over unreliable network such as mobile

- Suitable to industrial environment with computationally "challenged" devices, with restricted power due to solar or battery power, on extremely low bandwidth and often brittle RF communications including satellite

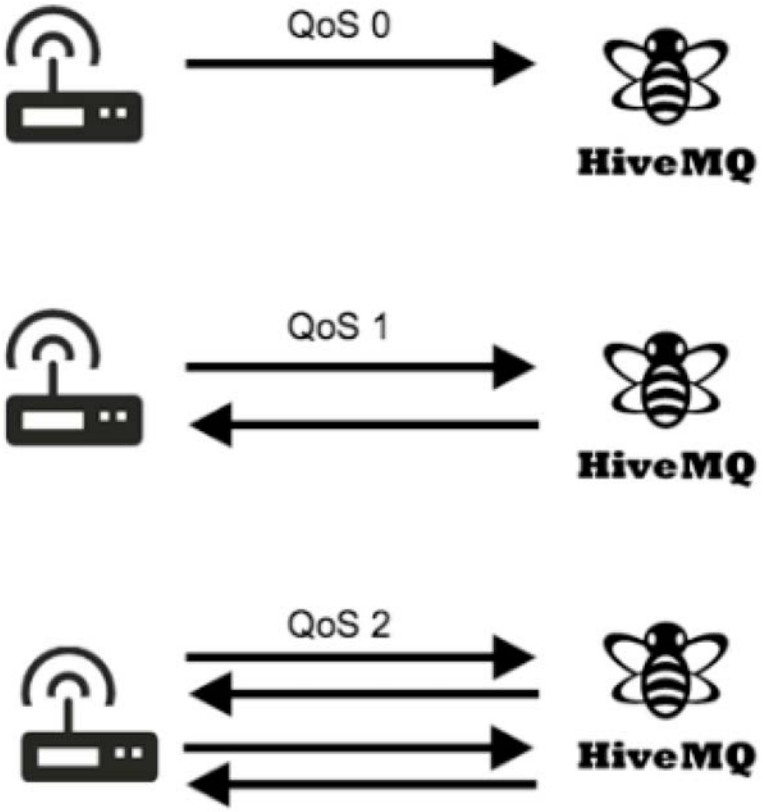
- It follows a pub/sub so to avoid heavy unnecessary workloads and bandwidth consumption due to all the request/response polling-based monitoring

- The protocol enables devices to open a connection, keep it open using very little power and receive events or commands with as little as 2 bytes of overhead.
  - reliable behavior in an unreliable or intermittently connected wireless environments.

Publish/Subscribe Pattern



Publish/Subscribe: QoS



QoS 0

At most once delivery

The message is delivered once or never.

QoS 1

At least once delivery

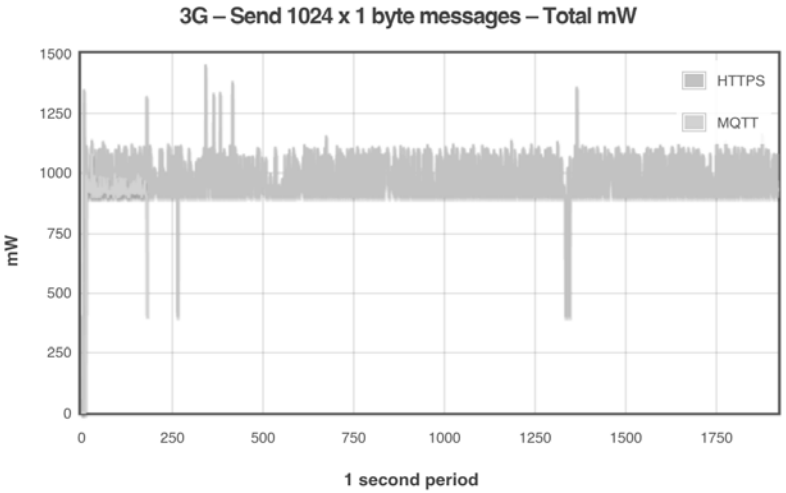
The message is delivered once or more.

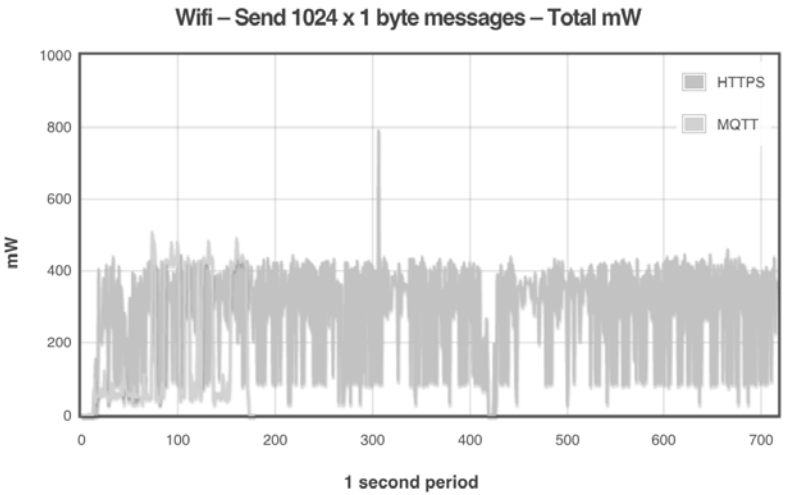
QoS 2

Exactly once delivery

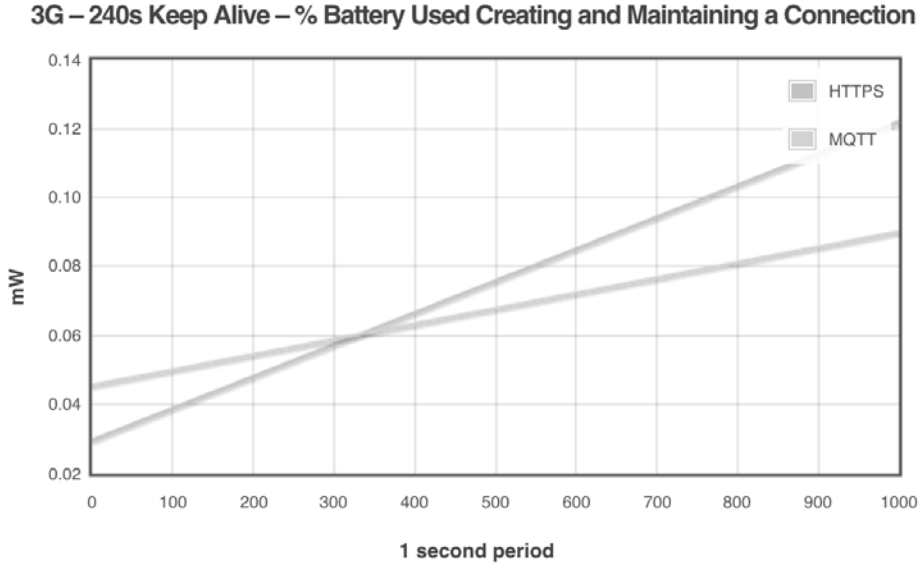
The message is delivered exactly once.

MQTT vs HTTP - Power





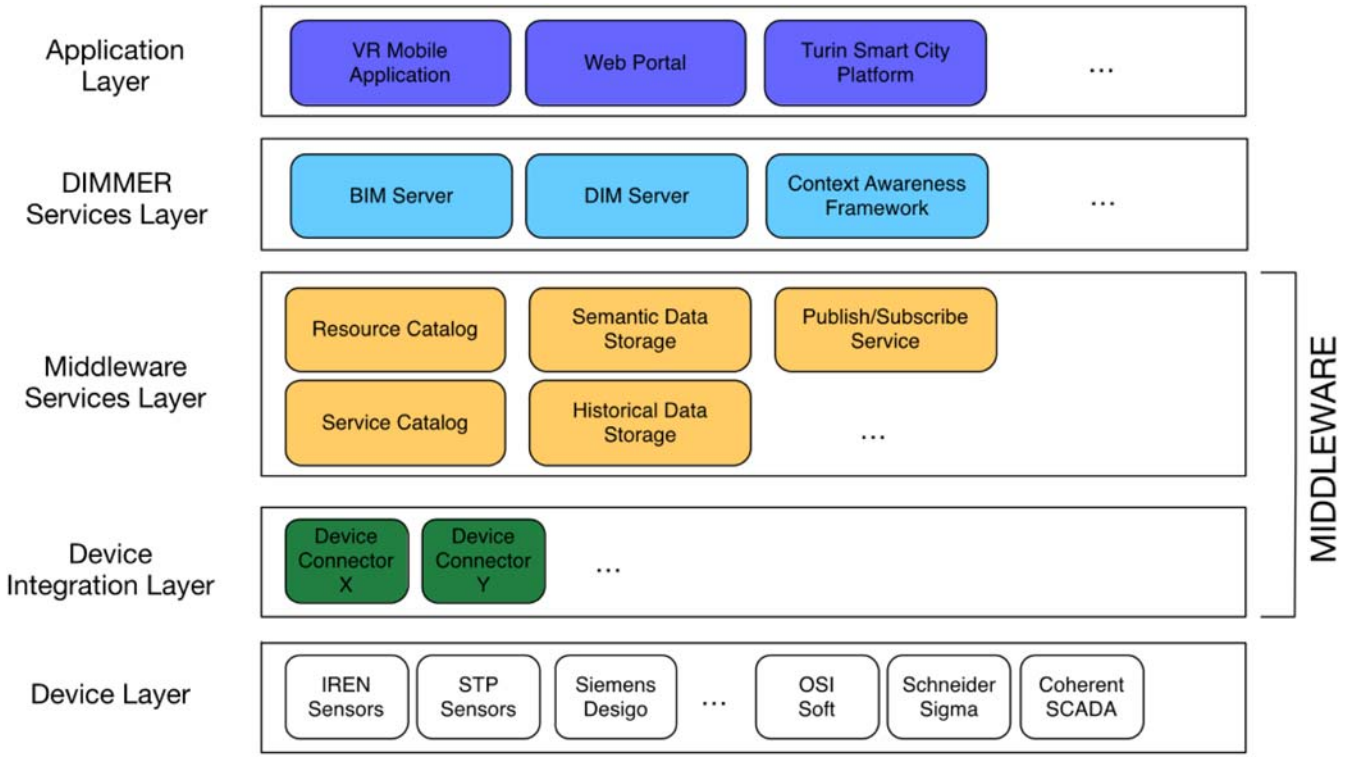
% Battery Used			
3G		Wifi	
HTTPS	MQTT	HTTPS	MQTT
0.34645	**0.27239**	0.04817	**0.00411**



Mosquitto

- Mosquitto provides a lightweight server implementation of the MQTT and MQTT-SN protocols, written in C. The reason for writing it in C is to enable the server to run on machines which do not even have capacity for running a JVM. Sensors and actuators, which are often the sources and destinations of MQTT and MQTT-SN messages, can be very small and lacking in power. This also applies to the embedded machines to which they are connected, which is where Mosquitto could be run.
- Typically, the current implementation of Roger Light's Mosquitto has an executable in the order of 120kB that consumes around 3MB RAM with 1000 clients connected. There have been reports of successful tests with 100,000 connected clients at modest message rates.
- https://www.eclipse.org/proposals/technology.mosquitto/

Middleware Services: an example



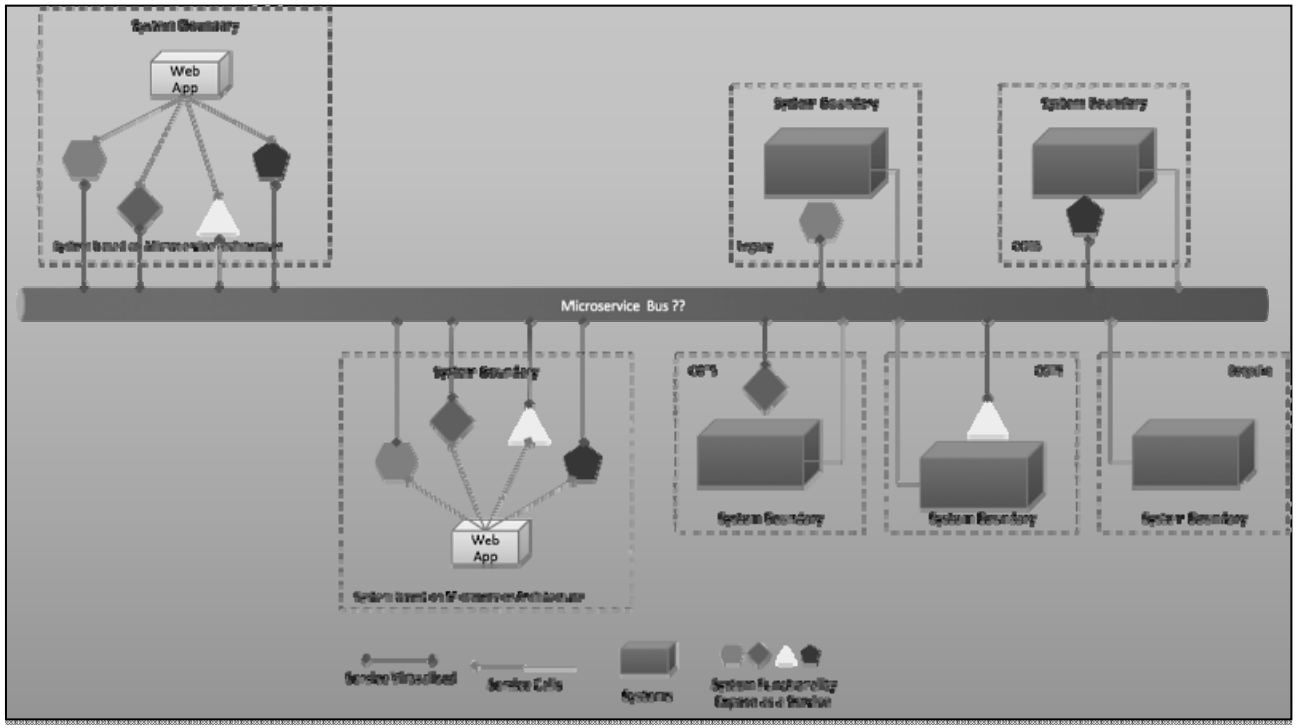
© DIMMER consortium

Micro-services

Micro-service architecture:

•the system is decomposed into a collection of independent and specialized services each of which exhibits its own interface to access to other services of the system

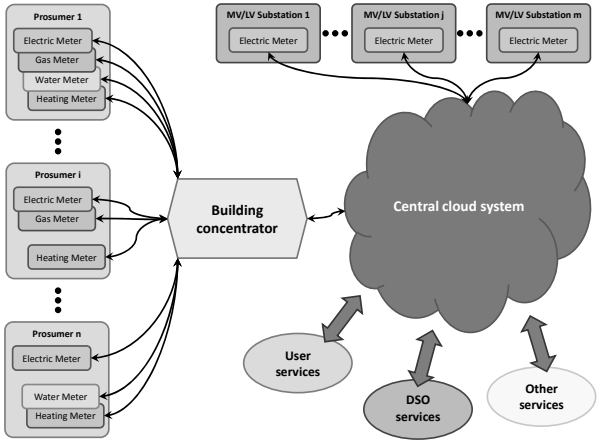
•Each service can be managed independently for deploy / start / stop operations.



Benefits of micro-services

- Unlike monolithic applications, micro-service-based applications can be selectively scaled out.
- Since each micro-service is autonomous and independent, it is easy to monitor and replace a faulty service without impacting any other.
- The micro-service development can be done in self handling application and deployed in the cloud area.
- Each micro-service can be updated without impacting the other services.

Example application: Smart Buildings



© FLEXMETER consortium

- The thing here is the power plug that becomes a "smart plug"
- Data are processed in the CLOUD and visualized in A/R or V/R through context aware technologies on a mobile device

© SEEMPubS consortium

Design Environments

- IBM Bluemix
- The thing box