BETaaS: an Open Framework for Accessing Things as a Service

Enzo Mingozzi
University of Pisa, IT

IEEE IoT-SoS – Sydney, June 16, 2014

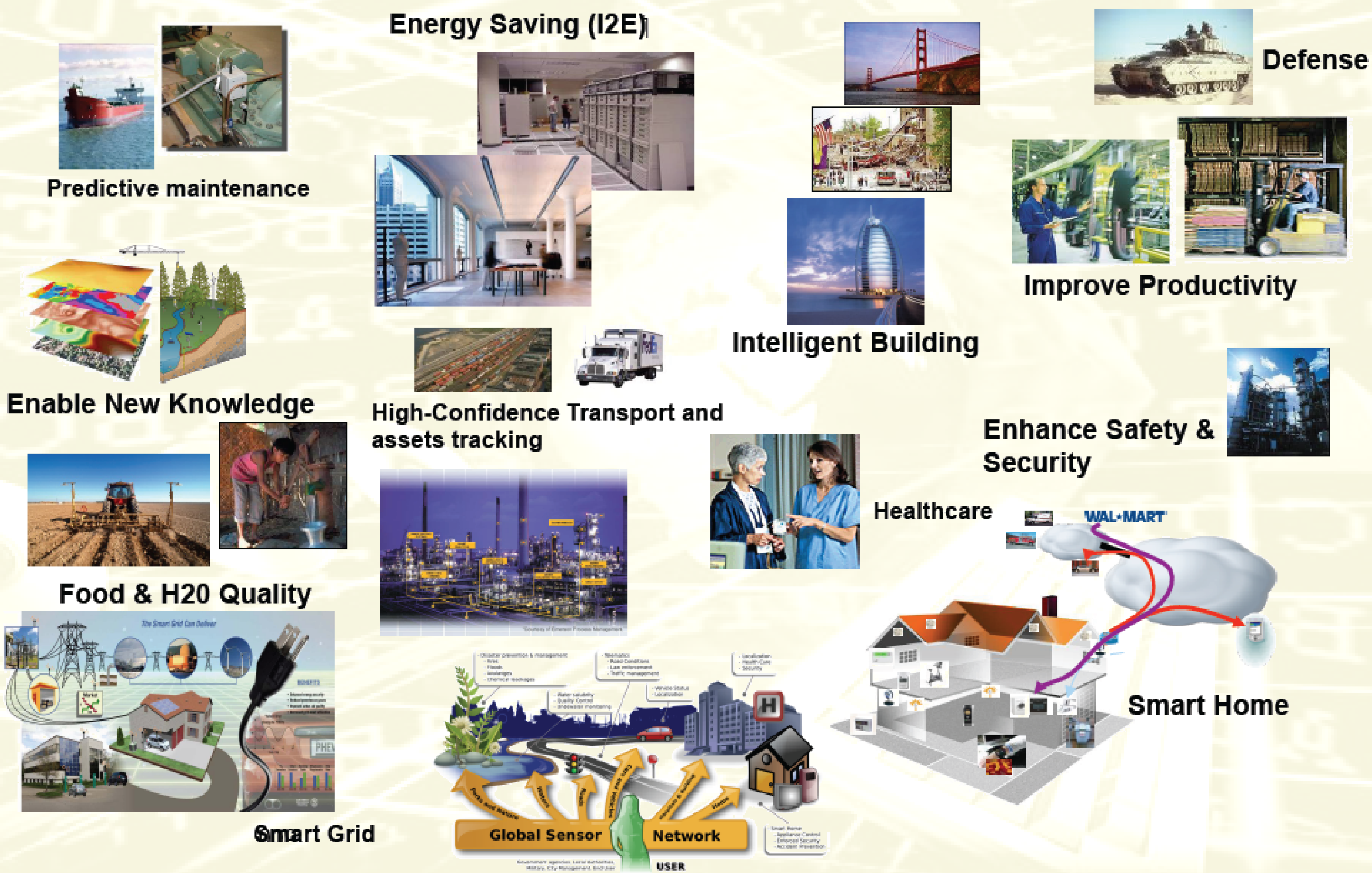
IoT: a visionary paradigm

"The next logical step in the technological revolution connecting people anytime, anywhere is to connect inanimate objects. This is the vision underlying the **Internet of things: anytime, anywhere, by anyone and anything**" – ITU, Nov. 2005

Each object can be addressed

Objects can be linked and communicate

New opportunities …

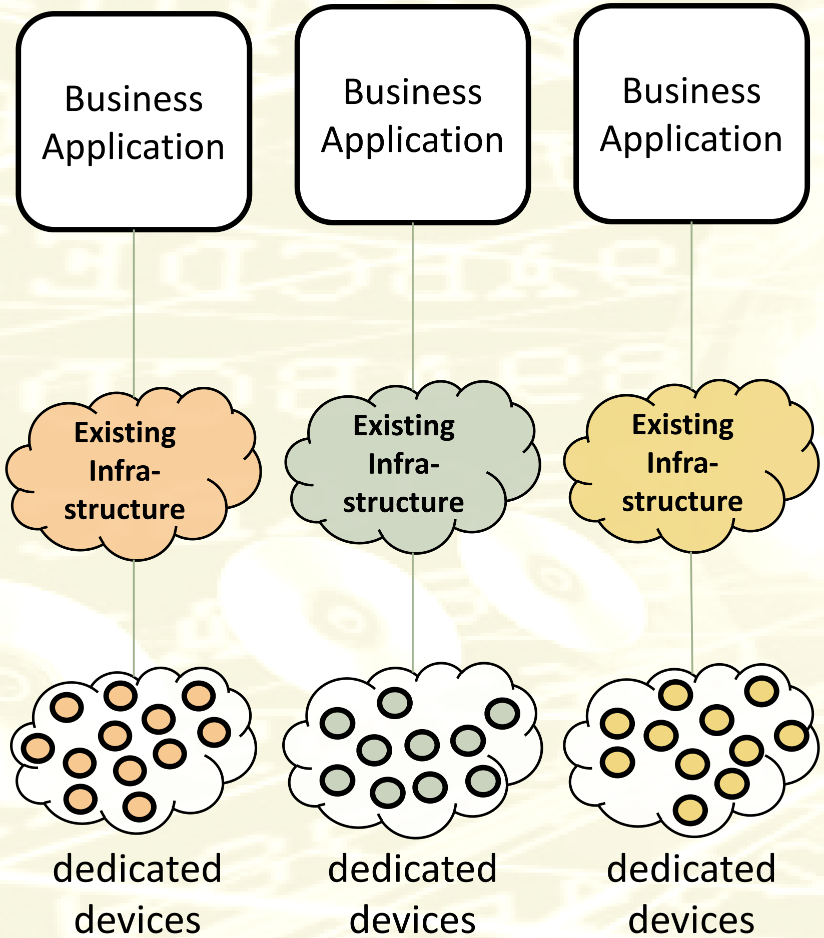


… but also new challenges

- Scalability
  - Number of nodes in the system
  - Amount of data generated by each node
- Diversity of applications
- Diversity of communication technologies
  - Potentially lossy if wireless
- Interoperability
- Low-power consumption
- Lifetime
- Context-awareness
- Security, trust
- …

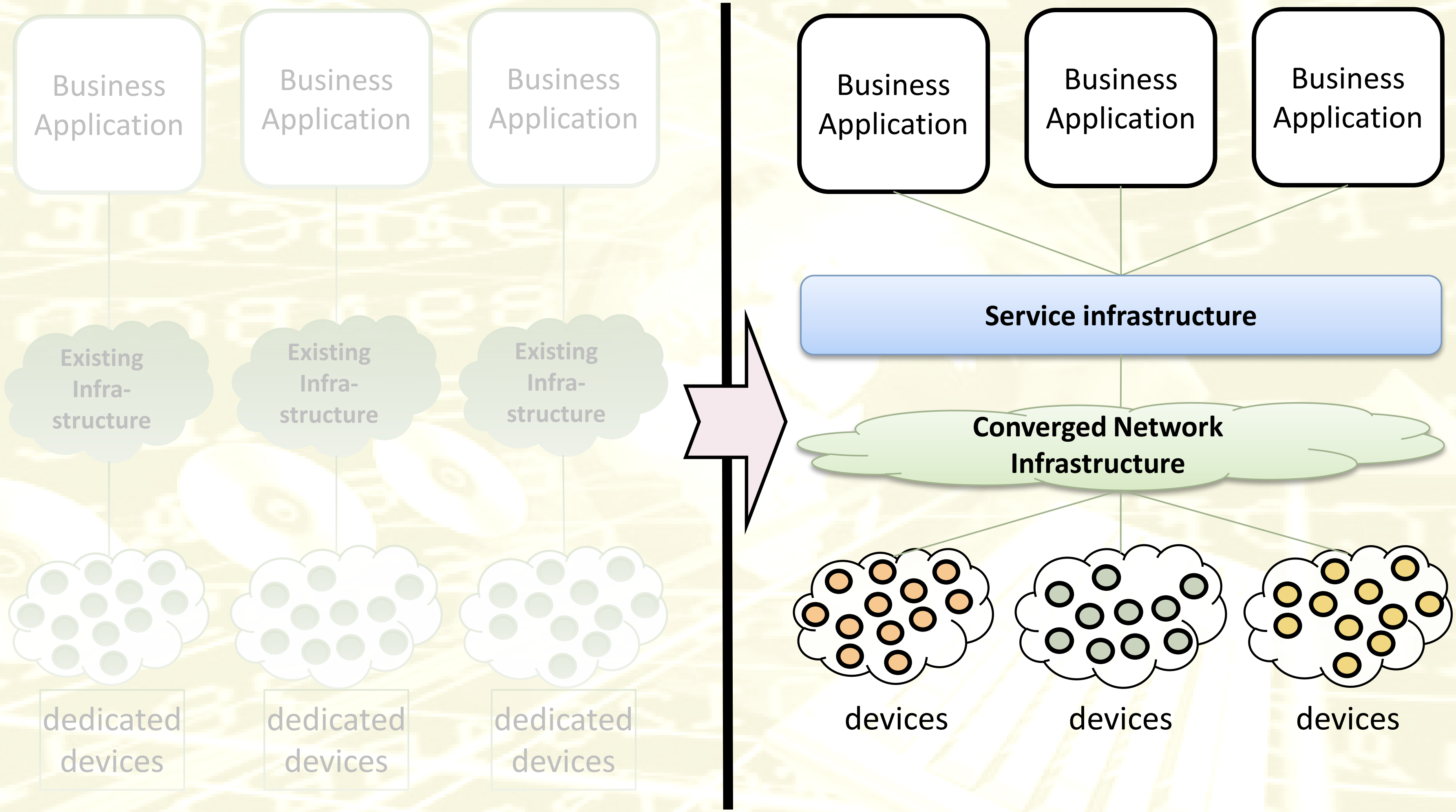
Still a vision, or real already?

The present: ‘verticals’

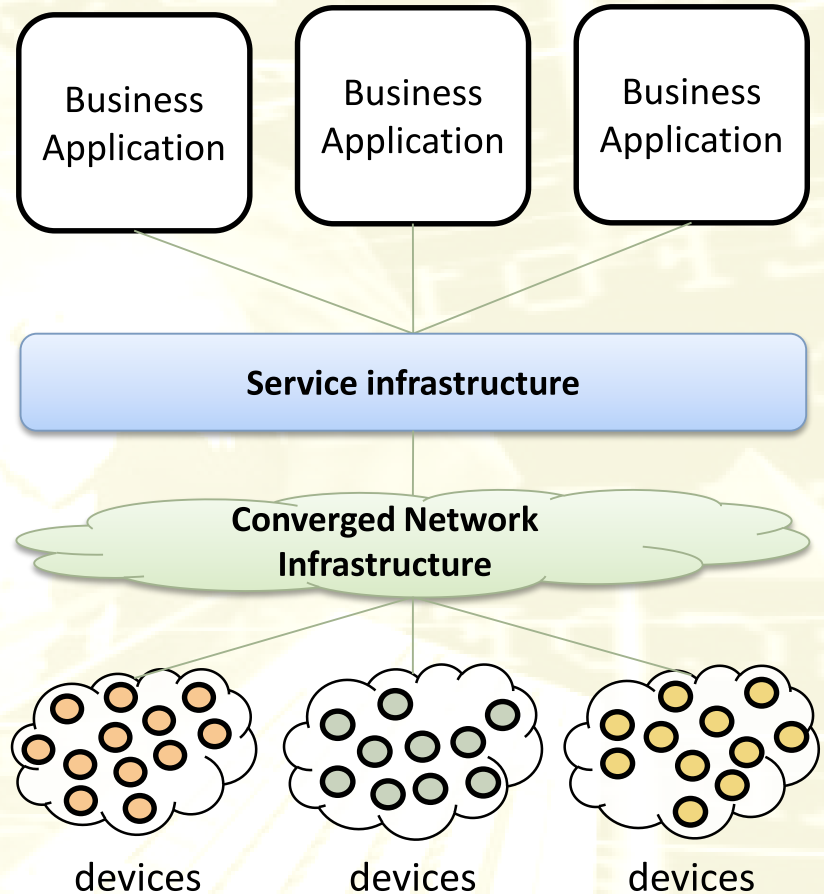


- Operate in isolation: no (or very limited) cooperation
- Inefficient: each device is dedicated to a single application
- Do not scale well

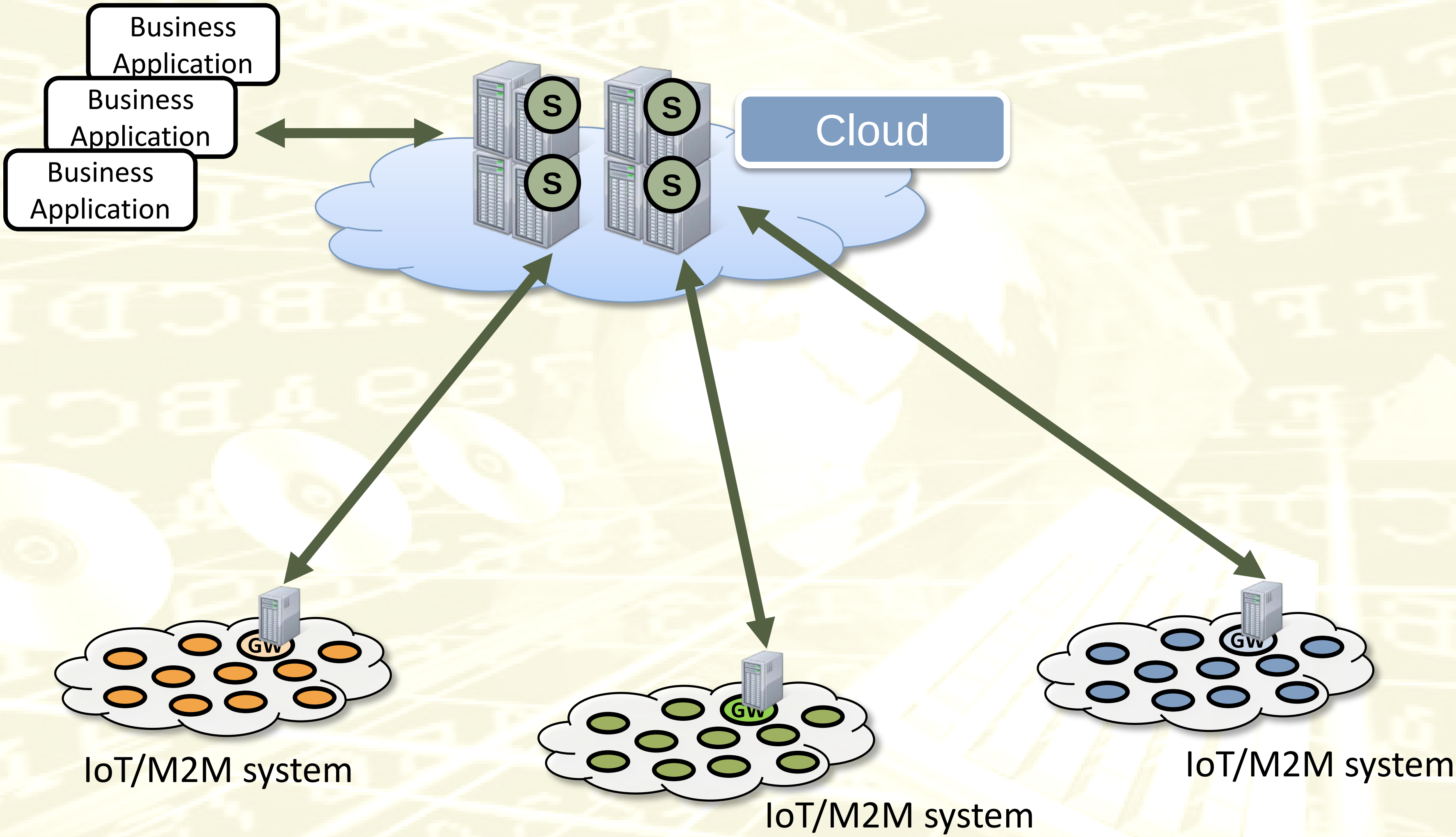
The future: go horizontal



The future: go horizontal

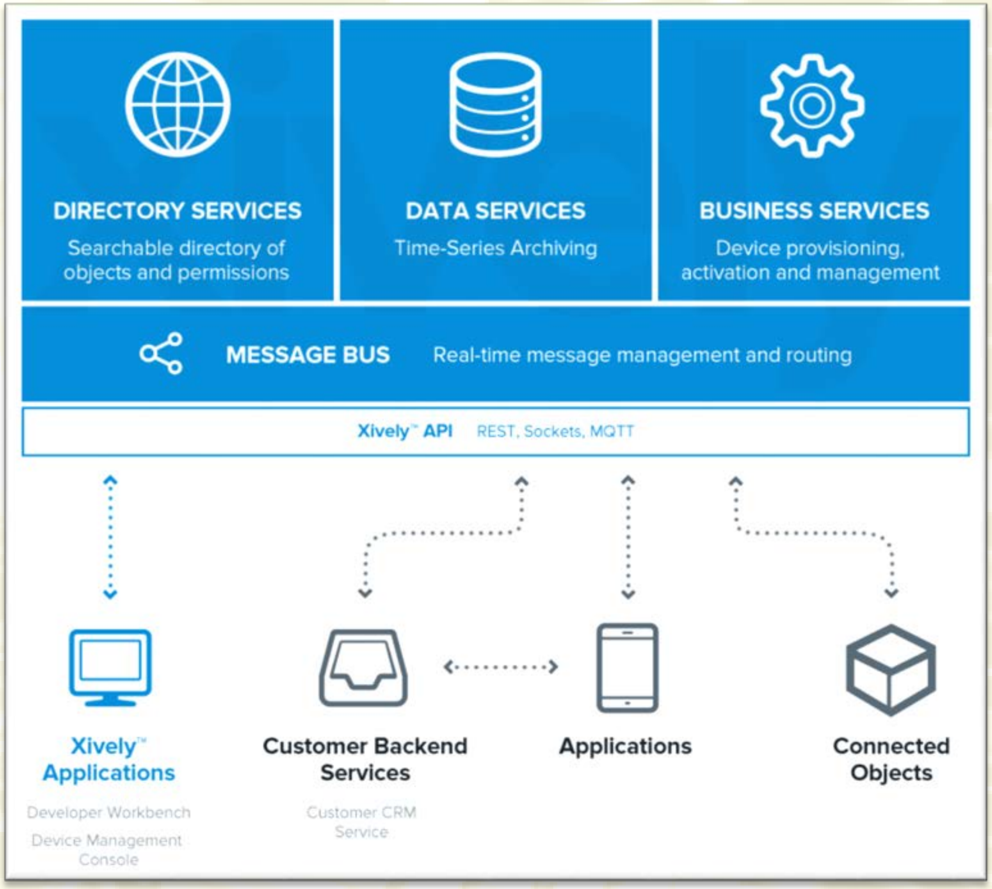


The present: centralized

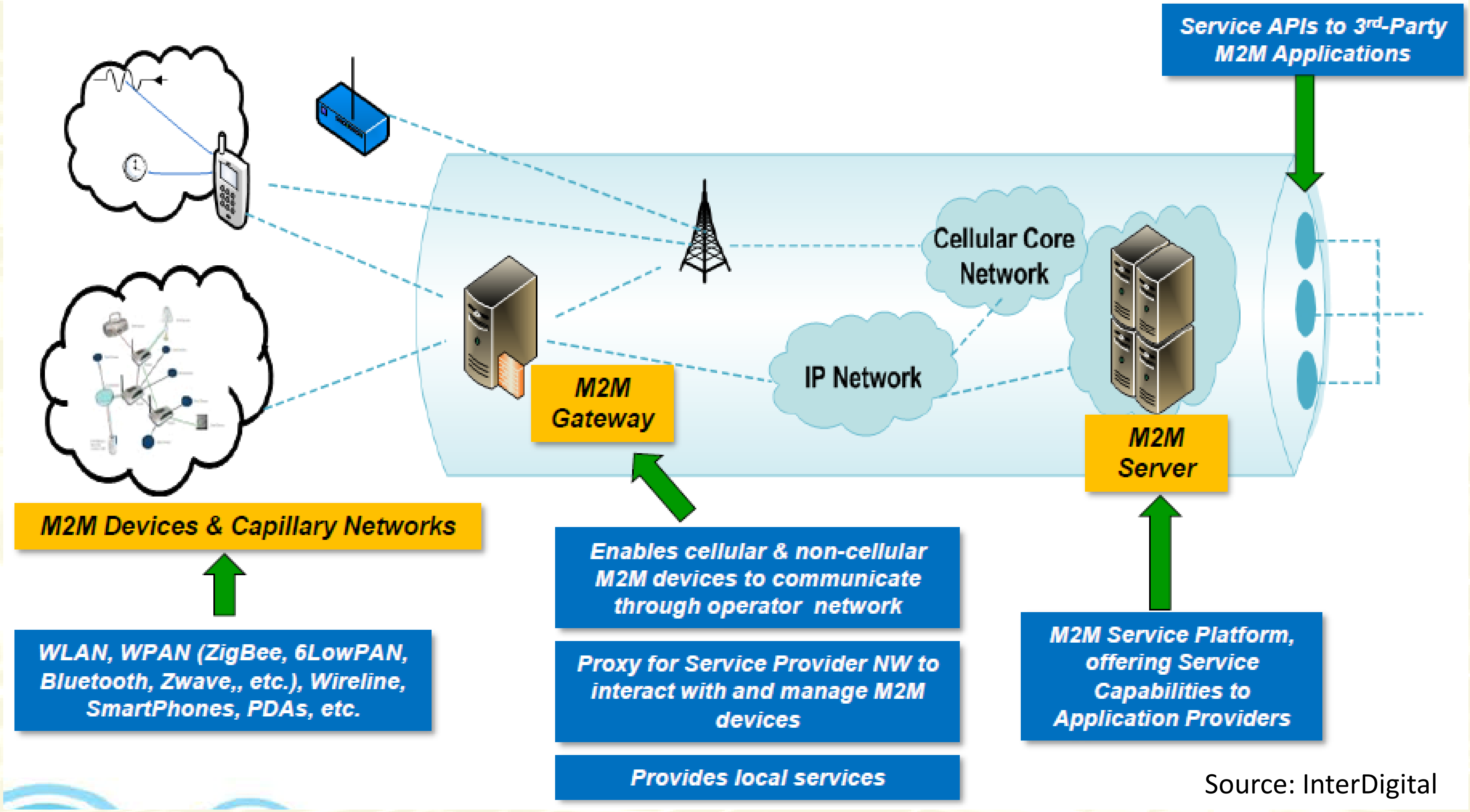


Pachube (now Xively)

- PaaS providing tools and services for IoT products development

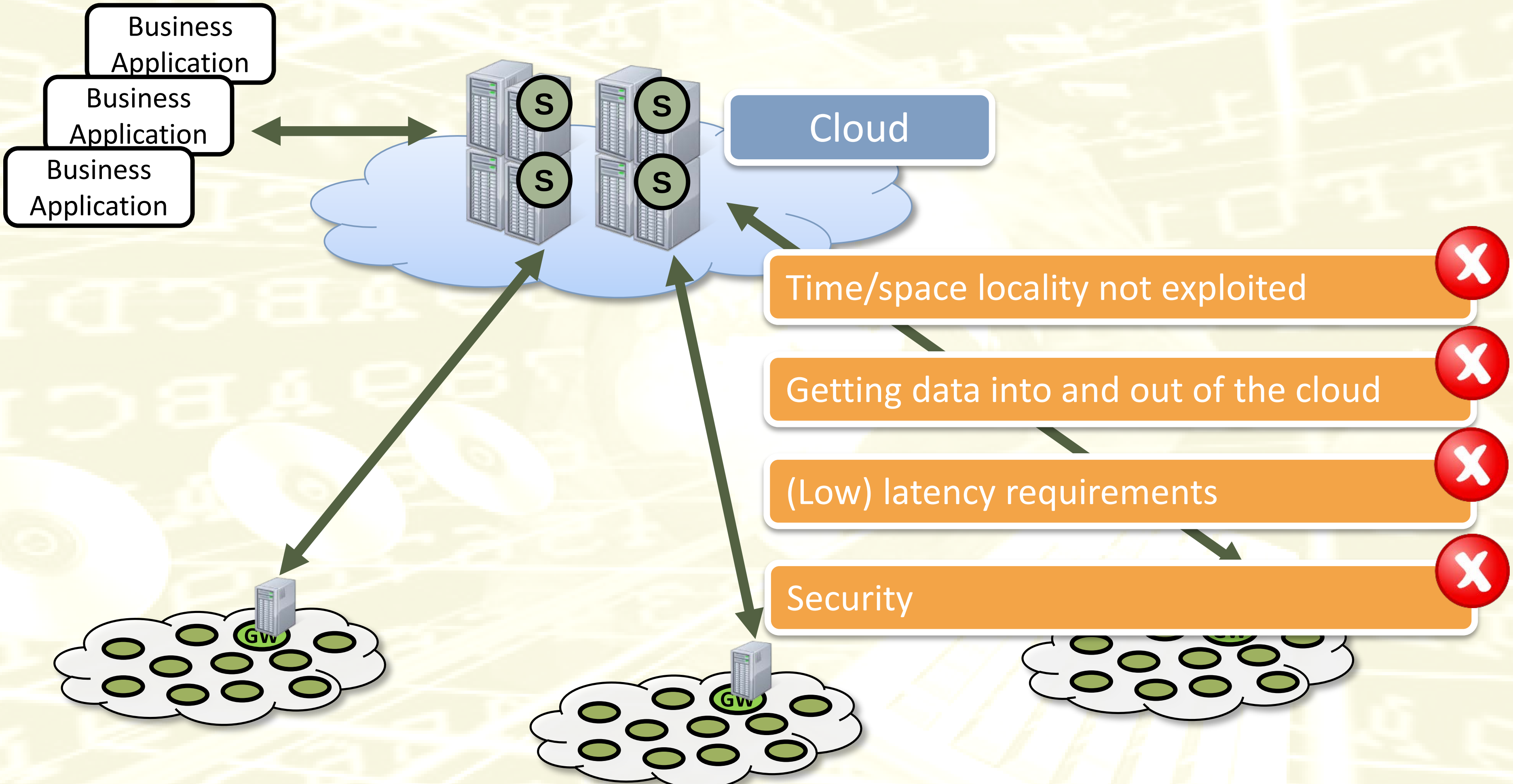


ETSI M2M



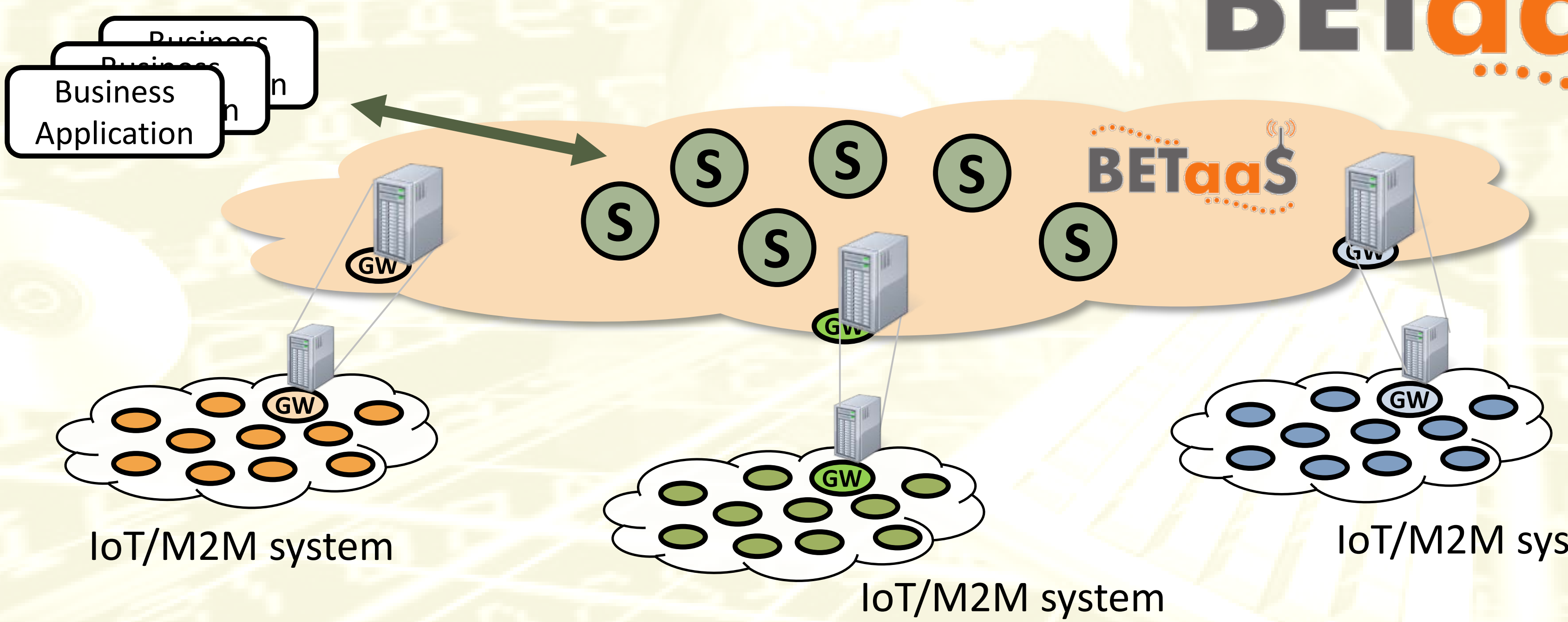
Source: InterDigital

Is the cloud always appropriate?



BETaaS approach

- **Move/distribute the intelligence to the edge!!!**
  - *BETaaS gateways*: Networking devices, set-top boxes, RSUs , …



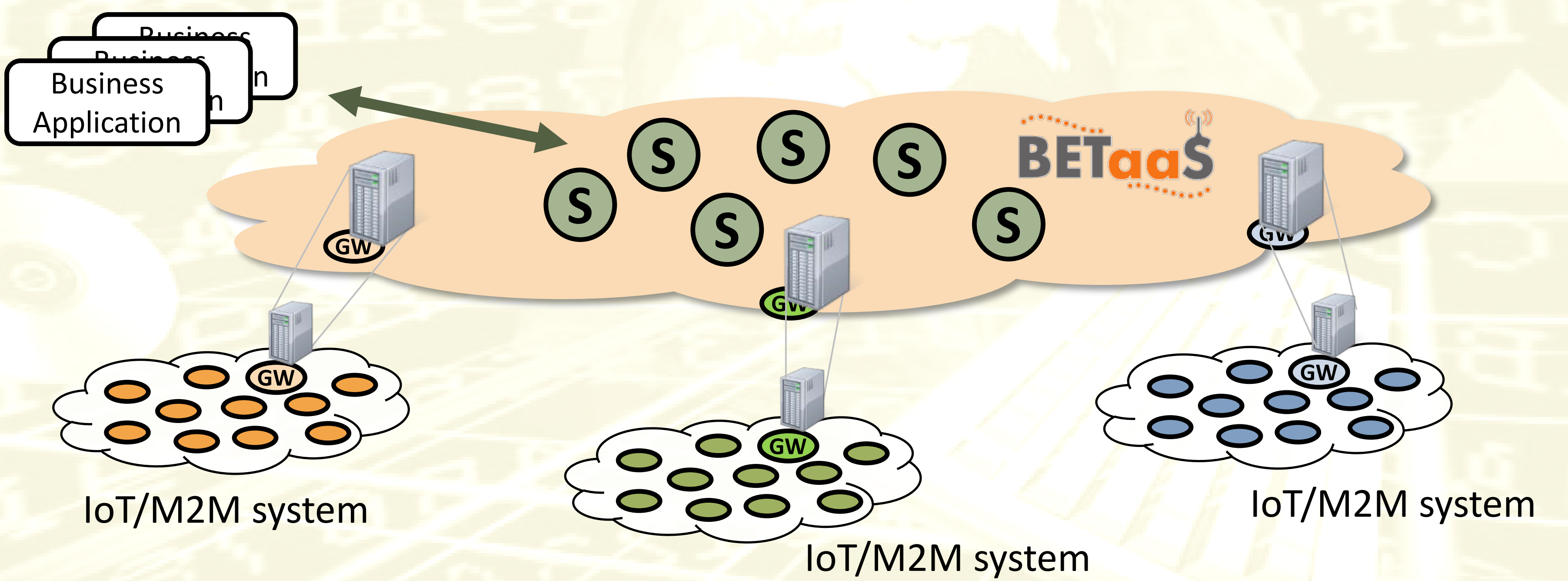
BETaaS approach

- **Move/distribute the intelligence to the edge!!!**

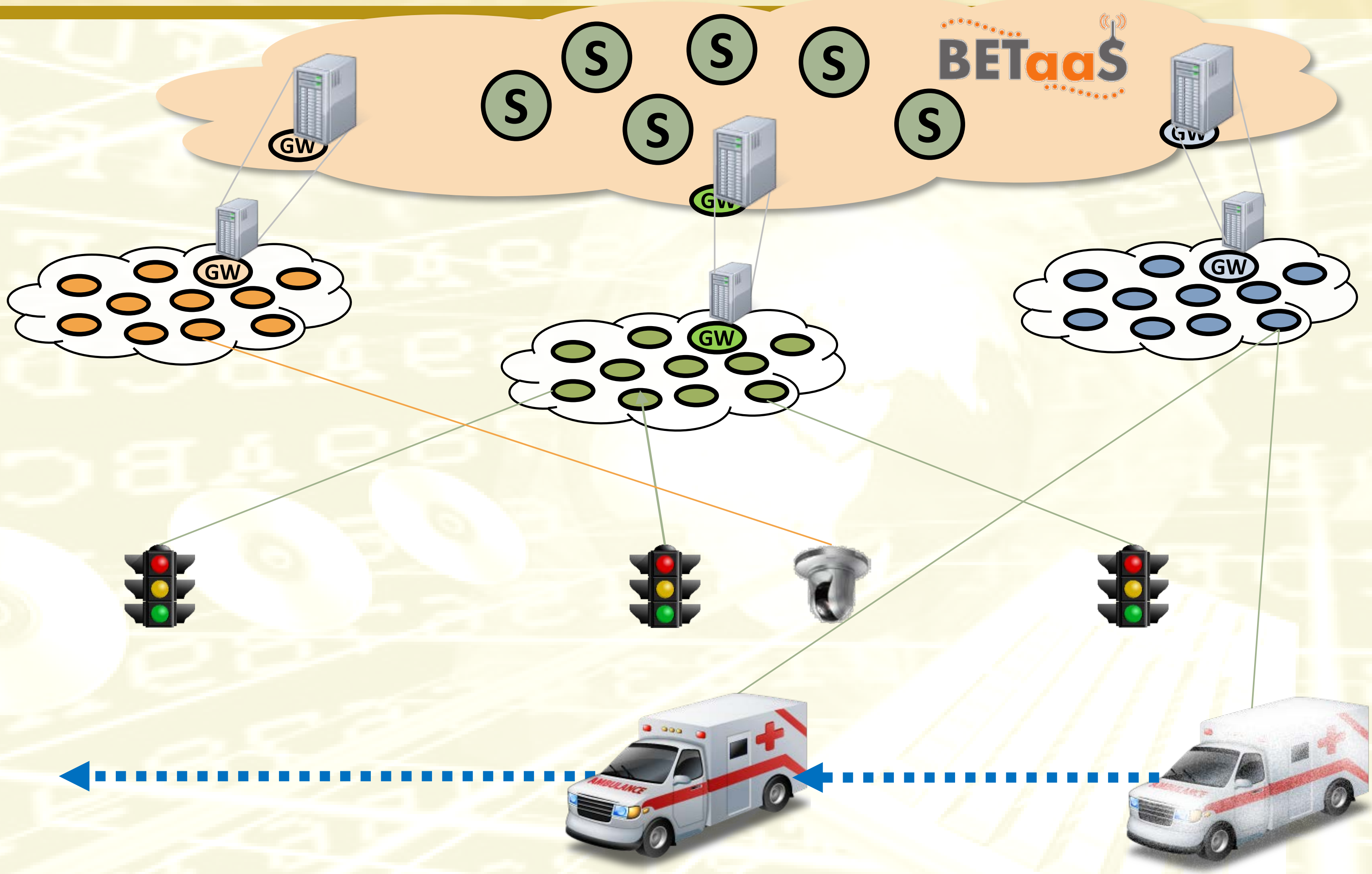
Data storage and processing **close** (in space and time) to where it is generated

Reduced latency

Resource pooling/optimization

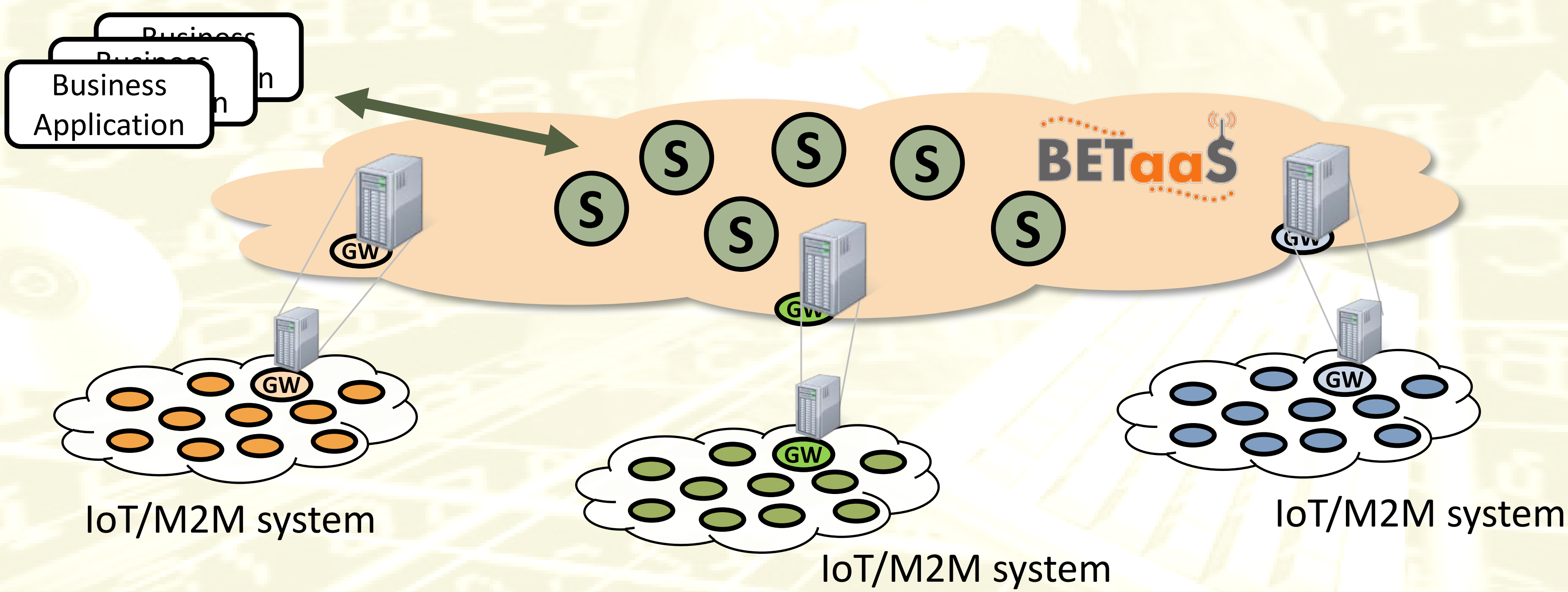


Example



“Local cloud” of gateways

- The set of computational resources hosting the BETaaS runtime environment



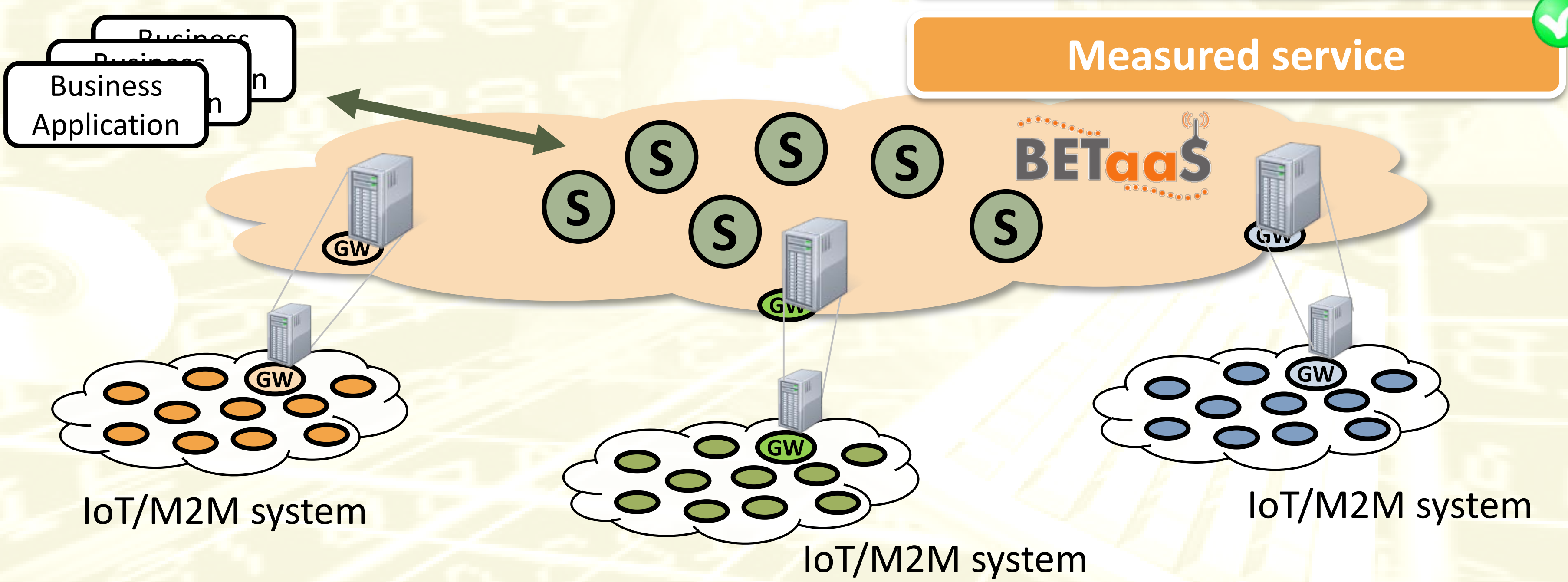
“Local cloud” of gateways

- The set of computational resources hosting the BETaaS runtime environme

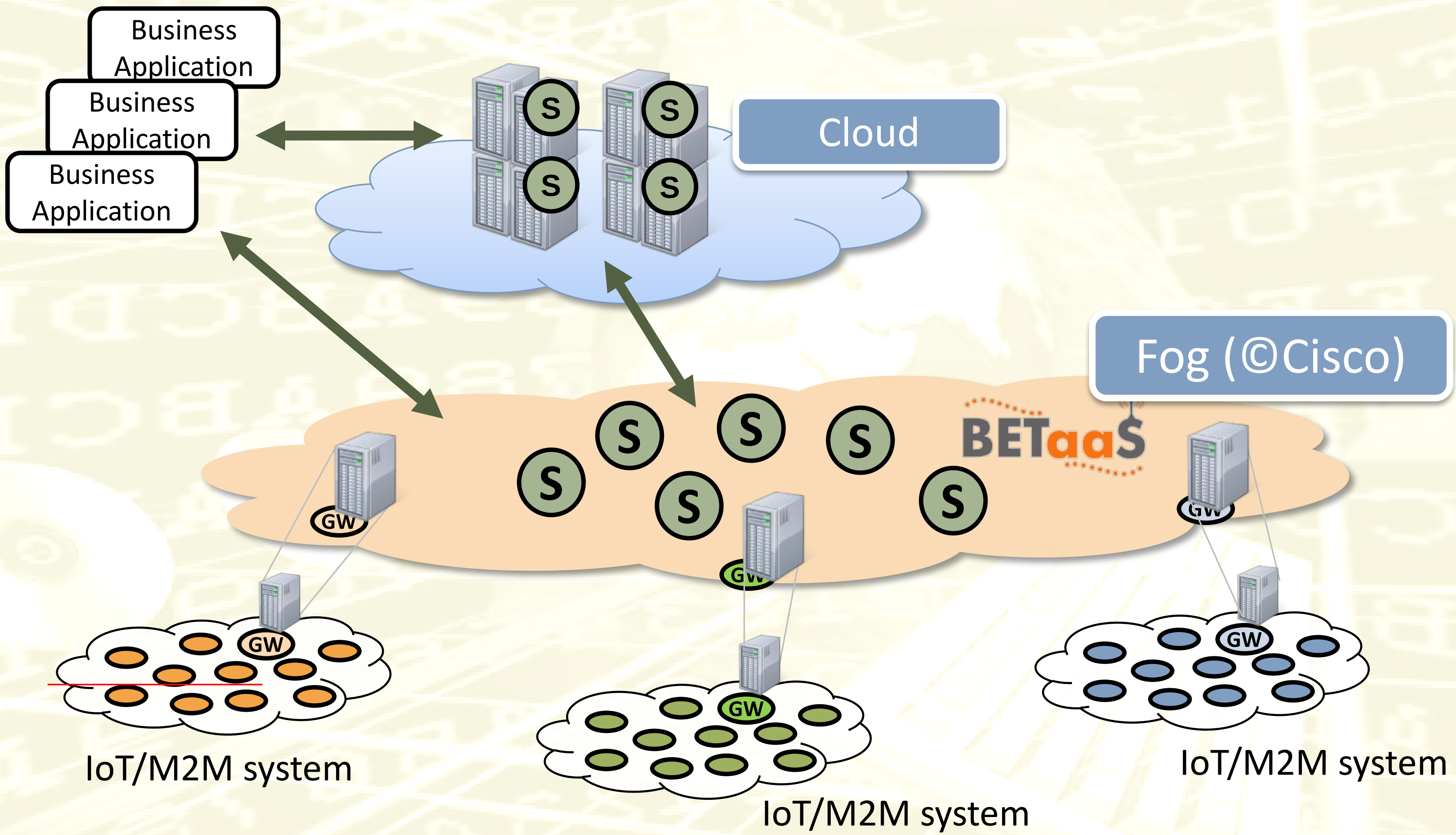
Resource pooling

Rapid elasticity

Measured service



Fog + Cloud computing



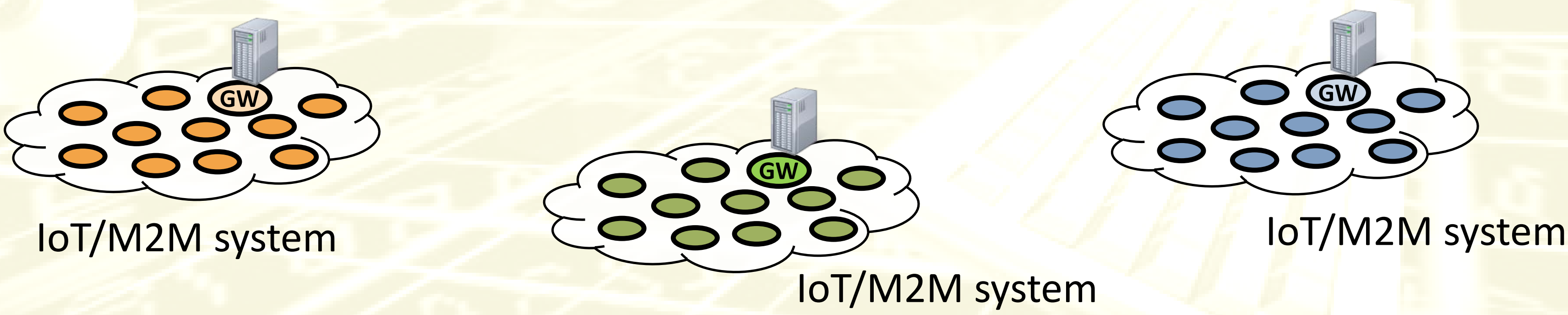
The BETaaS framework

- A reference framework enabling interoperable (**horizontal**) M2M application deployment
  - A **distributed runtime environment** based on a "local cloud" of gateways for the services to be deployed so as to fulfill high-level M2M applications
  - Content-centric Things-as-a-Service layered model
  - Built-in support for non-functional requirements (QoS, big data management, dependability, security)
  - Seamless integration of existing IoT/M2M systems

- Work carried under the EU FP7 project "***BETaaS: Building the Environment for the Things as a Service***"

Existing IoT/M2M systems

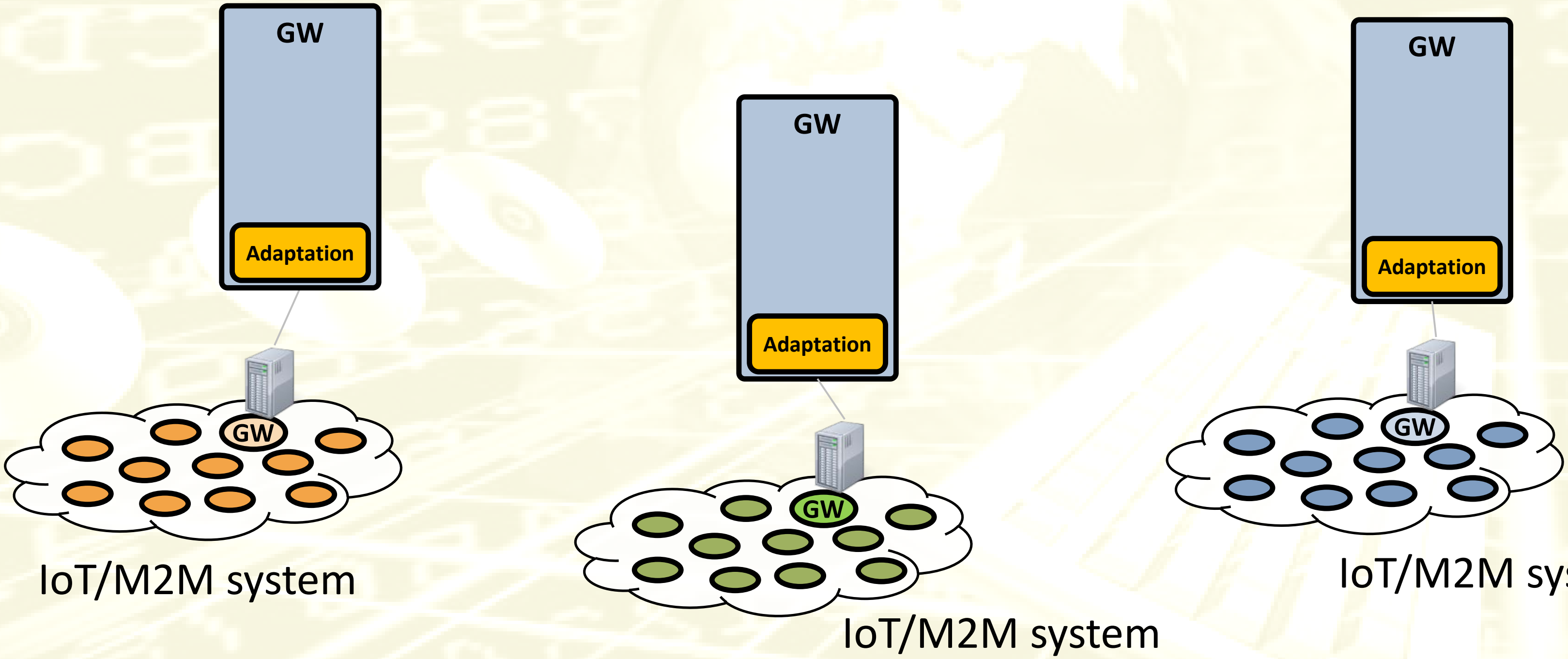
- Heterogeneous physical devices and protocols
- Several data formats and structures
- No common semantic for resource description



Enable integration

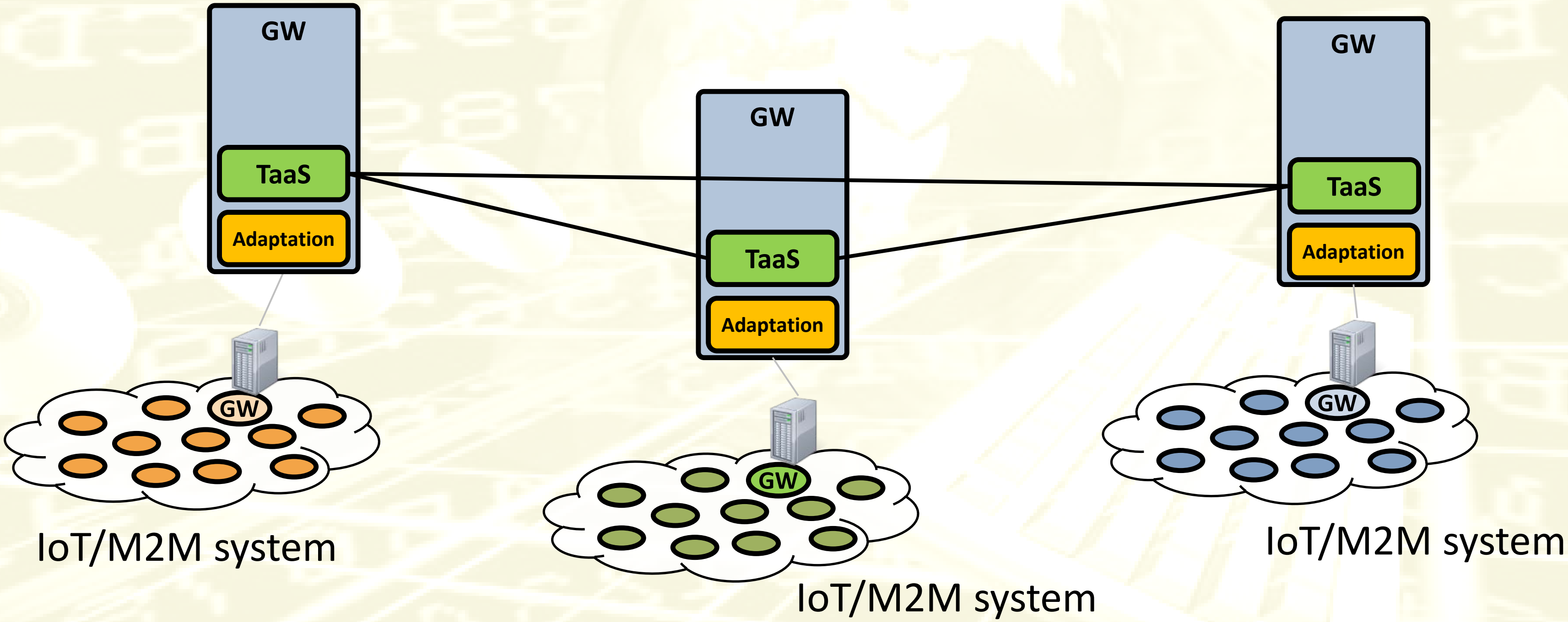
Standardize access through a common interface and data representation

Guarantee that a basic set of functionalities is provided by the plugged-in IoT/M2M system



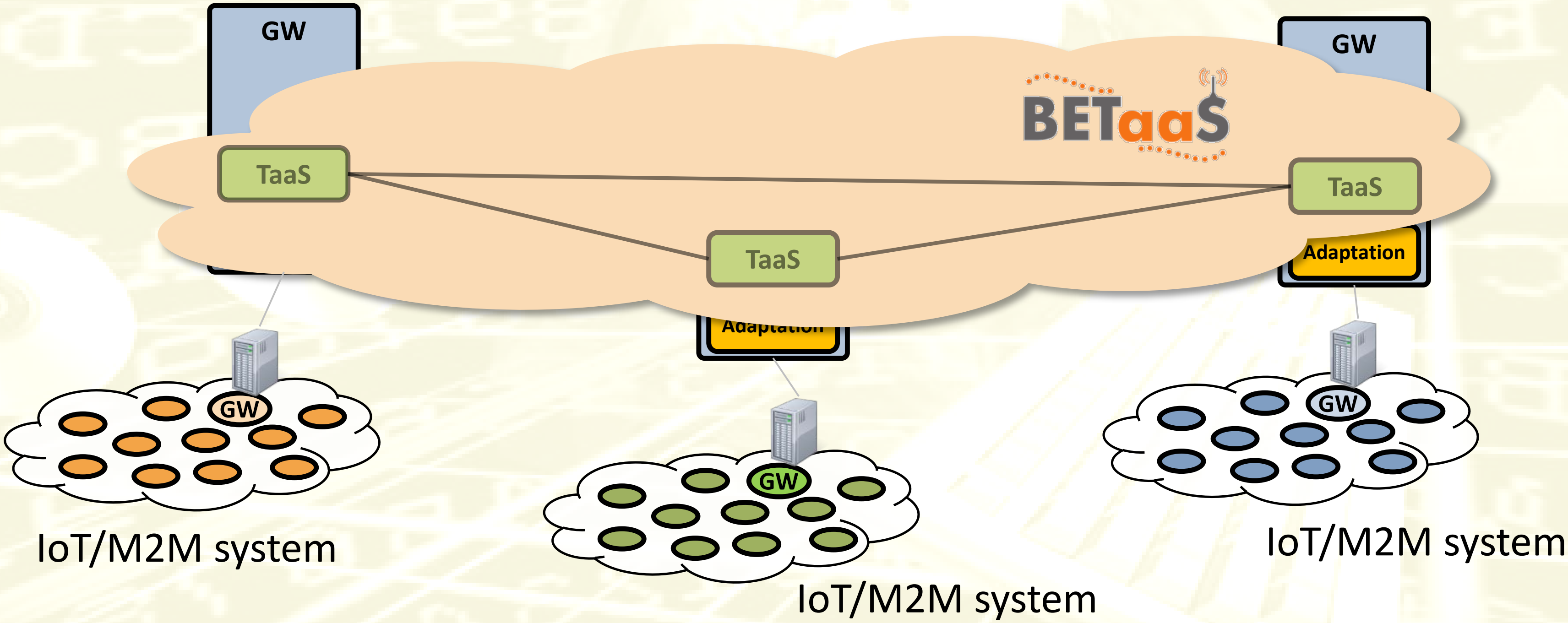
Realize integration: TaaS model

- Seamless service-oriented access to things irrespectively of the location
- Common semantics to enable context-aware lookup
- Support for non-functional requirements (QoS, security, dependability)



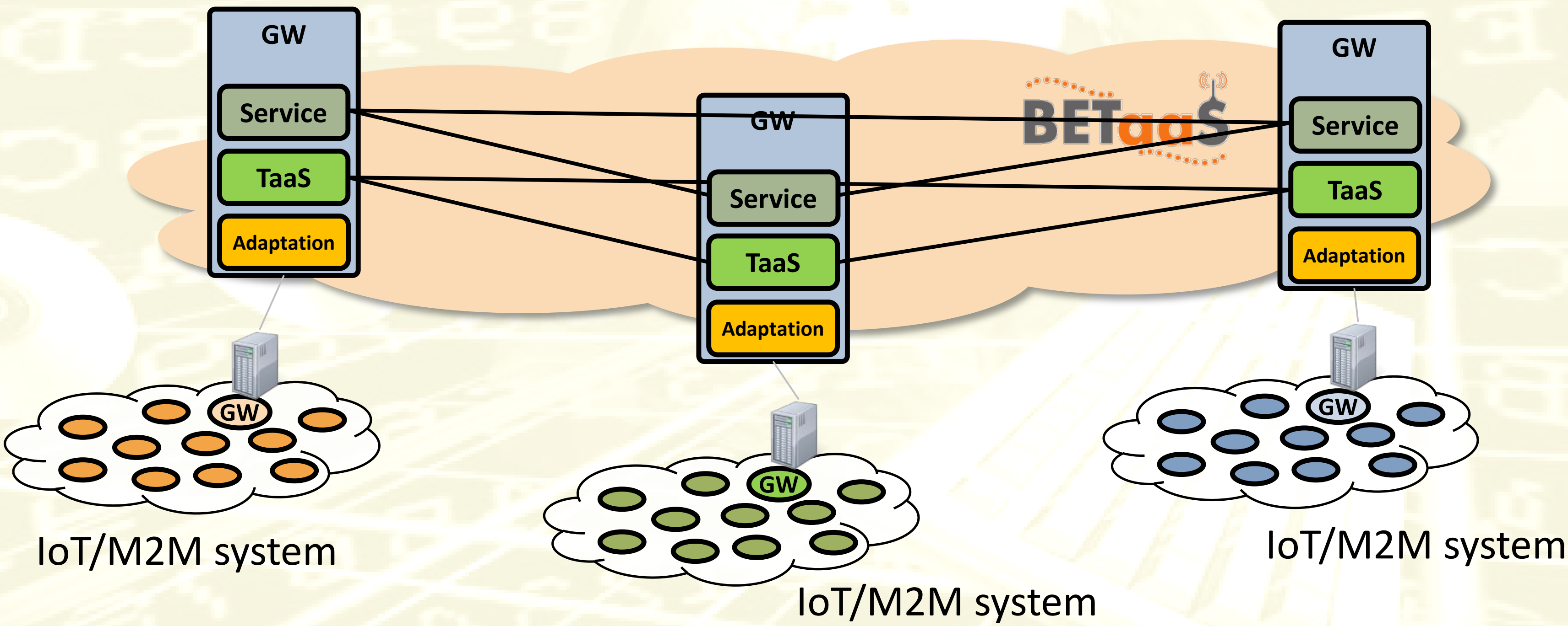
Realize integration: TaaS model

- Seamless service-oriented access to things irrespectively of the location
- Common semantic to enable context-aware lookup
- Support for non-functional requirements (QoS, security, dependability)

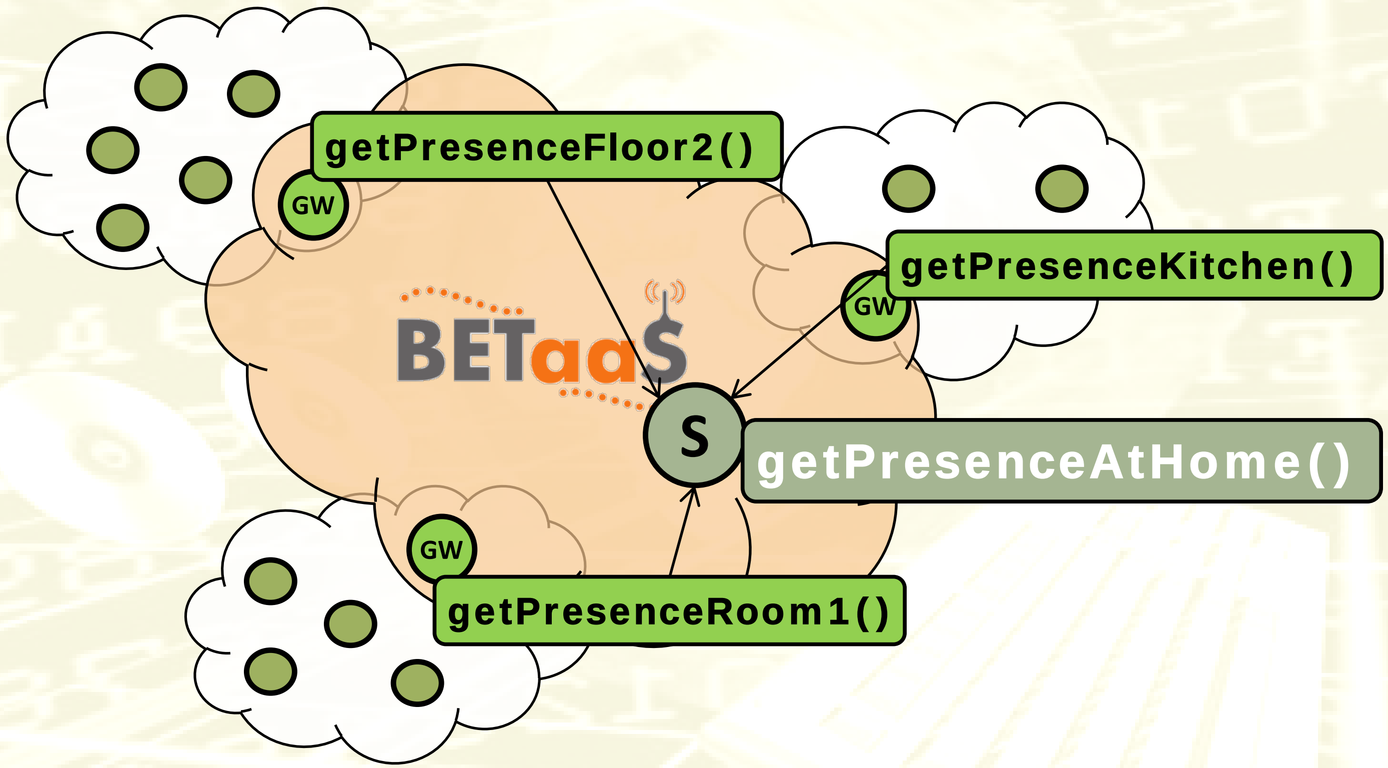


M2M service deployment

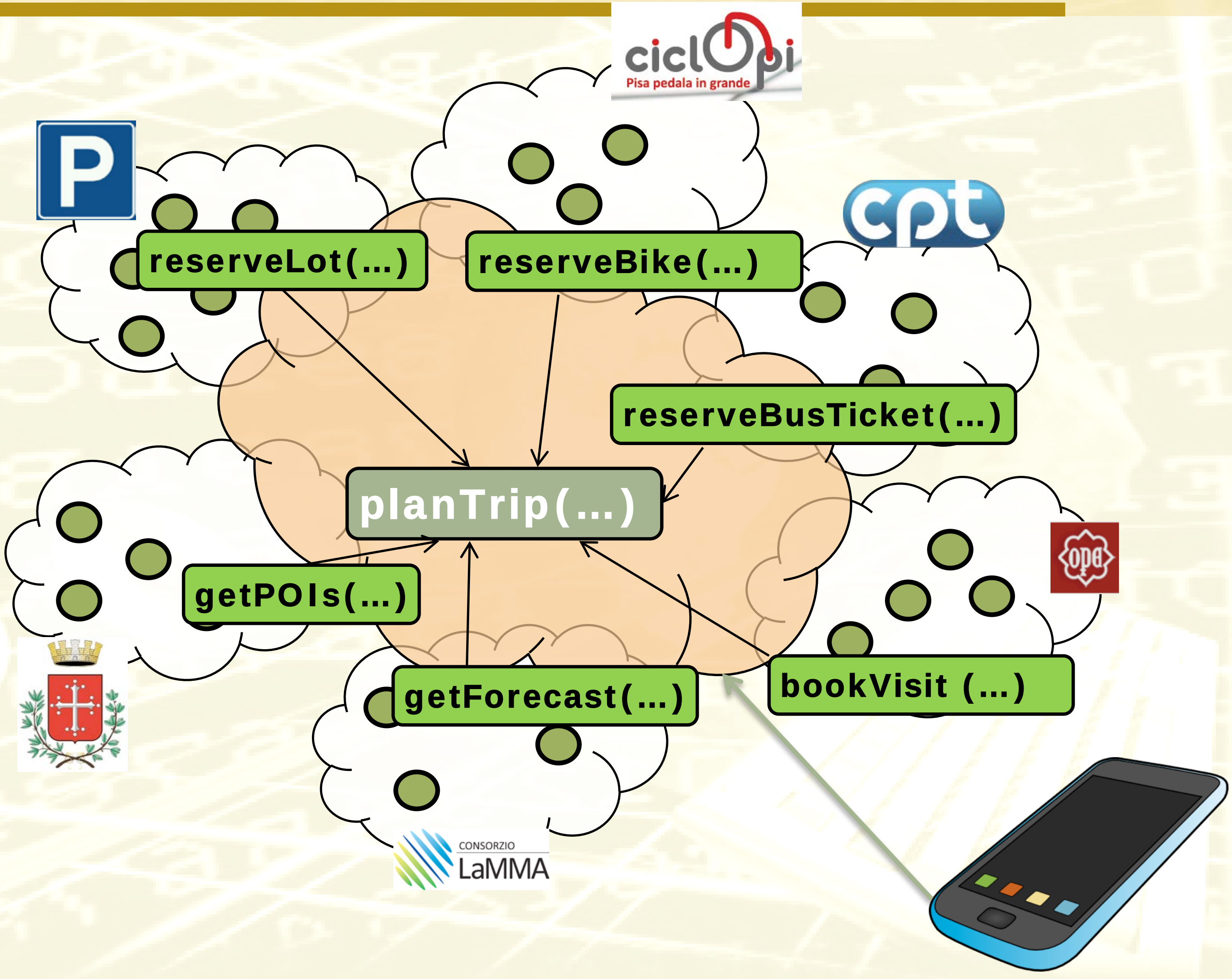
Manage M2M services built on top of TaaS



Basic M2M services

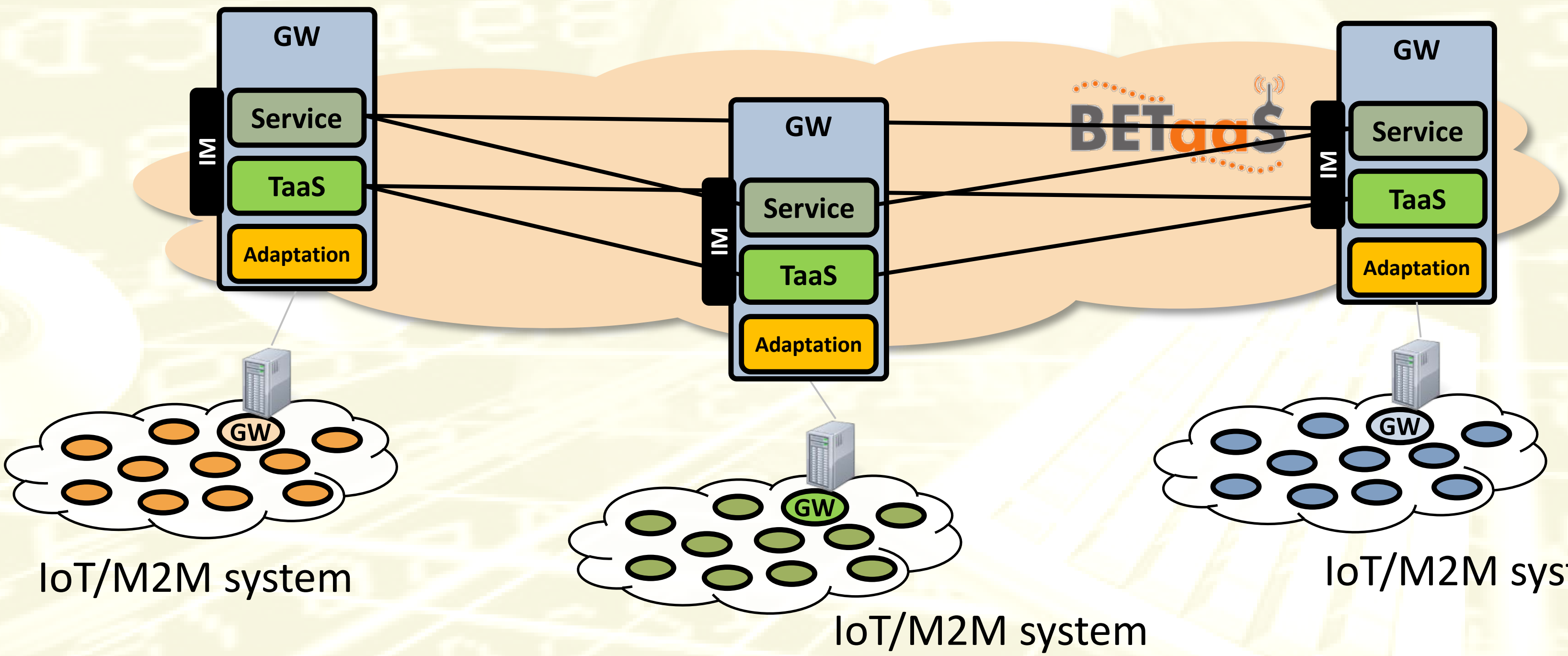


Extended M2M services

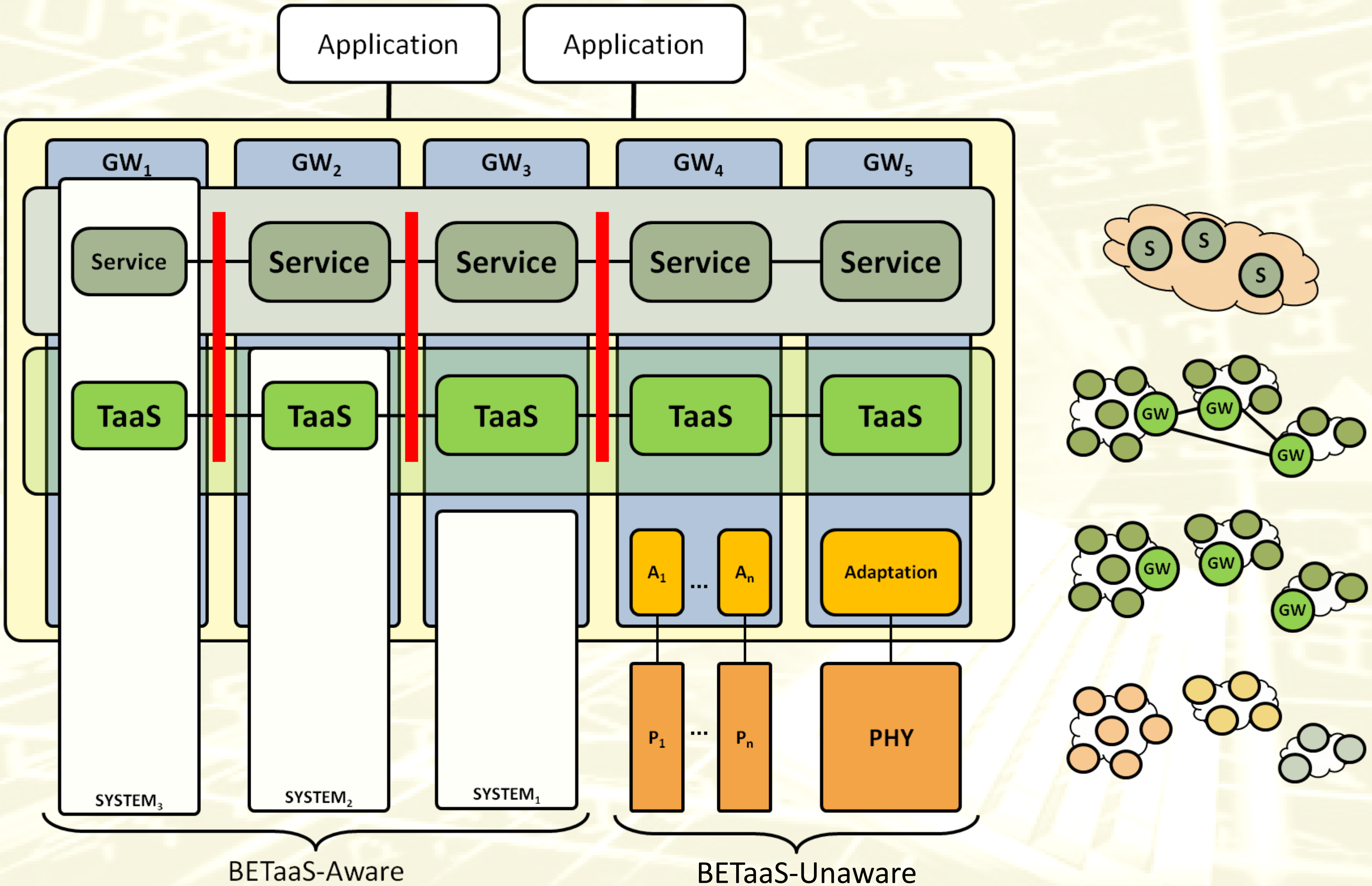


BETaaS "instance"

BETaaS instance management (GW join/disjoin, GW discovery, etc.)

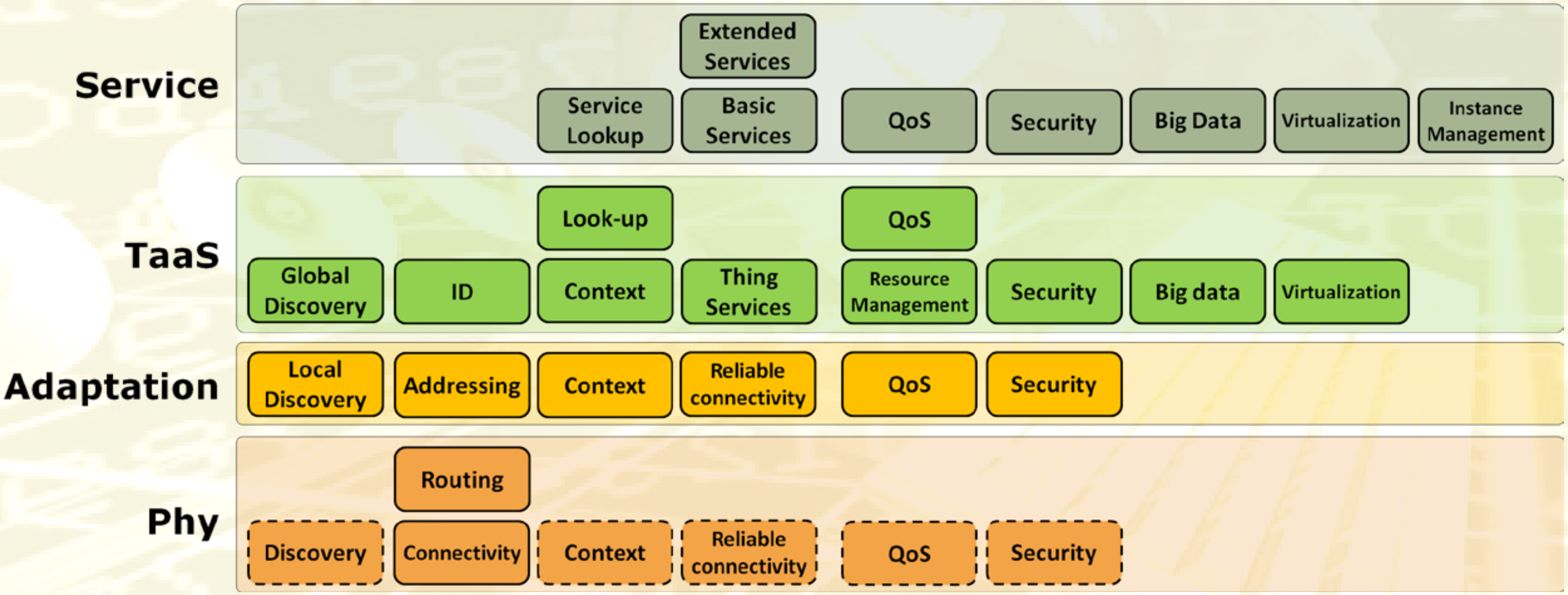


BETaaS aware vs. unaware systems

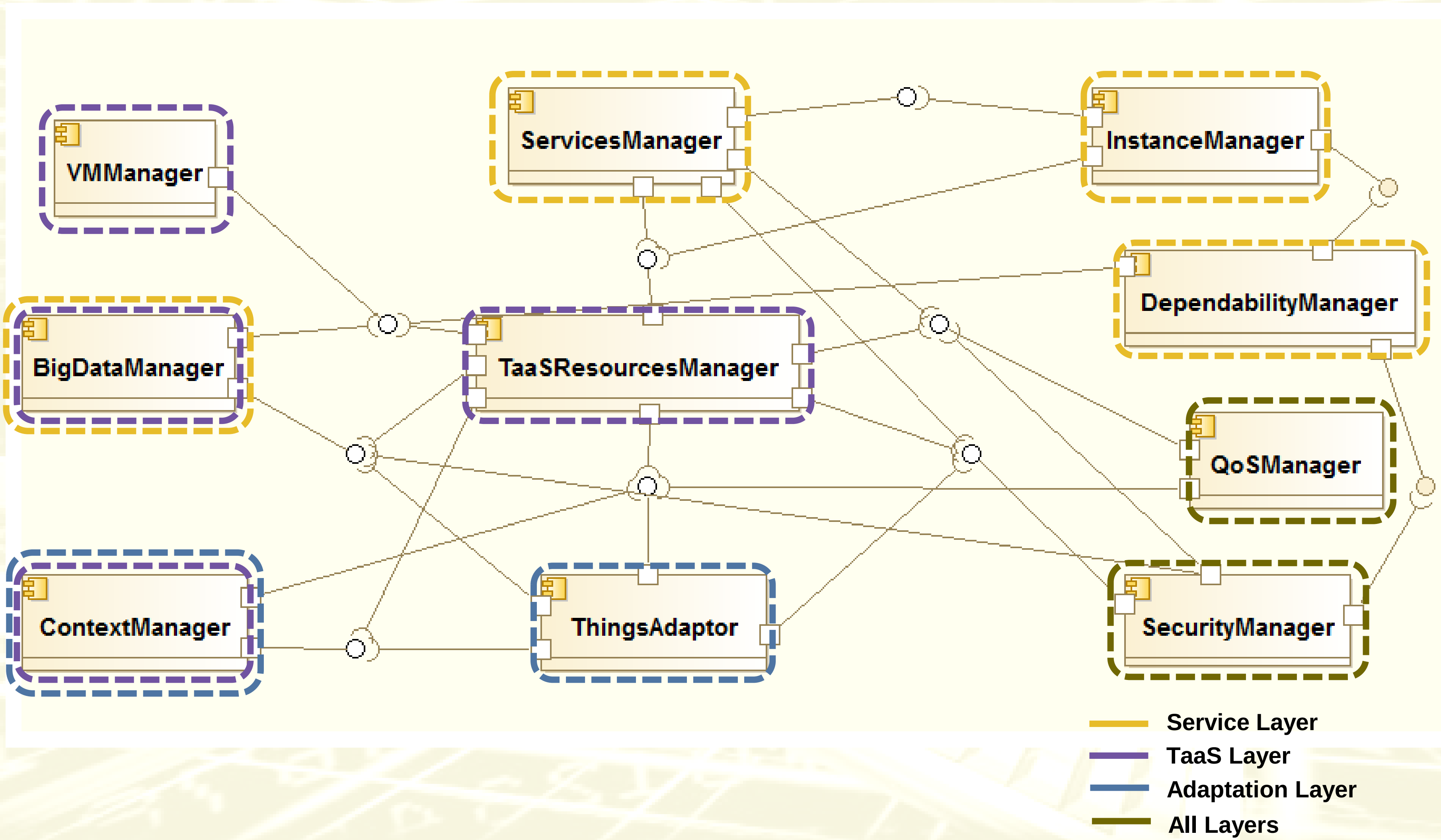


BETaaS functional view

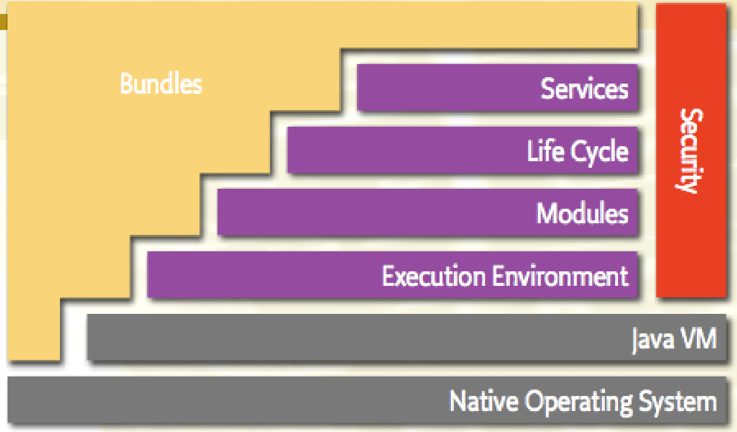
- Built-in support for extended features
  - Context-aware lookup
  - QoS, Security, Big Data, Virtualization



GW component architecture



BETaaS implementation

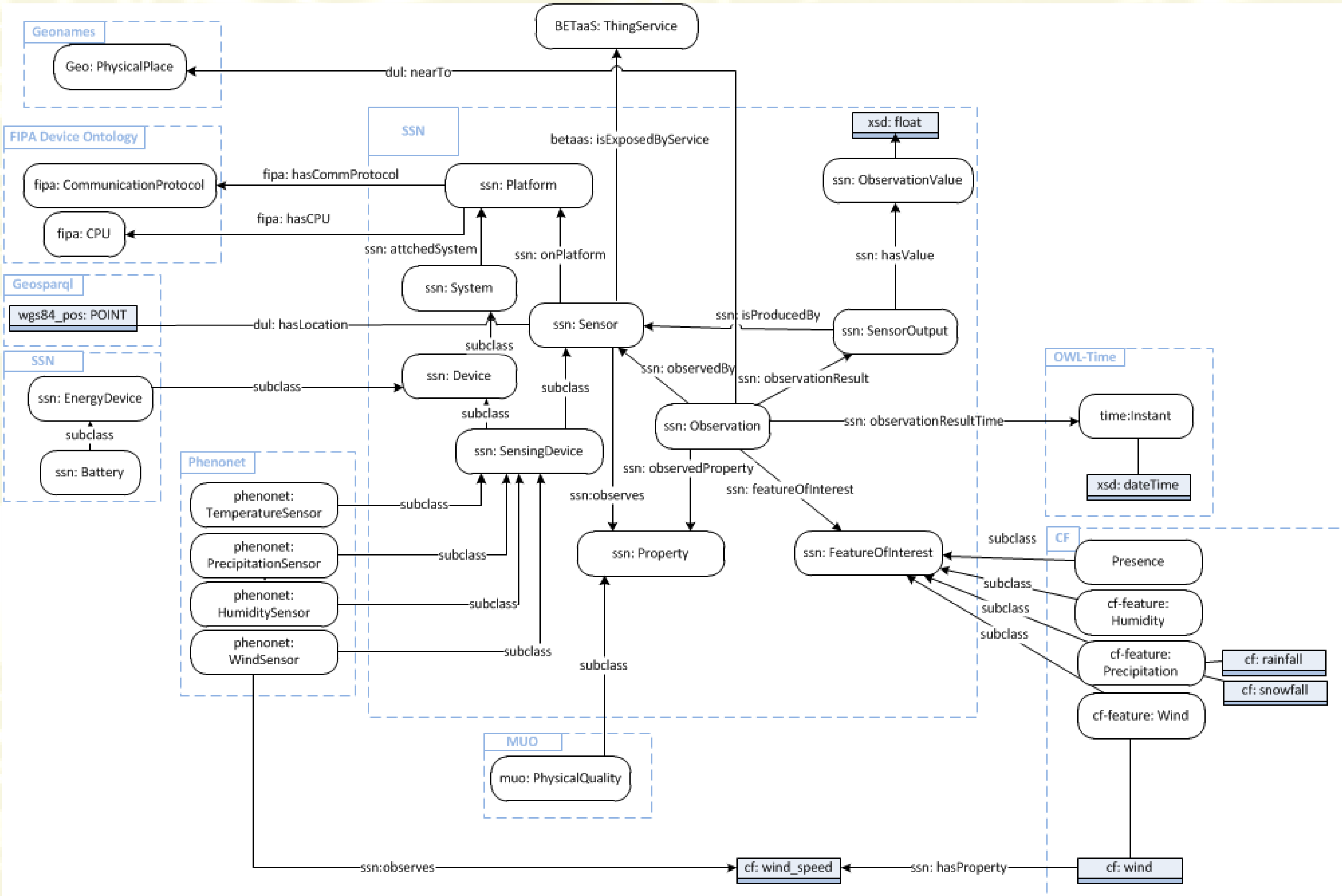


- Baseline technologies
  - OSGi & D(istributed) OSGi
- Benefits:
  - **Reduced Complexity** (component-based service-oriented architecture: *bundles* that communicate through well defined *services*)
  - **Reuse (**third party components)
  - **Easy Deployment** (standardized management API - how components are installed and managed)
  - **Dynamic Updates (**Bundles can be installed, started, stopped, updated, and uninstalled without bringing down the whole system**)**
  - **Simple (**OSGi API**)**, **Small (**runs on a large range of devices: from very small, to small, to mainframes, only asks for a minimal Java VM to run**)**, **Fast**, **Lazy (**do things only when they are really needed**)**
  - **Widely Used**
  - **Supported by Key Companies (**Oracle, IBM, Samsung, Nokia, Progress, Motorola, NTT, Siemens, Hitachi, Deutsche Telekom, Redhat, Ericsson, and many more)

Semantic support

- Goal
  - **Unify** the **information** that comes from heterogeneous resources and applications
  - **Infer knowledge** from raw data in a **context-aware** fashion
- How
  - Build a **network of ontologies** based on existing ontologies: **BETaaS Ontology**
  - Introduce **context** associated to things
  - Infer information not explicitly reflected in the ontologies
    - Equivalent Things Services
    - Combined Thing Services into on-the-fly Services

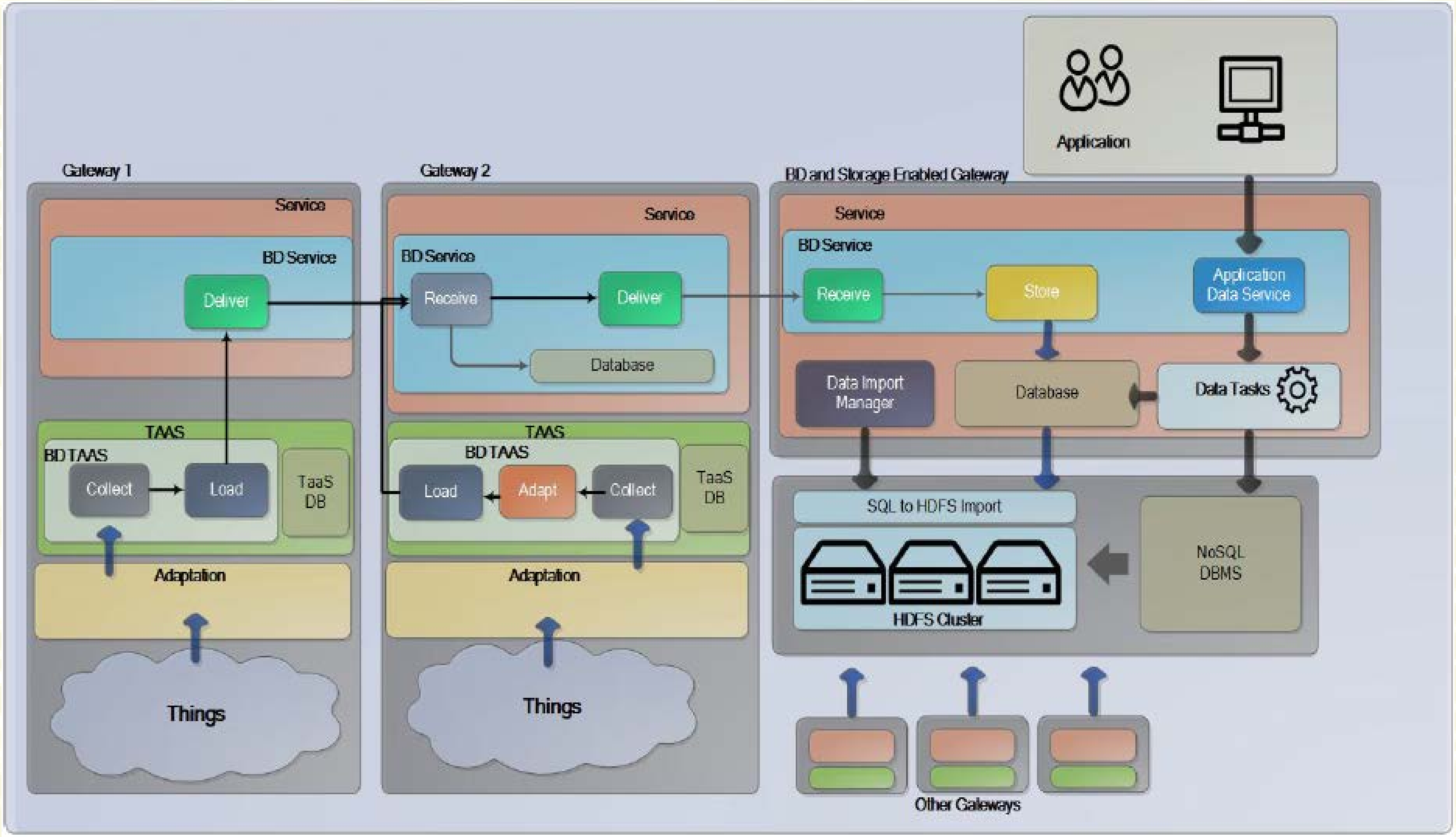
Semantic support



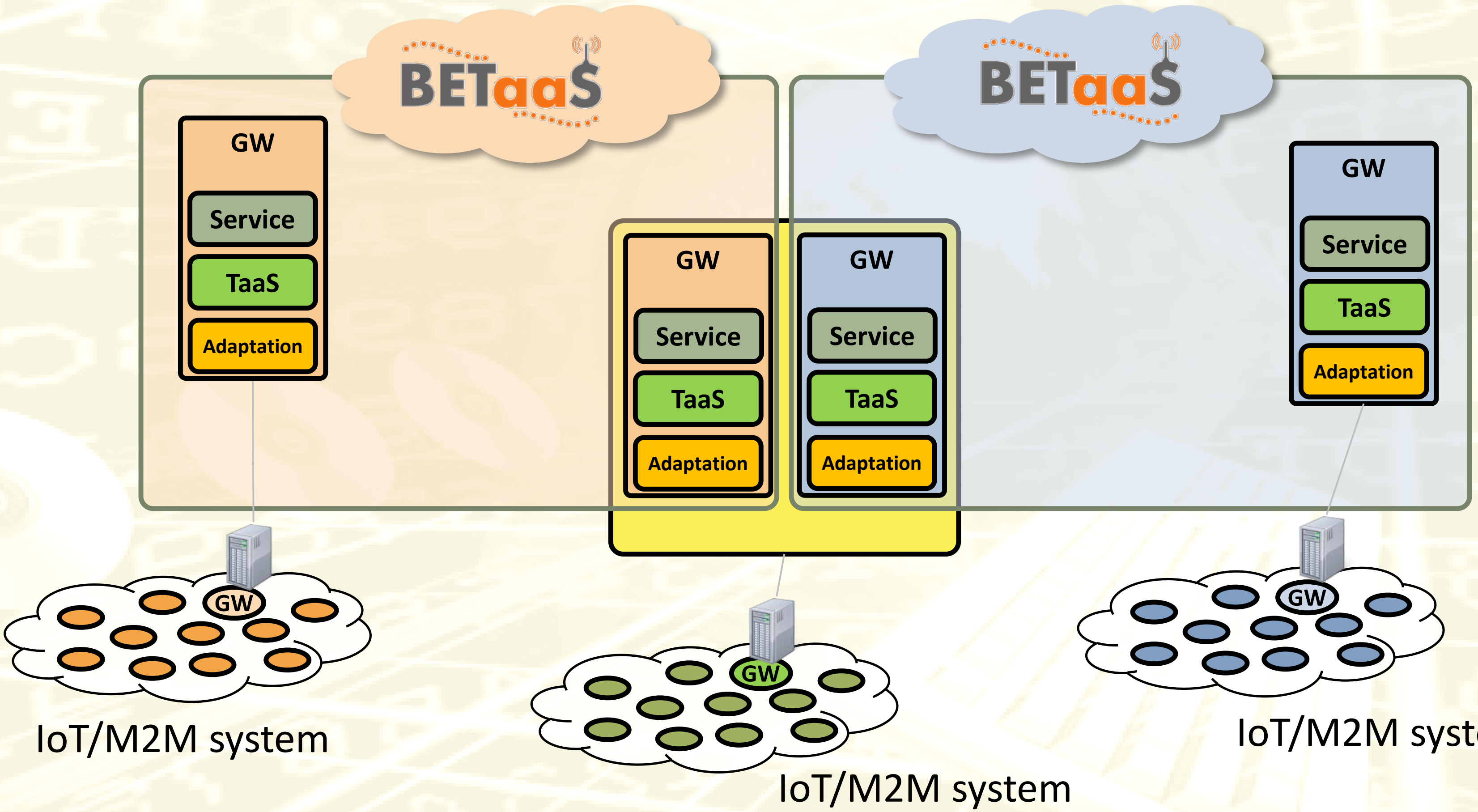
Big Data management

- Relevant (more) for **large scale data scenarios**
  - e.g., IoT Smart City scenario: traffic trends analysis, tourists attractions preferences and typical paths discovery
- BD **functional components** and **services**
  - Distributed file system (e.g. HDFS)
    - Required for storing distributed data
  - Data processing services (e.g. Hadoop Map/Reduce jobs)
    - Applications can perform batch-jobs queries over the collected data
  - Real-time query services
    - Applications can perform (almost) real-time queries over the collected data
- Deployed at **both** TaaS and Service layers
  - Flexible and different deployment configurations according to single gateways resources capabilities
  - General capabilities
    - Gather data from source (things) (@TaaS)
    - Store data (@TaaS/Service)
    - Process data (@Service)

Big Data management



GW virtualization



Quality of Service support

- M2M application scenarios may have very different and unique requirements
- Provide services with associated guarantees on performance
  - Allow applications to negotiate a Service Level Agreement (SLA)
    - QoS model
  - Enforce QoS through the efficient management of resources
    - Resource reservation and optimized allocation
  - Monitor SLA fulfillment

QoS model

- Classes of service
  - **Real-Time**: Applications with hard response time requirements (deterministic)
    - E.g., Surveillance system, Traffic light management
  - **Assured Services**: Applications with soft response time requirements (e.g., probabilistic)
    - E.g., Road traffic alerts, Vehicle tracking
  - **Best-Effort**: Applications without requirements
    - E.g., Meter data collection
- SLA templates defined accordingly

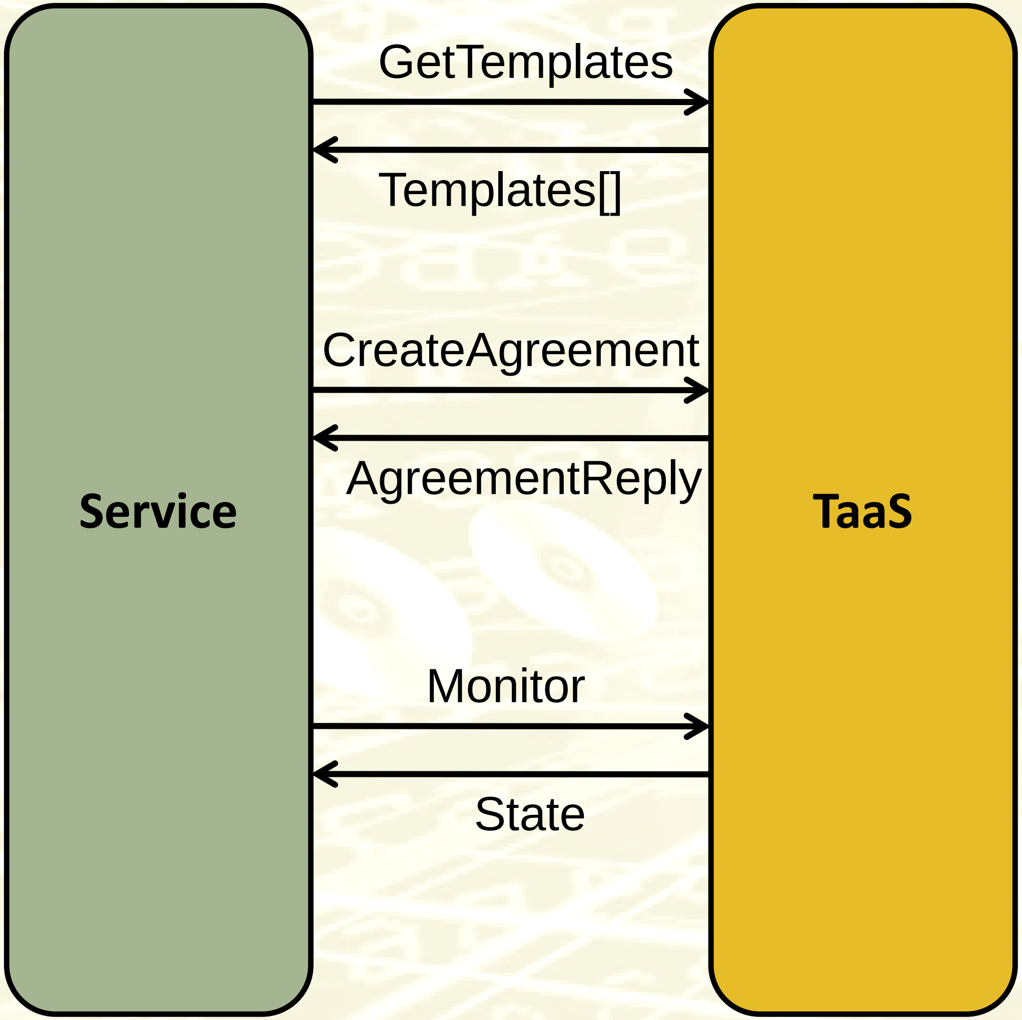
QoS negotiation procedure

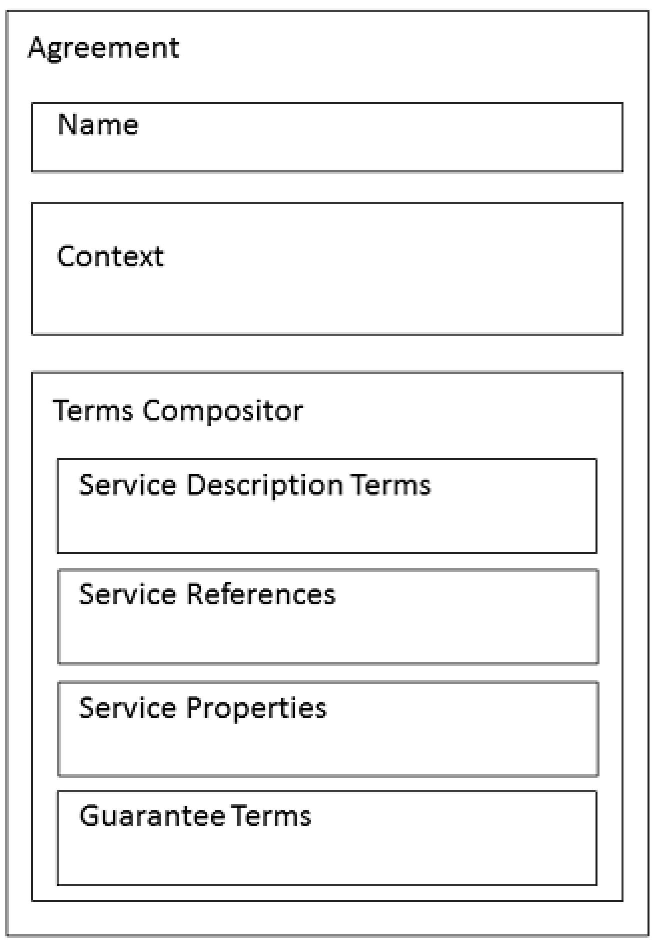
- Application negotiate QoS requirements with the service layer
  - Lightweight: manifest with acceptable QoS parameters (range values)

  or

  - WS-Agreement
- Service layer negotiate with the TaaS layer on behalf of applications
  - WS-Agreement

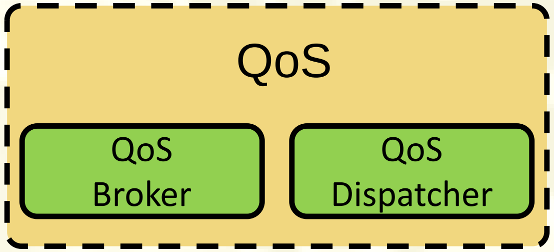
WS-Agreement





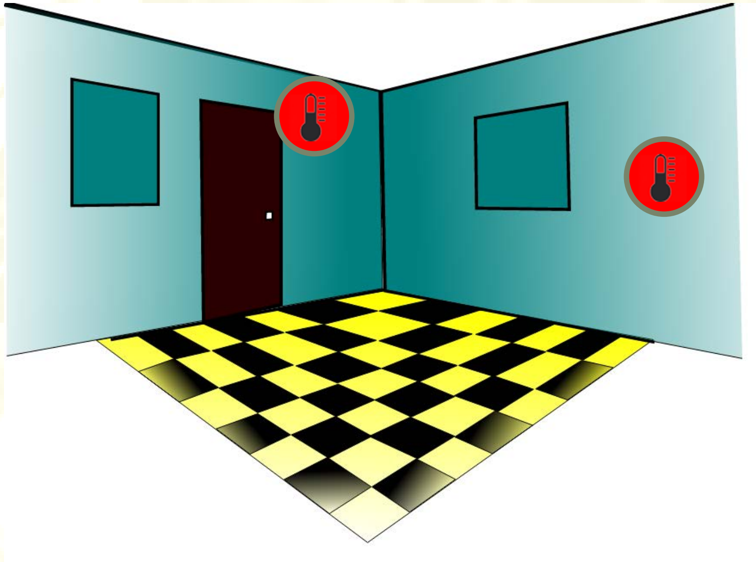
Reservation and Allocation

- To exploit equivalent things a two-step procedure is adopted in BETaaS
- Reservation (QoSBroker)
  - Manages QoS-based negotiation
  - Performs admission control based on QoS requirements
  - Manages QoS-based resource reservation
  - Generates EPRs to authorize Thing Service invocation
- Allocation (QoSDispatcher)
  - Optimizes allocation of resources at service invocation time
  - Based on dynamic state of resources, e.g., in terms of energy efficiency among a set of equivalent things



Context-based TS equivalence

- Service exposed by different things, that can be used interchangeably in a given context to provide the Thing Service required by the Service Layer
- Example: two temperature sensors in the same room
  - Equivalence is context-dependent!
- **Which Thing will the Thing Service be allocated to?**



TS selection optimization

- Mathematical formalization
  - Allocate **K** request to **N** Thing Services to maximize the lifetime of the system
  - The cost of allocating a request ***k*** to a TS ***n*** depends on ***n***
  - Each request ***k*** can be allocated only to a subset of the **N** TSs
  - Each request ***k*** has time completion requirements
  - (A subset of) the **N** TSs are battery constrained
- The problem can be formulated with a Bottleneck version of the Generalized Assignment Problem (BGAP)

Aknowledgements & credits

- **BETaaS: Building the Environment for the Things as a Service**

http://www.betaas.eu

Thanks!

Enzo Mingozzi
Associate Professor @ University of Pisa
e.mingozzi@iet.unipi.it