CISTER – Research Centre in
Real-Time & Embedded Computing Systems

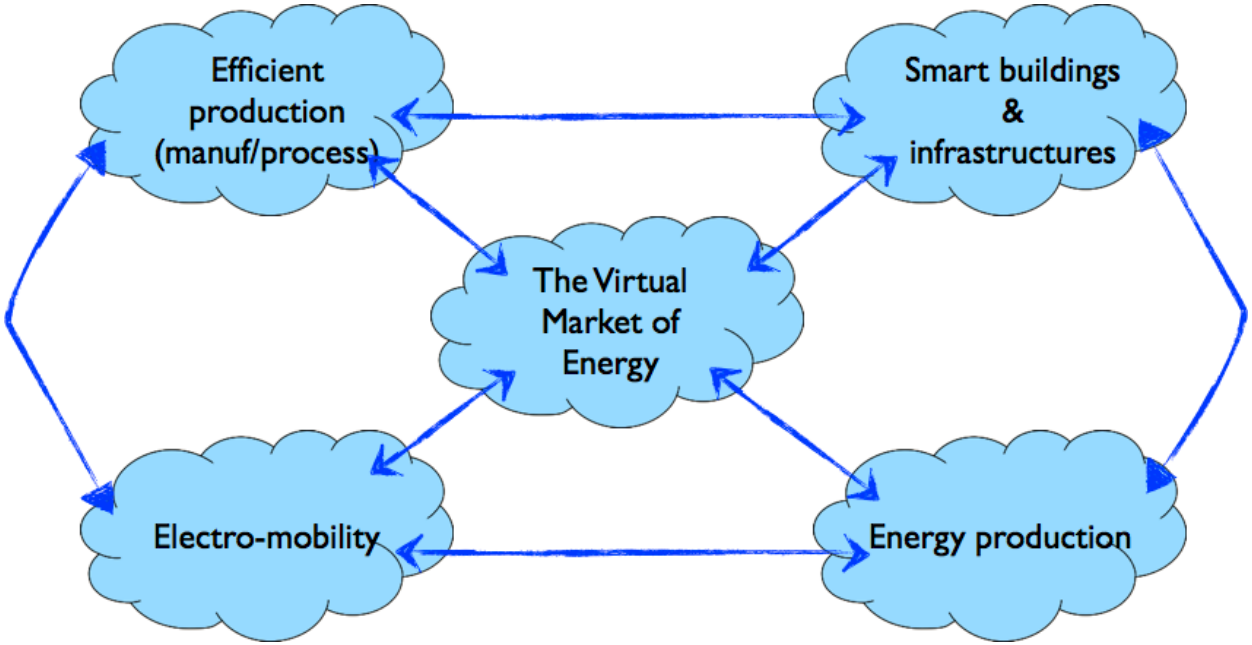
QoS-as-a-Service in the Local Cloud

Luis Lino Ferreira,
Michele Albano,
Jerker Delsing

The Arrowhead Project

A European initiative (79 partners, 15 countries) aiming at normalizing all interactions between embedded systems by means of a Software Oriented Architecture (SOA)

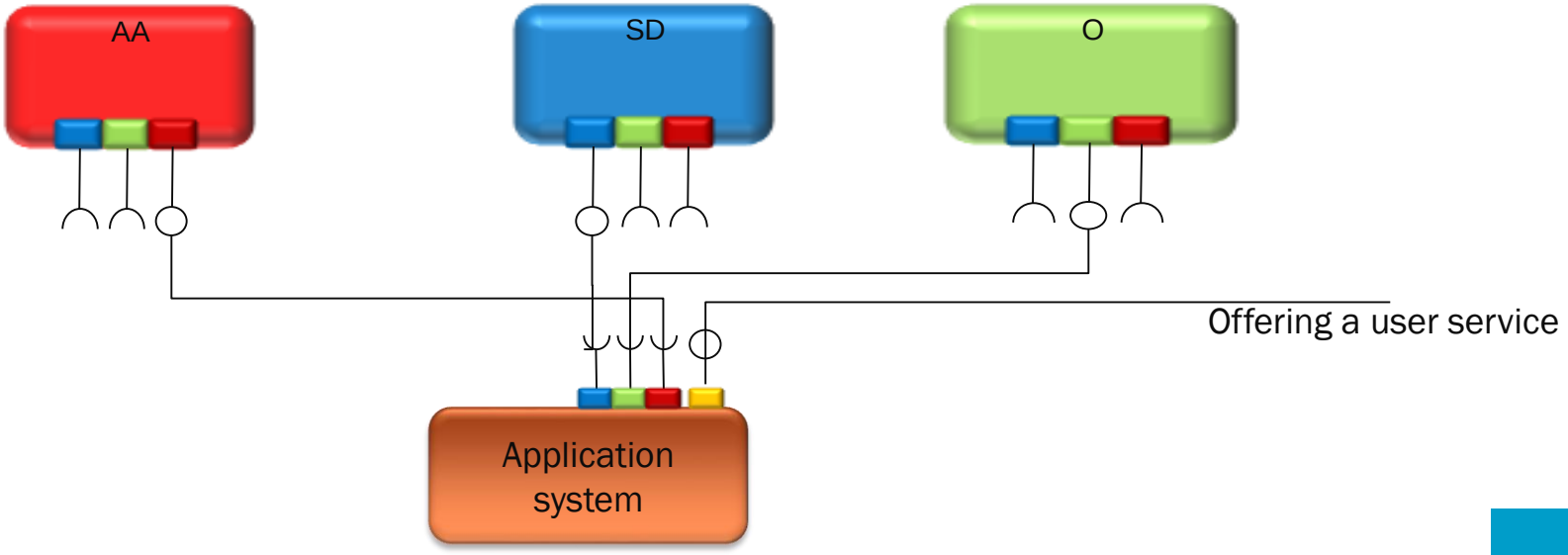
- Any use case is a System-of-Systems (SoS), made up of devices that host systems, which produce and consume services



The Arrowhead Framework

Services are either

- User Services, providing the application functionalities on each particular scenario (business logic)
- Core Services, provided by the Arrowhead Framework and satisfying non-functional requirements (housekeeping)
  - At least: Authentication Service for devices, Registration Service for devices and services, and Orchestration Service to look-up for devices/services, and to create more complex services

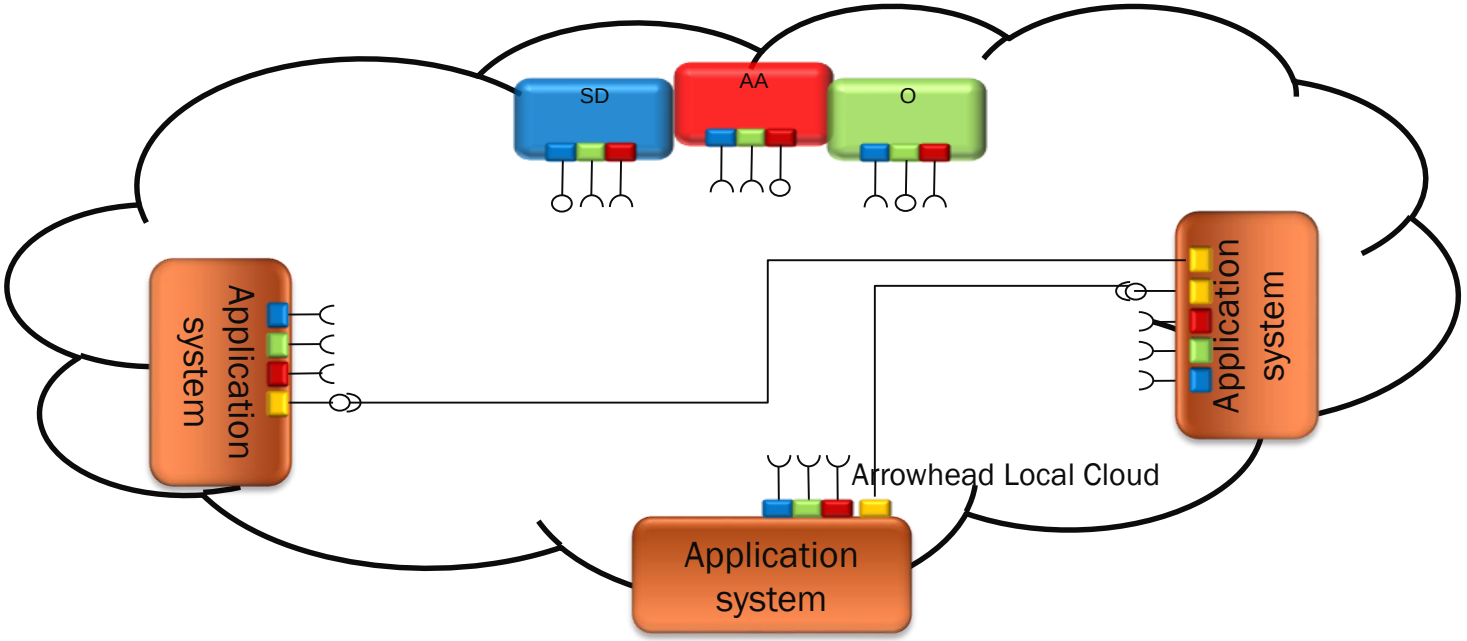


The Local Cloud

An Arrowhead-enabled SoS is based on the concept of Local Cloud:

- A bounded set of computational resources used by stakeholders to attain a goal
- Controls security
  - Restricts access to authenticated Devices/Systems/Services.
- Autonomous
  - Contains all core services needed to be able to function

Relating QoS to a Local Cloud simplifies QoS monitoring and reduces the QoS managing complexity.



Orchestration Process(es)

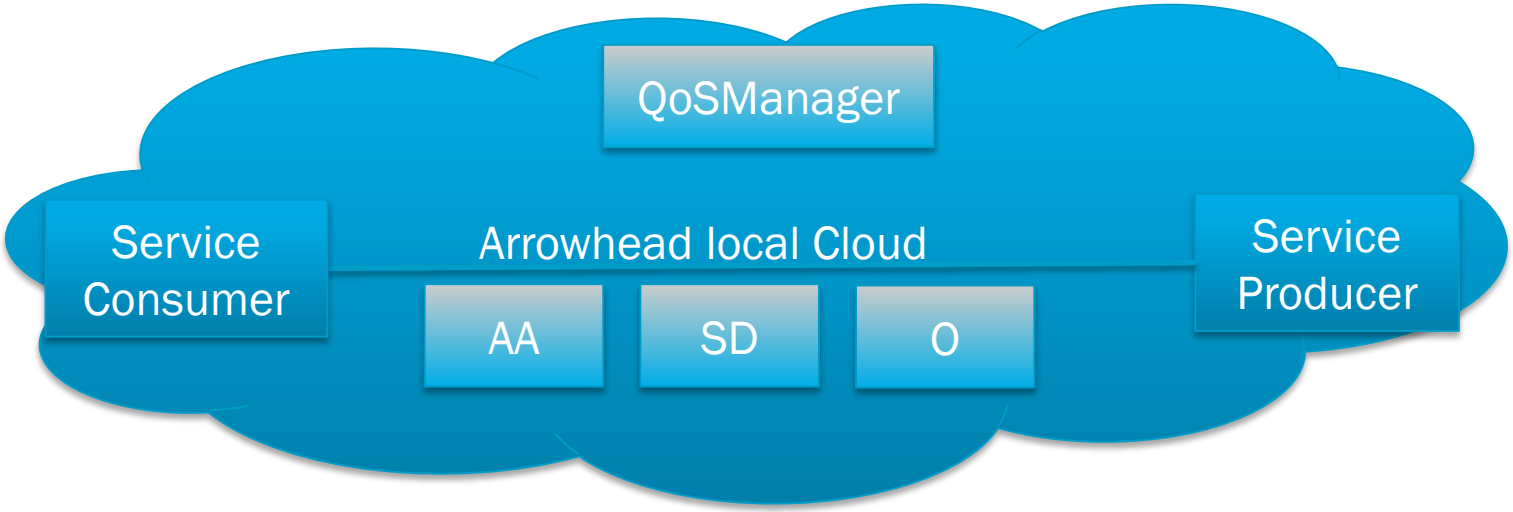
Two orchestration approaches are envisioned:

- An imperative “pull” approach:
  - Devices, systems, services are registered into the SoS
  - Systems access the Orchestrator service of the Orchestrator system with their functional requirements
  - The Orchestrator retrieves topology and constraints from the registries
  - The Orchestrator computes the best options for the systems
  - The Orchestrator answers to the systems with the new orchestrations
- A declarative “push” approach:
  - Devices, systems, services are registered into the SoS. Systems register also the functional requirements for consumed services
  - The Orchestrator system periodically – or after configuration changes – wakes up
  - The Orchestrator retrieves topology and constraints from the registries
  - The Orchestrator computes the best options for the systems
  - The Orchestrator initiates interactions with the systems to provide them with orchestrations

QoS-as-a-Service

Some scenarios present QoS constraints for effective service fruition

- Let us apply the SOA paradigm
  - Allow request for QoS for (orchestrated) services
  - QoS requirements are requested as services from the framework
    - can be dynamic needs, negotiated dynamically during runtime
    - QoS is a non-functional requirement, thus a core service
- Let us add the QoSManager to the Core Systems
  - QoSSetup core service to verify QoS feasilibity, and configure SoS to grant it
  - QoSMonitor core service to monitor QoS at runtime

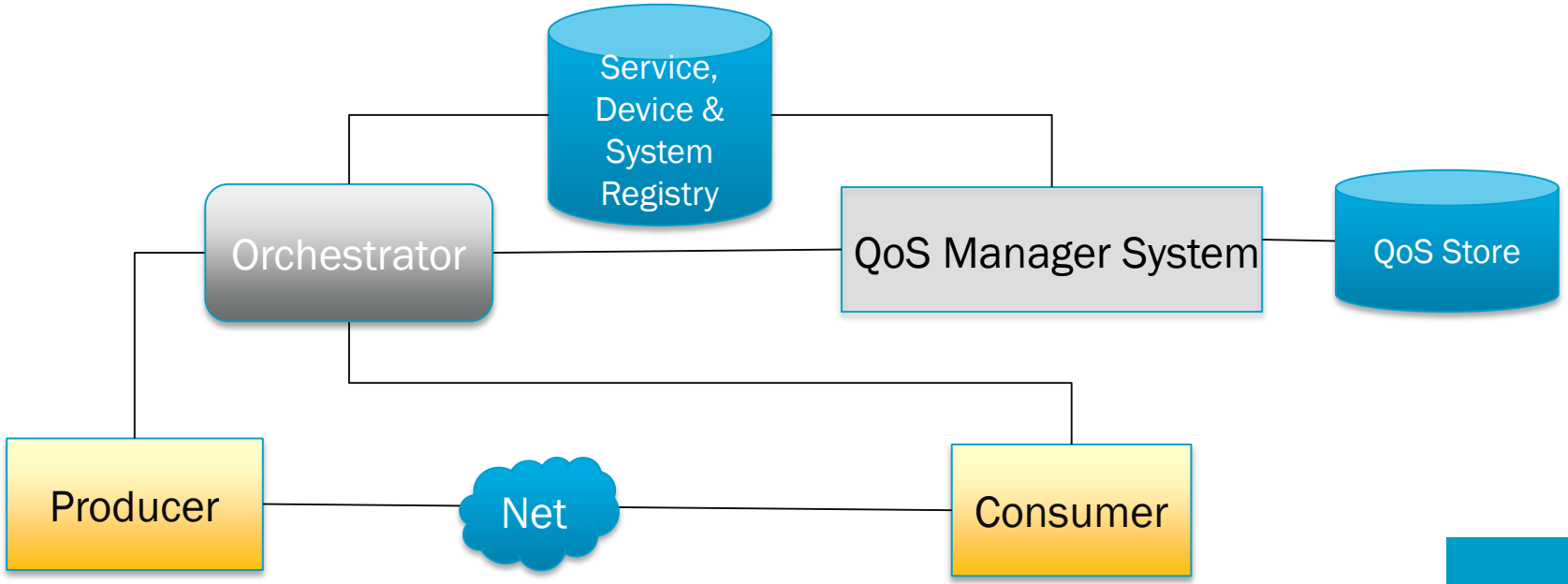


Types of QoS

- Arrowhead service fruition considers a few QoS types
  - Time-related (Hard real-time, Soft real-time, Relative priorities)
  - Bandwidth (service requests handled per second, network bandwidth)
  - Communication semantics (delivery guarantees, message ordering)
- QoS capabilities depend on the underlying support.
  - Hard real-time
    - Time Triggered Ethernet
    - IEEE 802.15.4 with GTS
    - Real-time Fieldbus (e.g.: CAN)
  - Soft real-time
    - Ethernet with limit on acceptance of new requests
    - IEEE 802.15.4 with adapted CSMA MAC
  - Best effort (with prioritization)
    - Internet technologies (with DiffServ, IP TOS)

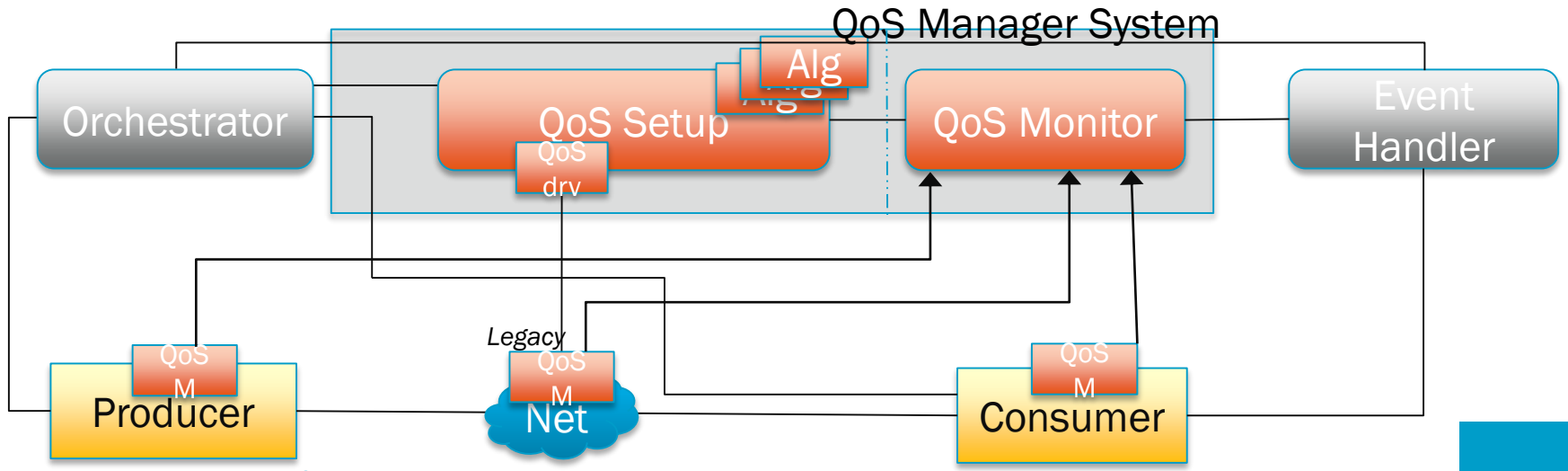
Architectural components 1/2

- Producer, Consumer, Orchestrator and QoS Manager are systems connected through a network
- The Registries (Device, System and Service) provide information, for example for service orchestration
- The QoS Store contains all data regarding QoS: capabilities, requirements, reservations

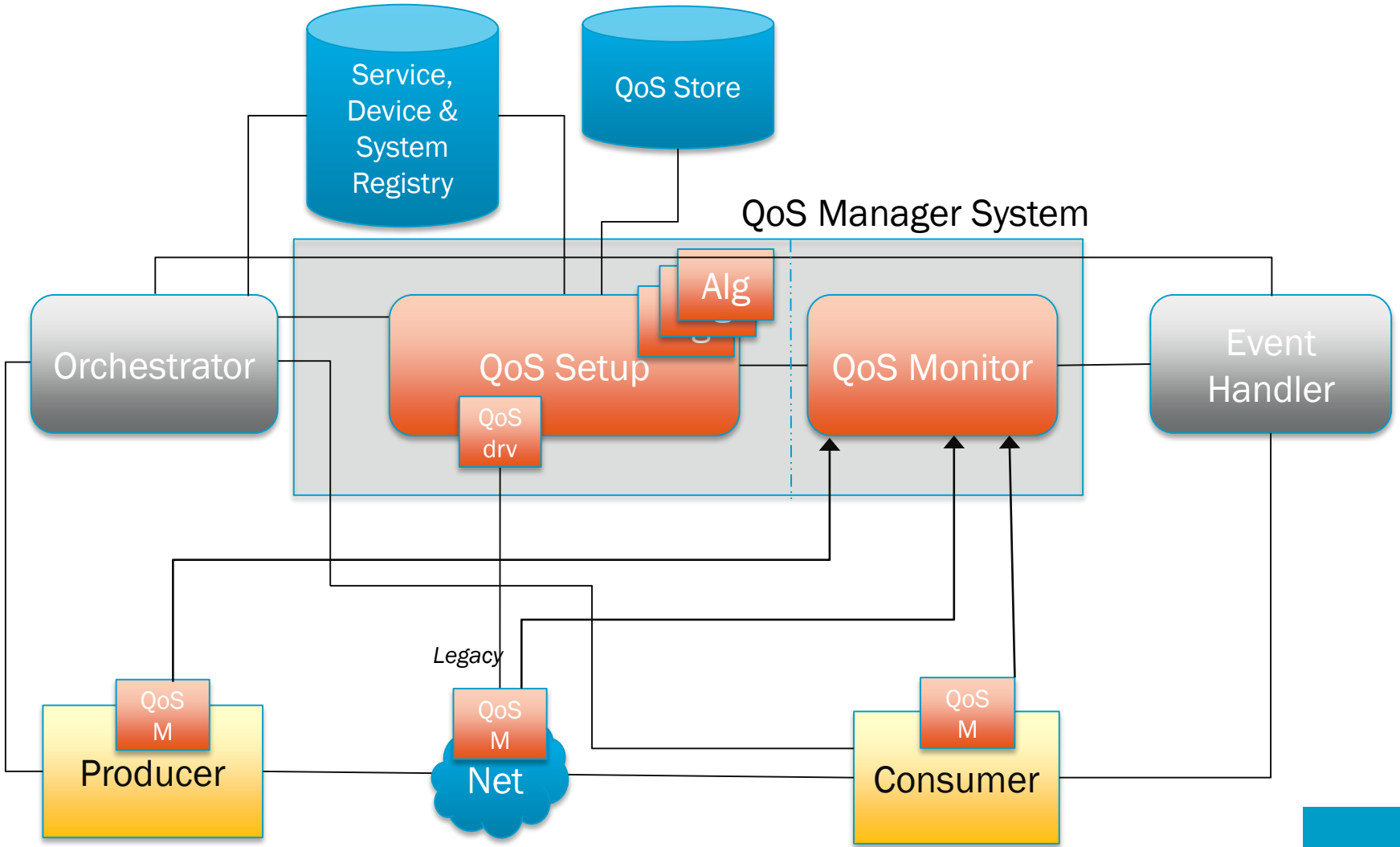


Architectural components 2/2

- QoS services are produced either by one system – QoS Manager – or by different systems.
- The QoS Setup service uses a set of algorithms (Alg blocks) to verify QoS feasibility, and the QoS Drv to set up systems and networks
  - It interacts with Orchestrator only
  - QoS Drivers are use to interact with not Arrowhead-compliant routers/devices via legacy protocols
- QoS Monitor receive monitor data from the QoS Modules (QoS M)
  - On critical events (e.g.: QoS fault), it sends a message to interested (subscribed) parties through the Event Handler service/system
  - Interested parties are usually the Orchestrator (when it is "pushy") and the service consumer



Full Architecture for QoS

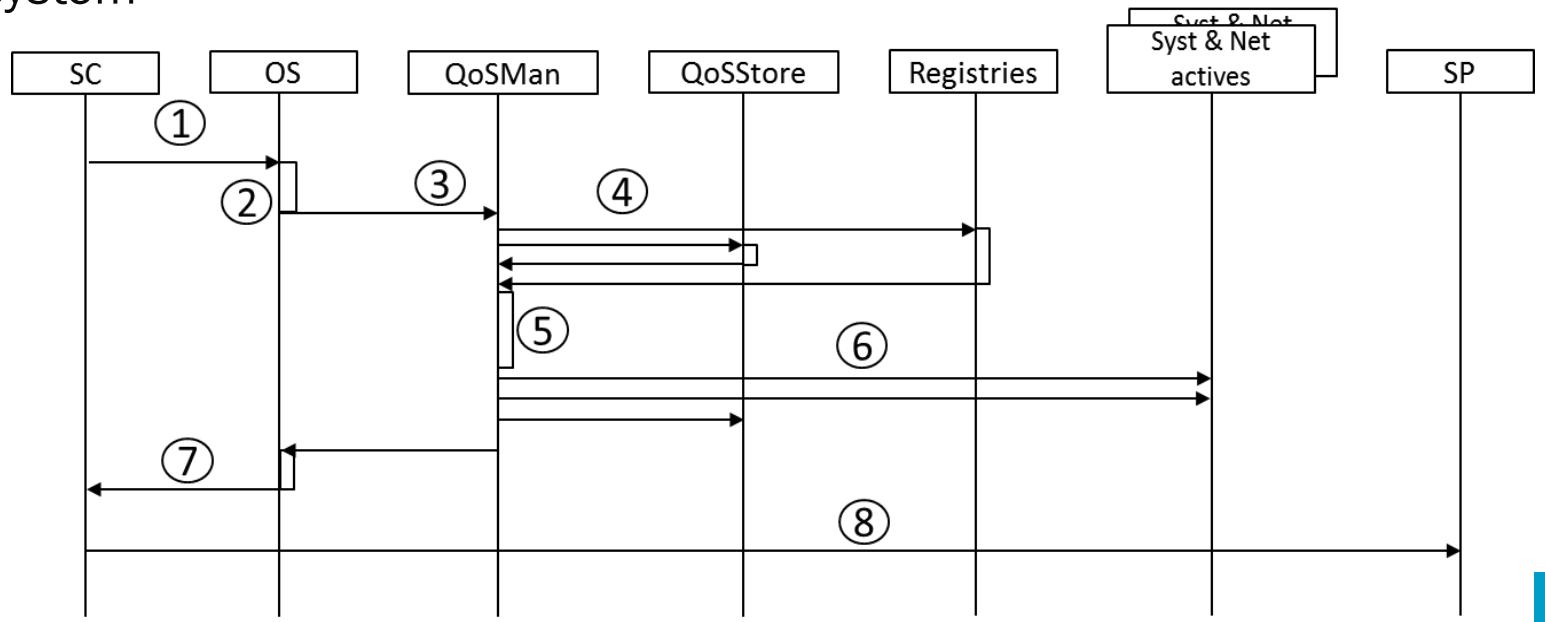


Push or pull?

- Orchestration can work in push (declarative, spontaneous) or pull (imperative, on demand) way
- QoS Manager can act push or pull with respect to the Orchestrator
  - Push: QoS Manager uploads on the registries a set of constraints, which correspond to QoS capabilities and requirements, to be used by the Orchestrator
  - Pull: when Orchestrator computes orchestrated services, it verifies them by contacting the QoS Manager before sending them (in push or pull manner) to the systems

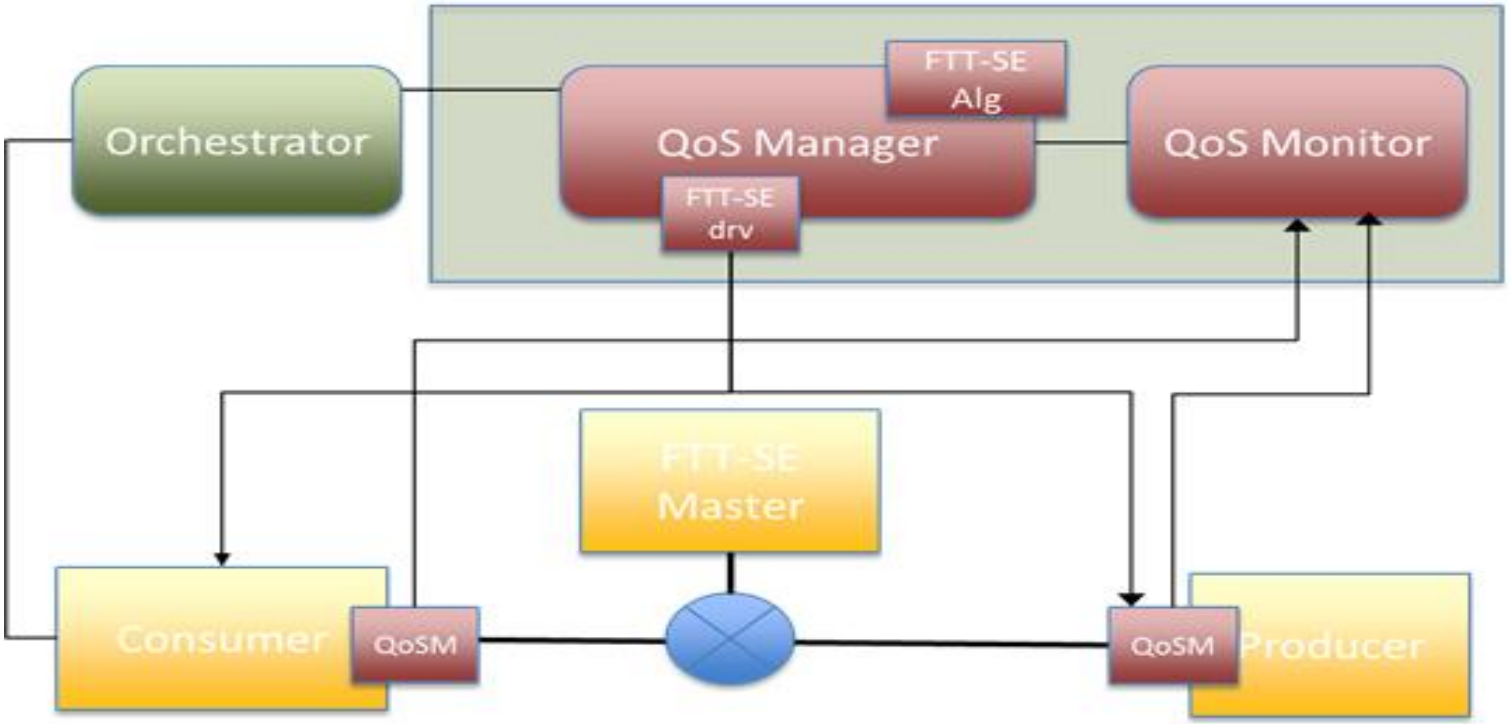
Better to pull

- Three drawbacks of the declarative approach:
  - Race conditions in dynamic systems
  - Reservations must be computed in advance for each possible set of services, thus many rules
  - Reservations must be computed in advance for each possible set of active reservations, thus many MANY rules
- Better to pull, and have the QoS Setup act as plugin for Orchestrator system



Concrete Architecture

- The Reference Architecture for QoS was mapped on a FTT-SE scenario
  - Master/slave protocol, consumers and producers have to reserve bandwidth / request to send data with the master
  - The QoS drivers are used to configure the QoS on the master, before allowing producers/consumers to start interacting



Future Work

- Proof that Declarative QoS is NP-complete
- Implement the concrete FTT-SE architecture (please wait for next slide)
- Apply the architecture to a 802.15.4 scenario
- Study the impact of QoS-as-a-Service on security and scalability
- Extend to multiple local clouds
  - But beware of the Internet, and other inter-cloud networks not in control of Arrowhead stakeholders

FTT-SE scenario

- The scenario was actually implemented a couple of weeks ago
  - Let us see a video of what is implemented, and what was measured
  - The video is available on the youtube channel of the Arrowhead project:

https://www.youtube.com/user/ArrowheadProject

Thank you for your attention!

Any questions?

Supplementary material: Arrowhead use cases

Supplementary material: Arrowhead WPs and pilots