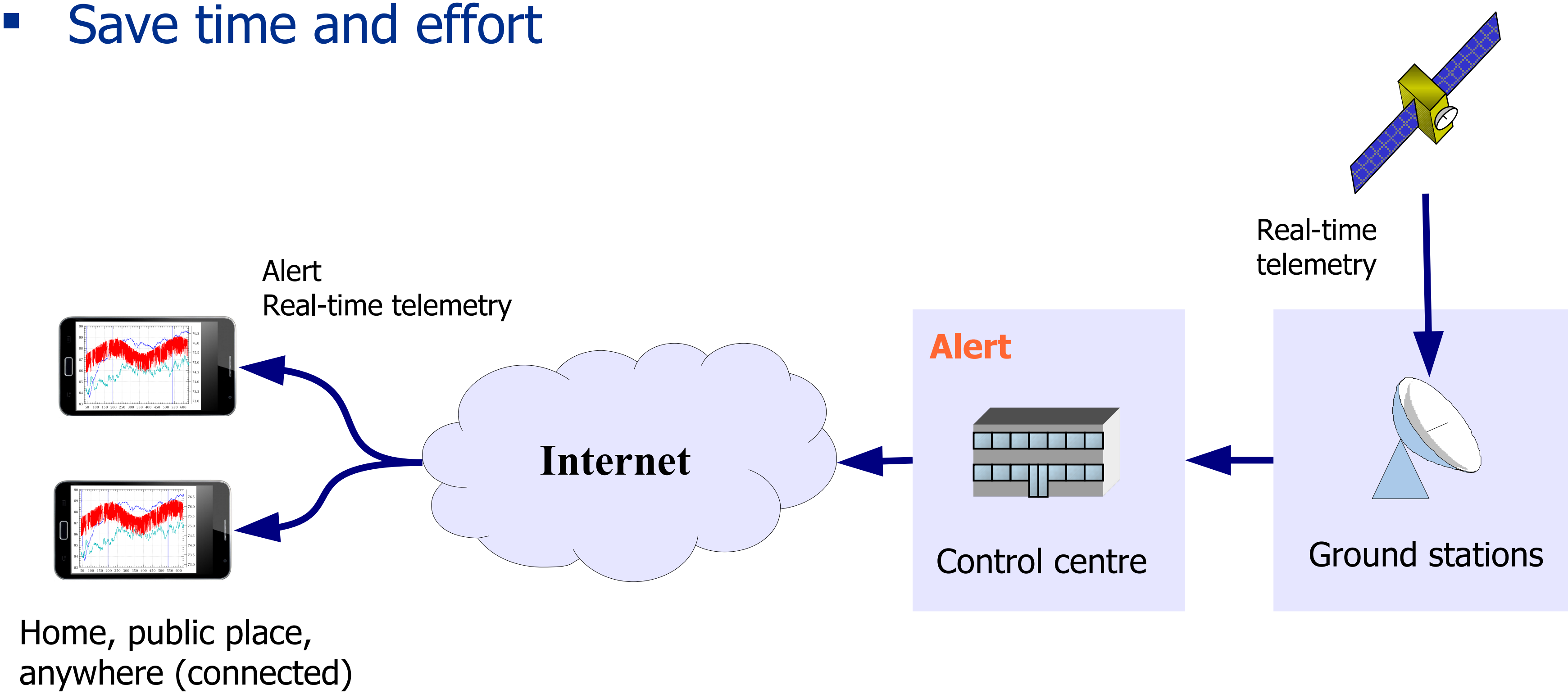
Monitoring a spacecraft from your smartphone using MQTT with Joram

joram.ow2.org
mqtt.jorammq.com
www.scalagent.com

David Féliot

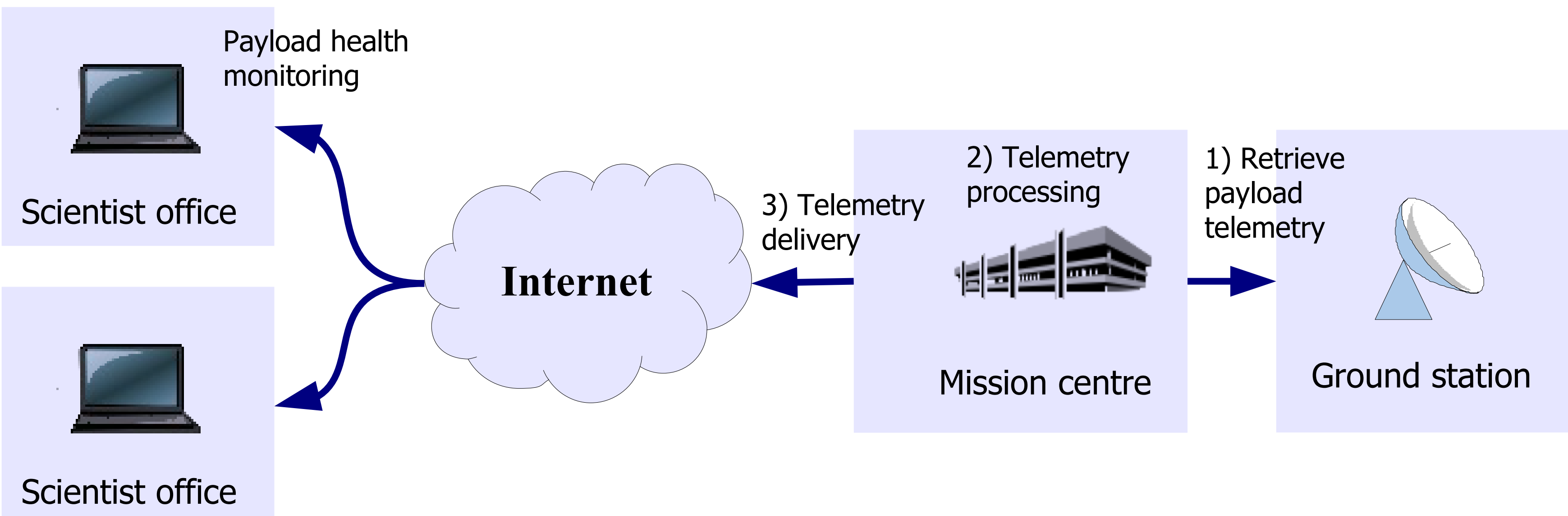
Use case #1: on-call operators

- On-call operators (working outside the control centre)
  - ⇨ Receive alert reports describing anomalies
  - ⇨ First level analysis by checking the real-time telemetry
- Save time and effort



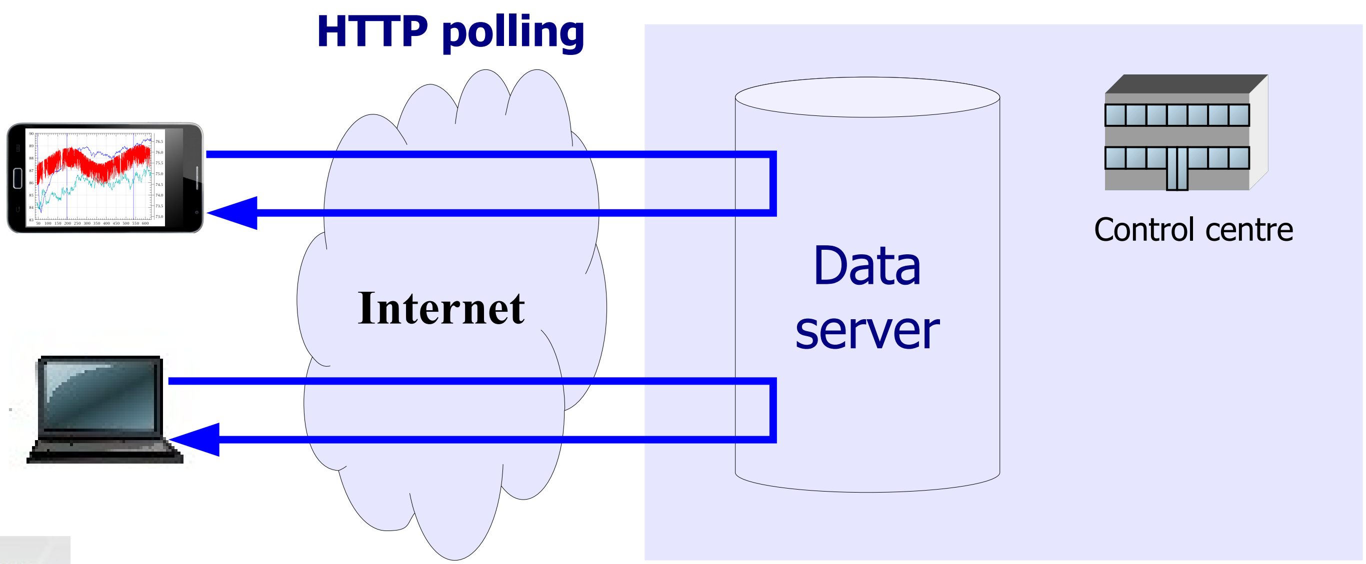
Use case #2: distributed scientists

- Science team members remotely working
  - ⇨ Their office is outside the mission centre that controls the scientific instruments
  - ⇨ Need to monitor the instruments health
  - ⇨ Situation happens in internationally cooperative space missions



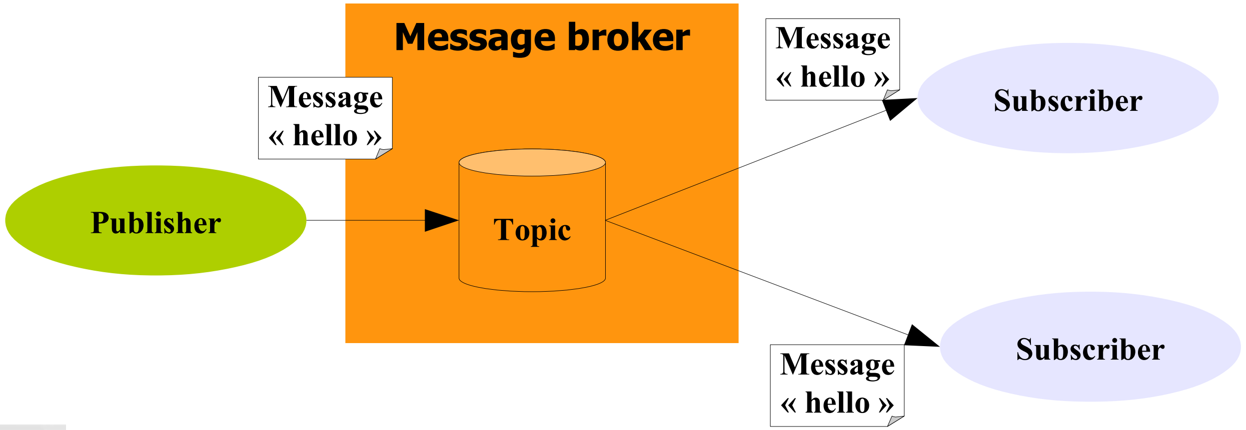
Existing solution

- Client/server architecture
  - ⇨ Using HTTP to send requests to a data server
  - ⇨ Alerts and telemetry are received by polling the server
- Security can be easily handled
  - ⇨ One-way data flow (low interaction level)
    - ↳ From a secured zone (control centre) to a less secured zone (client)
    - ↳ No data sent by the clients to the control centre (e.g. commands)



MQTT (Message Queuing Telemetry Transport)

- Lightweight message queuing protocol
  - ⇨ Devices with limited resources
  - ⇨ Constrained networks (bandwidth, connectivity)
- Provides a Publish/Subscribe interaction pattern
  - ⇨ Message published once on a given topic (subject of interest)
  - ⇨ Every consumer registered to this topic receives a copy of the message
- Relies on a message broker
  - ⇨ Time decoupling and reliable message delivery

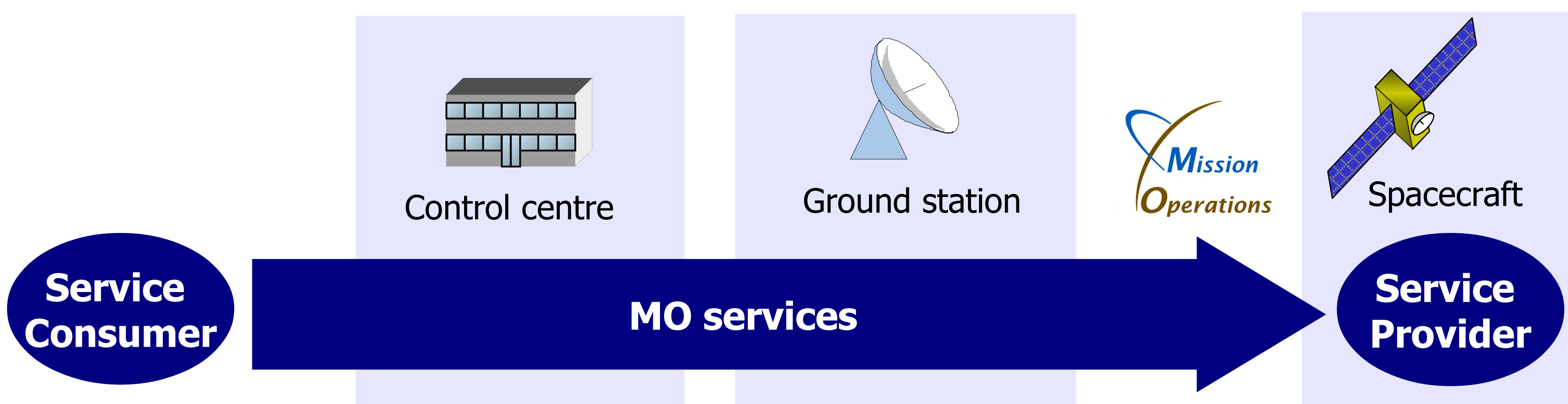


Solution using MQTT

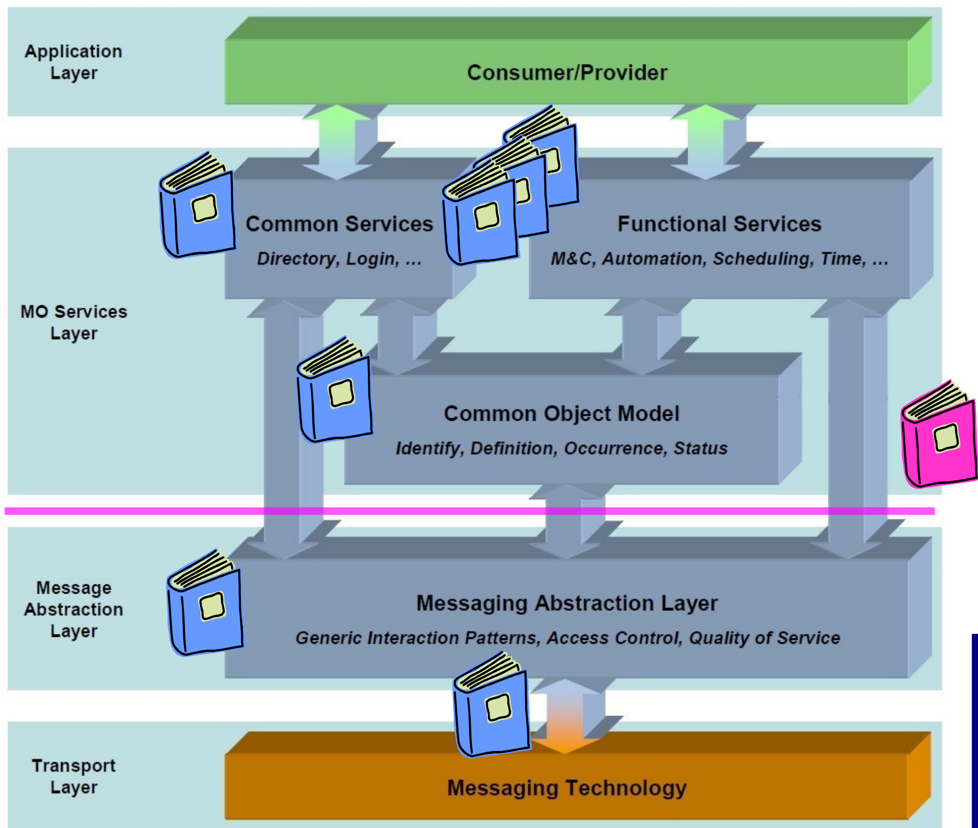
- Event driven architecture
  - ⇨ Publish/Subscribe allows to push real-time data
    - From the control centre to the clients (one-to-many)
    - Low message transmission latency
- MQTT should be more efficient than HTTP
  - ⇨ Less bandwidth and power usage
- Time decoupling and reliable message delivery
  - ⇨ Use case #2: scientist disconnected when the payload telemetry is published

CCSDS Mission Operations (MO)

- Service Oriented Architecture for space activities
  - ⇨ Standard end-to-end services that can be used on ground, ground to space, and in space
- Decoupling Consumer/Provider implementations
  - ⇨ MO services specify the meaningful information (semantic level)
    - Exchanged between a consumer and a provider
    - No dependency on the different links and transport protocols used underneath



MO service framework



CCSDS Recommended Standard

CCSDS Recommended Practice

Monitoring and Control: Parameter, Alert, Action

MAL Java API

Data model

Interaction patterns: req/resp, pub/sub

Data encoding format

Message sending/receiving, pub/sub

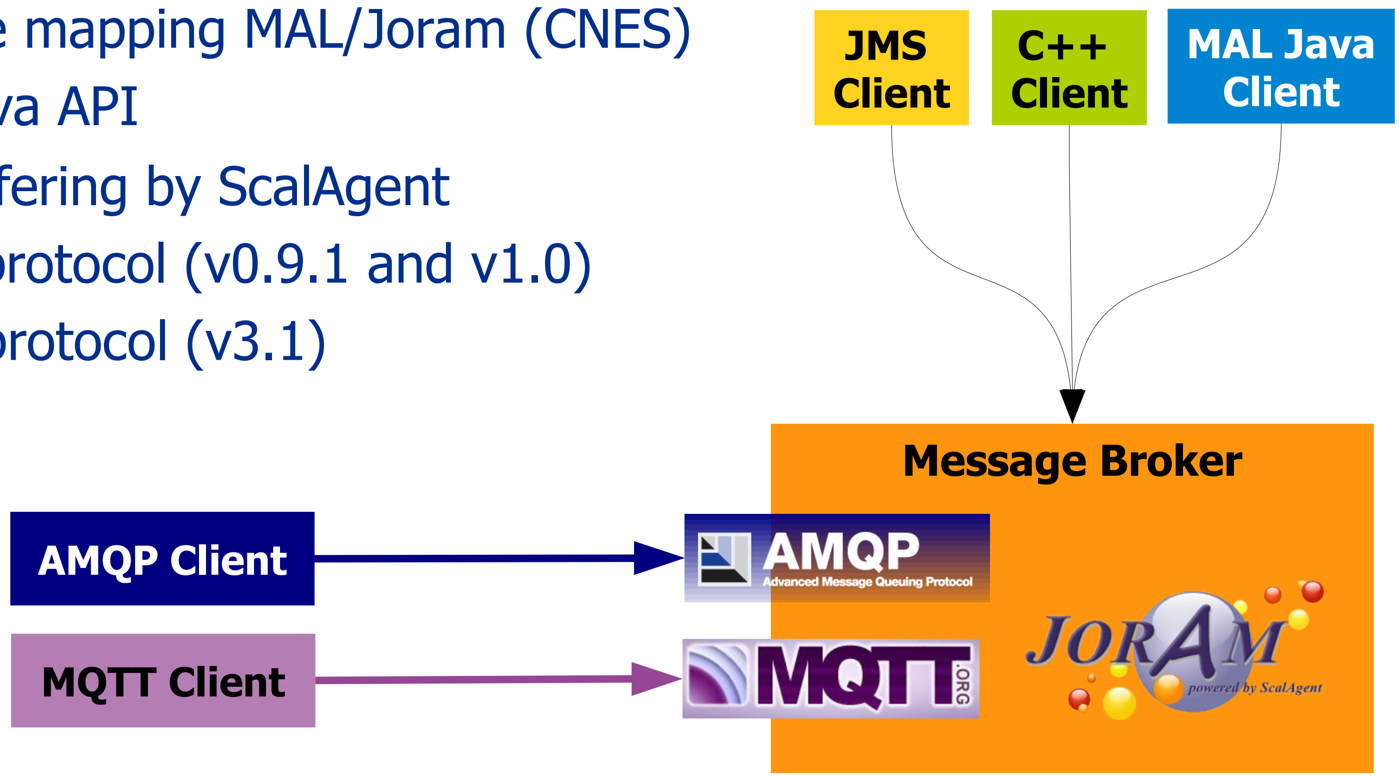
JMS

MO framework implementations

- Two open-source implementations of the MO standard in Java, compliant with the MAL Java API
  - ⇨ CNES
  - ⇨ ESA
- The CNES implementation is used by a prototype of Mission Control System (MCS)
  - ⇨ Developed by CNES
  - ⇨ Relies on Java, OSGi and Joram
- MO component platform experimented
  - ⇨ Based on Distributed OSGi and iPOJO

Joram, MAL/Joram and JoramMQ

- Open-source message broker written in Java (http://joram.ow2.org)
  - ⇨ Client APIs
    - JMS API (v2.0)
    - C++ API
- Open-source mapping MAL/Joram (CNES)
  - ⇨ MAL Java API
- JoramMQ offering by ScalAgent
  - ⇨ AMQP protocol (v0.9.1 and v1.0)
  - ⇨ MQTT protocol (v3.1)



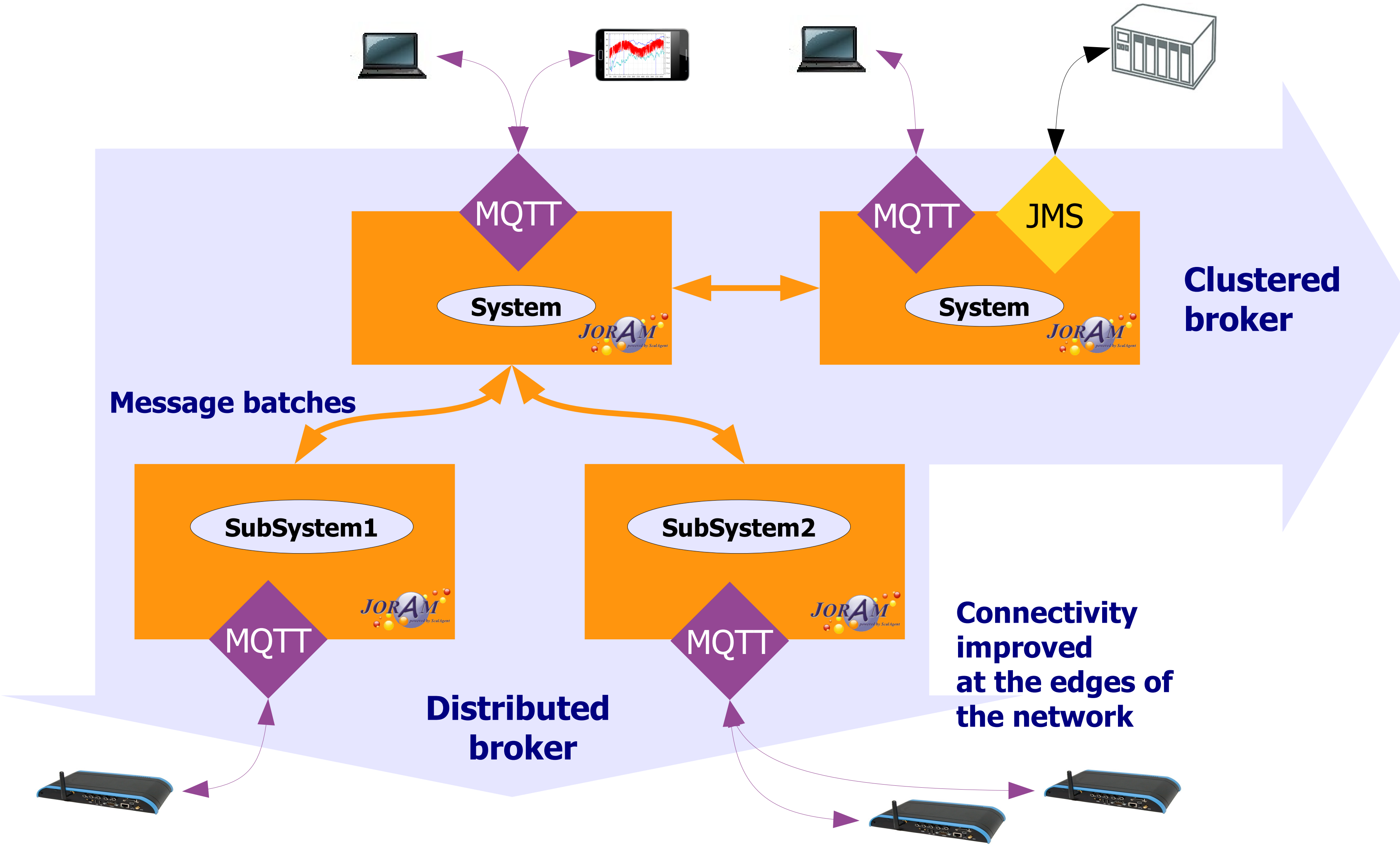
MAL/Joram key features

- Time decoupling provided by the message broker
  - ⇨ Message producers not tied to message consumers
  - ⇨ Slow consumers are handled by the message broker
    - Do not directly affect the producers
- Publish/Subscribe interaction pattern
  - ⇨ Real space decoupling provided by the message broker
    - Publishers do not need to know the network addresses of subscribers
  - ⇨ Scalability with the number of publishers and subscribers
    - By distributing the broker across multiple servers
- Message delivery reliability (no message loss)
  - ⇨ Messages delivered to the data store of the control centre
  - ⇨ Alerts transmission
- Interactions multiplexing (single connection) and flow control

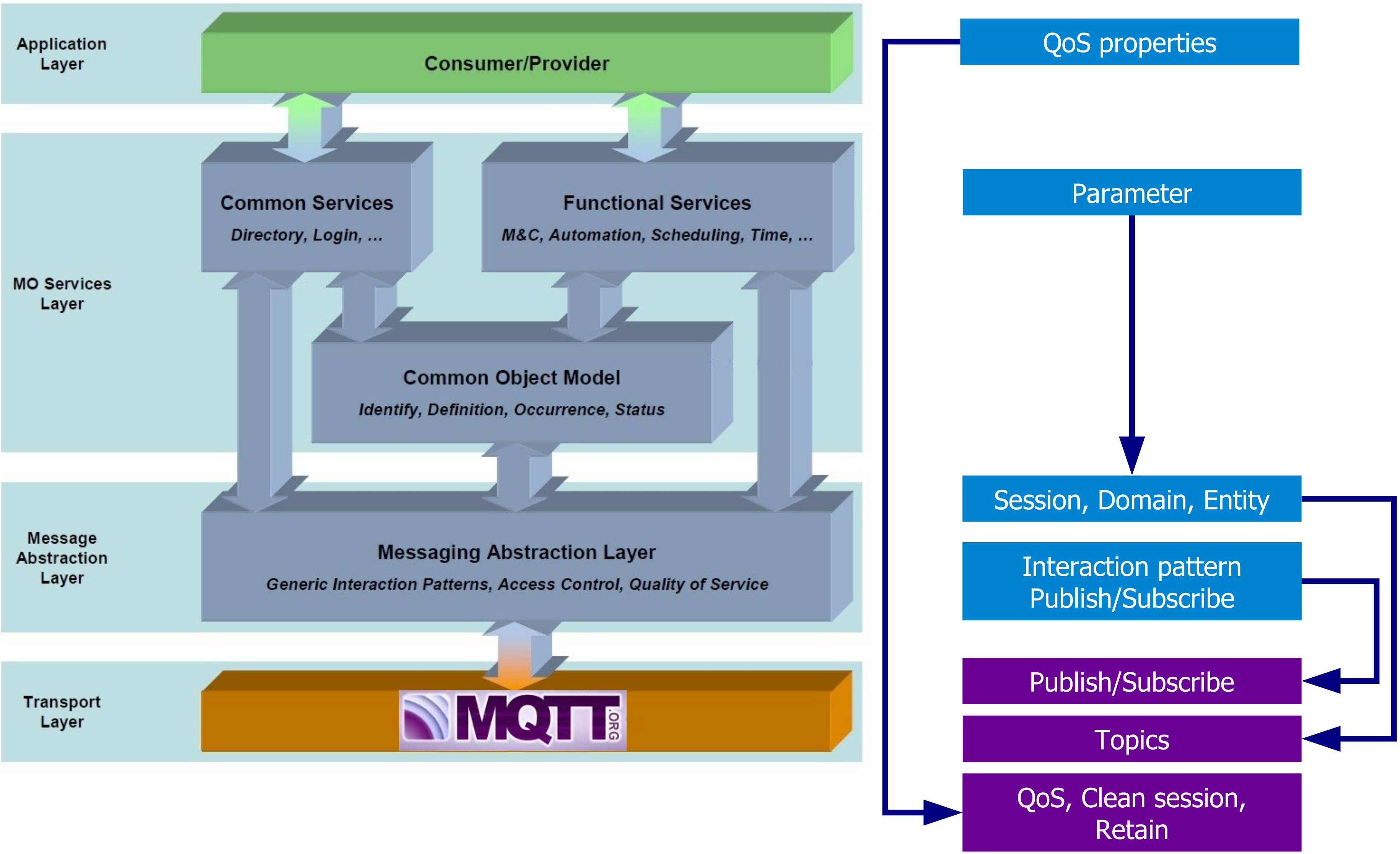
MQTT/Joram

- Provided by JoramMQ
- Fully supports MQTT v3.1 (and upcoming v3.1.1)
  - ⇨ QoS levels
    - QoS 0, 1, 2 and the Clean Session flag
  - ⇨ Topic
    - Hierarchies, wildcards, dynamic topic
    - Retained messages
- Administration tools and security mechanisms
  - ⇨ Topic access rights
- MQTT clients interoperate with JMS and AMQP clients
  - ⇨ Example: publish with MQTT and subscribe with JMS
    - Benefit from JMS 2.0 “shared subscriptions” (parallel consumers)

Scalability with the number of clients



Mapping from MO to MQTT



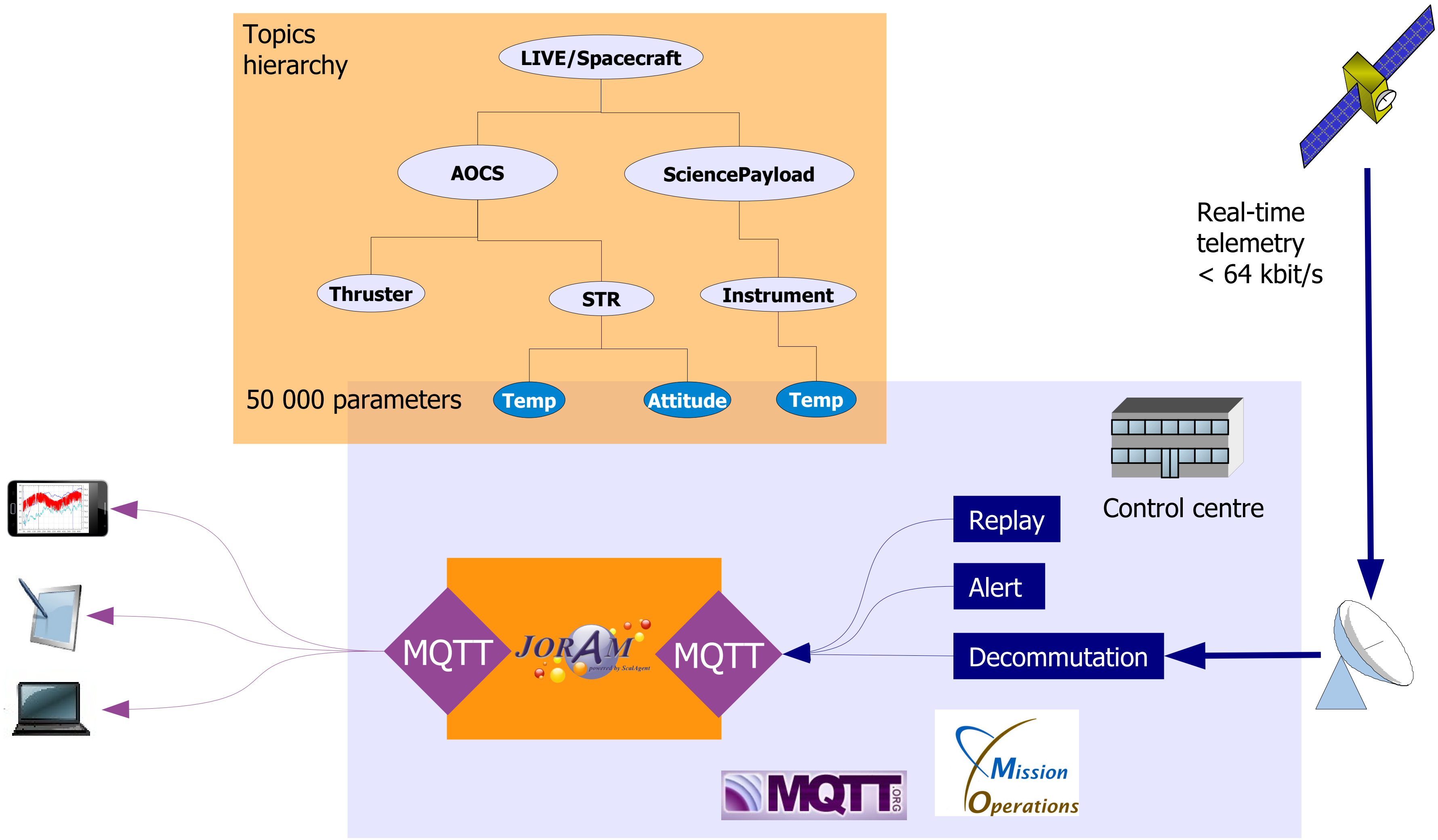
Mapping to MQTT topics

- MO data are published in a *domain* for a given *session*
  - ⇨ A domain is a path, similar to an MQTT topic name
    - ↳ Identifies a subsystem or device
      - Example: “spacecraft/AOCS/STR”
  - ⇨ A session is a name identifying the execution context
    - ↳ LIVE, REPLAY (historical data), SIMUL (test data)
- Published parameters have a name and a definition id
  - ⇨ Definition: type and unit of a parameter
- Resulting MQTT topic format:
  - ⇨ <session>/<domain>/<param>/<def>
  - ⇨ “LIVE/spacecraft/AOCS/STR/Attitude/671”

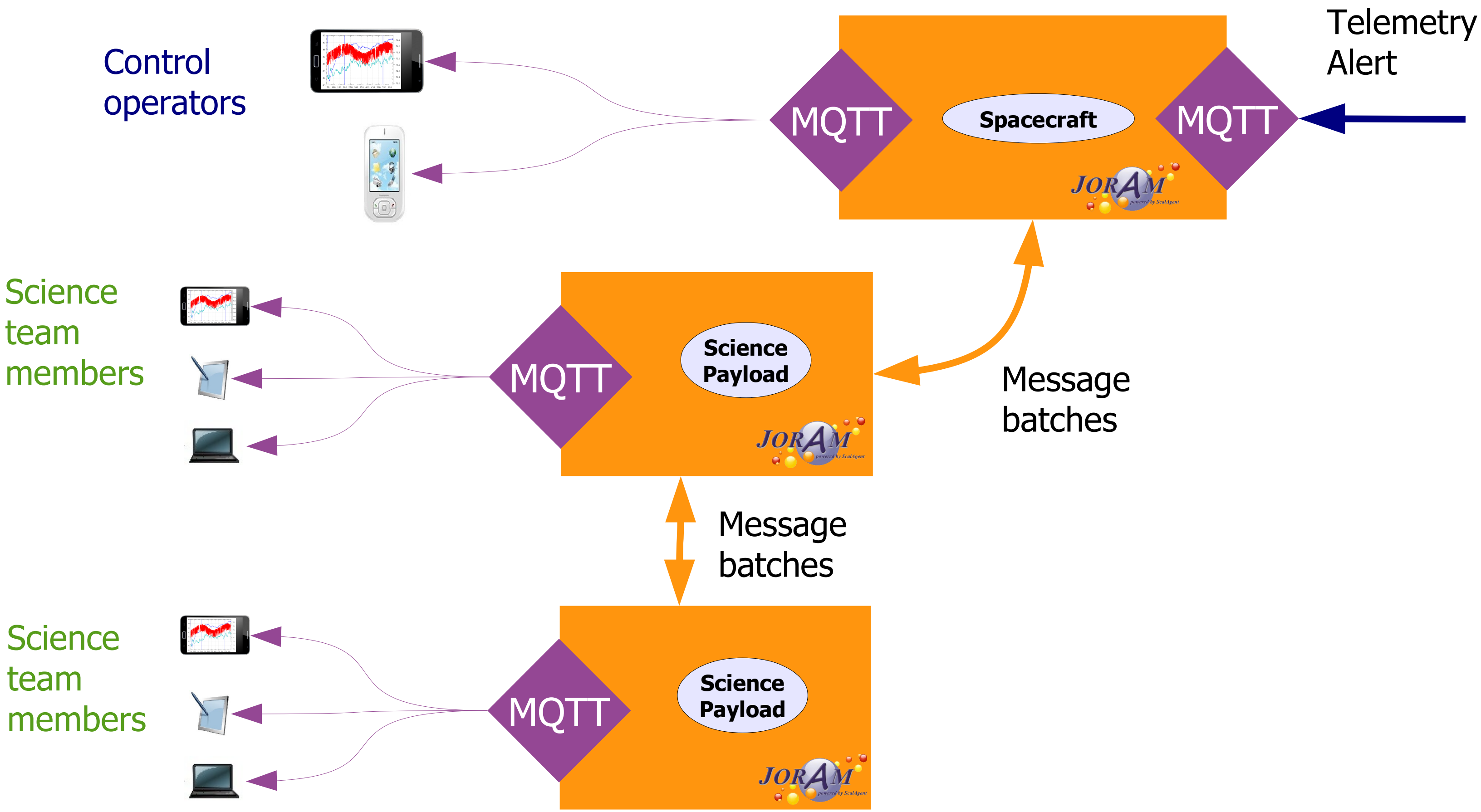
Mapping to MQTT QoS, clean session, retain

- Best Effort
  - ⇨ Real-time telemetry data (use case #1)
    - ↳ Data may be dropped in order to keep up with real-time
- At least once
  - ⇨ Payload telemetry (use case #2)
  - ⇨ Alerts (use case #1)
- Exactly once
  - ⇨ Alerts (use case #1)
    - ↳ If alert not idempotent
- Sessions should not be *cleaned*
  - ⇨ Benefit from time decoupling and durable subscriptions
- All messages should be *retained*
  - ⇨ No need to retrieve a snapshot to have the current data values

Real-time telemetry published with MQTT



Scalability with the number of MQTT clients



More information about MQTT with Joram

- JoramMQ offering by ScalAgent
  - ⇨ http://mqtt.jorammq.com