Upgrade of the CERN RADE Framework Architecture
Using RabbitMQ and MQTT
M. K. Miskowiec, O. O. Andreassen, F. Marazita
CERN, Geneva, Switzerland

AMQP (Advanced Message Queuing Protocol) was originally developed for the finance community as an open way to communicate the vastly increasing over-the-counter trace, risk and clearing market data, without the need for a proprietary protocol and expensive license. In this paper, we explore the possibility to use AMQP with MQTT (Message Queue Telemetry Transport) extensions in a cross platform, cross language environment, where the communication bus becomes an extendible framework in which simple/thin software clients can leverage the many expert libraries at CERN.

The RADE Framework
- Interface LabVIEW based equipment and software with CERN technical infrastructure
- Distributed architecture with several application servers hosting analysis libraries and dedicated communication

RabbitMQ
- Open source, lightweight message broker
- Supports various communication channels
- Data shared using protocols like AMQP, MQTT
- Benefits: load balancing, job distribution, cross-communication, cross-platform

AMQP and MQTT Protocols
- Openly published wire specification for asynchronous messaging
- Standard protocol for message-oriented middleware
- Machine-to-machine connectivity protocol
- Lightweight messaging transport
- Used in industry for endpoint communication

Service Layer
The RADE CORE Service is a generic Java based virtual class that extends all the communication layers needed to connect to the CERN technical infrastructure.

Data Layer
The data layer contains five main queues:
- Error queue
- RADE Server status log queue
- Core request queue
- MQTT adapter queue
- Data queue

Application Layer
Clients from the application layer communicate with the RADE CORE service via the entity layer.

Entity Layer
The entity layer proxies’ data to a free broker using a designated routing key.

The new system architecture:
- The service layer
- The application layer
- The entity layer
- The data layer

LabVIEW Client
In the test system, all LabVIEW clients are connected and send their request to dedicated broker using the MQTT protocol.

RabbitMQ Broker
Through the Broker the request is sent to an appropriate service executed depending on the request message payload.

RADE Java Services
The requested information is sent back to the broker using AMQP protocol, and then relayed to the requesting client.

- Improved scalability, redundancy, performance and development flexibility for our applications.
- Lightweight, open source protocols provide cross platform and cross language communication.

- Cluster based backend such as RabbitMQ gives us a safer and more robust architecture.
- Moving forward towards more standardized and efficient applications with smaller footprints.