Development of MQTT-Channel Access Bridge

Jiro Fujita, Michael Cherney, Creighton University, Omaha, Nebraska
Dmitry Arkhipkin, Jerome Lauret, Brookhaven National Laboratory, Upton, New York

Abstract

The integration of the Data Acquisition, Offline Processing and Hardware Controls using MQTT has been proposed for the STAR Experiment at Brookhaven National Laboratory. Since the majority of the Control System for the STAR Experiment uses EPICS, this created the need to develop a way to bridge MQTT and Channel Access bidirectionally. Using CAFE C++ Channel Access library from PSI/SLS, we were able to develop such a MQTT-Channel Access bridge fairly easily. The prototype development for MQTT-Channel Access bridge is discussed here.

STAR Detector

Most of the STAR (Solenoidal Tracker At RHIC) Experiment Control System has been based upon the EPICS (Experimental Physics and Industrial Control System) from the beginning. Currently roughly 60,000 parameters are controlled and monitored with EPICS.

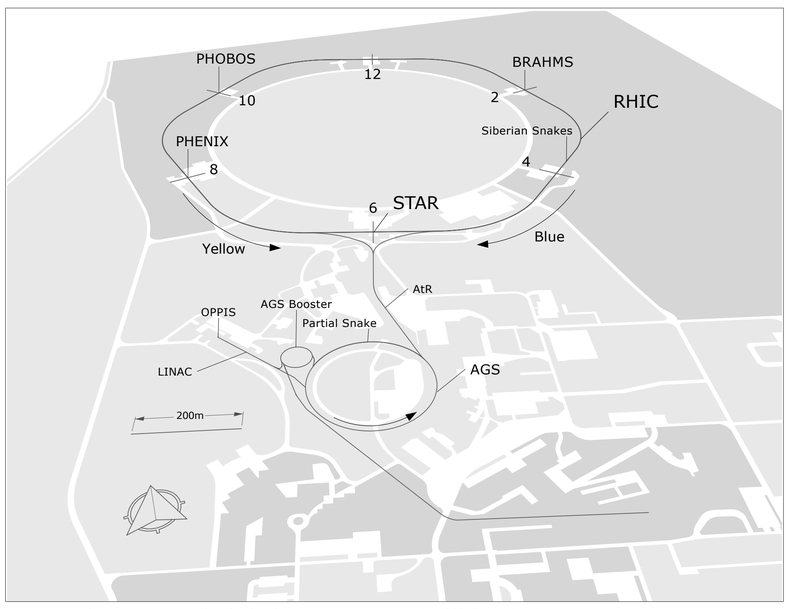

Figure 1: Relativistic Heavy Ion Collider & STAR Detector at Brookhaven National Laboratory, Upton, NY, USA

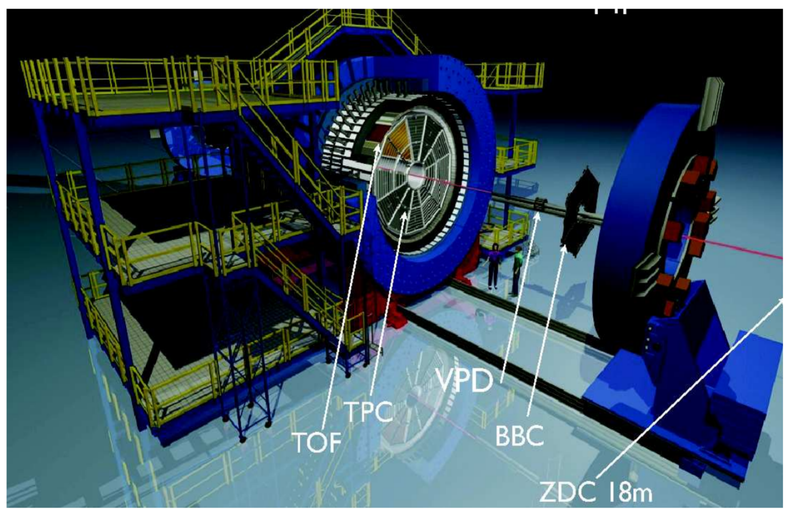

Figure 2: Schematic Drawing of the STAR Detector

Motivation

- STAR had adapted DAQ/Offline/Slow Control integration based on MQTT [1]
- Current Slow Control is based mostly on EPICS since the beginning of the Experiment
  - Bridge EPICS Channel Access and MQTT in both direction
- EPICS stays as it is for the existing control systems
- General concept of what has been proposed at STAR Experiment
  - Send/Receive MQTT messages in JSON
  - Receive/Send Control Data in Channel Access
- It is planned to use MQTT natively for new detector subsystems.

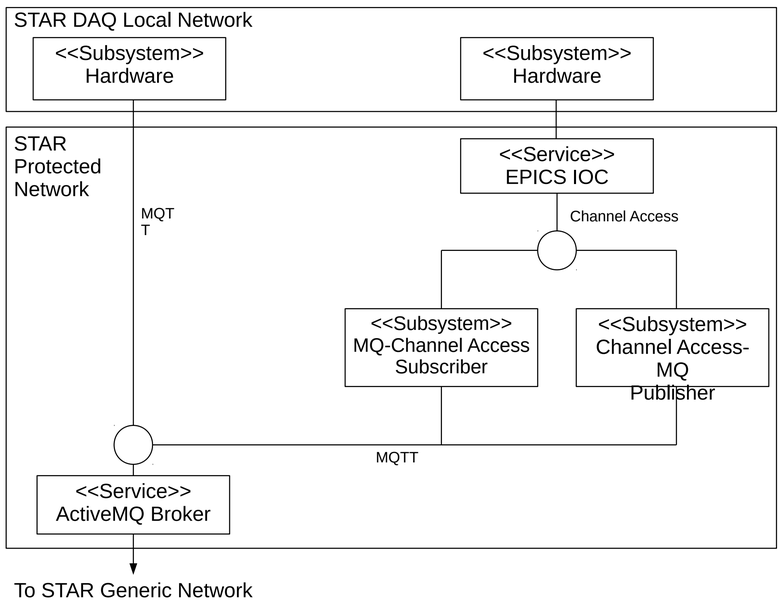

Figure 3: Schematic Drawing of the STAR Integration Plan

MQTT [2]

- Originally developed by IBM and Eurotech in 1999
  - IBM & Eurotech donated MQTT to Eclipse project in 2011
  - MQTT v3.1.1 is ISO standard as of 2016 (ISO/IEC PRF 20922)
- Runs on top of TCP/IP (as well as UDP and ZigBee)
- Relatively simple and easy to write/work with
- Low overhead & easy to work with
- Quality of Service
- Widely used by Internet of Things (IoT) and other places
- MQTT can easily handle very large volume of data (beyond 100M/sec)
- Very different from Channel Access

MQTT Concept

Message Broker

- Acts as a communication point
- Can be authenticated
- Quality of Service (QoS)
  - QoS0 — no checking
    - Fast delivery, but no guarantee
  - QoS1 — guarantees to deliver at least once
    - Safe delivery, but slow
  - QoS2 — guarantees to deliver once and once only
    - Safest delivery, but slowest
- Message
  - The “data” that clients publish/subscribes
  - Only in ASCII format
  - payload agnostic; MQTT does not enforce the format of the message it is carrying
  - Grouped by topic
- Topic
  - Routing information to the broker (e.g. “Status”, “Voltage”, “TPC”, “Beamline1”)

Clients

- Publish
  - Devices publish specific message grouped by topic
- Subscribe
  - Devices could also subscribe to specific message by topic

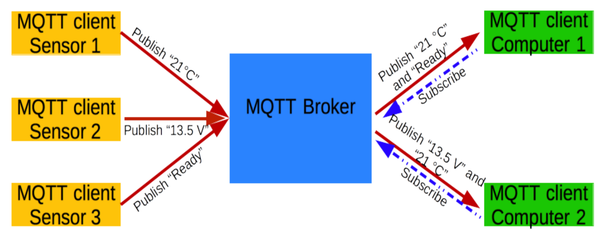

Figure 4: MQTT Concept Diagram
MQTT Broker acts as the Central Hub to all the clients

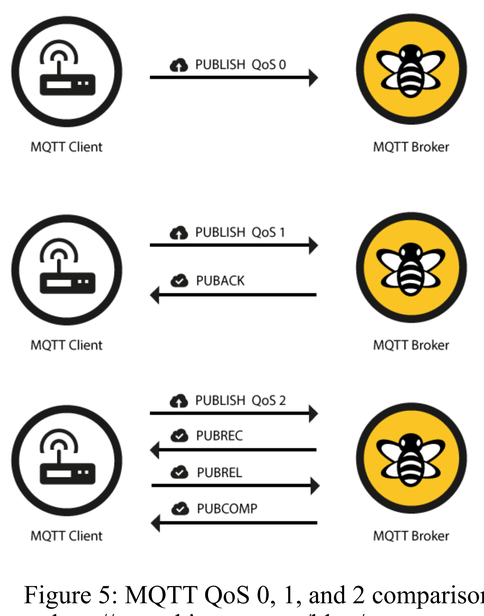

Figure 5: MQTT QoS 0, 1, and 2 comparison
Image from: http://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels

Requirement & Tools Used

- C (or C++)
  - This was a request from the DAQ expert to the control group
  - CAFE library was chosen over standard portable Channel Access Library
- Paho [3] MQTT Library
  - Appears to support C/C++ well http://www.eclipse.org/paho
- Apache ActiveMQ Apollo [4] for the message broker
  - The Offline/DAQ integration has already been using Apollo as the message broker https://activemq.apache.org/apollo

MQTT-Channel Access Bridge

- Written in about 2 weeks, including setting up the broker and MQTT test programs
- No prior knowledge/experience of MQTT
- Fairly easy to write a program using MQTT
- First Portable Channel Access Application written by the STAR Control Group
- Two different components
  - Publisher — Channel Access to MQTT
  - Subscriber — MQTT to Channel Access

Publisher Component

- Publishes Channel Access data to MQTT broker
- Written in C using Portable Channel Access library
- Sends Channel Access data in MQTT in JSON
  - { "PV": "PVname", "Host": "Hostname", "data": "PV value"}

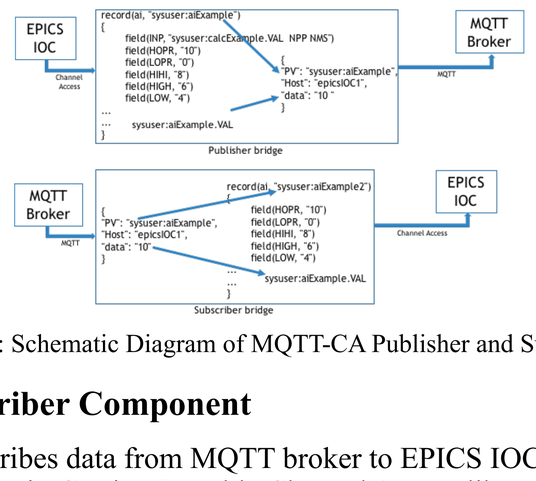

Figure 6: Schematic Diagram of MQTT-CA Publisher and Subscriber

Subscriber Component

- Subscribes data from MQTT broker to EPICS IOC
- Written in C using Portable Channel Access library
- Receives MQTT in JSON, broadcasts in Channel Access
  - { "PV": "PVname", "Host": "Hostname", "data": "PV value"}

CAFE Library

- Modern C++ Interface for Channel Access from PSI [5]
  - Easy to write compared to Standard Channel Access Library
- Making it easy for students to participate in the project
- Hope for synergy and increased acceptance within the EPICS community...

```
#include <stdio.h>

#include <cafe.h>

int main(int argc,char **argv)
{

    using namespace std;
    CAFE * cafe = new CAFE();
    int status;
    CAFEStatus cstat;
        try {
     status=cafe->init( );
    }
    catch (CAFEException_init &e) {
        cout << e.what() << endl;

        exit(1);
    }

    if(argc != 2) {
            fprintf(stderr,"usage: cafeExample pvname\n");
            exit(1);
    }

    status=cafe->set(argv[1], 100);
    if (status!=ICAFE_NORMAL) {
      cstat.report(status);
    }
    cafe->printStatusIfError(cafe->getInfo().getHandleFromPV(argv[1]),status);

    double d;
    status=cafe->get(argv[1], d);
    if (status==ICAFE_NORMAL) {
        cout << argv[1] << " has value= " << d << endl;
    }
    else {
        cafe->printStatusIfError(cafe->getInfo().getHandleFromPV(argv[1]),status);
    }
    return(0);

}
```

```
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include "cadef.h"

int main(int argc,char **argv)
{
    double      data;
    chid  mychid;

    if(argc != 2) {
            fprintf(stderr,"usage: caExample pvname\n");
            exit(1);
    }

SEVCHK(ca_context_create(ca_disable_preemptive_callback),"ca_context_c
reate");

SEVCHK(ca_create_channel(argv[1],NULL,NULL,10,&mychid),"ca_create_c
hannel failure");
    SEVCHK(ca_pend_io(5.0),"ca_pend_io failure");
    SEVCHK(ca_get(DBR_DOUBLE,mychid,(void *)&data),"ca_get failure");
    SEVCHK(ca_pend_io(5.0),"ca_pend_io failure");
    printf("%s %f\n",argv[1],data);
    return(0);
}
```

Figure 7: Comparison of a simple caget equivalent in CAFE and Standard Channel Access library. While CAFE code is longer, it is much easier to understand.

Performance Evaluation

- Initial testing was done in sending thousands of MQTT data and/or Channel Access data per seconds
  - Several computers were involved (all in the local network, some in different subnet)
    - EPICS IOC host computers
    - More than one IOC computers were used for testing
    - Computer running MQTT bridge programs
    - MQTT data sender/receiver computer running simple MQTT publisher/subscriber programs
    - MQTT broker (Apache Apollo) computer in different subnet
- It appears to withstand both directions at least up to about 1000 messages/second or so — more that what STAR needs
  - Not very quantitative measure, as we had no idea how to quantify the number easily, as there are many different factors involved (computer performance, network speed, etc)
- For the system used for production currently, it can withstand around 5000 messages/second at peak with around 1000 messages/second average

Future Plan

- More stability testing
- Integration with the rest of the STAR MQTT project

References

[1] D Arkhipkin and J Lauret, “STAR Online Framework: from Metadata Collection to Event Analysis and System Control” 2015 *J. Phys.: Conf. Ser.* **608** 012036 https://doi.org/10.1088/1742-6596/608/1/012036
[2] MQTT: http://mqtt.org/
[3] Paho: http://www.eclipse.org/paho/
[4] Apache ActiveMQ Apollo: https://activemq.apache.org/apollo/
[5] CAFE: https://ados.web.psi.ch/cafe/index.html

Acknowledgements

- US Department of Energy Office of Science
- STAR Collaboration
- Creighton University College of Arts & Science
- Dr. Jan Chrin for CAFE
- EPICS Community