A Messaging Based Data Access Layer for Client Applications

James Patrick (patrick@fnal.gov), Accelerator Division, Fermilab, Batavia, IL

Abstract

The Fermilab Accelerator Control System has recently integrated use of a publish/subscribe infrastructure as a means of communication between Java client applications and data acquisition middleware. This supersedes a previous implementation based on Java Remote Method Invocation (RMI). The RMI implementation had issues with network firewalls, misbehaving client applications affecting the middleware, portability to other platforms, and lack of authentication. The new system uses the RabbitMQ implementation of the AMQP messaging protocol and broker architecture. This decouples the client and middleware, is more portable to other languages, and has proven to be much more reliable. A Java client library provides for single synchronous operations as well as periodic data subscriptions. This new system is now used by the synoptic display manager application as well as a number of new custom applications.

Fermilab

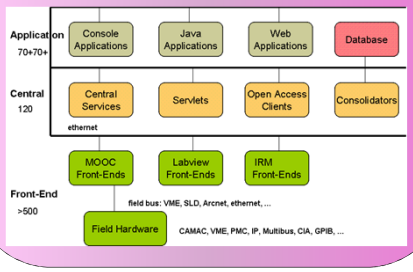
-Large multipurpose accelerator complex
-Neutrino beams to experiments on-site and in Minnesota
-Hadron beams to other fixed-target experiments + test beams
-Standalone superconducting electron linear accelerator under construction (ASTA)
-Standalone superconding proton linear accelerator also under construction (PXIE)
-Muon beams under construction using old anti-proton source
-Neutrino beam to LBNE experiment in South Dakota in planning stage
-High intensity superconditing proton accelerator (Project X) in planning stage

Control System

-Single unified control system for the entire complex
  -Including ASTA and PXIE superconducting test accelerators
-Known as **ACNET**
-Three-tier system
-Front-end computers attached to field hardware
-Middle layer services
  -alarms, database, archving and other non-GUI services
  -Acquire data from front-ends via UDP based "ACNET" communication protocoll
    -Java based Data Acquisition Engines (DAEs) for Java applications
-Applications
  -C++ applications run under integrated console environment
  -Custom Java applications not restricted to the control system network
  -Synoptic display manager applications
    -Viewer and drag and drop builder written in Java

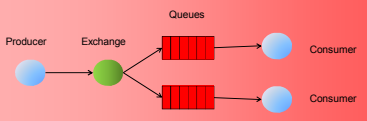
-Data transmitted between middle layer DAEs to Java applications via Java RMI
  -Serious issues with firewall configuration, load balancing, authentication & authorization, automatic failover in case of DAE failure, etc.

-A new method for this communication based on the AMQP standard has been developed to address all of these issues



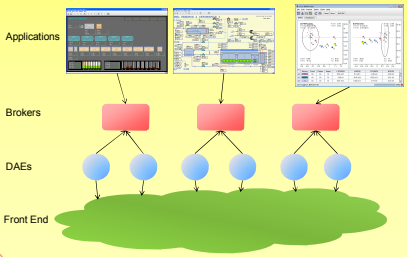
AMQP

-Advanced Message Queing Protocol
-Originated in financial industry
-Standard that defines wire protocol, queuing, routine, reliability and security
-Language independent, unlike Java Messaging Service (JMS)
-Different implementations should interoperate, also unlike JMS
-Producers, Consumers, Brokers
-Producers publish messages to an "Exchange" in the broker
-Broker routes messages to "Queues"
  -Fanout, direct, round-robin, topic based strategies
-Consumers bind to queues and receive messages
-AMQP 1.0 is much different and more limited than < 1.0
  -We use 0.8/0.9.1
-Producers and Consumers only communicate with the broker
  -They don't need to know each others identity or where they are running
  -Messages are routed to queues by the broker based on a routing key
-Hence very simple for firewalls



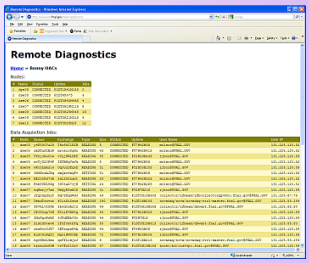
ACNET Implementation

-Applications, Data Acquisition Engines connect to broker
-Application publishes data request to DAE
  -Creates dedicated exchange/queue for reply
  -Sends Kerberos ticket in message header for authorization
-DAE acquires data by standard ACNET methods
-DAE publishes data back to application via broker
-Pair of DAEs per broker, currently operate with 3 sets
-"Loose" cluster, we currently don't use AMQP clustering
-Heartbeat mechanism detects stalled or failed DAE,
  -Resubmits request to different one
-Load balancing by random distribution of requests
-Structured data serialized via ACNET Protocol Buffers rather than AMQP



Monitoring

-The DAEs make available internal statistics, overall state information, and information about all the requests it is processing via Java Management Extensions (JMX)
-A servlet periodically reads this information and makes it available via a web page
-A user may drill down from the summary page below to get detailed information
-Newer versions of the RabbitMQ broker expose internal information via http
  -We are in the process of migrating and will create web pages with this information



Experience

-This system is now used for all synoptic display manager applications
-A modest number of new custom applications use it
-Older applications using Java RMI have not been rewritten
-Overall the system performs very well
-RabbitMQ is very reliable and robust
  -However on the rare occasions when it isn't messages are sparse and cryptic
-Recently usage has dramatically increased and resulted in overload problems
  -An additional broker/DAE set was added
  -Work is in progress on better load balancing. Data requests can vary widely in their required data throughput
-Although the AMQP 0.8/0.9.1 protocol we use has been superceded by the very non-backwards compatible AMQP 1.0, RabbitMQ continues to support and enhance their products for this older version.

Conclusions

The Fermilab Accelerator Control System ACNET has recently introduced the RabbitMQ implementation of the AMQP messaging system to transmit data between the middle layer and Java applications. This has solved major issues with the previous method based on Java RMI such as robustness, firewall configuration, load balancing, and authorization. Work continues on migration to newer versions of the data broker, and improved load balancing and monitoring of the system.