EPICS-DDS*

N.Malitsky, J.Shah, BNL, Upton, NY 11973, U.S.A.
N.Hasabnis, Stony Brook University, Stony Brook, NY 11794, U.S.A.

Abstract

This paper presents a new extension to EPICS, approaching the Data Distributed Service (DDS) interface based on the Channel Access protocol. DDS is the next generation of middleware industrial standards, bringing a data-centric publish-subscribe paradigm to distributed control systems. In comparison with existing middleware technologies, the data-centric approach is able to provide a consistent consolidated model supporting different data dissemination scenarios and integrating many important issues, such as quality of service, user-specific data structures, and others. The paper considers different features of the EPICS-DDS layer in the context of the high-level accelerator environment.

RATIONALE

The Accelerator Online Model (AOM) is a collection of characteristics and associated theoretical approaches required for the successful analysis and control of accelerator performance. Its particular implementation depends on many factors, such as scope and complexity of operational tasks, type of accelerator control system, and others. Usually, AOM is built as one or several middle-layer servers of a three-tier application environment encompassing a low-layer with distributed front-end computers controlling physical devices and an open collection of high-level thick and thin client applications. Despite the common infrastructure, requirements of each layer are different. As a result, the middle-layer servers in present accelerator facilities work additionally as gateways connecting at least two communication protocols and interfaces.

This paper proposes a new extension named EPICS-DDS for developing a homogeneous high-level application environment (see Figure 1) based on the EPICS distributed infrastructure [1] and the Data Distribution Service (DDS [2]) data-centric model. In this approach, different layers and components communicate via the common EPICS low level protocol, Channel Access (CA), and the high-level interface between servers and applications is implemented at the additional DDS-oriented layer. In context of the DDS specification, the different middle layers servers are considered as the corresponding DDS publishers designed to provide the states of the associated data structures shared by high-level client-subscribers.

The rest of this report is broken into two parts. Section 2 gives a brief overview of the DDS model and section 3 describes the different features of the EPICS-DDS extension using the list of dedicated examples.

*Work supported by DOE contract DE-AC02-98CH10886

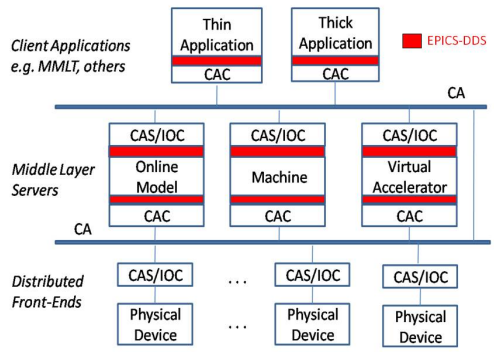

Figure 1: High-Level Application Environment based on the EPICS-DDS extension.

DDS DATA-CENTRIC ARCHITECTURE

DDS is the next generation of middleware industrial standards, bringing a data-centric publish-subscribe (DCPS) architecture to distributed control systems. The overall conceptual model is shown in Figure 2 and encapsulates the following major concepts:

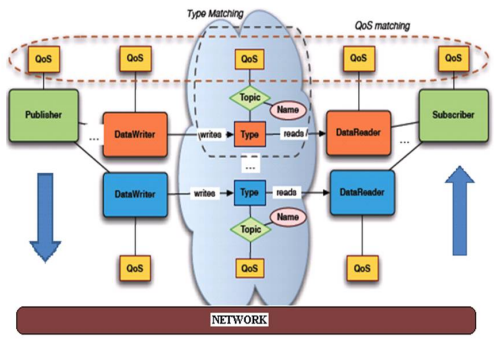

Figure 2: Data-Centric Publish-Subscribe Model [3].

- Topics of Typed Global Data Space: a logical data space in which applications read and write data decoupled in space and time.
- Data Writer: data producer of the given topic.
- Data Reader: data consumer of the given topic. Data reader can obtain data via two ways: (1) listener-based asynchronous mechanism and (2) waitset-based synchronous approach that blocks the application until designated conditions are met.
- QoS policies: a rich set of characteristics that define the behavior of the DDS systems (such as reliability, liveliness, durability, latency budget, etc.)

Similar to the OMG CORBA specification, DDS is language neutral and can be implemented in programming languages like Java and C++. In contrast with CORBA, the DDS interface is also protocol neutral, facilitating its deployment in different distributed systems. Moreover, the DDS technology extends present run-time environments with the relational-oriented model. It creates a basis for developing consistent run-time interfaces to complex hierarchical structures using the well-established software engineering techniques.

EPICS-DDS MIDDLEWARE

Both EPICS Channel Access and DDS represent unique approaches and our primary task was to identify the common concepts decorated by these different terminologies. The developer's intuition and optimism was based on the fact that both technologies have a long history of successful projects in neighboring domains. Moreover, the overall goal was more practical: understanding how the DDS data-centric model can extend the CA interface for developing high-level applications. As a result, an initial scope of the EPICS-DDS middleware was narrowed to the following three DDS concepts expressed in the CA terms:

- Topic: collection of values (or other record fields) belonging to the different PV instances of the same record type.
- Data Reader: maintainer of the CA channels for getting data from the collection of PV fields associated with the Data Reader's topic.
- Data Writer: maintainer of the CA channels for putting data to the collection of the PV fields associated with the Data Writer's topic.

These suggested associations explicitly identify the core architecture of EPICS-DDS setting the DDS Typed Global Data Space onto the EPICS I/O Controllers and considering the DDS participants as wrappers of the CA clients. Starting from this point, we consistently and incrementally continued to tackle different features of the CA interface towards the three-tier high level application environment. The present list of these features is illustrated by the following EPICS-DDS examples:

caMonitorApp: classical asynchronous example from the EPICS Application Developer's Guide [4] demonstrating the implementation of the DDS DataReaderListener based on the CA callback function. According to the DDS specification, DataReaderListener has several methods which can be selected with a corresponding status mask. This mask-based approach provides direct mapping from the DDS object-oriented model to a list of CA C-functions. Following the CA approach, all DDS DataReaderListener's methods can be divided into two uneven categories. The first category includes only one method *on_data_available* dealing with data acquisition and implemented by the CA *ca_create_subscription*. The second category is associated with monitoring of QoS policies. The current EPICS-DDS version controls the liveliness of DataWriter's with the CA function *ca_add_exception_event*.

caExampleApp: extension of another classical example from EPICS Application Developer's Guide[4], demonstrating the DDS synchronous interface and a carcass of the EPICS-DDS middle layer server. In DDS, synchronous access is represented by a WaitSet which can attach a set of ReadCondition's and waits for them to return. Each ReadCondition is associated with an individual DataReader. From this perspective, the DDS synchronous model directly corresponds to one of the major CA communication approaches allowing the accumulation of numerous requests into a single message. The original caExample was implemented as a client program. In our version, we added an IOC server extended with the application object running in the dedicated thread. This example demonstrates the most straightforward approach [5] for building the middle layer servers in the EPICS-DDS environment.

caTimeApp: implementation of the corresponding CA command line utility for benchmarking the CA-based communication. The benchmark measures the latency of receiving sequences of values. This application also highlights the limitation of the present DDS specification treating the sequences with different sizes as the different data types. The next application shows how this problem can be resolved with the PvData approach introduced in the EPICS 4 version.

PvDataApp: PvData-based implementation of the caTimeApp example. PvData is a hierarchical tree of containers designed after the Composite pattern. The leaf nodes are represented by scalars and arrays of the primitive data types, such as integer, float, etc. The composite nodes serve to accomodate nested structures. As a result, the PvData approach provides a *flexible* mechanism for building the PV-specific *fixed* data types in a run-time environment. Particularly, PvDataApp prototypes the PVFloatArray container where the number of elements is defined after creating the CA channel. This approach addresses the recent OMG request of proposals "Extensible and Dynamic Topic Types for DDS" and we expect to consider and apply it to the different user-specific types of accelerator applications.

TwissApp: example which illustrates the extension of the EPICS 3 data types with the application-specific data structures, e.g. Twiss. The approach is based on the EPICS waveform record hosting an array of characters which can be serialized/deserialized in the EPICS-DDS layer. The serialization algorithm is encapsulated in the ByteBufferCAC class designed after the corresponding class from the Java NIO package. Currently, a byte buffer is created from values of primitive types (and their compositions) by directly copying their memory blocks using *memcpy*. There are also several portable approaches, for example OMG Common Data Representation (CDR) or Google's Protocol Buffers. The

final variant however will be determined after finalizing the EPICS 4 protocol.

BuiltinTopicApp: implementation of one of the built-in topic data structures with information about the DDS participants. This example identifies a direction in the development of the EPICS-DDS full-scale directory service based on its own framework without use of the additional communication systems, such as CORBA. Moreover, the approach allows to automatically inherit the distributed and publish/subscribe features of the DDS infrastructure.

We plan to further extend this list of features with other DDS elements, such as QoS, content filtered topic, etc. However, the present EPICS-DDS version already creates a necessary basis for starting the development of the multi-tier high level application environment.

CONCLUDING REMARKS

This paper presents a new extension to EPICS, approaching the Data Distributed Service (DDS) interface based on the Channel Access protocol. The integration of these two technologies addresses five major tasks. First, DDS brings an industrial standard interface to the accelerator online environment allowing to decouple a variety of high-level applications and toolkits from the underlying low-level control systems, such as EPICS, TINE, TANGO, and others. Second, the DDS topic-oriented approach elevates the EPICS Channel Access protocol to the high-level applications replacing the additional RPC-like communication interfaces. Third, the DDS specification introduces some guidance for extending the EPICS infrastructure with the relevant set of quality of service. Fourth, DDS creates a basis of Service-Oriented Architecture (SOA) promoting decoupling of the service interfaces from their project-oriented implementations. In context of the high-level application environment, it means flexibility in selecting and connecting the most appropriate modeling algorithms and programs. Finally, the DDS technology extends the EPICS run-time environment with the relational model creating a platform for adding relational queries and integration of full-scale Data Stream Management Systems (DSMS) for data stream processing and archiving. Moreover, adherence to the relational approach facilitates the design of consistent run-time interfaces to the complex hierarchical structures according to the well-established software engineering techniques, such as object-relational mapping.

From the other side, EPICS represents *de facto* standard open-source software with a multi-year history of numerous successful projects. As a result, it creates a solid basis for developing the open source implementation of the DDS specification. Moreover, the special features of the Channel Access approach provide the advantageous means for solving the complex DDS issues, for example server-based event filtering. The new PVData concept from the coming EPICS 4 version introduces another important idea addressing the recent OMG RFP: Extensible and Dynamic Topic Types for DDS.

The positive experience gained from this project encourages us to further explore and extend the EPICS-DDS middleware in the development of the full-scale high-level accelerator application environment.

ACKNOWLEDGEMENTS

The EPICS-DDS design has been shaped and consolidated from the numerous discussions and valuable inputs of members of Control and Accelerator Physics groups of the NSLS-II project. We would like also to thank B. Dalesio and S. Stoller for their support and S. Shasharina and N.Wang from the Tech-X Corporation for their contributions.

REFERENCES

[1] L. Dalesio *et al*., "The Experimental Physics and Industrial Control System Architecture," ICALEPCS'93, Berlin, Germany, October 1993, http://www.aps.anl.gov/epics/

[2] OMG, "Data Distribution Service for Real-time Systems, Version 1.2," formal/07-01-01, http://www.omg.org/cgi-bin/doc?formal/07-01-01

[3] A. Corsaro, "Advanced DDS Tutorial," Workshop on Distributed Object Computing for Real-Time and Embedded Systems, July 2008, Washington, DC, USA

[4] M.Kraimer *et al*, "EPICS Application Developer's Guide," January 2009.

[5] J.Hill, EPICS Tech-Talk.