OPC UNIFIED ARCHITECTURE WITHIN THE CONTROL SYSTEM OF THE ATLAS EXPERIMENT

Piotr Nikiel, Ben Farnham, Sebastien Franz, Stefan Schlenker, CERN, Geneva, Switzerland
Henk Boterenbrood, NIKHEF, Amsterdam, The Netherlands
Viatcheslav Filimonov, PNPI, Gatchina, Leningrad District, Russia

Abstract

The Detector Control System (DCS) of the ATLAS experiment at the LHC has been using the OPC DA standard as an interface for controlling various standard and custom hardware components and their integration into the SCADA layer. Due to its platform restrictions and expiring long-term support, OPC DA will be replaced by the succeeding OPC Unified Architecture (UA) standard. OPC UA offers powerful object-oriented information modelling capabilities, platform independence, secure communication and allows server embedding into custom electronics. We present an OPC UA server implementation for CANopen devices which is used in the ATLAS DCS to control dedicated IO boards distributed within and outside the detector. Architecture and server configuration aspects are detailed and the server performance is evaluated and compared with the previous OPC DA server. Furthermore, based on the experience with the first server implementation, OPC UA is evaluated as standard middleware solution for future use in the ATLAS DCS and beyond.

OPC DA USE IN THE ATLAS DCS

The ATLAS DCS has been using OPC DA (OPC Data Access) since initial implementation of the DCS [1]. The OPC DA protocol has been used to provide:

- communication between the SCADA layer and the CANopen [2] OPC DA server (which would later on communicate through CAN bus with numerous (more than 5000) IO nodes called ELMB (Embedded Local Monitoring Boards [1]) using the CANopen protocol
- communication between the SCADA layer and various types of commercially available, off-the-shelf equipment (e.g. industrial power supplies sold by WIENER Plein&Baus GmbH, ISEG and CAEN companies)

The OPC DA is approaching obsolescence with dwindling development activity and decreasing level of support. The fact that OPC DA is based on MS Windows proprietary technology (i.e. COM/DCOM [3]) is considered one of the biggest obstacles in modernisation of the ATLAS DCS. Furthermore, this reduction in active development on the OPC-DA stack and OPC-DA toolkits impedes its evolution to adapt to industry trends such as multi-core processing etc.

Therefore both its scalability and performance lags behind what could be obtained using the same hardware.

OPC UA IN THE DCS: MOTIVATION AND APPLICATIONS

Natural update from OPC DA

Many of the unfavourable properties of OPC DA have been addressed and improved in OPC UA[4]. Therefore a migration to OPC UA is a natural modernisation path.

Benefits of OPC UA for the DCS

In addition to modernising a system, OPC UA brings many qualities that the ATLAS DCS middleware will be able to profit from:

- independence from operating system. OPC UA Reference Stack (providing lower layers of OPC UA internally to OPC UA toolkits) is delivered in standard C. OPC UA toolkits (which de facto provide OPC UA connectivity in OPC UA servers) are delivered in many programming languages (standard C++, Java, .NET, Python and other [5]). Moreover ubiquitous TCP/IP is communication technology used by OPC UA, so there's full portability of OPC UA to any modern network aware operating system one may think of.
- robust data modelling capabilities – with OPC UA structured information can be stored in bespoke datatypes. These structures are created using complex data types that may involve object-oriented techniques including type hierarchies and inheritance [6]. Moreover type information is fully exposed to the client and may be accessed like type instances [6].
- enabling OPC UA connectivity directly to custom hardware – thanks to dependence only on standard C and C++, some common cryptography functions and TCP/IP stack, one can run OPC UA servers on any (even resource constrained) embedded computer that has enough memory and TCP/IP connectivity. A number of successful miniaturised and embedded controllers with OPC UA servers are already on the market.
- thanks to PKI (Public Key Infrastructure) support in OPC UA, one can use state-of-the-art security in OPC UA based control infrastructure. Potentially a successful attack on SCADA system could cause loss of data and may damage both hardware and reputation.

ISBN 978-3-95450-139-7

Applications of OPC UA in the DCS

OPC UA components used in the ATLAS DCS are:

- the OPC UA CANopen server (presented in detail below),
- the SCADA layer OPC UA client (standard component of WinCC OA SCADA system)
- custom OPC UA clients (for maintenance and debug)
- the UaExpert software from Unified Automation (for maintenance and debug)
- OPC UA as the DCS-wide communication technology (planned)

THE OPC UA CANOPEN SERVER

Typical Use Case of the OPC UA CANopen Server

In the most common case the server would be a link between SCADA system (by OPC UA interface) and one or many hardware devices on (possibly multiple) CAN buses communicating using the CANopen protocol. Such a typical arrangement is presented in Figure 1.

Requirements

Principal design decisions were made before development of the server was started. The following requirements were identified:

- portability (Linux and MS Windows compatibility)
- performance (at least 10 thousand readout values processed per second with CPU load less than 50%)
- independence of hardware interfaces (plug-in support is required for various CAN bus gateway hardware provided by various vendors.)
- conformance with the CANopen standard
- flexible configuration
- use of all of OPC UA core advantages

Architecture

Figure 2 shows software components of the server and data flow among them. The salient points are:

- The configuration module parses XML configuration file and sets up the AddressSpace accordingly. At setup time the module establishes bindings of items of the AddressSpace with particular objects from the CANopen object dictionary.
- The AddressSpace stores data (e.g. most up to date value of given readout channel) and metadata (e.g. type of a given readout channel is double-precision floating point and it is read-only).
- The NodeManager handles node management: NodeGuard protocol (which periodically checks connectivity and state of configured CANopen nodes), Sync protocol (which requests updates of measurement), start/stop/reset requests.
- CANopen interface module composes transmit messages in the CANopen frame format and decomposes received messages from the CANopen frame format.
- CAN Interface modules communicate via hardware components.
- Hardware components are plug-ins for various types of CAN hardware interfaces.

Configuration System

The configuration system of the server:

- can handle any CANopen object dictionary
- bases on XML (eXtensible Markup Language) and XSD (XML Schema Definition) industry technologies
- the part responsible for AddressSpace setup may be reused in any further OPC UA server project
- uses automatic XML Schema Definition to C++ code generation, which ensures coherency and minimises coding effort

OPC UA Toolkit Selection

We evaluated two C++ OPC UA Toolkits: Softing OPC UA Toolkit and Unified Automation OPC UA Toolkit. At the time of evaluation Unified Automation's toolkit was chosen because it offered more complete implementation of OPC UA Standard and because its source code was available.

PERFORMANCE MEASUREMENTS

It was essential to verify performance of our OPC UA based readout chain and identify bottlenecks. The measurements were done in almost full production environment, the only difference was that data updates were not coming from real CANopen devices but generated by software in realtime with controlled throughput. We took CPU load at given throughput as a figure of merit.

The following conclusions were drawn from the measurements:

- the server itself satisfied performance requirements (about 20% measured CPU load for 10k updates per second).
- we couldn't observe any performance bottlenecks of the server itself. It scaled beyond one CPU.
- our subscription client used very little CPU processing power (about 5% for 10k updates per second) and at least few times less than the production client. It

ISBN 978-3-95450-139-7

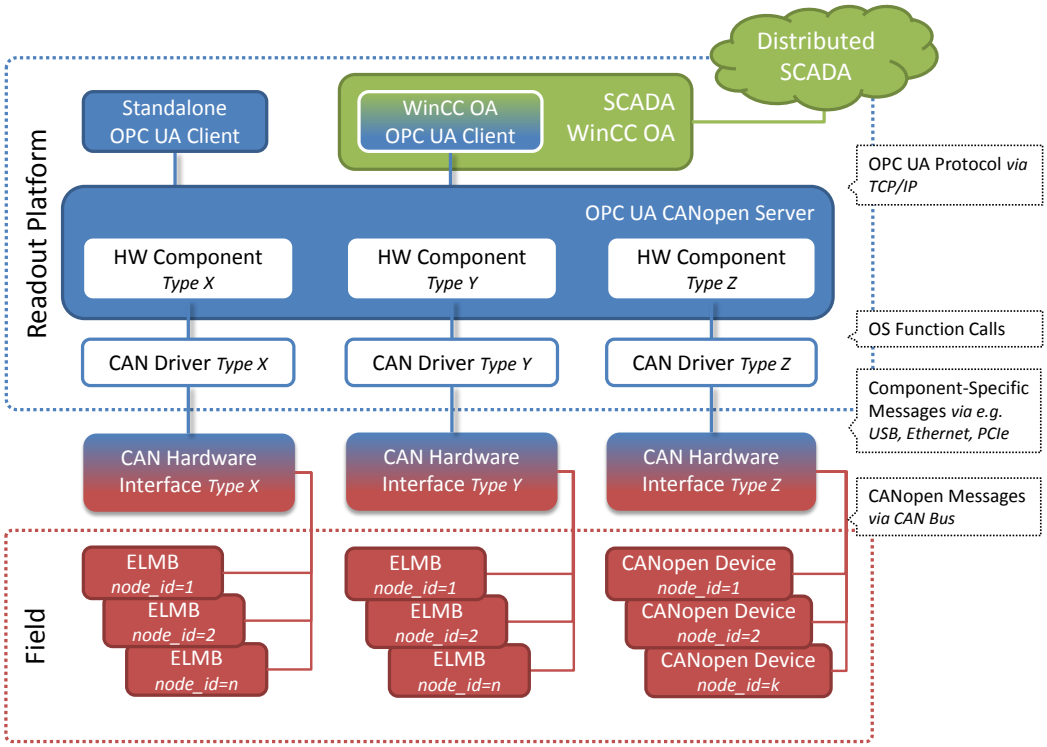

Figure 1: A schematics showing typical use case of OPC UA in the ATLAS DCS.

proves that the CPU power observed on the production client is spent on interfacing the SCADA layer itself rather than on handling OPC UA communication.

- we observed performance bottlenecks in the production client. First performance bottleneck was visible at update rate of around 40k updates per second when single production client reached 100% CPU consumption and buffered excess traffic instead of processing it. We then split OPC UA data items equally among 3 production clients instead of one. That helped us to push performance limit up to about 120k updates per second when again all three clients reached combined 300% CPU consumption. Both limits are far above the requirements though.

ONGOING DEVELOPMENT WORK

Further expansion of OPC UA use in the ATLAS DCS is being currently evaluated.

OPC UA as a Common Middleware Communication Protocol

Currently the most common middleware communication protocol in the DCS is the DIM protocol [7]. OPC UA could be considered an alternative to the DIM protocol bringing software components unification, better modelling capabilities and security. Moreover configuration of all unified systems OPC UA servers could be stored as XML files. The configuration component of the server (described in the paper) could be reused to generate OPC UA AddressSpace from XML config files of common format.

The next step in unification is implementation of OPC UA servers embedded in numerous FPGA-based electronic modules used in the ATLAS. Nowadays they use custom protocols to communicate with the DCS. Contradictory preferences are identified:

- FPGA-based modules favour simple protocols, because these modules are optimised for extremely high-throughput processing rather than for elaborate logic (and implementing an advanced soft core CPU may not be possible)

- the DCS favours advanced protocols (like OPC UA) which supports rich address space, high reliability, security and availability of adapters for communication with the SCADA layer

Since reliability, security and smooth integration with the SCADA layer are of priority, the efforts of OPC UA integration in the modules are ongoing.

CONCLUSIONS

OPC UA passed both functional and performance tests in the ATLAS DCS. It has already replaced OPC DA technology in the ELMB-based parts of the DCS. Thanks to its features (especially versatility, portability, standards compliance, SCADA connectivity and security) it may become the dominant communication technology of the DCS.

ISBN 978-3-95450-139-7

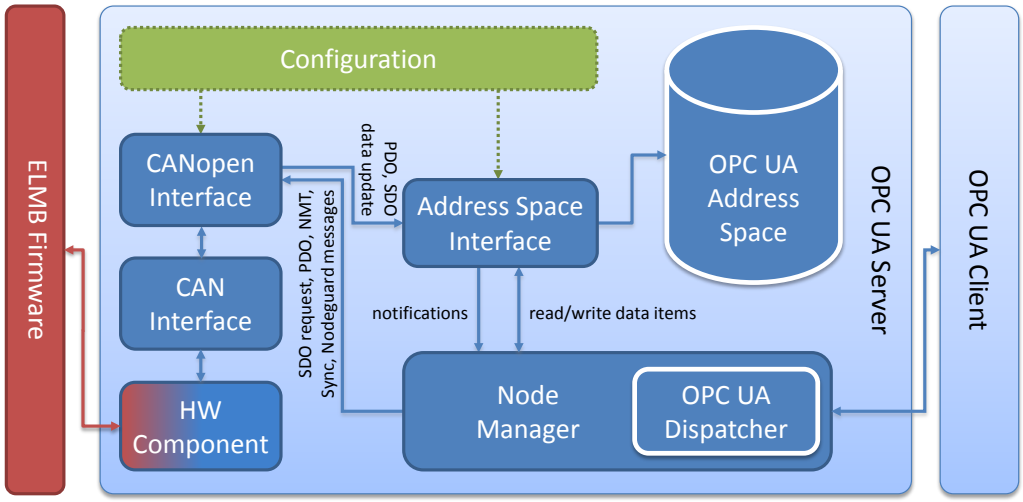

Figure 2: Software components of the server and data flow among them. CANopen specific terms like SDO, PDO, NMT are described in [2].

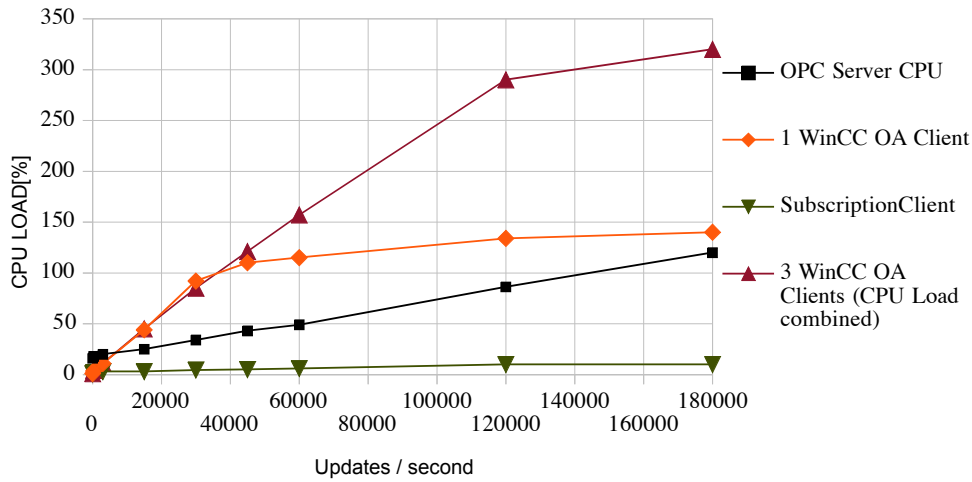

Figure 3: CPU load of OPC UA software components vs few update rate values.

REFERENCES

[1] A. Barriuso Poy, H. Boterenbrood, H. J. Burckhart, J. Cook, V. Filimonov, S. Franz, O. Gutzwiller, B. Hallgren, V. Khomutnikov, S. Schlenker, F. Varela. "The detector control system of the ATLAS experiment", Journal of Instrumentation, Vol. 3, May 2008, doi:10.1088/1748-0221/3/05/P05006

[2] CAN-in-Automation, CAL, CAN Application Layer for Industrial Applications, CiA Draft Standard DS-201 to DS-207, Version 1.1, Feb 1996

[3] Microsoft Corporation, The Component Object Model Specification, http://www.microsoft.com/Com/resources/comdocs.asp (1995)

[4] W. Mahnke, S.H. Leitner, "OPC Unified Architecture - The future standard for communication and information modeling in automation", 3/2009 ABB Review 3/2009, page 56-61

[5] Unified Automation website, http://www.unified-automation.com, access 16th Sep 2013

[6] W. Mahnke, S.H. Leitner, M. Damm, "OPC Unified Architecture", ISBN 978-3-540-68898-3, 2009 Springer-Verlag Berlin Heidelberg

[7] C. Gaspar, M. Donszelmann, Ph. Charpentier, "DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication"

ISBN 978-3-95450-139-7