ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPA122

DATA ACQUISITION AND SUPERVISION FOR THE HL-LHC QUENCH PROTECTION SYSTEM - PART II THE SOFTWARE STACK*

M.-A. Galilée†, M. Christensen, G. M. Garcia, J. C. Garnier, M. M. Moya, T. Podzorny, L. R. Ropero, and E. Thaller, European Organization for Nuclear Research, Geneva, Switzerland

Abstract

The Quench Protection System (QPS) of the LHC is crucial for ensuring the integrity of the superconducting magnet circuit elements. It also plays an important role in the acquisition of data from these elements during the magnet qualification, equipment commissioning and accelerator operation. The new superconducting circuits for the HL-LHC era, which will be assembled and operated in the first place in the IT String facility, require finer and more comprehensive measurements during all of these steps than the currently operated magnets of the LHC. The required data throughput of such measurements cannot be achieved with the current QPS data acquisition technology. Therefore, a new data acquisition stack called *EDAQ* has been developed to address this issue and provide further improvements, including microsecond precision timing synchronisation down to the individual field agents. This contribution presents the technologies chosen for this new stack, their additional benefits, their assembly into a robust and high-performance prototype, its integration into the existing controls environment and the ongoing validation in successive steps towards the HL-LHC installation.

QPS FOR HL-LHC MAGNETS, REQUIREMENTS

Hardware is the main factor driving the technological choices for the software solution and in particular the data acquisition stack. The hardware baseline for HL-LHC is the Universal Quench Detection System (UQDS), which is an FPGA board based, versatile digital platform. The UQDS system is detailed in [1]. Four key aspects of the UQDS are particularly relevant to the adequate design of its acquisition stack: the amount of data it can provide, the radiation levels in the location where it will be deployed, the computational capacity of the embedded communication controller, and the ultimate timing accuracy.

UQDS is able to provide high-definition data, up to 11 Mbps per unit, which is of high interest in the context of the new Nb_3Sn magnets development, qualification and commissioning phases. It will be installed in radiation free testbenches and underground areas of the LHC machine. Lastly, the Micro Controller Unit (MCU) used comes with very little memory (up to 256 kB).

A NEW STACK FROM THE GROUND UP

The current QPS data acquisition stack is not able to fulfill these requirements; based on the WorldFIP [2] technology, it provides only up to 1 Mbps of data throughput for a whole acquisition segment, which accommodates up to 50 field agents (functionally equivalent to the new UQDS agents). The WorldFIP technology is obsolete, with spare parts and maintenance not provided anymore by industry. Furthermore, the existing data acquisition stack grew organically with the extension of QPS in the LHC, which spread over more than 15 years from design to current exploitation through numerous renovations. This makes it a complex software system, with significant costs for maintenance and evolution. These circumstances prompted the design of a new acquisition stack from the ground up, addressing the future requirements for quench protection in the LHC, and facilitating the QPS evolution in the longer term. Nonetheless, a new data acquisition must fit into the same controls environment as the current one, in particular the set of existing expert and operation tools. This implies further constraints on the outward communication interfaces and control procedures.

In order to enable modularity of the stack components and reduce eventual evolution costs, the new stack has to be compliant with the Open Systems Interconnection model (OSI) [3] model. In the first place, *Ethernet* was chosen for the Physical and Data link layers, *IP* and *UDP* for the Network and Transport layers, with a dedicated protocol implementing the Session and Presentation layers. Although the use of *Ethernet* and *IP* enables the integration of the network elements into an existing network, e.g. CERN's Technical Network (TN) [4], in the interest of simpler configuration and security, the initial network topology is that of a private network, on which reside only a local subset of all the UQDS devices relevant to the protection of the magnet circuits. So several of these private networks compose the overall acquisition stack, each connected to the TN via a gateway computer. Similarly, a plain star topology was chosen, with an Ethernet router linking the gateway and the UQDS agents.

EDAQ: NETWORK AND APPLICATION PROTOCOL

This new dedicated protocol is called *EDAQ* and is built on top of *UDP/IP*, as visible in Fig. 1. It is described in its own engineering specification. Due to the limited amount of computational resources of the MCU, and the interest in enabling real-time emission, *UDP* was chosen over *TCP* as the basis for *EDAQ*, at the cost of specifying and im-

* Research supported by the HL-LHC project
† mgalilee@cern.ch

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPA122

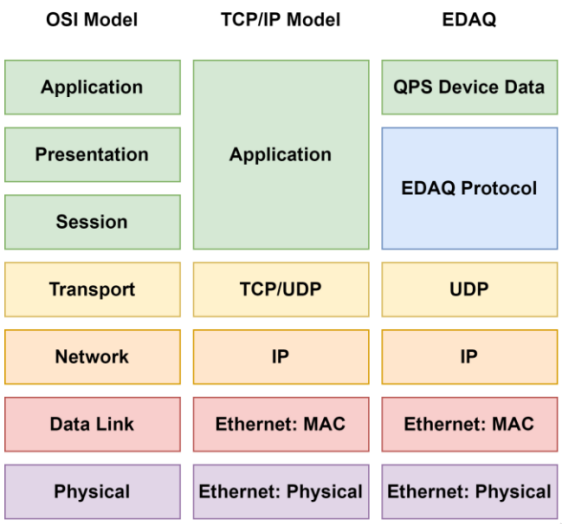

Figure 1: EDAQ in the OSI model.

plementing cherry picked features of *TCP*, adapted to the requirements of our use-cases. These include on-demand reliability, buffering and ordering of the transmitted messages. *EDAQ* defines three network traffic modes:

- Unreliable; virtually bare *UDP*.
- Buffered unreliable; the receiver can request data retransmission.
- Reliable; reliability ensured by an ACK/NACK mechanism.

These modes are non exclusive, so a field agent and a gateway can in principle communicate simultaneously over all three modes, selecting either mode for a given piece of data to transmit, depending on the expected load it puts on the infrastructure and its criticality. Examples are: data transmitted in nominal circuit conditions, detailed record of a magnet circuit event, commands.

While its design is driven by the QPS use case, *EDAQ* remains effectively application agnostic. *EDAQ* payloads are left to the Application layer to specify and handle, as seen in Fig. 1 and Fig. 2 where different application payloads are showcased.

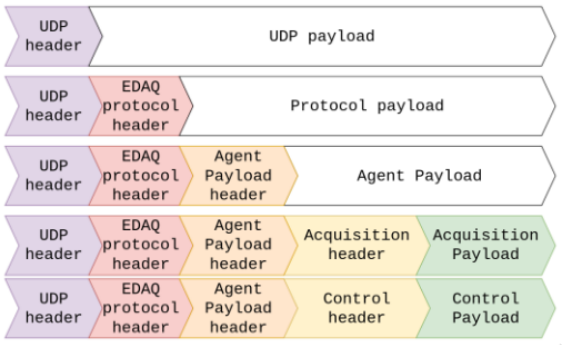

Figure 2: EDAQ encapsulation model.

SAFETY AND PERFORMANCE WITH RUST

With regard to the implementation of the protocol, several programming languages were considered and reviewed: *C++17/20*, *Go* [5] and *Rust* [6]. The main evaluation criteria were:

- Safety,
- Performance,
- Long term community/industry support,
- Tooling quality,
- Attractiveness to young professionals.

While *C++* benefits from decades of extensive industry usage and will likely stay around for many more decades, even its more modern revisions carry the complexity burden of a very extensive language and of the backward compatibility requirement, increasing the difficulty to audit the code and ensure the actual safety of the program. *Go* fares reasonably well in all aspects, but loses to *Rust* with regard to performance and safety, according to the scientific press review we carried out at the project start, in early 2021. Notably, *Rust* boasts safe and efficient concurrent programming as one of its major goals, using the motto 'fearless concurrency' [7]. It as well benefits from an outstanding appreciation in the software developers community [8]. Consequently, we selected *Rust*, to be further evaluated as part of the prototyping of the protocol and the gateway application embedding it.

The design of the gateway application is as follows:

1. Field I/O leveraging *Tokio* [9], using a dedicated UDP socket for each field agent;
2. An internal, elastic, concurrent queues system for tracking, ordering and buffering messages;
3. A *gRPC* [10] interface for higher level (controls) clients.

After several iterations, and witnessing the reliability of *Rust* and its tooling, the code quality they enable, the performance achieved, the *Rust* implementation became the canonical implementation of the protocol and gateway application. Performance measurements include the stress testing of the application with four simulated field agents emitting data at up to 20 kHz, for a total throughput of 1 Gbps with overall CPU usage well below 50 % on a common modern computer. Figure 3 highlights the throughput and transmission reliability scaling using the actual hardware infrastructure and agents.

During the exploratory phase, we experimented with sharing the *Rust* code base across the network boundary, re-using the protocol software implementation in the micro-controller firmware, with the intent of reducing the maintenance costs from having two parallel implementations and benefiting from *Rust*'s safety model. While the effort delivered promising results, we decided against this firmware solution due to the initial development overhead, the perceived low adoption of *Rust* in the embedded world, and the concerns it raises about longer term support.

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPA122

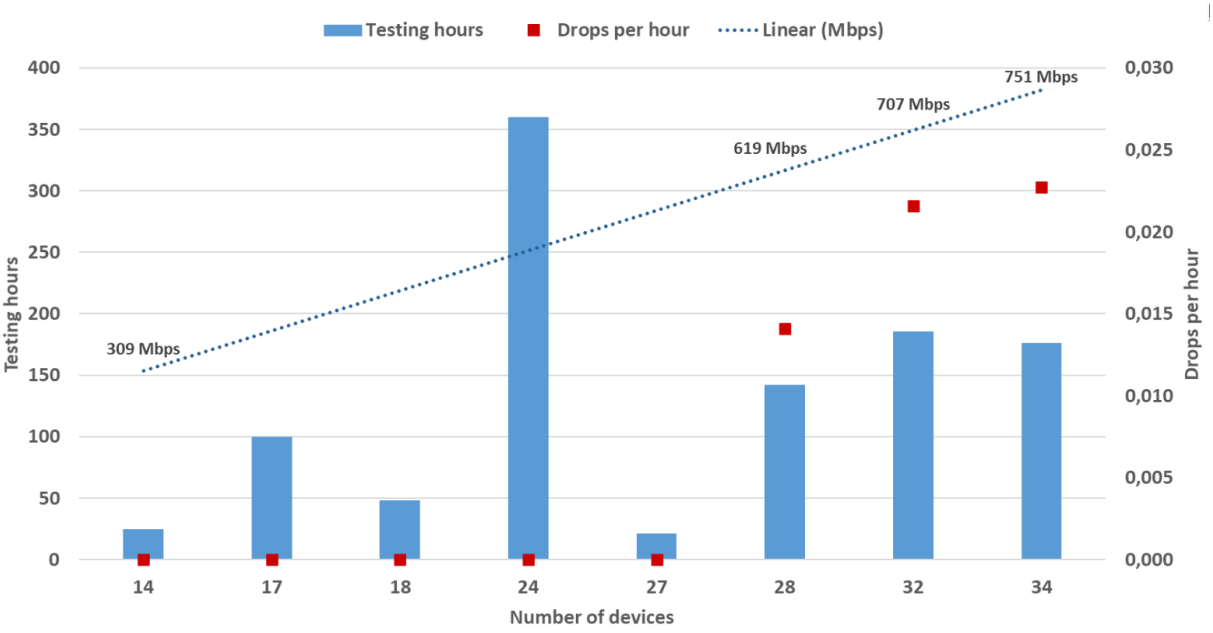

Figure 3: Data throughput and scaling. Data and visualization courtesy of M. Murillo Moya.

TIMING SYNCHRONIZATION

Timing synchronization is achieved with the Precision Time Protocol (PTP), for which traffic runs along that of the data acquisition on the same private network. The master time is acquired from the gateway computer, already synchronized to the CERN global timing, and set on a dedicated Network Interface Card (NIC), a *Syn1588* from Oregano Systems, which is then synchronized via a Pulse Per Second signal (PPS) to the CERN timing infrastructure. A *QFX5110-48S* network switch from Juniper Networks is used and set up to handle PTP traffic transparently. This topic is detailed further in another contribution to this conference [1], as well as in [11].

CONTROLS INTEGRATION AND DEPLOYMENTS

The *EDAQ* gateway application is domain agnostic, and so its outward *gRPC* interface does not describe domain level data, in particular that of the QPS. A set of QPS specific client libraries instead provides the means to interpret *EDAQ* payloads as QPS data and reciprocally. These libraries are also written in *Rust* but are packaged as *C++* and *Python* libraries as well, respectively via the *Rust CXX* [12] and *Rust PyO3* [13] bindings. This enables their integration with the usual technologies and tools of CERN accelerator controls.

The full acquisition stack has now been deployed in various facilities (lab, hardware testbed) and been integrated and tested against different field agents (pure software simulation, mock hardware, actual hardware), with positive results. The next stage in the validation of the stack is its operational use at CERN's magnet test facility, used for the initial qualification of HL-LHC magnets [14], and in the IT String facility [15], to validate the integration of the new HL-LHC cryo-assemblies and superconducting link cold powering system [16]. The former is imminent, the latter will occur in 2024.

CONCLUSION

Building an entirely new data acquisition stack from the grounds up is an extensive endeavor, which encompasses many different domains and requires to apprehend their respective technologies in order to compose them into a cohesive, functional structure. Careful choice of the technologies has a major impact on the development and maintenance costs of a project, and on the performance of the solution. In the case of the *EDAQ* stack, the choice of *Rust* as the foundation for its software layers enabled us to achieve early on high reliability and performance of the critical gateway application, with a level of trust in the software we built much higher than what we are used to achieve with other technologies.

The *EDAQ* stack development also benefited from the close coordination between the hardware and software experts of the domain, and the collaborative establishing of specifications for the network protocol and field agent behavior. This enabled clear, short feedback loops and efficient reactivity from both parts.

Actual field validation with the operation of the HL-LHC magnets testing facilities is a key milestone for this new stack, and will provide key insights on its longer term maintenance, and possible evolution and scaling to more use-cases and application domains.

REFERENCES

[1] T. Podzorny, A. Hollos, M. Christensen, M. A. Galilée, G. M. Garcia, J. Spasic, J. Steckert, M. M. Moya, R. Denz, T. Pridii,

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPA122

A. Skoczen, "Data acquisition and supervision systems for the HL-LHC quench protection system", in these proceedings

[2] WorldFIP, https://ohwr.org/project/cern-fip/wikis/worldfip

[3] OSI model, http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.200-199407-I!!PDF-E&type=items

[4] U. Epting, "Computing and Network Infrastructure for Controls CNIC", ICALEPCS'2005, https://accelconf.web.cern.ch/ica05/proceedings/pdf/O2_009.pdf

[5] The Go Programming Language, https://go.dev/

[6] The Rust Programming Language, https://www.rust-lang.org/

[7] The Rust Programming Language book - Fearless concurrency, https://doc.rust-lang.org/book/ch16-00-concurrency.html

[8] Stack Overflow Developer Survey 2022, https://survey.stackoverflow.co/2022/#programming-scripting-and-markup-languages

[9] Tokio - An asynchronous Rust runtime, https://tokio.rs/

[10] gRPC Remote Procedure Calls, https://grpc.io/

[11] Magnus B.B. Christensen, "Packet Based Time Synchronisation", Master Thesis, Aalborg University., Aalborg, 2021, https://projekter.aau.dk/projekter/en/studentthesis/packet-based-time-synchronisation(39025748-0b2b-4c6c-bbe9-6317870edc97).html

[12] CXX - safe interop between Rust and C++, https://cxx.rs/

[13] PyO3 - Rust bindings for Python, https://github.com/PyO3/pyo3

[14] The SM18 test facility in the HL-LHC era, https://home.cern/news/news/engineering/sm18-test-facility-hl-lhc-era

[15] M. Bajko, S. Bertolasi, C. Bertone, S. Blanchard, D. Bozzini, O. Brüning, P. Cruikshank, D. De Luca, N. Dos Santos, F. Dragoni, N. H. Garcia, A. Herty, A. Kosmicki, W. Maan, A. M. Selles, P. M. Urios, S. Le Naour, P. Orlandi, A. Perin, M. Pojer, F. R. Mateos, G. Rolando, L. Rossi, H. Thiesen, E. Todesco, E. Vergara, D. Wollmann, S. Yammine, J. Zawilinski, M. Zerlauth, "The Inner Triplet String Facility for HL-LHC: design and planning", IPAC2021 https://accelconf.web.cern.ch/ipac2021/papers/wepab376.pdf

[16] I. Béjar Alonso, O. Brüning, P. Fessia, L. Rossi, L. Tavian, M. Zerlauth, "High-Luminosity Large Hadron Collider (HL-LHC): Technical design report", Geneva, CERN, 2020, https://cds.cern.ch/record/2749422