EPJ Web of Conferences **116**, 05001 (2016)
DOI: 10.1051/epjconf/201611605001

The Control Unit of KM3NeT data acquisition

Cristiano Bozza[a] for the KM3NeT Collaboration

University of Salerno, Dipartimento di Fisica, via Giovanni Paolo II 132, Fisciano 84084, Italy

Abstract. The KM3NeT Collaboration is building a new generation of neutrino telescopes in the Mediterranean Sea. With the telescopes, scientists will search for cosmic neutrinos to study highly energetic objects in the Universe, while one neutrino detector will be dedicated to measure the properties of the high-energy neutrino particles themselves. Control of the KM3NeT data acquisition processes is handled by the KM3NeT Control Unit, which has been designed to maximise the detector live time. The Control Unit features software programs with different roles, following the philosophy of having no single point of failure. While all programs are interconnected, each one can also work alone for most of the time in case other services are unavailable. All services run on the Common Language Runtime, which ensures portability, flexibility and automatic memory management. Each service has an embedded Web server, providing a user interface as well as programmatic access to data and functions. Data to and from detector components for monitoring and management purposes are transmitted using a custom designed protocol. The Control Unit is interfaced to one or more Message Dispatchers to control the data acquisition chain. A Data Base Interface provides fast and fault-tolerant connection to a remote Data Base.

1. Overview

The KM3NeT Collaboration [1] is installing water-Cherenkov neutrino telescopes [2] at multiple locations in the Mediterranean Sea. Data taking is expected to start by the end of 2015 when the first sensors will be installed. One telescope will be devoted to the search for astrophysical sources of neutrinos, whereas another will be used to study features of neutrinos themselves. Each detector is built of detection units, i.e. vertical lines supporting 18 Digital Optical Modules. Each optical module hosts 31 photomultipliers, monitoring instruments and piezo elements for the acoustic positioning of the module. A detector *building block* will be made of 115 detection units supporting in total 64,170 optical modules. The telescopes should be operational for at least 15 years. The detector and the Trigger and Data Acquisition System (TriDAS) [3] are controlled by a scalable software infrastructure called Control Unit that is already being used in test and validation benches to characterise and tune detector components.

Interaction events both of primary cosmic neutrinos and of secondary neutrinos from cosmic rays may occur at any time. Maximising the detector live time is one of the main priorities in the design of the Control Unit. It should have no single point of failure, and should be able to operate also in

[a] e-mail: cbozza@unisa.it

downgraded mode with one or more services stopped. Long time scales demand easy operation and configuration. The software is designed relying on widely adopted standards with a large development and user base, whereas the code of all non-standard components is completely under the control of the KM3NeT Collaboration. The Control Unit can run with relatively small hardware resources, which helps keeping the overall cost low.

All the controllable elements in the detector as well as in the TriDAS implement the same State Machine (SM), so the control logic is in principle the same for all of them. At any time, a target state is defined that each SM should reach, corresponding to an overall detector target:

Off: all systems are idle, photomultipliers are not powered and they do not generate data;
On: all systems active, photomultipliers are powered but they do not send data;
Run: detector taking data with all systems.

In principle, data acquisition should start when all subsystems are ready. However, after several years of operation, it is to be expected that some optical modules might be damaged or only partly functional; also in this case the acquisition must work without useless waits: the Control Unit software is tolerant with respect to failing hardware components. If software components fail in the TriDAS the acquisition cannot work because data cannot be processed; in that case, data acquisition becomes pointless.

The programs the Control Unit is composed of are called Control Unit Services. In the following, the shortened term *services* will often be used.

2. Networking protocols

Several networking protocols are used by the Control Unit, reflecting different criticality levels of the communication. For internal communications the following protocols are in place:

1) HTTP(S) for GUI interfaces to services mostly for human users but also for programmatic access.

2) Server Application with Web Interface (SAWI): remote calls among services are implemented on top of HTTP(S) on specific pages. SAWI transports data, method calls and exceptions in a transparent way as if everything were occurring in the same process. Since it runs on HTTP(S), it works on top of an existing HTTP(S) Web server library. Among the benefits of this approach, it is worth recalling that Web browsers can be used to mimic remote calls, so they can be used for debugging purposes. SAWI is mostly used for light messages, whereas for massive data transfer NFS is used (see below).

3) Simple Retransmission Protocol (SRP): this is a UDP-based protocol used to communicate with the Central Logic Boards (CLBs) [4] of the optical modules. Commands flow from the Detector Manager to CLBs, monitoring data go in the opposite direction.

4) One or more Message Dispatchers use stable TCP links to TriDAS programs to send control commands, receive reports, and distribute data streams.

5) Network File System (NFS): data to be transferred to the database are saved to a local shared directory so that they can be checked in case of needs. Because their storage relies on the file system, even in the unlikely case of a failure of either the sender or receiver service they are not lost.

For external communications, HTTP(S) is used for GUI interfaces, while SQLNet is used to connect the Data Base Interface (see 3.5) to the central database ([5], [6]) running Oracle Database Server [7].

3. Components

The modularity in the Control Unit design has two goals:

1) Resilience by avoiding single points of failure and explicitly foreseeing activity in downgraded conditions with one or more services unavailable;

2) Scalability by allowing the same service to work in multiple instances, sharing the workload.

3.1 Service structure

All services are designed to run on all major operating systems. Linux SLC6 is used for production, whereas Windows and MacOS are used for development. Neither recompilation nor cross-compilation is needed to change the operating system. This is possible thanks to the adoption of the Common Language Runtime (CLR) [8], supported by Mono [9] for Linux, Windows and MacOS, and by .NET on Windows. So far all programs have been written in C# [10], but any other language with CLR support may be used to add features. The core of all services provides four basic features:

1) Persistence/resilience tools tracking the current state of the service by a configuration file and a session file to allow restart with minimal or no human action in case of a power cut or sudden failure;

2) An HTTP server library (Web server) supporting a GUI interface; the SAWI protocol uses this HTTP server to expose methods for remote calls.

3) A Virtual Directory that allows for representing all the quantities and parameters that need to be exposed for monitoring in a convenient directory structure for access through the HTTP server;

4) Human-readable log generation.

Built on top of these features, each service has its own specific commands (from text-mode console), GUI, set of SAWI-exposed methods and control behaviour.

3.2 The Master Control Program

The Master Control Program (MCP) is the central reference point to define the current state, i.e. detector configuration, *runsetup* (set of operational parameters), run number and target (*Off*/*On*/*Run* as stated above). When the state changes, other services are notified (*push* message mode). They also periodically query the MCP to check the state (*pull* message mode). The MCP can also execute scheduled jobs with predefined start-end time and *runsetup*. State information is kept encrypted in the persistent files to preserve integrity.

3.3 The Detector Manager

The task of the Detector Manager (DM) is to control each Central Logic Board (CLB) of an optical module to fulfil the general detector state defined by the MCP. The SM of each CLB is driven according to the current target and operational parameters applied as specified in the current *runsetup*, defined in the KM3NeT database. Monitoring data about photomultiplier high voltage, temperature, humidity, compass, accelerometer, etc. are continuously collected in so-called DataLog files that are uploaded to the database. Communication with the CLBs is accomplished through SRP. A *swarm* of threads shares the load of controlling the CLBs optimising CPU occupancy. A single control thread manages reactions to external commands. The control logic and operational parameters can be fine-tuned for each single CLB if needed, which is useful to add test equipment to otherwise standard data-taking devices. The DM also features a special mode in which it merely routes and translates commands received by the HTTP server to CLBs. This is useful for automatic testing of large stocks of devices by running minimal installations of the Control Unit software driven by lightweight scripts.

3.4 The TriDAS Manager

The programs in the Data Acquisition Systems devoted to trigger and on-line data processing need to be driven and configured as a whole entity. This is the purpose of the TriDAS Manager (TM). In principle, there may be hundred instances of such programs running when the detector reaches the size of a *building block*. Because also TriDAS programs have the same SM as hardware components, the control logic is very similar. All communication from/to the TriDAS programs occurs using the Control

Host protocol. TM has notion of time and retry count to distinguish occasional response delays from stuck programs. If needed, it can start/restart/stop remotely each DAS program.

3.5 Database Interface and Database Stream Writer

Control Unit services do not connect directly to the KM3NeT database. Within each data-taking station, which can be a shore station of an underwater detector or a test-bench for product characterisation, all data exchange goes through HTTP (SAWI) calls and file exchange. The Data Base Interface (DBI) optimizes resource usage by hosting local data caches to avoid connections to remote databases that can be time-consuming. The list of users, detector definitions and *runsetups* are all automatically synchronized with the KM3NeT database. DataLog files generated by the DM are staged and buffered in the upload cache. The DBI runs data upload code that optimizes the physical arrangement of data on database disks. The Data Stream Writer (DSW), currently under development, has a similar approach to writing large data sets into the database. Calibration parameters and acoustic data are extracted from the serial flow of the Dispatcher and are written in batches to the database.

3.6 The Local Authentication Provider

In each data-taking station, a locally centralized authentication service provides users with access on the basis of their unique credentials, which are stored encrypted. With a similar logic, *service accounts* are used to provide a local directory service (i.e. maintaining the list of *which machine does what*) without having to specify other services' network addresses and port numbers in the configuration file of each member of the Control Unit. Only the Local Authentication Provider (LAP) address needs to be known to each service. Every time a user or service logs onto the Control Unit, it receives an access token. In addition, because all the GUIs are built using Web browsers, the LAP tracks all the tokens assigned, and allows handing out a token from one browser session to another so that users do not need to log in at each program window.

4. Performances

Services that come into action only during overall state changes raise no concerns about performance. Those that actively do work need some attention to correctly tune the hardware resources needed.

In tests run on the first detection units, each with 18 optical modules and a base controller, with DM controlling detector operation, the monitoring data polling frequency was set to 0.1 Hz, which was 10 times higher than envisaged for normal workload; 1.2 cores were used on average. This should scale to 13.8 cores for a full *building block* of detection units monitored at 10^{-2} Hz. However, this is expected to be a conservative guess, considering that in the aforementioned test up to 0.6 cores were used only to support the GUI.

The DBI and DSW will continuously feed the database with new data. While the expected data rate for a building block at 10^{-2} Hz is about 0.3 Mbps, tests of the SQLNet protocol with very simple tables, needing almost no ordering and indexing between two upload shots, showed it can use all the available network transfer speed if needed. The way data are stored in the database and their organization during the upload are known to affect the transfer speed by large factors, and, at the time of writing this report, work is focused on the optimization of this process, with promising preliminary results.

References

[1] `http://www.km3net.org`
[2] S. Adrián-Martínez et al., Eur. Phys. J. C **74**, 1 (2014)

[3] C. Pellegrino, T. Chiarusi, these Proceedings
[4] D. Real et al., these Proceedings
[5] A. Albert, C. Bozza, AIP Conf. Proc. **1631** 167 (2014)
[6] A. Albert, C. Bozza, these Proceedings
[7] http://www.oracle.com
[8] http://www.ecma-international.org/publications/standards/Ecma-335.htm
[9] http://www.go-mono.com
[10] http://www.ecma-international.org/publications/standards/Ecma-334.htm