Computer Physics Communications 256 (2020) 107433

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

The Control Unit of the KM3NeT Data Acquisition System☆

S. Aiello [1], F. Ameli [2], M. Andre [3], G. Androulakis [4], M. Anghinolfi [5], G. Anton [6], M. Ardid [7], J. Aublin [8], C. Bagatelas [4], G. Barbarino [9,10], B. Baret [8], S. Basegmez du Pree [11], M. Bendahman [12], E. Berbee [11], A.M. van den Berg [13], V. Bertin [14], V. van Beveren [11], S. Biagi [15], A. Biagioni [2], M. Bissinger [6], J. Boumaaza [12], S. Bourret [8], M. Bouta [16], G. Bouvet [17], M. Bouwhuis [11], C. Bozza [18,*], H. Brânzaş [19], M. Bruchner [6], R. Bruijn [11,20], J. Brunner [14], E. Buis [21], R. Buompane [9,22], J. Busto [14], D. Calvo [23], A. Capone [24,2], S. Celli [24,2,49], M. Chabab [25], N. Chau [8], S. Cherubini [15,26], V. Chiarella [27], T. Chiarusi [28,*], M. Circella [29], R. Cocimano [15], J.A.B. Coelho [8], A. Coleiro [23], M. Colomer Molla [8,23], S. Colonges [8], R. Coniglione [15], P. Coyle [14], A. Creusot [8], G. Cuttone [15], A. D'Onofrio [9,22], R. Dallier [17], M. De Palma [29,30], I. Di Palma [24,2], A.F. Díaz [31], D. Diego-Tortosa [7], C. Distefano [15], A. Domi [5,14,32], R. Donà [28,33], C. Donzaud [8], D. Dornic [14], M. Dörr [34], M. Durocher [15,49], T. Eberl [6], I. El Bojaddaini [16], H. Eljarrari [12], D. Elsaesser [34], A. Enzenhöfer [14], P. Fermani [24,2], G. Ferrara [15,26], M.D. Filipović [35], A. Franco [29], L.A. Fusco [8], T. Gal [6], A. Garcia Soto [11], F. Garufi [9,10], L. Gialanella [9,22], E. Giorgio [15], S.R. Gozzini [23], R. Gracia [6], K. Graf [6], D. Grasso [36], T. Grégoire [8], G. Grella [18], D. Guderian [50], C. Guidi [5,32], S. Hallmann [6], H. Hamdaoui [12], H. van Haren [37], A. Heijboer [11], A. Hekalo [34], J.J. Hernández-Rey [23], J. Hofestädt [6], F. Huang [38], G. Illuminati [23], C.W. James [39], P. Jansweijer [11], M. de Jong [11], P. de Jong [11,20], M. Kadler [34], P. Kalaczyński [40], O. Kalekin [6], U.F. Katz [6], N.R. Khan Chowdhury [23], F. van der Knaap [21], E.N. Koffeman [11,20], P. Kooijman [20,51], A. Kouchner [8,41], V. Kulikovskiy [5], R. Lahmann [6], G. Larosa [15], R. Le Breton [8], F. Leone [15,26], E. Leonora [1], G. Levi [28,33], M. Lincetto [14], M. Lindsey Clark [8], A. Lonardo [2], F. Longhitano [1], D. Lopez-Coto [42], G. Maggi [14], J. Mańczak [23], K. Mannheim [34], A. Margiotta [28,33], A. Marinelli [43,36], C. Markou [4], G. Martignac [17], L. Martin [17], J.A. Martínez-Mora [7], A. Martini [27], F. Marzaioli [9,22], S. Mazzou [25], R. Mele [9,10], K.W. Melis [11], P. Migliozzi [9], E. Migneco [15], P. Mijakowski [40], L.S. Miranda [44], C.M. Mollo [9], M. Morganti [36,52], M. Moser [6], A. Moussa [16], R. Muller [11], M. Musumeci [15], L. Nauta [11], S. Navas [42], C.A. Nicolau [2], C. Nielsen [8], B. Ó Fearraigh [11,20], M. Organokov [38], A. Orlando [15], V. Panagopoulos [4], G. Papalashvili [45], R. Papaleo [15], C. Pastore [29], G.E. Păvălaş [19], C. Pellegrino [33,53], M. Perrin-Terrin [14], P. Piattelli [15], C. Pieterse [23], K. Pikounis [4], O. Pisanti [9,10], C. Poirè [7], G. Polydefki [4], V. Popa [19], M. Post [20], T. Pradier [38], G. Pühlhofer [46], S. Pulvirenti [15], L. Quinn [14], F. Raffaelli [36], N. Randazzo [1], A. Rapicavoli [26], S. Razzaque [44], D. Real [23], S. Reck [6], J. Reubelt [6], G. Riccobene [15], M. Richer [38], L. Rigalleau [17], A. Rovelli [15], I. Salvadori [14], D.F.E. Samtleben [11,47], A. Sánchez Losa [29], M. Sanguineti [5,32], A. Santangelo [46], D. Santonocito [15], P. Sapienza [15], J. Schnabel [6], V. Sciacca [15], J. Seneca [11], I. Sgura [29], R. Shanidze [45], A. Sharma [43], F. Simeone [2], A. Sinopoulou [4], B. Spisso [18,9], M. Spurio [28,33], D. Stavropoulos [4], J. Steijger [11], S.M. Stellacci [18,9], B. Strandberg [11], D. Stransky [6], M. Taiuti [5,32], Y. Tayalati [12], E. Tenllado [42], T. Thakore [23], S. Tingay [39], E. Tzamariudaki [4], D. Tzanetatos [4], V. Van Elewyck [8,41],

☆ The review of this paper was arranged by Prof. Z. Was.

* Corresponding authors.
E-mail addresses: cbozza@unisa.it (C. Bozza), tommaso.chiarusi@bo.infn.it (T. Chiarusi).

https://doi.org/10.1016/j.cpc.2020.107433

G. Vannoye [5], F. Versari [28,33], S. Viola [15], D. Vivolo [9,10], G. de Wasseige [8], J. Wilms [48], R. Wojaczyński [40], E. de Wolf [11,20], D. Zaborov [14,54], A. Zegarelli [24,2], J.D. Zornoza [23], J. Zúñiga [23]

[1] INFN, Sezione di Catania, Via Santa Sofia 64, Catania, 95123, Italy
[2] INFN, Sezione di Roma, Piazzale Aldo Moro 2, Roma, 00185, Italy
[3] Universitat Politè,cnica de Catalunya, Laboratori d'Aplicacions Bioacústiques, Centre Tecnològic de Vilanova i la Geltrú, Avda. Rambla Exposició, s/n, Vilanova i la Geltrú, 08800, Spain
[4] NCSR Demokritos, Institute of Nuclear and Particle Physics, Ag. Paraskevi Attikis, Athens, 15310, Greece
[5] INFN, Sezione di Genova, Via Dodecaneso 33, Genova, 16146, Italy
[6] Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen Centre for Astroparticle Physics, Erwin-Rommel-Straße 1, 91058 Erlangen, Germany
[7] Universitat Politècnica de València, Instituto de Investigación para la Gestión Integrada de las Zonas Costeras, C/ Paranimf, 1, Gandia, 46730, Spain
[8] APC, Université Paris Diderot, CNRS/IN2P3, CEA/IRFU, Observatoire de Paris, Sorbonne Paris Cité, 75205 Paris, France
[9] INFN, Sezione di Napoli, Complesso Universitario di Monte S. Angelo, Via Cintia ed. G, Napoli, 80126, Italy
[10] Università di Napoli "Federico II", Dip. Scienze Fisiche "E. Pancini", Complesso Universitario di Monte S. Angelo, Via Cintia ed. G, Napoli, 80126, Italy
[11] Nikhef, National Institute for Subatomic Physics, PO Box 41882, Amsterdam, 1009 DB, Netherlands
[12] University Mohammed V in Rabat, Faculty of Sciences, 4 av. Ibn Battouta, B.P. 1014, R.P. 10000 Rabat, Morocco
[13] KVI-CART University of Groningen, Groningen, Netherlands
[14] Aix Marseille Univ, CNRS/IN2P3, CPPM, Marseille, France
[15] INFN, Laboratori Nazionali del Sud, Via S. Sofia 62, Catania, 95123, Italy
[16] University Mohammed I, Faculty of Sciences, BV Mohammed VI, B.P. 717, R.P. 60000 Oujda, Morocco
[17] Subatech, IMT Atlantique, IN2P3-CNRS, Université de Nantes, 4 rue Alfred Kastler - La Chantrerie, Nantes, BP 20722 44307, France
[18] Università di Salerno e INFN Gruppo Collegato di Salerno, Dipartimento di Fisica, Via Giovanni Paolo II 132, Fisciano, 84084, Italy
[19] ISS, Atomistilor 409, Măgurele, RO-077125, Romania
[20] University of Amsterdam, Institute of Physics/IHEF, PO Box 94216, Amsterdam, 1090 GE, Netherlands
[21] TNO, Technical Sciences, PO Box 155, Delft, 2600 AD, Netherlands
[22] Università degli Studi della Campania "Luigi Vanvitelli", Dipartimento di Matematica e Fisica, viale Lincoln 5, Caserta, 81100, Italy
[23] IFIC - Instituto de Física Corpuscular (CSIC - Universitat de València), c/Catedrático José, Beltrán, 2, 46980 Paterna, Valencia, Spain
[24] Università La Sapienza, Dipartimento di Fisica, Piazzale Aldo Moro 2, Roma, 00185, Italy
[25] Cadi Ayyad University, Physics Department, Faculty of Science Semlalia, Av. My Abdellah, P.O.B. 2390, Marrakech, 40000, Morocco
[26] Università di Catania, Dipartimento di Fisica e Astronomia, Via Santa Sofia 64, Catania, 95123, Italy
[27] INFN, LNF, Via Enrico Fermi, 40, Frascati, 00044, Italy
[28] INFN, Sezione di Bologna, v.le C. Berti-Pichat, 6/2, Bologna, 40127, Italy
[29] INFN, Sezione di Bari, Via Amendola 173, Bari, 70126, Italy
[30] University of Bari, Via Amendola 173, Bari, 70126, Italy
[31] University of Granada, Department of Computer Architecture and Technology/CITIC, 18071 Granada, Spain
[32] Università di Genova, Via Dodecaneso 33, Genova, 16146, Italy
[33] Università di Bologna, Dipartimento di Fisica e Astronomia, v.le C. Berti-Pichat, 6/2, Bologna, 40127, Italy
[34] University Würzburg, Emil-Fischer-Straße 31, Würzburg, 97074, Germany
[35] Western Sydney University, School of Computing, Engineering and Mathematics, Locked Bag 1797, Penrith, NSW 2751, Australia
[36] INFN, Sezione di Pisa, Largo Bruno Pontecorvo 3, Pisa, 56127, Italy
[37] NIOZ (Royal Netherlands Institute for Sea Research) and Utrecht University, PO Box 59, Den Burg, Texel, 1790 AB, Netherlands
[38] Université de Strasbourg, CNRS, IPHC, 23 rue du Loess, Strasbourg, 67037, France
[39] Curtin University, Curtin Institute of Radio Astronomy, GPO Box U1987, Perth, WA 6845, Australia
[40] National Centre for Nuclear Research, 02-093 Warsaw, Poland
[41] Institut Universitaire de France, 1 rue Descartes, Paris, 75005, France
[42] University of Granada, Dpto. de Física Teórica y del Cosmos & C.A.F.P.E., 18071 Granada, Spain
[43] Università di Pisa, Dipartimento di Fisica, Largo Bruno Pontecorvo 3, Pisa, 56127, Italy
[44] University of Johannesburg, Department Physics, PO Box 524 Auckland Park, 2006, South Africa
[45] Tbilisi State University, Department of Physics, 3, Chavchavadze Ave., Tbilisi, 0179, Georgia
[46] Eberhard Karls Universität Tübingen, Institut für Astronomie und Astrophysik, Sand 1, Tübingen, 72076, Germany
[47] Leiden University, Leiden Institute of Physics, PO Box 9504, Leiden, 2300 RA, Netherlands
[48] Friedrich-Alexander-Universität Erlangen-Nürnberg, Remeis Sternwarte, Sternwartstraße 7, 96049 Bamberg, Germany
[49] Gran Sasso Science Institute, GSSI, Viale Francesco Crispi 7, L'Aquila, 67100, Italy
[50] University of Münster, Institut für Kernphysik, Wilhelm-Klemm-Str. 9, Münster, 48149, Germany
[51] Utrecht University, Department of Physics and Astronomy, PO Box 80000, Utrecht, 3508 TA, Netherlands
[52] Accademia Navale di Livorno, Viale Italia 72, Livorno, 57100, Italy
[53] INFN, CNAF, v.le C. Berti-Pichat, 6/2, Bologna, 40127, Italy
[54] NRC "Kurchatov Institute", A.I. Alikhanov Institute for Theoretical and Experimental Physics, Bolshaya Cheremushkinskaya ulitsa 25, Moscow, 117218, Russia

ARTICLE INFO

Article history:
Received 21 October 2019
Received in revised form 20 May 2020
Accepted 31 May 2020
Available online 10 June 2020

Keywords:
KM3NeT
Data acquisition control
Neutrino detector
Astroparticle detector

ABSTRACT

The KM3NeT Collaboration runs a multi-site neutrino observatory in the Mediterranean Sea. Water Cherenkov particle detectors, deep in the sea and far off the coasts of France and Italy, are already taking data while incremental construction progresses. Data Acquisition Control software is operating off-shore detectors as well as testing and qualification stations for their components. The software, named *Control Unit*, is highly modular. It can undergo upgrades and reconfiguration with the acquisition running. Interplay with the central database of the Collaboration is obtained in a way that allows for data taking even if Internet links fail. In order to simplify the management of computing resources in the long term, and to cope with possible hardware failures of one or more computers, the KM3NeT Control Unit software features a custom dynamic resource provisioning and failover technology, which is especially important for ensuring continuity in case of rare transient events in multi-messenger

07.05.Hd
29.85.Ca

astronomy. The software architecture relies on ubiquitous tools and broadly adopted technologies and has been successfully tested on several operating systems.

1. Introduction

The KM3NeT neutrino detectors are complex objects [1] designed for neutrino astrophysics [2] and the study of atmospheric neutrino oscillations [3]. They are being built at the bottom of the Mediterranean Sea in a phased installation scheme. The infrastructure will consist of three-dimensional arrays of photosensors also called *building blocks*. Each building block will comprise 115 vertical instrumented detection lines (Detection Unit, DU) equipped with 18 optical sensors (Digital Optical Module, DOM). Each DOM [4] contains 31 photo-multiplier tubes (PMTs) that detect the Cherenkov light induced by relativistic particles emerging from neutrino interactions. The French site will host one such building block (Oscillation Research with Cosmics in the Abyss — ORCA) and the Italian site will host two building blocks (Astroparticle Research with Cosmics in the Abyss — ARCA). In each DOM, the data recorded by the PMTs are digitized and transferred to the shore station by the Central Logic Board (CLB). The settings and performance of PMTs need to be controlled and monitored. The first DUs have been successfully deployed and operated in the sea [5] after the positive outcomes of prototype DOM [6] and DU [7] campaigns. The DOMs host also other instruments devoted to monitoring and dynamic position reconstruction, as the detector shape in water currents is constantly changing. In particular, an acoustic positioning system is in place taking data from hydrophones that listen to known emitters. At the base of each DU there is a module that contains some instruments and a CLB. The Trigger and Data Acquisition System (TriDAS) [8] relies on a distributed and scalable architecture. The computing processes that implement the TriDAS have a number of running instances that may grow as needed, exceeding a few hundreds on tens of servers in a single installation. Each detector can run different tasks, with varying data taking strategies. The Control Unit (CU), a suite of computer processes exposing distributed services, has the task of directing all such hardware and software components to work together. The Control Unit is also in charge of collecting and storing logs of operations that are suitable both for machine processing and human access.

In addition to the above stated needs, the qualification and certification procedures for single PMTs, DOMs or whole DUs require running one or more data acquisition tasks in controlled environments and with multiple testing protocols [9] to ensure that all devices operate within specifications. The software running in detector operation is also used for production and testing of components. Test bench stations [10] actually work in a way that is very similar to shore stations of detectors for physics data taking.

KM3NeT searches for rare events – interactions both of primary cosmic neutrinos and of secondary neutrinos from cosmic rays – that may occur at any time. Maximizing the detector livetime is a key requirement to collect high statistics. Hence the reliability of the Control Unit and the possibility to operate continuously despite hardware or software failures have a direct impact on the statistical significance of data taking results.

The detectors are designed to operate at least for 10 years in the sea. The software makes use of widely adopted standards (see ahead in the text) at its foundation, with a large development and user base that should ensure support for a long time scale. All of the custom code is completely under the control of the KM3NeT Collaboration, whose software quality plan includes long-term software preservation.

After an overview in Section 2 of the distributed architecture, the present paper describes the various services it consists of. Authentication and identification of users and services are the functional backbone of the architecture and are described in Section 3. Run control and overall supervision are described in Section 4. Details about the representation of detectors and operational parameters in the database are given in Section 5. Interaction with the database is described in Section 6. Control of detector devices and instruments is described in Section 7. In Section 8 it is shown how the software components of the data processing chain are controlled. Details of the networking protocols and services are given in Section 9. Dynamic resource provisioning and fault tolerance are described in Section 10. Conclusions are given in Section 11. For convenience, all acronyms are listed in Appendix A. For the reader's convenience, a short summary of the concepts and structure of the data taking in KM3NeT is given in Appendix B. A dedicated paper with extensive discussion is in preparation, although an updated summary was recently released in [8].

2. Software components

The Control Unit consists of five different services that can run independently of each other:

1. Local Authentication Provider (LAP);
2. Master Control Program (MCP);
3. Database Interface (DBI);
4. Detector Manager (DM);
5. TriDAS Manager (TM).

The services can run on the same machine or on different servers, in the case of installations with failover functions. All programs are written in C#, as specified in the standards ECMA-334:2003–2006, ISO/IEC 23270:2003–2006 and following. The language was chosen because it has a large user base and good support; it is actively developed and evolving; it generally produces efficient code; many libraries are available, although the number of different external dependencies is kept small to avoid future obsolescence. The executables are encoded in a machine-independent language that is JIT-compiled by the Mono[1] compiler and can then run on different operating systems such as flavors of GNU/Linux[2], Microsoft Windows[3] and OS X. In the KM3NeT context, the Control Unit is hosted by servers running CentOS 7. Development and maintenance of the code are managed through GitLab for source code repository, continuous integration and automatic testing. Deployment uses a toolset based on Ansible[4]. Containerization has been technically tested but is not needed so far because of the inherent portability of Mono/.NET binaries.

All services have been developed to have a small footprint in terms of CPU and memory usage. They may run in more

1 https://www.mono-project.com.

2 The distributions tested include Fedora 24, SLC6, CentOS 7, Debian 8 "Jessie", Debian 9 "Stretch", LMDE 2, Linux Mint 19 and Ubuntu 16.04 LTS, but there are no evident reasons for incompatibility on others.

3 Windows 7/2008 or higher, all desktop and server versions.

4 https://www.ansible.com.

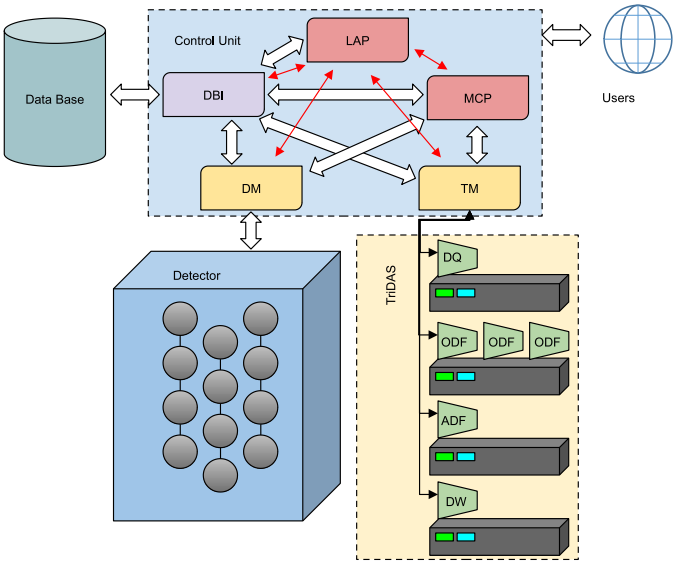

Fig. 1. The Control Unit components and their relationships. White and black arrows represent flows of control and monitoring information. Red arrows show the flow of authentication/authorization information. The flow of PMT and acoustic data from the detector to the TriDAS and hence to the final storage is not shown.

than one process on different machines for failover purposes (see Section 10) or to meet high demands in terms of workload. The latter case is foreseen for DM controlling large detectors, e.g. the full ARCA installation with a total such as 230 DUs.

Each service has a unique access point through HTTP.[5] The Graphical User Interface (GUI), when present, is offered as a Web-like service. This allows using a Web browser to perform most tasks and avoids adding software dependencies on graphical/interactive libraries. The GUI can be accessed by HTTP on VPN (Virtual Private Network) from remote controllers that pass both VPN authentication and CU authentication (see Section 3). For highly critical management purposes and basic configuration, a local console accessible only by administrators through terminal is provided. The risk of misconfiguration is assessed to be higher if coming from inexperienced users than from remote attackers, because the only ways to harm the detector and the ability to take data are power functions and system setup. Any other mistake would be quickly solved by switching to the correct set of operational parameters. Fig. 1 shows the logical connections among the services and with the detector and TriDAS components. In addition to control and logging, the Control Unit is also the bridge between the users, the central KM3NeT database [11], and the off-shore detector and the online trigger system.

3. Authentication and identification — LAP

Access to detector control and management is given to users on the basis of an authentication system, which is managed by the Local Authentication Provider. The LAP uses accounts and session tokens to manage identification and authentication. All accounts are kept in an encrypted local file together with the security credentials and privileges. When a login request is accepted for an account, the corresponding privileges are copied to a new session token that is then kept active until it expires or is deleted because of an explicit logout. The LAP uses the logical scheme of account management shown in Table 1.

[5] Secure communication on SSL/TLS could also be supported, but in a local private network of the KM3NeT ICT infrastructure this is overkill.

Table 1
Account types.

User account	Local account	The unique identifier, name and password are created locally in the detector/test bench control station and are meaningless outside of it
	Global account	The unique identifier, name and password are managed on the central database of KM3NeT and are periodically synchronized with a local encrypted cache
Service account		Defines a common name for a CU service

Operating privileges are given to user and service accounts to enable specific functions such as controlling the whole station in terms of jobs (high level) or tuning single parameters (low level). It is worth noticing that the function of a service depends on its privileges rather than on its name. This allows flexibility in the design: in the future, a single process may incorporate more than one function and this would just need a change in the registration on the LAP rather than statically hard-coding an association between a name and a function.

A user can be granted privileges one by one or in well-defined groups named roles. Because detectors take data 365 days a year, 24 h a day, the KM3NeT Collaboration follows a shift plan to share the load of detector control. Each shift lasts seven days and a shift team includes a *Shifter* and a *Shift Leader*, with the tasks of monitoring the detector operation and checking data quality. The *Run Coordinator* stays in charge for a longer timespan (usually four-eight weeks), connecting the activity of each shift team to the next and overseeing the optimization of the detector performance. The role system is especially useful for shift management: when a user is registered on the central database for a shift, he/she gets automatically and for the corresponding time window all the privileges that are defined in the *Shifter*/*Shift Leader*/*Run Coordinator* role. They are all revoked when the shift ends. A user that is registered as a *DAQ Expert* (Data AcQuisition expert, usually among the lead developers of hardware or software components) or *Detector Operation Manager* (responsible for detector management, usually for several years) on the central database automatically gets all the related privileges on all installations. For example, shifters are supposed to operate the detectors using predefined configurations, whereas experts are allowed to tune single parameters for diagnostic and testing purposes.

While the concept of a user login is quite intuitive, a service login deserves some explanation. The mere fact that a program is installed and running on a server is not enough for it to be known to the LAP (and hence to other CU services). When the program logs in on the LAP it gets its own security token and becomes known to all other CU services. This explicit login requirement ensures that the hardware resource usage can be optimized and services can be moved from one machine to another according to the needs. This also makes the initial configuration easier, because there is no need to handcraft a static configuration file. The LAP itself maintains a local database of hardware resources as an XML file. Administrators can build the configuration indirectly by issuing incremental commands to the LAPs to register new instances of services.

4. Run control — MCP

The Master Control Program is in charge of maintaining the run status of the detector and TriDAS. The complete information of the run status consists of the following pieces:

1. Current detector: a detector changes when DUs are added or removed or for a failover reconfiguration (see Section 10).
2. Current *runsetup*: the coherent set of input parameter values controlling the detector, such as PMT supply voltage, and quantities to be read out for logging. See Sections 7 and 8 for more details.
3. Current run number: a run is a timespan during which a detector is operated with the same *runsetup*; for practical reasons a long run may be split in two or more with the same *runsetup* to have smaller output datafiles.
4. Current target: the overall target of the CU can be one of the following (notice that a target change does not imply a run switch):
   - Off: all PMTs are turned off, data taking is off, no triggering or post-processing.
   - On: all PMTs are on, data taking is off, no triggering or post-processing.
   - Run: all PMTs are on, data taking is active, triggering and post-processing run.
5. Current time/position calibration: the set of adjusted positions and time offsets for individual PMTs that is being used for online triggering.
6. Current job: a job is a run schedule with a priority grade. A run may start with or without a predefined schedule because the MCP may be commanded to immediately switch the run number. A job is an entry in a schedule specifying that at some time a new run will start with a *runsetup* that is defined in advance and that will last for a certain timespan, unless preempted by higher priority jobs. One job may correspond to one or more runs. Some examples on job management are shown in Fig. 2: the baseline job is usually defined to use a *runsetup* with tuned PMT voltages and the detector in "On" state; jobs J1–J8 might be routine data taking jobs with priority 1 and the detector in Run state; job J9 might be a calibration run and job J10 might be running a special data taking. Routine jobs J6 and J7 will produce no runs because they will be overridden by J10. J3 will produce two runs because the MCP will start with it, switch to J9 after J3 has started and then fall back to J3 again when J9 ends.

Jobs may be modified before they begin and they can be truncated when they have started. The run status, run switch history and job addition/deletion/modification history are kept in a dedicated local file, which acts as a transaction log. Such information is periodically pulled by the DBI to be recorded in the central database.[6] Only after the information has been successfully written in the database, the file is purged. In addition, all run switches are recorded to a human-readable log file, but the syntax is such that, in case of loss or corruption of the run status file, it is possible to reconstruct the latter from the former. A file-based local storage is a better option than a local database instance for several reasons:

- it is faster than a full-fledged database;
- it requires almost no expertise to manage;
- it requires no licensing costs;
- it avoids introducing additional dependencies on external software components that may become obsolete or unsupported.

[6] Remotely hosted in the computing center CC-IN2P3 in Lyon — https://cc.in2p3.fr.

In standard operation, a detector may be required to run for months with the same operating parameters. For this purpose, it is possible to use the "auto-schedule" feature that automatically fills a priority line with jobs of equal duration and a specified *runsetup* and target. This frees shifters from error-prone repetitive tasks.

Whenever the run state changes, the MCP notifies all the services that are registered in the LAP with the *Status_Notification_Privilege*) privilege, which usually means at least DM and TM. This is a "push" type notification, aimed at fast communication. Fault tolerance is ensured by the "pull" communication mode: the DM and TM periodically update their knowledge of the run state by retrieving such information from the MCP. A finite time will elapse between the run switch by the MCP and the reaction in the DM and TM. All of this is logged and it is possible to precisely identify the run switch latency time in each case.

A run switch is also triggered by a system reconfiguration after a fault (more detail in Section 10). It is worth pointing out that the MCP alone is responsible for providing a unique pair of detector and run number for each run. Different detectors in different KM3NeT sites can use the same run number without clashes.

The MCP offers a Web-based GUI (Fig. 3) to perform all routine tasks, with the exception of service configuration and disaster recovery. The GUI enforces user privilege compliance: job scheduling is not allowed to users that are neither on shift nor *DAQ Expert* privilege owners. An additional security check layer involving LAP queries is able to filter out possible HTTP forged queries that may try to circumvent or bypass the GUI. In this context, HTTPS would be possible but overkill because security is focused on ensuring compliance, by users and automated processes, to data taking procedures. All communications already occur on a private network and users connect through a VPN.

5. Detector description and *runsetups*

5.1. Detectors

KM3NeT detectors are described and defined in the central database. A detector always has a location and a start timestamp, which is the first time it is connected and can provide signals. The end timestamp is set on its final disconnection. The same physical detector, located in the same place and reconnected, would have a different detector identifier. From the point of view of the CU, a detector is a list of previously integrated DUs and TriDAS processes, namely Data Queues (DQs) to rearrange data packets from single DOMs into events, Optical/Acoustic Data Filters (ODFs/ADFs) to run triggering algorithms and Data Writers (DWs) to write data to disk. In a basic implementation, TriDAS processes are defined in the central database, the process map is static and there can be no failover plan. In a more evolved view that supports dynamic provisioning and failover, the set of TriDAS processes might change during the lifetime of a detector and even several times per day in case of failures or addition of computing power. Nevertheless, from the point of view of MCP, DM and TM, there is only one definition of a detector that is provided at a certain time, and it always includes TriDAS processes.

5.2. Runsetups

PMTs need their operating high voltage (HV) to be fine-tuned in order to provide uniform performance. The optimal value might also change over time. Likewise, functions may need to be enabled or disabled on certain DOMs, especially for testing and calibration purposes. *Runsetups* define the input to each DOM and the output for feedback, monitoring and data logging, and all of them depend on the purpose each *runsetup* was defined

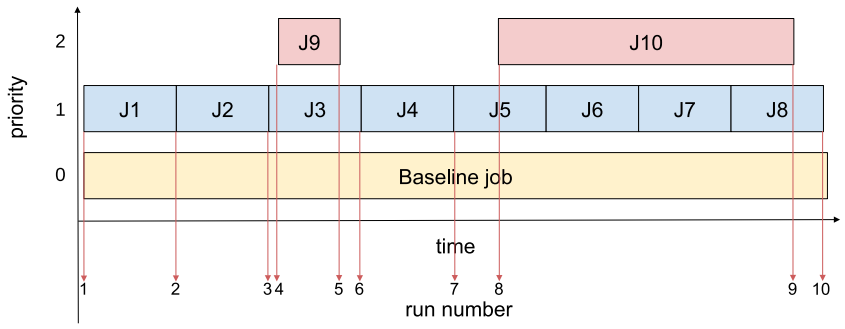

Fig. 2. Example of job chart. See explanations in the text.

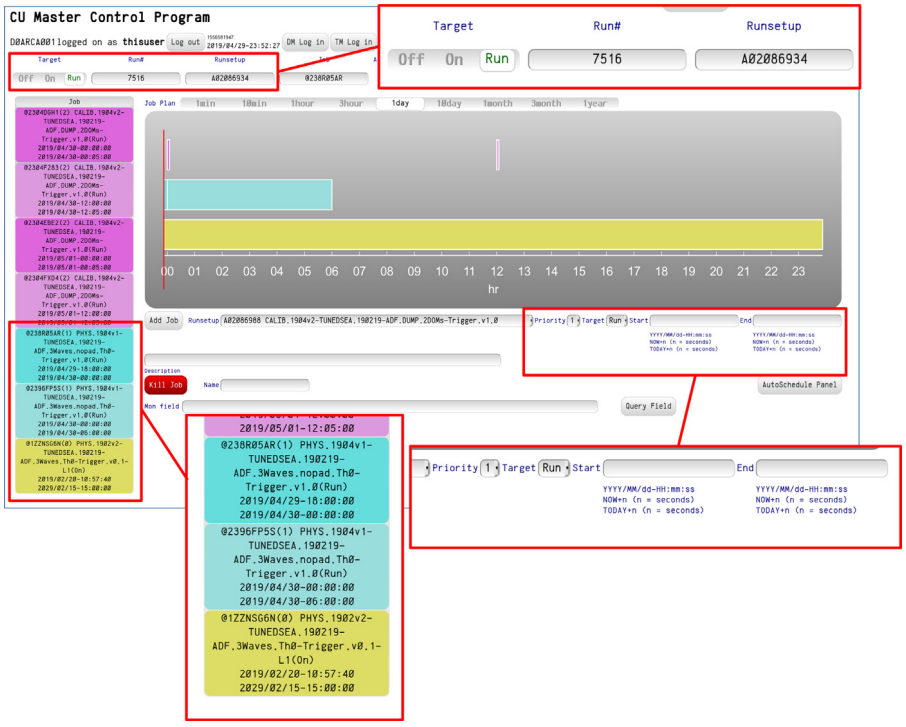

Fig. 3. Screenshot of the MCP graphical user interface (framed in blue), with several jobs scheduled at different priorities. Selected details, framed in red, are enlarged to improve readability.

for (e.g. minimum data filtering, timing tuning, HV tuning, etc.). Many *runsetups* differ only for some sets of parameters. Parameters with correlated meanings and purposes (e.g. PMT HVs, threshold and activation state) coalesce into *configuration groups*. Each *runsetup* is an ordered list of configuration groups, which are picked at various levels as referring to a whole category of items, subcategory or individual items.

6. Interaction with the database — DBI

The service named Database Interface (DBI) is devoted to handling the interaction with the central database. Its main operating principle is to work as a file buffer to replace SQL/DML interaction of programs with the database as sketched in Fig. 4. The main reasons to implement a DBI are:

- To avoid redundancy, database access credentials are stored in a single place at CU installation time and encrypted for safety.
- Decoupling CU code and database code/schema. SQL queries and/or DML statements need not be written in any code outside of the DBI itself. All the complications of handling and converting database data types are handled by the DBI and the client code is written in terms of CU data structures. This allows for refactoring on either side, i.e. the necessary evolution over time of both the CU and database will not affect each other.
- Coping with remote connection instability. The connection with the central database uses a Wide Area Network, which is intrinsically unreliable. The DBI stores all the datasets that are needed for CU operation in a local cache, speeding up access and improving reliability. On the other hand, the DBI buffers write operations and replays them if they fail because the database is not accessible.

Information sets that have been downloaded from the database are hosted in the download cache in XML format. As shown in Fig. 4, they include:

- The current detector definition. It is downloaded only when the detector definition changes, by authorized users.
- All *runsetups* written for the current detector (a one-to-many relationship). The DBI regularly polls the database for appearance of new *runsetups*, but on-demand access is tried

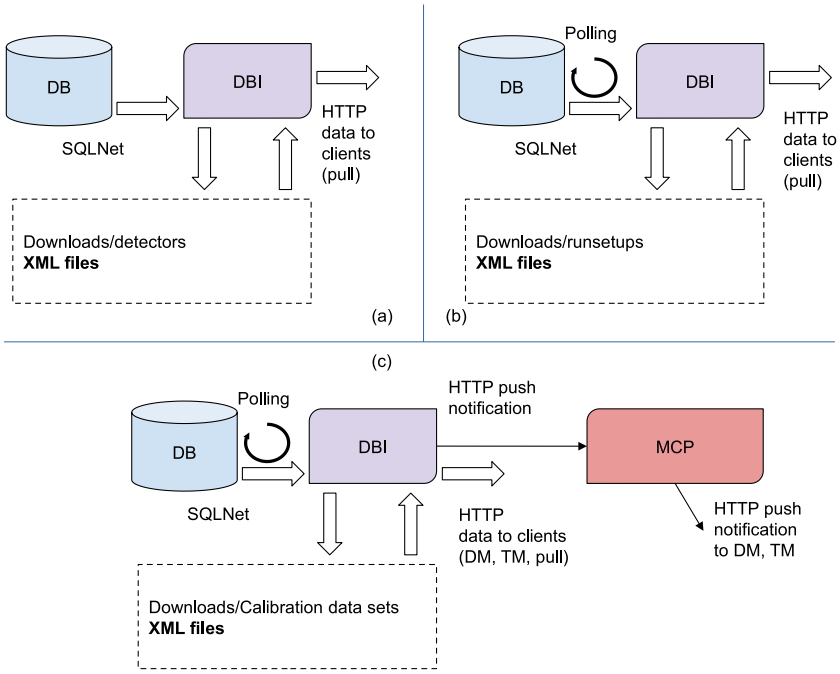

Fig. 4. Logic and network protocols involved in data download from the database. (a) Detector data flow. (b) Runsetup data flow. (c) Calibration data flow.

for *runsetups* required by the MCP/DM/TM that are not yet in the cache.
- The current sets of calibration data. These data are continuously polled for updated versions and immediately pushed to the MCP and other services.

Runsetups are usually created by humans, so the time of their creation is well separated from the time they are used. Calibration datasets are instead supposed to be updated regularly and automatically to have optimal detector operation. As soon as a new set is available and has been successfully downloaded, the DBI notifies the MCP which decides when to switch the run and broadcasts the signal to other services. In this sense, the DBI is an active part in data taking.

The upload cache stores data queued to be written in the database, usually flushed upon successful transfer. In this case, binary files are expected in the native format generated by writer programs. The DBI handles the needed conversions. At present, the following types of data are hosted in the upload cache:

- DM *datalogs* that contain detector monitoring data and notification of management events, such as the real time of run start for each CLB, which is different from the time the MCP issues the command to change the run number; see Section 7 for more details.
- TM *datalogs* containing logs of TriDAS activity, documenting actual starting–stop times of each run process-by-process, possible crashes, etc.; see Section 8 for more details.
- Times-Of-Arrival (TOAs) of acoustic wave pulses found by the ADF(s).

Run book-keeping information is "pulled" by the DBI querying the MCP and written to the database without going through a local cache. This reflects the fact that *datalog* and TOA tables in the database have foreign keys to the table of runs: an error in *datalogs* or TOAs remains confined to that dataset, but an error in run book-keeping would have a cascade effect of errors on other tables. The DBI will send a "purge" command to the MCP for runs and jobs that have been successfully written. *Datalogs* and TOAs for the runs and jobs that have been already communicated to the database and staged in the upload cache are cleared for writing to the database, whereas all other data therein are kept standing by. Fig. 5 shows the different information flows.

When a *datalog* or TOA set write fails, it is not retried until another write of *datalog* or TOA set succeeds. This copes with the case of Wide Area Network failure: for a certain timespan all writes fail, but each dataset is tried only once. As soon as the database can be reached again, all queued writes are executed. If a dataset cannot be written multiple times (usually the threshold is set to 5), it is flagged as "failed" and must be reviewed by a DAQ expert.

7. Detector management — DM

Detector subsystems work according to the state machine depicted in Fig. 6. The three states "Idle" (corresponding to the "Off" target), "Ready" (corresponding to the "On" target) and "Running" (corresponding to the "Run" target) are stable, in the sense that they are supposed to be kept throughout the duration of a run job lasting several hours. The "Standby" and "Paused" states are transitional. The DM drives the state machine of each CLB issuing events that are transported over the network. The task of the Detector Manager is threefold:

1. setting the input parameters of DOMs as specified in the current *runsetup*;
2. driving the state machines of the CLBs according to the current target;
3. reading out and logging the output parameters of DOMs producing the *datalog* files ready to be written to the database by the DBI (see Section 6).

The DM is indeed the most critical component of the CU from the point of view of scalability to the size of a 115-DU block and beyond. It receives messages from all CLBs and sends messages to all CLBs, so it is expected to have CPU and memory footprints that are linearly dependent on the overall number of DOMs.

The DM is expected to receive all notifications from the MCP when the run state changes. As mentioned above, even if the

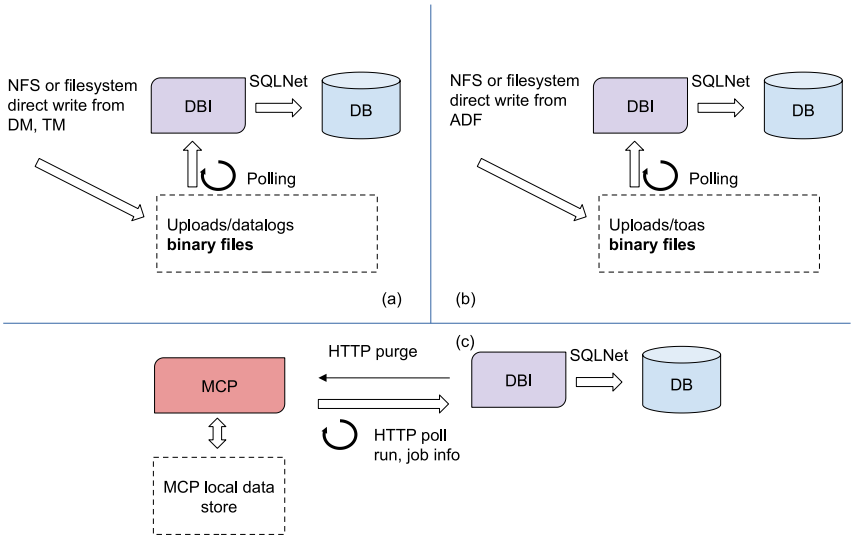

Fig. 5. Logic and network protocols involved in data upload to the database. (a) Datalog data flow. (b) TOA data flow. (c) Run and job book-keeping information flow.

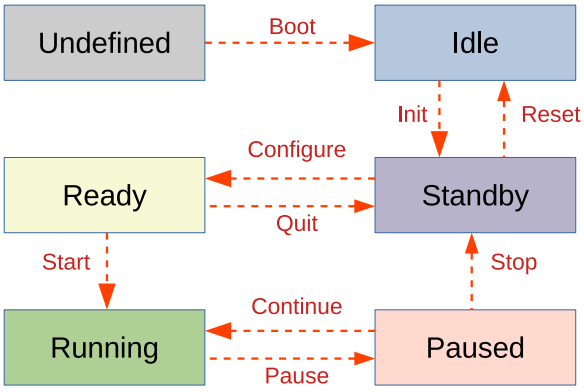

Fig. 6. The state machine for the data acquisition as implemented in both the CU and the CLBs: the states are boxed, while the events are paired to the dashed arrows that indicate the related state transitions.

"push" mode misses a beat or a communication error occurs, the DM regularly polls the MCP to know the run state. In fact, this allows the DM to work even if the system is being reconfigured while data acquisition goes on. Any run state change will be logged so that it can be written to the database.

Every time the detector or the *runsetup* changes, the DM goes through all DOMs to reconfigure them. This means working out the full list of input parameters and their values and output parameters. The list is customized to the level of a single PMT. The DM communicates with the CLBs by means of the Simple Retransmission Protocol (SRP — see Section 9.4), a UDP (User Datagram Protocol)-based protocol. It includes functions to set up and establish the link between the DM server and the CLBs. This is useful both when the DM first starts up, when DUs are rebooted or when a CLB needs to be restarted. One of the purposes of the DM is monitoring the activity of the CLBs and regaining control of those that may stop communicating, thus minimizing the need for human interventions. SRP allows point-to-point messages from DM to the CLBs and back, broadcast messages and subscription-based data transmission, so that the DM asks once for the set of parameters to be monitored and receives regular updates (1 Hz or 0.1 Hz) without the need for further polling. Each CLB exposes the following subsystems:

- System (SYS)
- Network (NET)
- Optics (OPT, only for CLBs hosted in DOMs)
- Acoustics (ACS)
- Instrumentation (INS)
- Base (BSE, only for DU base modules)

Each subsystem is controlled independently of the others. However, except during the short timespans of transitions, all subsystems should be in the same state. The DM takes actions when it receives a new CLB status report: it is compared to the currently expected state and, only if they are not in agreement, a new event is generated so the state machine moves to another state. Parameter setting is only allowed in the Configure event that connects the Standby state to the Ready state. Hence, any change in parameters implies driving the CLB state machine to the Standby state, setting the parameters and then putting the state machine in the state that is consistent with the current target. In doing so, also the run number is compared to the corresponding monitoring variable shown by the CLB. If they differ, the DM directs the CLB to go through all the states needed, until the CLB run number matches the current run number defined by the MCP.

For testing and troubleshooting, the DM also provides a manual mode that is reserved to users that hold the *Detector Control* privilege (usually *Run Coordinators* and *DAQ experts*). The manual mode can be activated on single CLBs and allows operators to tweak every single parameter and to control the state machine issuing events manually via a GUI. When the "automatic" control mode is restored, the CLB goes back to normal operation, but newly set input parameters are not restored until the next run switch. The ability to control parameters manually is useful to fix possibly critical conditions while a new *runsetup* is being prepared and a new run is ready to start.

For DU base modules, it is also possible to use the GUI to toggle the DU power. This function is reserved to holders of *Detector Control* privilege. Some parameters can be tuned only through the DM console command line, as they may cause severe damage to the detector, such as overcurrent.

As shown in Fig. 7, the DM has one Control Thread to handle a serial queue of external commands (mostly from the MCP, but also from shifters and console commands by administrators). There is one CLB Controller per CLB, but this does not have

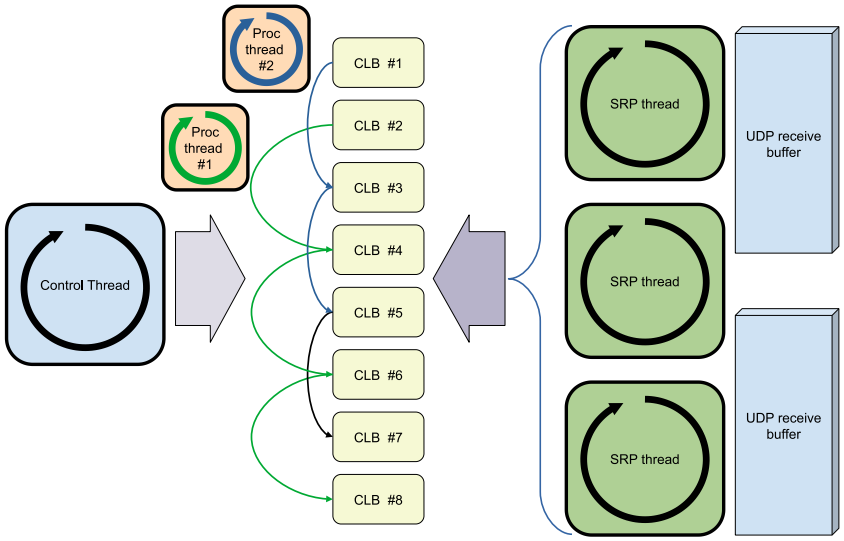

Fig. 7. Sketch of the threading structure of the DM. The HTTP thread pool is not shown. The Control Thread sends messages to all CLB Controllers, which have no thread of their own. Processing threads (two in the sketch) power the CLB Controllers by sharing the workload. SRP threads (three in the sketch) read the messages found in two UDP socket buffers and convert them into events for CLB Controllers. Large arrows show the communication flow towards CLB Controllers. Small arrows show the sharing of CLB control workload among processing threads.

its own thread: the usage of computing resources by the DM has to be carefully controlled. Although it is a naturally multi-threaded application, the usage of thread pools is limited to the HTTP interface. The allocation of memory and threads for SRP communications and for CLB action processing is statically configured. It can be changed by explicitly setting configuration parameters in the DM console, but it cannot change during a run. The UDP receive-buffer size can be statically configured. In case of oversubscription, i.e. when too many SRP messages arrive, a fraction of them is automatically dropped. Monitoring messages are grouped by type and source; in case of excess load on the processing thread, subsampling occurs by dropping a suitable fraction of messages. Such loss of information results in a decrease of the average sampling frequency of the detector monitoring. The DM provides counters to diagnose the communication and computing load, so that DAQ experts can adjust the allocation of resources. As a reference, sampling a DU at 1 Hz uses about 10% of one typical CPU core (Intel Xeon Silver 4116 at 2.1 GHz). This implies that about 12 cores should be enough for the monitoring of a whole block of 115 DUs. It has been shown that a single socket with 64 KiB receive-buffer can monitor at least three DUs. The number of sockets can be tuned according to the needs, allowing to scale to a full detector of multiple blocks.

The same program for DM is used in the various KM3NeT environments of detector control, such as shore stations, qualification test benches and development installations. In some cases, specific actions that are normal in other contexts may carry high risks because of peculiarities in earlier hardware components (e.g. first DUs deployed, old DOMs, etc.). The DM has a standard blacklist of such actions (mostly related to power control functions) that need to be individually allowed. An additional module, called "Authorization Block", which is compiled to run on a well-identified machine in a single geographic place, enables those actions that are potentially dangerous. The Authorization Block makes sure that an administrator has explicitly unlocked all permitted functions. A DM without an Authorization Block or with a locked one would filter all actions in the blacklist.

Two outputs are continuously generated by the DM: one is a human-readable log and the other is a binary formatted *datalog*. The latter is produced at regular intervals (usually 10 min) or when it reaches a certain size (32 MiB in memory). It contains a chunk of monitoring data ready for database insertion. Usually it is written in the upload cache of the DBI (see Section 6). Subsampled snapshots are exposed in the Virtual Directory (see Section 9.3) that is available via the HTTP, mostly for GUI purposes. An example of a screenshot of the GUI with live monitoring data is shown in Fig. 8. In addition, other programs may read them if needed.

8. TriDAS management — TM

The TriDAS is a set of programs developed in compliance with the requirements of the KM3NeT data taking and processing framework. In most scenarios there is at least one Dispatcher, one or more Optoacoustic Data Queues, one or more Optical Data Filters, one or more Acoustic Data Filters and one or more Data Writers. All the programs need to be driven in a coordinated way, consistently with the current operational target. They all feature a state machine that is identical to the one implemented in the CLBs. As a general guideline, normally all TriDAS components should be in the same state as a generic CLB. Like in the case of the DM, each TriDAS element has its own TriDAS Element Controller. In practice, control and communication are so different for CLBs and TriDAS programs that there are very few similarities in the inner structure of DM and TM. The inner structure of TM is shown in Fig. 9.

If a CLB suddenly stops responding, data taking by the remaining ones goes on unperturbed. In the case of a crash of a TriDAS program, bringing it quickly up again is important to minimize data loss, depending on how critical is its task. A crashing ADF is almost harmless if it comes back again within a few hours. An ODF that suddenly disappears leads to a proportional loss ($1/N_{ODFs}$) in detector livetime for the duration of the restart procedure. An Optoacoustic Data Queue that crashes leads to total data loss until it is up and running again. Of course, the Data Writer is also crucial because data need to be saved. It is worth mentioning that, counting all instances of the various processes, we have witnessed so far stable operation beyond 30 years, crashes occurring only immediately and repeatably on wrong configurations.

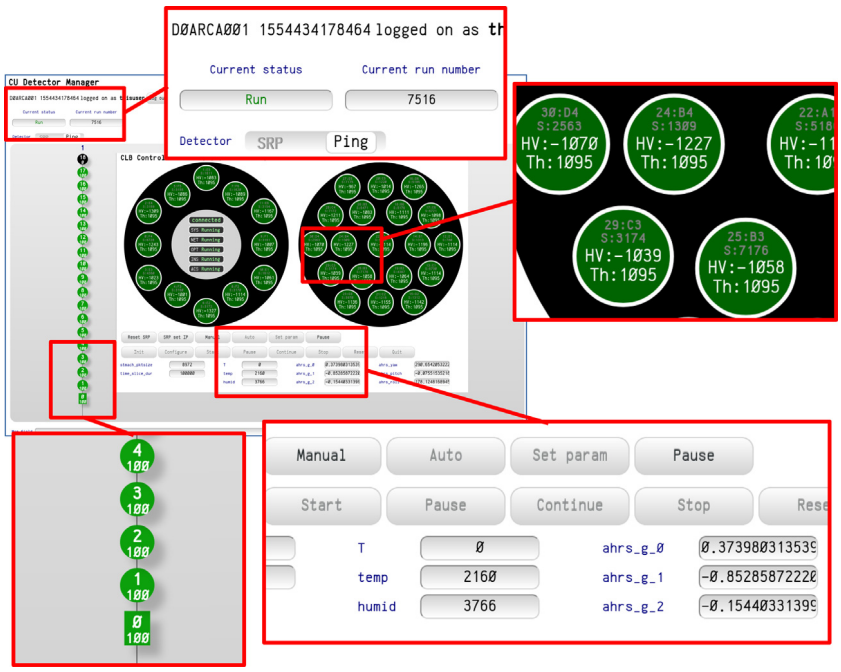

Fig. 8. Graphical user interface of the DM for one DU and one DOM (superimposed). The screenshot is framed in blue. Live monitoring data obtained via HTTP are shown. To improve readability of the screenshot, selected details are enlarged and framed in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

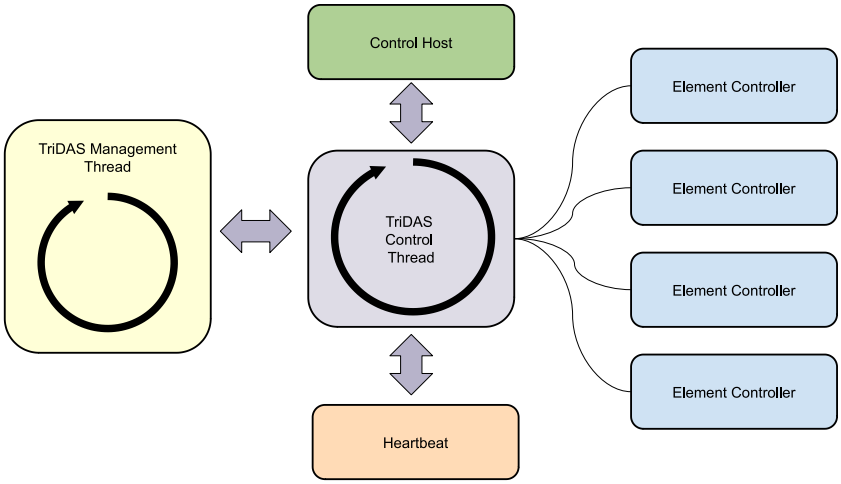

Fig. 9. Sketch of the threading structure of the TM. The TM Control Thread receives commands from HTTP and from the console. The Control Thread powers the Element Controllers while the Control Host interface ensures the I/O and the Heartbeat provides a clock.

While the DM communicates with the CLBs directly one by one, TriDAS processes use the Control Host protocol[7] to communicate through the Dispatcher. As a result, the TM receives a time-ordered stream of messages from the TriDAS processes. This has some implications on the control process:

- The Dispatcher must be identified, contacted and a permanent TCP (Transmission Control Protocol) connection with the TM has to be established.
- The Dispatcher cannot be used to start processes (although it can be used to stop them).
- While the stream is time-ordered, it is not time-aware, in the sense that it does not produce timeouts like a point-to-point connection does. As a result, a command that is not answered will not automatically produce a timeout error.

A local agent (named TriDASManager Agent, or TM Agent) communicates with the TM to receive the requests to start or stop programs and uses the LAP to check that the requests are authorized. The TriDASManager Agent has a security system for credentials that is integrated with the CU. The internal TriDAS Element Controllers have a few more states in their state machine to handle the cases of a program that is starting up, but not responsive yet, or shutting down. A "Heartbeat" is introduced to

7 Originally developed by R. Gurin and A. Maslennikov (CASPUR, 1995).

measure time at a central level that is then broadcast to TriDAS Element Controllers.

The TM is a very lightweight application, with a CPU workload that is normally about 3 or 4% of a single core (Intel Xeon Silver 4116 at 2.1 GHz) and only ramps up a bit during run start. It also provides *datalogs* to document starting and stopping times and working conditions for each process.

9. Networking

The CU uses several protocols for communication. This is the result of matching diverse needs and complying with existing standards or practices.

9.1. HTTP-based access

All CU services use HTTP as a basic communication protocol. Each CU service uses a lightweight Web-server library that allows exposing an HTTP access. Notice that this is the opposite of what Web-hosting normally means, i.e. hosting the application inside a Web-server. In this case, the application has its own port and its own Web-server dedicated to it. As a matter of fact, a service that runs as a daemon needs a way to communicate with machine administrators, and this often goes through TCP. Using an HTTP interface allows reducing the needs for ports, because both the administrative traffic and user access can go through HTTP.

When applicable, the HTTP server hosts conventional HTML pages for a GUI. They are exposed in the `/gui` URL directory. Common image formats, CSS style sheets, JavaScript source files and AJAX are all supported.

9.2. SAWI remote calls

On top of the HTTP layer, the Server Application Web Interface (SAWI) provides a lightweight implementation of remote procedure call. SAWI exposes four virtual directories:

- `/listmethods` gets the list of callable methods;
- `/call` calls a method passing parameters;
- `/callret` gets the result of a long-running method call;
- `/listcalls` shows the list of calls and their completion status.

Subpages of these virtual directories are supposed to be called by programs and have no human-oriented formatting. The pages provided by default at the virtual directories instead show the available options: in practice, a skilled user can mimic remote procedure calls and use the Web browser as a debugger.

SAWI allows both blocking calls and asynchronous calls. The result of an asynchronous call is stored as a job object that is remembered for a set time (usually 10 min) after it ends. The caller is expected to poll the `/callret` virtual directory for completion, specifying the process ID and the ID of the call to get the result. The process ID is provided by the server: in principle, a client has to account for a server process to be restarted, so the call ID is not enough alone to uniquely identify a client–server call. If a server process ID changes, the client knows that the result of the method call is lost and there is no ambiguity. SAWI provides support only for simple datatypes (Bool, Int, Long, Double, String). However, it transports exceptions from the callee to the caller and distinguishes exceptions of the remote call protocol from functional exceptions.

From the point of view of the developer, usage of SAWI is very simple: the callee just has to flag the methods that have to be exposed with the *WSrvPage* attribute. The C# method is reflected at runtime and exposed over the network with the same parameter names. The caller has to include a *WSrvPageClient* object that declares the name and the type of parameters. The first member of all CU calls is a token string that is checked with the LAP to ensure that the caller has the right to call the procedure. Before the call, the server and port have to be set. The *WSrvPageClient* object can be used multiple times. Fig. 10 shows an example screenshot of the `/listmethods` steering page for a TriDASManager.

9.3. Virtual directory

Each CU service exposes a `/mon` directory that is meant to contain real-time monitoring data on the service application. They are organized in a virtual directory tree that does not correspond to any file on disk. Leaves in the tree are elementary data, i.e. JSON objects containing the data value and the time it was set. Fig. 11 shows an example of Virtual Directory path to real-time monitoring on the DM.

The implementation of the Virtual Directory structure contains several details that are relevant for optimized performance:

- each time a new leaf in the tree is created, the server gets a direct reference to that leaf, which can then be updated without browsing the full path, which would waste CPU power;
- HTTP clients are allowed to create shortcuts that gather a client-defined set of variables in a single shot: subsequent calls to the shortcut can retrieve unlimited groups of variables by direct access to their leaves;
- writers access direct references to the data leaves in the tree, so they do not need to traverse and lock the tree to repeatedly update values, avoiding mutual locking with readers.

Virtual Directory data can be accessed both for the purpose of creating GUI pages or to run monitoring scripts or applications. Using web clients as well as ubiquitous executables such as `wget`[8] or `cURL`[9] it is possible to write specific monitors.

9.4. Simple retransmission protocol

The UDP protocol on which the communication between DM and CLBs is based is the SRP. It tags messages and tracks message acknowledgments to allow re-transmission if needed. The DM uses a light version of the SRP library, written in C#, supporting the subset of the functions that are needed for routine duty. Some diagnostic and debugging features would not be useful in an automatic control context.

9.5. Control host interface

The Control Host library, which is used as the inter-process communication protocol among the data triggering and processing applications, is ported in C#. The Control Host protocol is used by the TM to connect to the Dispatcher and read/write messages to components of the TriDAS. Each message has a tag and is dispatched to all clients that subscribed for that tag. Since each client can subscribe both for specific tags and for its own unique identifier, both broadcast and one-to-one communication are possible. The Dispatcher collects all incoming messages and enqueues them into serial pipelines. Unlike SAWI, which is a connectionless protocol, the Control Host protocol is built for high-speed data transfer but requires a persistent TCP connection. A network error or a disconnection would be interpreted as the client program closing and reported as such to subscribers.

[8] https://www.gnu.org/software/wget.

[9] https://curl.haxx.se.

Available WebService pages http://127.0.0.1:1302/listmethods/

Available WebService pages - Process Instance 4785

Void	NotifyTargetChange	*token,mode*
String	GetTarget	*token*
Void	Disable	*token,clb,subsystem*
Void	Enable	*token,clb,subsystem*
Bool	IsEnabled	*token,clb,subsystem*
Void	NotifyRunsetupChange	*token,runsetup,runnumber,runstartunixtime,t0set*
Void	NotifyRunnumberChange	*token,runnumber,runstartunixtime,t0set*
String	CurrentDetector	*token*
String	CurrentRunsetup	*token*
Long	CurrentRunNumber	*token*
Void	NotifyDetectorChange	*token,detector*
Void	NotifyTriDASOverrideChange	*token*
Void	NotifyOpticalDataTargetChange	*token*
String	GetAuthenticationManager	*token*
Void	Terminate	*token*
	Call method	

Fig. 10. SAWI steering page for method calls from a DM. Clicking on each link would show a new page where the arguments to the call can be filled and it can be started. Clients using SAWI would jump directly to subpages, e.g. `/call/Current`*`Runsetup`*`?token=aabbbccc` ("aabbbccc" is meant to be the security token).

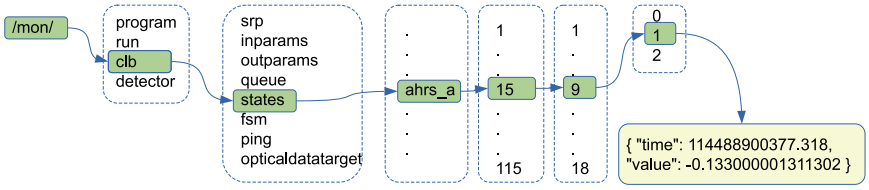

Fig. 11. The KM3NeT detectors change their shape under the action of water currents. The orientation and acceleration of the DOMs are constantly monitored. The example shows the Virtual Directory path to `/mon/clb/outparams/ahrs_a/15/9/2` obtaining the vertical acceleration value of the 9th DOM of the 15th DU.

10. Dynamic resource provisioning, failover and risk analysis

KM3NeT detectors are expected to operate for at least 10 years. During such timespan, TriDAS servers will be added and upgraded. Some servers will fail and will be replaced with newer ones. Adding, removing and replacing machines should be made easy to help system administrators, who may also change. The importance of maximizing the livetime of the detector has already been emphasized. It is worth noting that it is not only a matter of high percentages of integrated active time, but that even a few consecutive hours of downtime would prevent the observation of rare astrophysical transient events such as supernova neutrino bursts. In this respect, whenever the detector and shore station have enough resources to run, they should be running, even if not at 100% performance level. This is even more relevant if all powerful servers, which should host ODFs, are in service and just a CU machine has failed. For example, if the machine that hosts the TM fails, the acquisition does not even start, while all the real computing power is there just waiting for a command. A failure analysis has been performed to review the impacts of different failures and assess the corresponding service losses. Conservatively, the mean time between failures of hardware can be estimated to be of the order of five years, but services may be down because of software upgrades, which happens several times in a year and is largely the most common cause of temporary operation interruption, although for short time intervals, of the order of 10 min. The analysis is not limited to the CU but also includes the parts of the TriDAS that directly depend on the CU and considers failures caused by one or two concurrent events.

As shown in Table 2, disentangling different functions and putting them in different programs has already a positive impact on data taking stability, because the first five rows have low or medium severity. Indeed, it is common to upgrade the system during an ongoing run, shutting down services and restarting them one by one. Nevertheless, there are still other high severity scenarios due to multiple failures at the same time. A redundancy in all CU program instances can be introduced by having the same CU service running in multiple machines. This can be obtained with the "Dynamic Resource Provisioning and Failover" mode:

- The list of machines and services is not defined in the database but it is maintained by the LAP, and continuously logged to the database. This is a natural extension of the basic LAP function of recording users and services.
- For each CU service there are at least two installations, but only one is running while the others are kept in standing by.
- When running in Dynamic Resource Provisioning mode, every CU machine runs a LAP that hosts a Health Checker sub-service to perform basic tests. If a Health Checker determines that tests are not passed, it causes the automatic shutdown of the services that are locally hosted: one must avoid that there are conflicting managers, for example two DMs at the same time, one connected to the MCP and the other disconnected.
- LAPs poll the Health Checker service to get the status of the machine. A Health Checker answers the status polls that are issued regularly (e.g. 0.1 Hz) in normal conditions. If a machine crashes or fails, its Health Checker does not answer. If the Health Checker answers that the tests are not passed, the machine is not considered suitable to work as if it were failed. The Health Checker itself may fail, but given the fact that the code it runs is very simple, it can be assumed that there is a good (hardware) reason for its failure rather than a software problem.
- LAPs may reallocate CU services. When they decide to do so, they direct the MCP to switch to another run and the

Table 2
Single-condition and double-condition failure schemes.

First condition	Second condition	Loss of service	Impact on data loss
LAP down	N/A	GUI inaccessible	LOW
MCP down	N/A	Current run does not end	LOW
DBI down	N/A	*Datalog* + TOA upload pause	LOW
DM down	N/A	Missing *datalogs*	MEDIUM
TM down	N/A	None	LOW
DM down	Run switch	Data loss (run number lag), missing *datalogs*	HIGH
TM down	Run switch	Data loss (run number not set), missing *datalogs*	HIGH
OADQ server down	N/A	Partial data loss	HIGH
OADQ server down	Only available server	Total data loss	HIGH
ODF server down	N/A	Partial or total data loss	HIGH
ODF server down	Only available server	Total data loss	HIGH
ADF server down	Only available server	Total data loss	HIGH

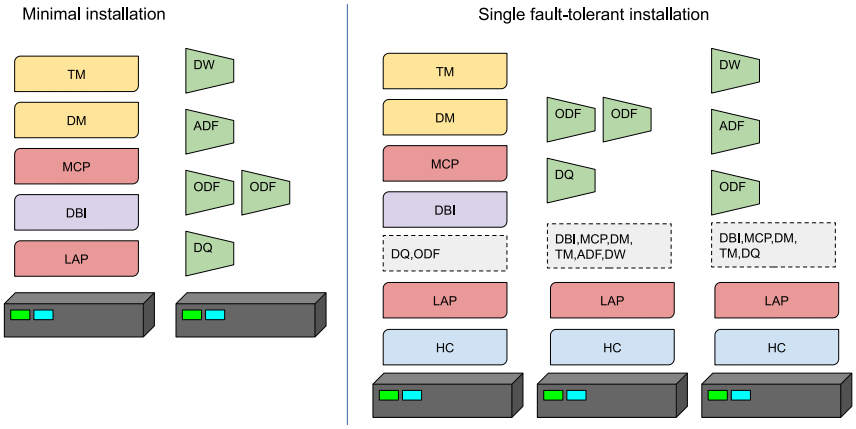

Fig. 12. Two configurations of the CU software stack. Left: minimal installation as used in testing stations without fault tolerance. Right: single fault-tolerant installation, as should be used in shore stations. Grayed areas show the services that are installed but kept standing by waiting to take over in case of failures.

(new) DM and (new) TM to reshape the detector definition and start a new one.

- LAPs may reallocate TriDAS computing power. When they decide to do so, they direct the MCP to switch to another run and the DM and TM to reshape the detector definition and start a new one.
- There is no central authority among LAPs. They synchronize their status continuously and services that must exist in single instances are automatically assigned to the available machine with the lowest IP address. Agreement is therefore not imposed by an authority that may itself run on a failed machine, but relies on algorithmic consistency.

In this approach, also TriDAS resources are recorded and managed in LAPs, which become a redundant set of local resource managers. The detector definition that MCPs, DMs and TMs get is the current one, i.e. one of all the configurations that are possible with the available resources. In logical compliance to this, at run start, the DM and TM record in *datalogs* the definition of detector that they are using for that run. A detector change always triggers a run change. Figs. 12 and 13 show the full CU stack in various configurations.

The number of Data Queues and Optical Data Filters may change with different processing configurations if computing machines fail. However, it is considered better to run with reduced resources than not running at all. Conversely, in this scenario the addition or replacement of a server is done by just registering the machine change on one of the LAP. It will then propagate the information to others and a detector change will soon allow the newly acquired computing power to enter data taking. Switching from one configuration to another should take place in less than 10 s, which is compatible with the duration of most astrophysical transient phenomena.

11. Summary and conclusions

The Control Unit of the KM3NeT data acquisition is a system built of several components that work together with the common goals of maximizing the live-time and data quality of the operated detectors, both in the deep sea as well as in component testing/qualification stations. Modularity helps achieving the target of reliability, because several parts of the Control Unit are able to continue their activity despite the temporary unavailability of others. The architecture used, based on the HTTP protocol for interprocess communications, ensures maximum openness of data and algorithms. Graphical user interfaces are provided through common Web technologies. It is possible to access the inner status of Control Unit programs by means of any Web browser. Scalability is guaranteed by performance optimization and careful design choices. Tests indicate that a single common server with 32 cores and 32 GB RAM can control a full detector made of two building blocks, with a total of 230 Detection Units. Although the Control Unit continuously reads and writes data to the remote authoritative database of KM3NeT, possible Internet downtimes are handled without interrupting the detector operations thanks to a dedicated caching system. As the Control Unit is usually accessed through private networks, safety practices are mostly focussed on avoiding mistakes that might affect data quality or detector functionality. In order to achieve that, the Control Unit implements a complete system of privileges for specific operator categories, integrated with the central database. To simplify the administration of the DAQ system and enhance fault tolerance, a Dynamic Resource Provisioning and Failover technology has been developed. It enables the Control Unit to cope even with hardware failures of the hosting servers: all the software services of either the Control Unit and the Trigger and Data Acquisition System can be automatically restarted on different machines,

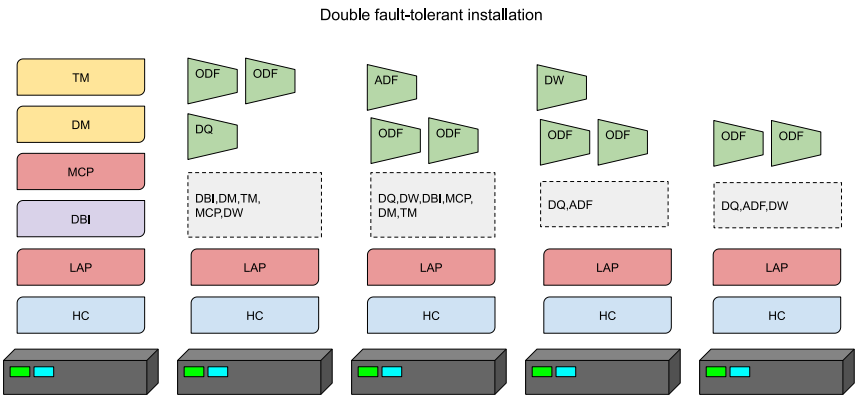

Fig. 13. Double fault-tolerant installation: up to two machines may fail at the same time without stopping acquisition, triggering and storage.

exploiting all the available computing resources coherently to the prefixed redundancy plan. The Control Unit is currently in service in more than ten sites, including the two shore stations for the ARCA and ORCA KM3NeT detectors in the Mediterranean Sea and various integration and testing stations. The project benefits of the increasing experience on the detector operations and on the continuous feedback from the users. This strategy allows for increasing the operational reliability of the Control Unit and provides with widespread knowledge for the lifetime of the KM3NeT scientific program.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors acknowledge the financial support of the funding agencies: Agence Nationale de la Recherche, France (contract ANR-15-CE31-0020), Centre National de la Recherche Scientifique (CNRS), France, Commission Européenne (FEDER fund and Marie Curie Program), Institut Universitaire de France (IUF), France, IdEx program and UnivEarthS Labex program at Sorbonne Paris Cité, France (ANR-10-LABX-0023 and ANR-11-IDEX-0005-02), Paris Île-de-France Region, France; Shota Rustaveli National Science Foundation of Georgia (SRNSFG, FR-18-1268), Georgia; Deutsche Forschungsgemeinschaft (DFG), Germany; The General Secretariat of Research and Technology (GSRT), Greece; Istituto Nazionale di Fisica Nucleare (INFN), Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR), PRIN 2017 program (Grant NAT-NET 2017W4HA7S) Italy; Ministry of Higher Education, Scientific Research and Professional Training, Morocco; Nederlandse organisatie voor Wetenschappelijk Onderzoek (NWO), the Netherlands; The National Science Centre, Poland (2015/18/E/ST2/00758); National Authority for Scientific Research (ANCS), Romania; Ministerio de Ciencia, Innovación, Investigación y Universidades (MCIU), Spain: Programa Estatal de Generación de Conocimiento (refs. PGC2018-096663-B-C41, -A-C42, -B-C43, -B-C44) (MCIU/FEDER), Severo Ochoa Centre of Excellence and MultiDark Consolider (MCIU), Spain, Junta de Andalucía, Spain (ref. SOMM17/6104/UGR), Generalitat Valenciana, Spain: Grisolía (ref. GRISOLIA/2018/119) and GenT, Spain (ref. CIDEGENT/2018/034) programs, La Caixa Foundation, Spain (ref. LCF/BQ/IN17/11620019), EU: MSC program (ref. 713673), Spain.

Appendix A

Acronym	Meaning
ADF	Acoustic Data Filter
AJAX	Asynchronous JavaScript and XML
CLB	Central Logic Board
CSS	Cascading Style Sheets
CU	Control Unit
CPU	Central Processing Unit
DAQ	Data Acquisition
DBI	Data Base Interface
DM	Detector Manager
DML	Data Management Language
DOM	Digital Optical Module
DQ	Data Queue
DU	Detection Unit
DW	Data Writer
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol — Secure
JIT	Just in time
JSON	JavaScript Object Notation
LAP	Local Authentication Provider
MCP	Master Control Program
ODF	Optical Data Filter
PMT	Photomultiplier Tube
SQL	Structured Query Language
SRP	Simple Retransmission Protocol
TCP	Transmission Control Protocol
TOA	Time of arrival
TM	TriDAS Manager
TriDAS	Trigger and Data Acquisition System
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VPN	Virtual Private Network
XML	Extensible Markup Language

Appendix B

The data acquisition of the KM3NeT neutrino telescopes is designed to be modular and scalable with the detector size. The acquired data are not filtered by any hardware trigger implemented in the underwater detector, but are all sent to shore, demanding the data reduction to an online selection performed by a pool of

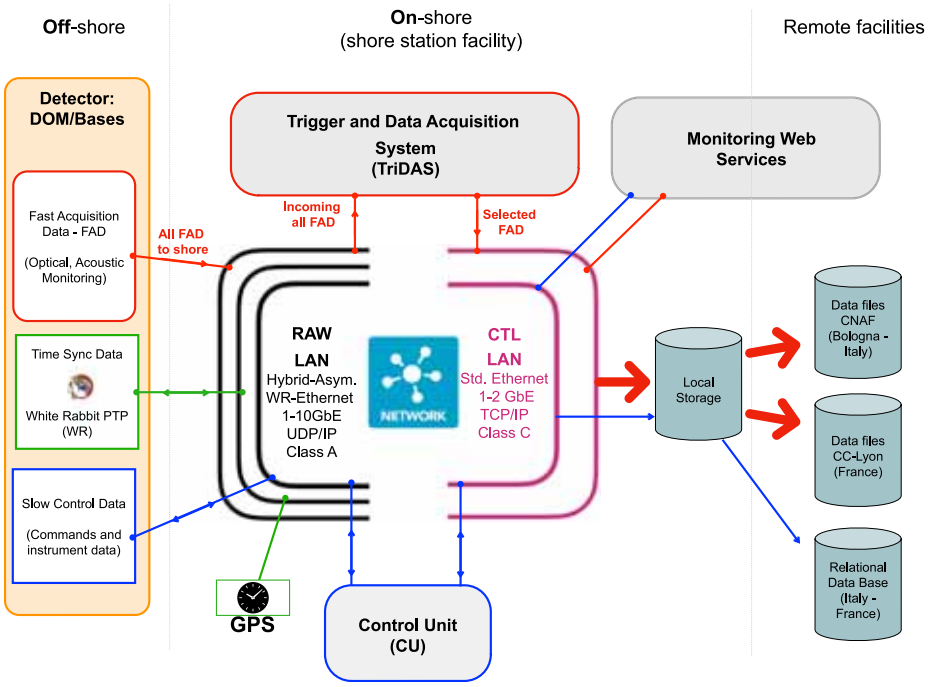

Fig. 14. Scheme of KM3NeT data acquisition. The network connections, data flows and computing facilities are sketched.

processes on a dedicated computing facility. This technique is also referred to as *streaming readout* approach. DOMs and computing resources on-shore are interconnected via a multi-LAN Ethernet-based network. The bandwidth requisite ranges from 1 Gbps up to 40 Gbps, according to the aggregation level of data. It starts from individual DOM interconnections to shore, up to the large throughput data-links between the shore station servers, when it is needed to route big data snapshots of information from the entire detector. The LAN that includes the DOMs and the front-end switches in the shore station is a synchronous Ethernet network. It uses a custom implementation of the White Rabbit protocol [12] to synchronize the off-shore detector with respect to a GPS reference on-shore. Each DOM handles different data streams. A sketch of the DAQ concepts is shown in Fig. 14.

Physics data streams, both optical and acoustic, are uni-directional streams from the DOMs to shore. They are called Fast Acquisition Data (FAD), and are made of continuous trains of UDP packets. The data collected by a DOM during any subsequent interval of time (called time-slice), 100 ms long, is called a frame. On shore, the first level of aggregation is done by the DQ processes. Each DQ manages the optical and acoustic data streams from an optimal number of DOMs (constrained by a trade-off between CPU and network bounds) reconstructing the various frames from the incoming datagrams. The next level of aggregation of optical data is done by the ODF processes. A single ODF receives, from all DQs, only the optical frames corresponding to a same time-slice, creating a super-frame, and then applies the trigger algorithms to it. Data corresponding to different time-slices are routed to different ODFs. The number of ODFs principally depends on the time needed for processing the super-frames, which has a non-linear connection to the size of the detector. Each ODF asynchronously processes the assigned snapshot of information, finds the candidates of particle tracks and sends the selected data to an unique DW process. The transmission between the ODFs and the DW is mediated by the Dispatcher process, based on the Control-Host protocol. The DW writes the received data into files to disk, on a local storage. Such recorded files are nightly transferred to the persistent storage repository in the CNAF and CC-Lyon data centers. As well as for the optical ones, the acoustic frames are routed from the DQs to a pool of ADFs. Each ADF handles separately the acoustic stream from each DOM. The number of ADFs linearly depends on the number of DOMs. Each ADF converts the beacon waveforms sampled by the served DOMs into time-of-arrival TOA information, which is stored on disk and off-line transferred to the central DB. The pool of DQs, ODFs, ADFs and DW constitutes the Trigger and Data Acquisition System, TriDAS, whose correct functioning is controlled via a monitoring system published through web services. While DOM optical and acoustic streaming as well as TriDAS processing are enabled on a run basis, for arbitrarily long duration data taking sessions, other DOM data-streams, like the slow control messaging and instruments monitoring are always available. The CU lays in between the detector and the on-shore resources, orchestrates the runs and provides useful data logs to the central DB.

References

[1] S. Adrián-Martínez, et al., (KM3NeT Collaboration), J. Phys. G 43 (2016) 1–131, http://dx.doi.org/10.1088/0954-3899/43/8/084001.
[2] S. Aiello, et al., (KM3NeT Collaboration), Astropart. Phys. 111 (2019) 100–110, http://dx.doi.org/10.1016/j.astropartphys.2019.04.002.
[3] S. Adrián-Martínez, et al., (KM3NeT Collaboration), J. High Energy Phys. 5 (2017) 1–39, http://dx.doi.org/10.1007/JHEP05(2017)008.
[4] R. Bruijn, (for the KM3NeT Collaboration), EPJ Web Conf. 207 https://doi.org/10.1051/epjconf/201920706002.
[5] M. Ageron, et al., (KM3NeT Collaboration), Eur. Phys. J. C 80 (2020) 99.
[6] S. Adrián-Martínez, et al., (KM3NeT Collaboration), Eur. Phys. J. C 74 (2014) 1–8, http://dx.doi.org/10.1140/epjc/s10052-014-3056-3.
[7] S. Adrián-Martínez, et al., (KM3NeT Collaboration), Eur. Phys. J. C 76 (2016) 54–65, http://dx.doi.org/10.1140/epjc/s10052-015-3868-9.
[8] R. Bruijn, (for the KM3NeT Collaboration), EPJ Web Conf. 207 https://doi.org/10.1051/epjconf/201920706007.

[9] S. Aiello, et al., (KM3NeT Collaboration), J. Inst. 13 (2018) P05035, http://dx.doi.org/10.1088/1748-0221/13/05/P05035.
[10] C.M. Mollo, et al., J. Inst. 11 (2016) T08002, http://dx.doi.org/10.1088/1748-0221/11/08/T08002.
[11] A. Albert, C. Bozza, EPJ Web Conf. 116 (2016) http://dx.doi.org/10.1051/epjconf/201611607004.
[12] T. Chiarusi, E. Giorgio, EPJ Web Conf. 207 (2019) http://dx.doi.org/10.1051/epjconf/201920706009.