Resumo

O sistema de controlo e aquisição de dados é uma parte importante de qualquer experiência moderna de física. À medida de as fronteiras do conhecimento avançam assim também crescem as especificações para estes sistemas e isto tem de ser tomado em conta ao desenhá-los. Além disso, visto que estas experiências podem durar vários anos ou mesmo décadas, é necessário considerar a futuras expansões e atualizações. Em investigação de energia de fusão é comum ver os subsistemas em torno do reator como fazendo parte de um todo e assim ter as tarefas de configuração, controlo, aquisição de dados, acesso a dados e segurança da máquina como parte dum sistema unificado. Tal sistema é chamado de CODAC, vindo do inglês *Control, Data Access and Communications* (controlo, acesso a dados e comunicações). Tipicamente estes seguem uma estrutura hierárquica em árvore e usam tecnologia proprietária. Esta tese é baseada no trabalho realizado no CODAC do Tokamak COMPASS, uma máquina de tamanho médio para investigação em fusão, que esteve instalada em Culham, Reino Unido e foi transferida para Praga, na República Checa. Este CODAC é baseado no sistema *FireSignal*, que está a ser utilizado no tokamak ISTTOK (Lisboa, Portugal) e foi usado pelo CASTOR, que antecedeu o COMPASS em Praga. A operação do COMPASS é baseada em placas de controlo e aquisição de dados, desenvolvidas pelo IPFN. Ao desenvolver *software* e *drivers* para integrar no CODAC diagnósticos e atuadores foi necessário não só ter em conta a física por detrás destes bem como também as capacidades do *hardware*. Durante este trabalho, um sistema de controlo em tempo real baseado na plataforma MARTe foi integrado com sucesso, tendo partes da operação sido automatizadas e permitindo que os dados fossem transferidos para a base de dados central. Diagnósticos, tais como a reflectometria, foram igualmente integrados com sucesso, permitindo aos operadores configurá-los usando uma interface comum, e automatizando a fase de preparação o mais possível. Foi também necessário desenvolver ferramentas de acesso à base de dados que tivessem em conta as características do COMPASS. Esta tese apresenta uma análise geral de desenho de CODAC para experiências de física. Desafios comuns como: crescimento de volume de dados, integração de eletrónica mais rápida e novas tecnologias são estudadas com o objetivo de propor um estrutura mais flexível e com expansão em mente.

Abstract

The control and data acquisition system is an important part of every modern physics experiment. As the boundaries of knowledge move forward, so do the requirements for these systems and this must be taken into account when designing them. Also, since experiments can span the course of several years or even decades, expansion and upgrade must also be considered. In fusion energy research it is common to view the sub-systems around the reactor as part of a whole and having the different tasks of configuration, control, data acquisition, data access and machine security as part of a unified system. Such a system is called a CODAC, standing for Control, Data Access and Communications. Typically they follow a hierarchical, tree-like structure and use proprietary technology. This thesis is based on the work developed on the CODAC of COMPASS Tokamak, a medium-sized fusion research machine that was installed in Culham, UK, and was transferred to Prague, in the Czech Republic. The CODAC is based on the FireSignal system that is being used at the ISTTOK tokamak (Lisbon, Portugal) and was used in CASTOR, the predecessor of COMPASS in Prague. The operation of COMPASS is based on the ATCA control and acquisition boards developed on IPFN. When developing software and drivers for integrating diagnostics and actuators on the CODAC, it was necessary not only to take into account the physics behind them, but also the hardware capabilities. During this work, a real-time control system based on MARTe was successfully integrated, with parts of the operation being automated and data being transferred to the main database. Diagnostics such as reflectometry were also successfully integrated on the system, allowing operators to configure them on a common interface and automating the setup phase as much as possible. It was also necessary to develop database access tools that took into account the characteristics of COMPASS. This thesis makes a broad analysis of CODAC design for physics experiments in general. Common challenges such as: increasing data volume, integration of faster electronics and new technologies; are taken into consideration to propose a structure that is more flexible and with expansion in mind.

Sommario

I sistemi di controllo e acquisizione dei dati sono una parte importante di ogni esperimento di fisica moderna. Mentri i confini del sapere avanzano anche i requisiti per questi sistemi crescono e questo deve essere tenuto in considerazione durante la loro progettazione. Inoltre, visto che questi esperimenti possono prolungarsi per molti anni è necessario considerare future espansioni e attualizzazioni. Negli esperimenti di fusione nucleare frequentemente si considerano i sottosistemi del reattore come un sistema fortemente integrato e perciò la configurazione, controllo, protezione e acquisizione dei dati viene spesso vista come un sistema unificato. Questo sistema si chiama CODAC (Control, Data Access and Communications) e normalmente segue una struttura ad albero gerarchica basata su tecnologie proprietarie. Questa tesi si basa sul lavoro svolto sul CODAC del Tokamak COMPASS, una macchina di medie dimensioni che è stato costruita, installata e operata prima in Culham, Regno Unito, e dopo alcuni anni trasferita a Praga, nella Repubblica Ceca. Il CODAC si basa sul sistema FireSignal che viene utilizzato al tokamak ISTTOK (Lisbona, Portogallo) ed è stato utilizzato in CASTOR, il predecessore di COMPASS a Praga. Il funzionamento di COMPASS si basa su delle schede di controllo e acquisizione dati sviluppate dal IPFN. Per lo sviluppo di software e per l'integrazione delle diagnostiche e attuatori di COMPASS sul CODAC, è stato necessario prendere in considerazione non solo la fisica, ma anche le capacità del hardware. Durante questo lavoro, un sistema di controllo in tempo reale basato su MARTe è stato integrato con successo, permettendo che parte dell'operazione fosse automatizzata e i dati trasferiti al database principale. Diagnostiche come la reflettometria sono stati integrati con successo sul sistema, consentendo agli operatori di configurarli su un'interfaccia comune, automatizzando la fase di configurazione per quanto possibile. Inoltre, è stato necessario sviluppare strumenti di accesso al database dei dati che prendessero in considerazione le caratteristiche di COMPASS. Questa tesi fa un'ampia analisi del design di un CODAC per esperimenti di fisica. Problemi comuni quali: l'aumento del volume di dati, l'integrazione dell'elettronica più veloce e di nuove tecnologie, sono presi in considerazione per proporre una struttura che è più flessibile e espandibile per integrare i requisiti che escono durante l'operazione dell'esperimento.

Palavras-chave/Keywords

CODAC	CODAC
Tempo real	Real-time
Armazenamento de dados	Data storage
Pesquisa de sinais	Signal query
Escalabilidade	Scalability

Acknowledgements

I would like to thank in first place to the Instituto de Plasmas e Fusão Nuclear and its president, Doctor Bruno Gonçalves for allowing me to work on this facility and for everything that was put at my disposal.

I would like to thank my supervisor, Prof. Bernardo Carvalho, for his support, advice and patience during all these years.

I would like to thank the Engineering team at IPFN, specially those whom I worked more closely, for their support, advices and also friendship. A heartfelt thank you to Prof. Horácio Fernandes, Doctor Jorge Sousa who were almost my co-supervisors. I also thank my colleagues André Neto, Ivo Carvalho, Daniel Valcárcel, Bruno Santos and João Fortunato, who helped me during this work and with whom I shared several work trips abroad.

I would like to thank the COMPASS team, namely Doctor Jan Stockel, Doctor Radomír Panék and Doctor Martin Hron, for inviting me to work with them at Prague and for all the support, not just professional, they gave me while I was working there. I would also like to extend a special thanks to those whom I worked more closely: Filip Janky, Jan Písačka, Jakub Urban and Richard Papřok. I wish them all successed on their projects.

I would also like to extend these acknowledgements, to my family and friends, specially my parents who suported me and gave me courage even in the worst times. Finally, I would like to remember my godfather who was one of my best friends during my entire life and always helped me anyway he could, but tragically passed away. May he rest in peace.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data acquisition is a key aspect of every experiment, since without it nothing can be learned. During the history of science, instrumentation and knowledge formed a feedback loop, where better instruments have provided humans with better understanding of natural fenomena which allows the develpment of more advanced instruments. This relation is cleary visible on the development of electronics during the past decades. Long running physics experiments are naturally sensitive to these developments and the design of their data acquisition systems must take this into account. As these systems become ever more complex, the concept of CODAC emerges[1]. CODAC stands for Control, Data Access and Communication and can be considered an extension of the CODAS (control and data acquisition system) as it treats all systems that are part of an experiment (namely control of the experiment, data collection, data access and plant security) as a whole. Although the specific needs of an experiment must be taken into account when designing a CODAC, there are several key features that should be taken into acount. This thesis uses the implementation of a CODAC for a specific experiment as the starting point for a study on how to design a CODAC for modern, large physicals experiments, taking into account not just their specific needs but also on a broader view.

1.1 Fusion Research and Engineering

Fusion is a nuclear reaction where light elements are merged (fused) into a another, heavier, element [2]. Although any element can be fused this way, it typicaly involves hydrogen and its isotopes, deuterium and tritium, being this the main reaction that powers our Sun. Since this reaction produces a great amount of energy, the main goal for fusion research is to apply it to produce electricity, as an efficient and clean alternative to existing power sources, namely fossil fules. Of all the fusion reactions available the deuterium-tritium technology is the most promissing: not only is the most energetic, it is also the one that occurs at faster rate. Besides, deuterium is abundant on ocean water, from where it can be easily extracted and tritium, although not occuring naturaly on Earth, can be obtained from Lithium which is a very common metal. However this reaction also generates fast neutrons that activate the structures of a fusion reactor, this being its biggest environmental issue. These

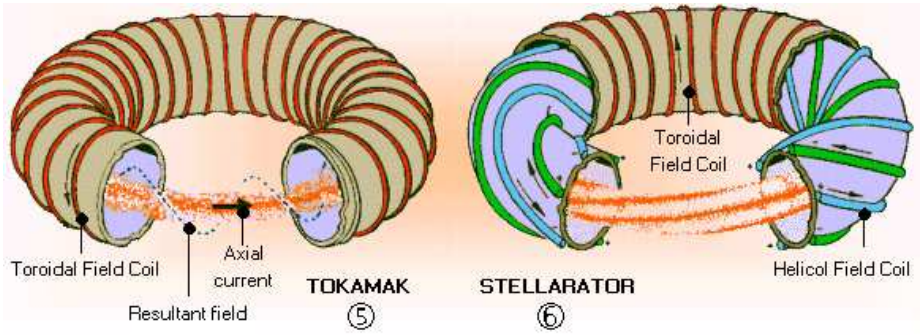

Figure 1.1: Drawing of typical Tokamak and Stellarator. Here we can see the different arrangements of their coils [5].

same neutrons, on the other hand, can be used to breed tritium from lithium, meaning that there should be no need to handle tritium (which is radioactive itself) outside of the fusion chamber [4].

In order for fusion reactions to occur certain densities and temperatures are required and for these to be fullfilled the elements must be on a confined plasma state. There are three ways to confine a plasma: gravitational, magnetic and inertial. Gravitational confinement is the one responsible for energy production in stars, so it is not viable on Earth. Inertial confinement uses powerful lasers to compress fusion fuel pellets during short periods of time. Magnetic confinement uses the fact that ions and electrons follow magnetic lines to keep the plasma inside a partial-vacuum chamber. The most common magnetic configurations are the Tokamak and the Stellarator. Both have toroidal shapes, their main difference being how a poloidal magnetic field, necessary to stabilize the plasma, is generated: in a tokamak a plasma current is induced by varying the magnetic flux on the central core (like in a transformer); in a stellarator by specially shaped external coils. Until now, the tokamak is the most successful magnetic geometry and is widely considered the best candidate for a commercial fusion power plant [3].

1.2 COMPASS

Tokamak COMPASS (COMPact ASSembly) [6] is installed in the Institute of Plasma Physics (IPP) of the Academy of Sciences of the Czech Republic since 2006. It was designed in the 1980s in the British Culham Science Centre to study plasma physics in circular and D shaped plasmas, with the purpose to prepare for ITER scenarios. In fact, the relations between its dimensions are very similar to those of ITER. At the turn of the 21st century, the Culham Science Centre started working with a new spherical tokamak, the Mega Ampere Spherical Tokamak (MAST), and so operation of COMPASS was discontinued. However, due to its relevance for the ITER project, it was offered to IPP - Prague which had experience with small tokamaks [7].

As before, the main objectives of COMPASS are related to ITER, specially:

- study of H-mode physics (there are only two other operational tokamaks in Europe capable of this regime);

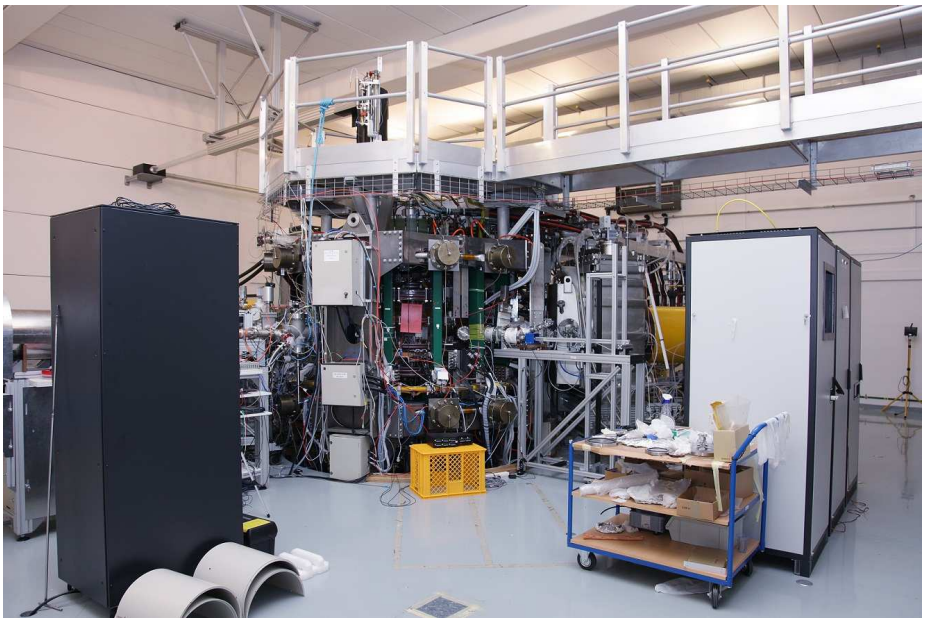

Figure 1.2: View of COMPASS and its surroundings.

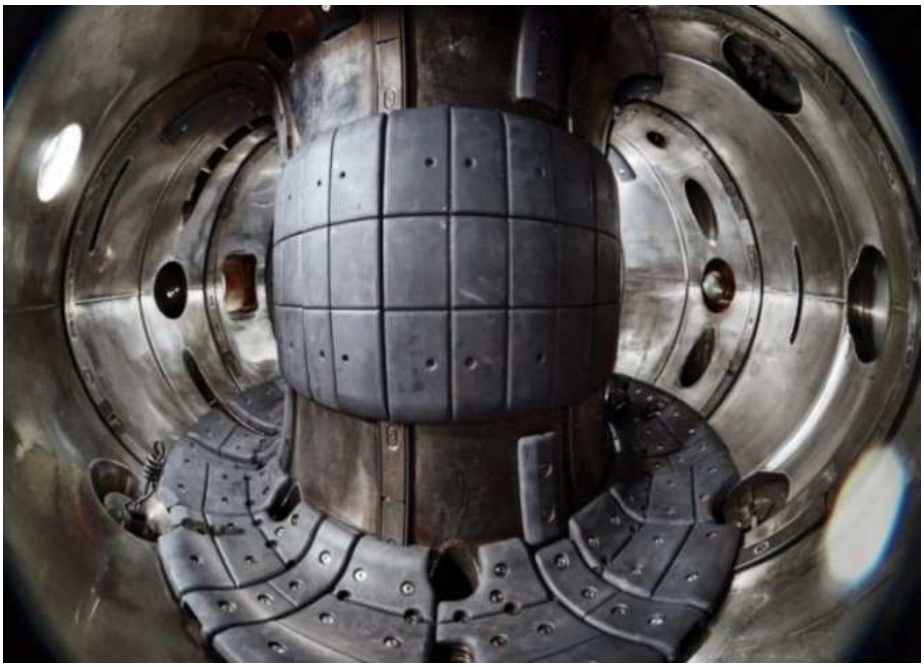

Figure 1.3: Inside the chamber of COMPASS.

Parameters	Values
Major radius	0.56 m
Minor radius	0.23 m
Plasma Current (max)	400 kA
Magnetic Field	0.9 - 2.1 T
Vaccum pressure	1×10^{-6}Pa
Elongation	1.8
Plasma Shape	D, SND, elliptical, circular
Pulse length	~ 1 s
Beam heating P_{NBI}40 KeV	2×0.4 MW

Table 1.1: The main parameters of COMPASS.

- MHD equilibrium and instabilities;
- Plasma-wall interaction;
- Physics of runaways and disruption;
- Developments of advance diagnostic methods;

1.3 ISTTOK

Tokamak ISTTOK (Instituto Superior Técnico TOKamak) [8] is installed in Lisbon, in the institute that gives it the name and is explored by IPFN. It is a small tokamak with a circular cross-section, a poloidal graphite limiter and an iron core transformer, that was built from the former TORTUR tokamak, which was de-commissioned by the Association EURATOM/FOM in 1988. The ISTTOK construction started, officially, on January 1st, 1990, date on which the contract of association EURATOM/IST enter into force. The operation of this experiment in an inductive regime started on February 1991. Its main objectives are:

- Teach and train personnel (mainly university students) in physics, engineering and technologies associated with nuclear fusion;
- Development of new diagnostic techniques and to develop and test digital instrumentation dedicated to control and data acquisition.

Since 2012, thanks to fast electronics developed at IPFN and real-time control, ISTTOK has been operating on AC discharges [29, 10], where the direction of the plasma current changes quickly enough to allow the pulse to be continued, keeping a somewhat constant plasma density without saturating the iron core. This has allowed discharge duration to go from 25-40 ms up to almost 1 s (although the typical discharge is around 200 ms).

1.4 Motivation

With the installation of the COMPASS tokamak in Prague, it was necessary to install a CODAC system for the experiment. Over the years, new diagnostics

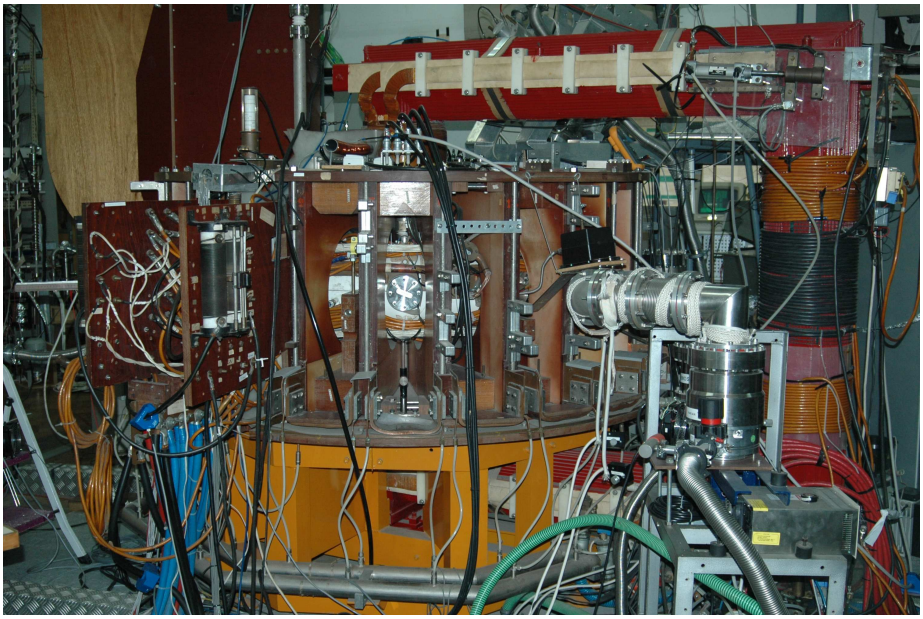

Figure 1.4: View of ISTTOK and its surroundings

Parameters	Value
Minor radius	0.085 m
Major radius	0.46 m
Plasma current	~7 kA
Pulse length	30 (DC) to 1000 ms (AC)
Maximum toroidal magnetic field	2.8 T
Nominal toroidal magnetic field	0.3-0.6 T

Table 1.2: Main parameters of the ISTTOK tokamak.

and actuators were expected to be installed and thus required to be integrated on the CODAC. The operation would be based on the novel ATCA [11] control and acquisition boards developed on IPFN (Instituto de Plasmas e Fusão Nuclear). These possess advanced technical features, namely state-of-art FPGA for signal processing, multi-Gigabit/s point-to point serial links for super-fast data sharing across all the input channels and a sub nano-second synchronous timing/event distribution network. This system was similar to the one used on the ISTTOK tokamak in Lisbon. The knowledge obtained during the design, installation and maintenace can be extrapolated to other experiments, future and current, thus becoming the basis for the work presented on this thesis.

Chapter 2

Control, data acquisition and retrieval in international experiments

International physics experiments are complex and require large investments in material, people and time. They require years of design and preparation of all its sytems especially the CODAC, as the experimental output handling is key for their success. We can thus use their experience and the challenges they ecountered to guide us in the design for new experiments. This chapter analyses the CODACs of four major physics experiments: JET (tokamak), W7-X (stellarator), NIF (inertial confinement) and ALICE (particle physics). JET has been operating since the 1980s so it is a good example for the challenges a long running experiments face; W7-X is a new experiment that allows us to look to a more modern design; NIF allows us to look at a different an experiment with different operation, requirements and challenges from magnetic confined plasmas, while still being related to fusion; ALICE, besides not being related to nuclear fusion, is a good example of an experiment that is just one of several experiments integrated in CERN, the largest physics laboratory in the world [12].

2.1 JET

The Joint European Torus (JET) [13] is currently the largest tokamak in operation in the world. It was designed in the mid 1970s and its operation began in 1983. It is installed near Culham, Oxfordshire in the UK and is managed through the European Consortium for the Development of Fusion Energy (EUROfusion). Not only was here that the first controlled release of fusion energy happened, but it also detains the world record in fusion power (16 MW). In 2006 operation was started with an ITER-like magnetic configuration and in 2011 an ITER-like wall was installed, thus becoming the key on the preparation for ITER operation. For the scope of this thesis, JET represents a good example of a large experiment that has been running for several decades, during a time of significant technological advancements.

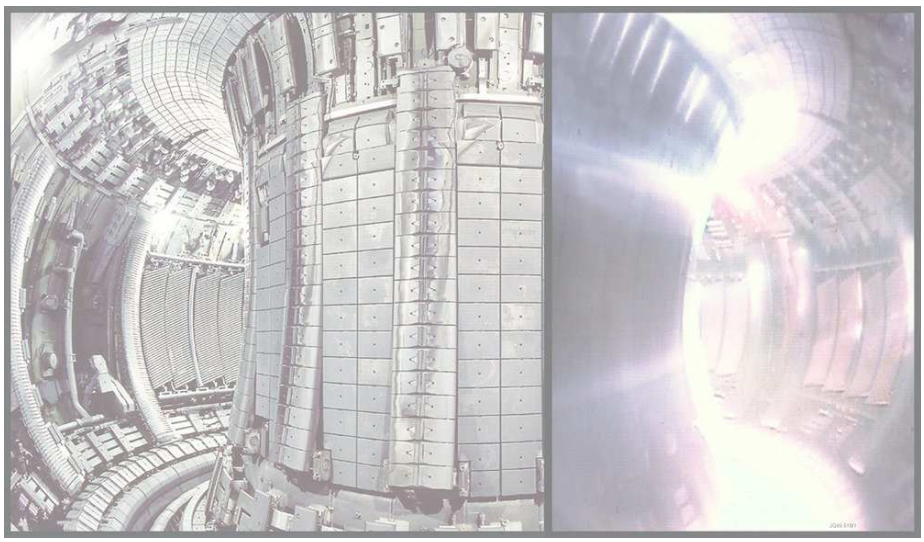

Figure 2.1: Inside the chamber of JET, without plasma on the left and with plasma on the right

At the begining [14], its CODAS was designed around a modular, tree-like structure, that consisted of three levels:

- Level 1: Jet-wide supervisory control
- Level 2: Subsystem supervisory control
- Level 3: Local unit or component

Level 2 and most of level 3 software ran on level 2 computers, with some real-time control on microprocessors in level 3. This structure was also used to identify data in the database. The technology used was cutting edge for the time, with NORD computers and CAMAC front-end electronics. Most software was written in Fortran, with time-critical software written in Assembly and NPL (NORD Programming Language). Around 1990, this technology was showing its age: more powerful hardware and modern software were available, but they were not supported on the NORD architecture. It was thus decided to switch to a UNIX based system (Solaris OS), to use ethernet networking and to replace the CAMAC system with VME and PC front-ends. This was no easy task: there were many CAMAC systems being used and they could not be replaced overnight, so they had to be interfaced with the new computers; tens of thousands of lines of code written in Assembly and NPL had to be ported to C; around half a million lines of Fortran code also had to be translated because of NORD extensions that were not supported by UNIX; some NORD features also had to be temporarly emulated; and the machine had to remain operational during the system upgrade. Only in 1994 and after a major shutdown phase, was the migration to the new CODAS system considered complete.

This, as expected, would not end here and by 2008 the system had received another overhaul, being by then mostly based on PCI eXtension for Instrumentation (PXI) and LabVIEW, with computers now using GNU/Linux operating

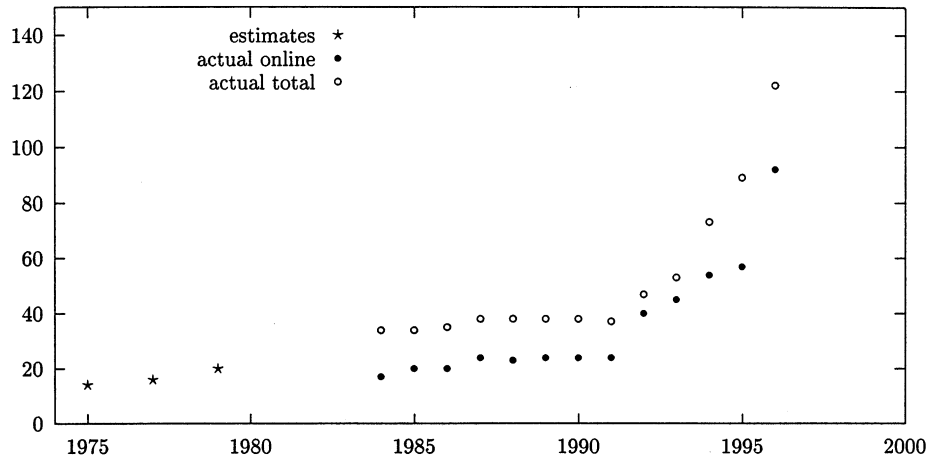

Figure 2.2: The number of control computers within JET CODAS until 1999. [14]

systems, specifically Linux RTAI for real-time applications [15]. More modern hardware architectures, like Advanced Telecommunications Computer Architecture (ATCA) here started to being used for new subsystems [27]. In 2010, a real-time control framework, called MARTe (Multi-threaded Application Real-Time executor), that can take advantage of multicore processors, was developed at JET and now is being used for some of its real-time control, including some critical subsystems like the vertical stabilisation control.

It is also interesting to look at how the system grew during its operational history. In the early design it was estimated that the CODAS would require about 15 computers, together with a large mainframe for data storage and analysis. It was also expected that this would be enough for the entire life of JET. But as soon as JET became operational this number had already grown to 34 NORD minicomputers (20 online), in 1999 there were 122 SPARC computers being used (92 online) and the number continued to grow to the point were it is not easy to make an estimation anymore. As for the data generated, it was first expected that each shot would produce almost 1 MB of data, value that was surpassed as soon as JET started regular operation. Over the years the amout of data collected has been steadily increasing, roughly doubling every two years, reaching 10 GB by 2008. In order to cope with this growth the first measure taken (besides improvements made on the network) was to classify data has either 'urgent' or 'non-urgent', with the first being collected in the first few minutes after a pulse and the other later in the day, during quiet periods. More solutions (now with ITER also in mind) like data compression and selective acquisition rates are also being studied and may be implemented in the near future [15].

2.2 Wendelstein 7-X

The Wendelstein 7-X (W7-X) reactor is an experimental stellarator built in Greifswald, Germany, by the Max Planck Institute of Plasma Physics (IPP),

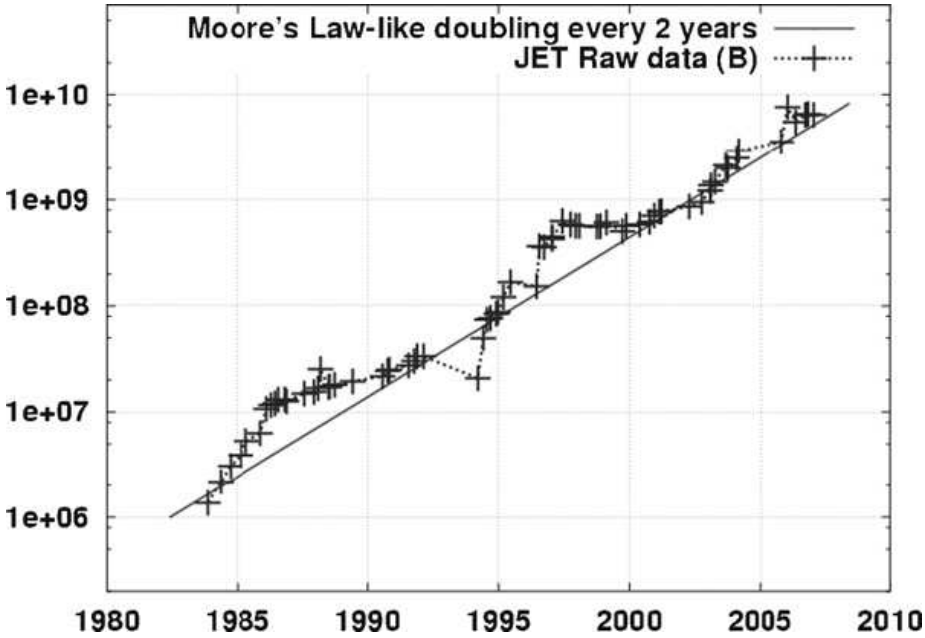

Figure 2.3: The data volume growth on JET. It closely follows a Moore-like law. [15]

and completed in October 2015 [16]. It is currently the largest stellarator in the world and it was designed to operate with up to 30 minutes of continuous plasma. Its first plasma was created in December 2015 and regular operation has yet to start.

The CODAC of W7-X is based around a modular system called Control and Data Acquisition Station (CoDaStation) [17]. It was designed with the specific needs of W7-X in mind and prototypes were tested on other installations (such as the WEGA stellerator) before deployment [18]. The core of the CoDaStation software is a generic framework that solely provides the necessary infrastructure to acquire and process signals. In order to achieve high modularity, there are many components on the system, each with well defined functions nad interations. On the data acquisition side there is:

- Buffers: signal packages containing data with the same *time group* (similar time-stamp)
- Signal providers and consumers: generate a signal and recieve one or mores signals, respectively
- Routers: transfer data from *signal providers* and *signal consumers*
- Routing jobs: define if routers act synchronously or asynchronously, according to urgency or special requirements

On the control side there is:

- Properties: the state of a CoDaStation component

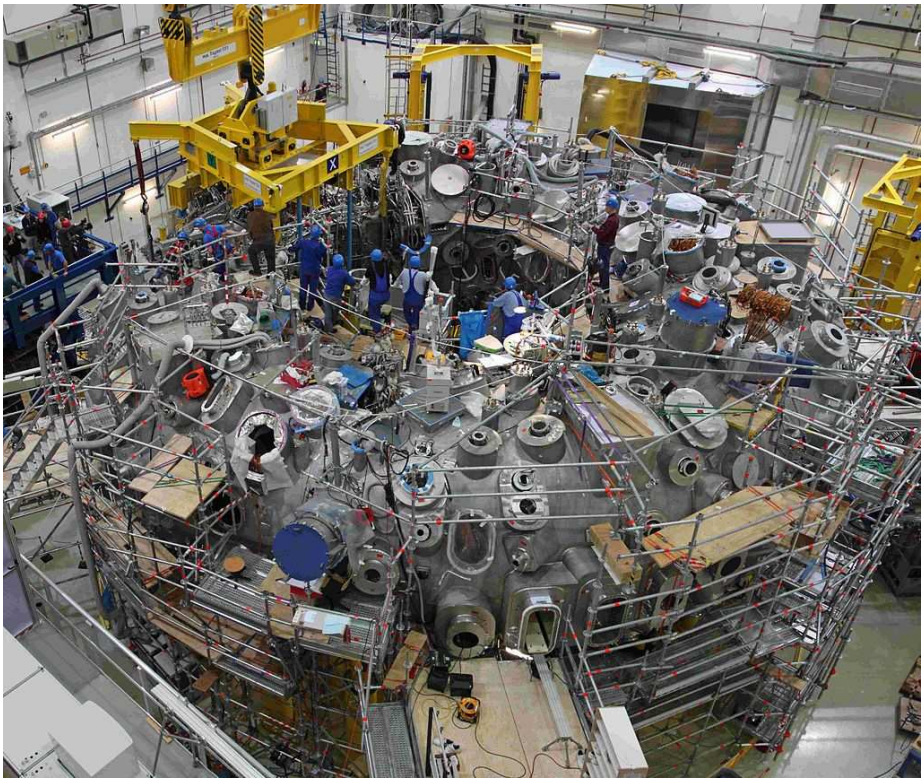

Figure 2.4: The torus hall of Wendelstein 7-X, during cosntruction.

- Controllables: retrieve *Properties* from the CoDaStation and translate them into component-specific actions
- Station states: define the overall state of the CoDaStation
- Controller: interface for external control systems to issue control commands

These come together on packages called AddOns, which are used to integrate subsystems thus providing CoDaStation with new functionalities. These can have *controllables*, *signal providers* and *signal consumers*, interacting with the main system through a fixed software interface. Thanks to this, new features can be easily tested outside W7-X and also CoDaStation components can be replaced by mockups. There are also *Supervisors*: software that operators use to monitor *Station* and *AddOn* states and act if necessary. This can also be done remotely, thanks to JXI based supervisors.

2.3 NIF

The National Ignition Facility (NIF) is a large laser-based inertial confinement fusion research device, located at the Lawrence Livermore National Laboratory in Livermore, California, in the United States of America [19]. Construction began in 1997, being ready for operation in 2009 and by 2010 the National Ignition Campaign started with the goal to produce more fusion energy than deposited on the target. However the campaign ended in 2012 without reaching its main objective and currently NIF is mostly devoted to materials studies.

At the heart of the control and data acquisition system of NIF is the NIF Data Repository (NDR) [20]. The NDR integrates many autonomous database systems thus being called a federated database. It allows the collection of all information relevant to the experiment, specifically campaign plans, machine configuration, calibration data, raw experimental results and processed results. There are three main events that trigger data collection and automated data analysis: laser alignment, shot time capture, and post shot optics damage inspection. Recent data is stored on-line, least recently used data is migrated to near-line storage and older data storage is subjected to different policies depending on the type of data. In 2009 NIF was generating approximately 66 TB of data per year, most of them 2-D images.

NDR uses industrial and scientific standards for its operation, such as:

- Web Services Business Process Execution Language (WS-BPEL) for preparing shots and organizing data collection, transfer and analysis;
- HDF5 files for storing calibration, background and diagnostic data;
- Uniform Resource Names (URN) to identify data and accessing it through Web services.

One of the goals of NDR is foster collaboration between universities and research institutes. To achieve this, all data and other content can be shared on the Web, with researchers being able to decide what they want to share with everyone, a restrited group of colleagues or to remain private.

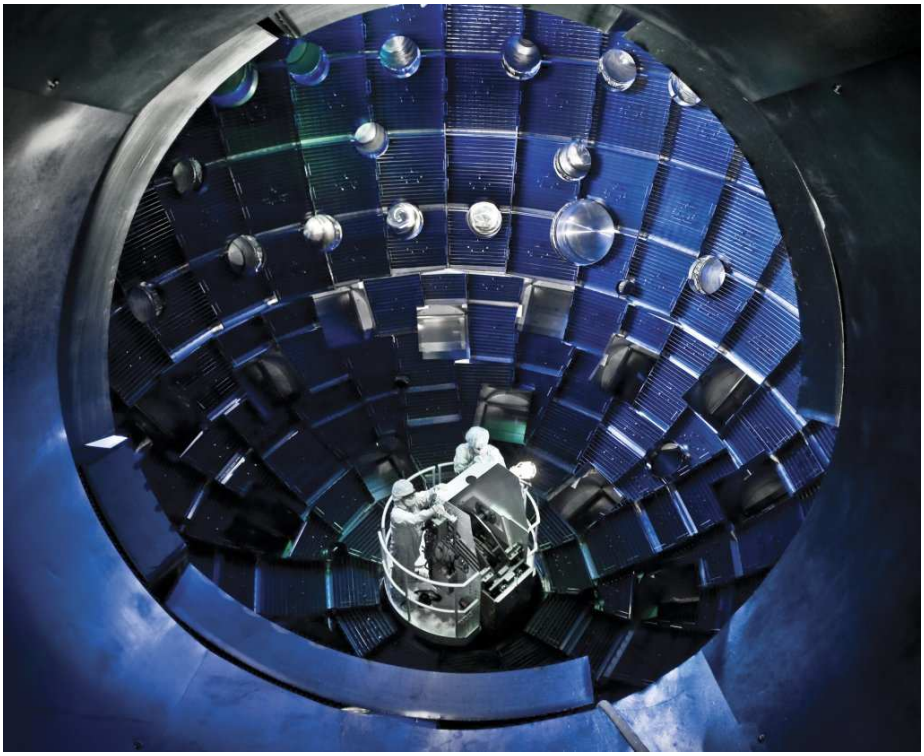

Figure 2.5: Inside the reaction chamber of NIF, while preparing for a shot.

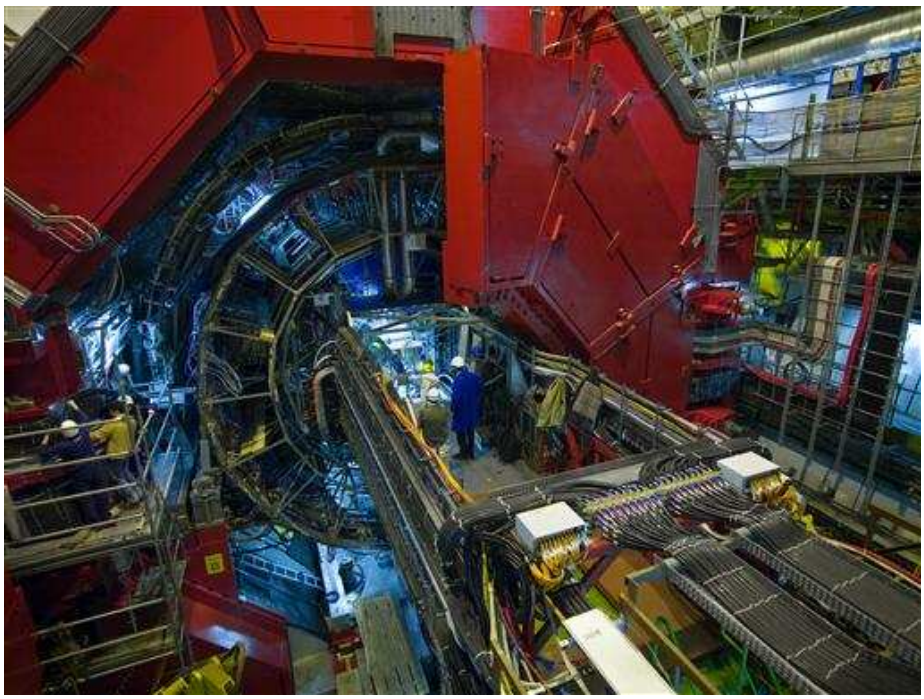

Figure 2.6: The ALICE detector in 2008. [22]

2.4 ALICE

One of several CERN LHC (Large Hadron Collider) experiments, ALICE (A Large Ion Collider Experiment) is a general-purpose, heavy-ion detector which focuses on Quantum chromodynamics (QCD), the strong-interaction sector of the Standard Model [21]. It is optimized to study heavy ion collisions (Pb-Pb), with temperature and energy densities high enough to produce quark–gluon plasma, which is the state of matter that the standard Big Bang model predicts to have existed in the early universe, from a few picoseconds to about 10 microseconds after the Big Bang event. This experiment has been running since 2008 on three main modes: heavy ion, proton-proton and proton-heavy ion collisions. These modes have different particle interaction rates, running time and data throughput.

The Data Acquisition system for ALICE (DAQ) was designed and implemented in the years between 1996 and 2005. The two main requirements for the DAQ were the maximum event rate and data-flow. The event rate was not considered at the time of design the most challenging part of the system, as the estimated rate of the order of 1 kHz was considered rather modest. On the other hand, the data-flow was dependent of the different running modes and scenarios, which made its estimate much more challenging, besides the fact that its limits were more dependent on the cost of media needed to store the data and of the computing resources needed to analyze them. At the beggining an estimate of 1.25 GBytes/s was accepted as a good balance regarding these constraints, but later the CERN management agreed on providing the maximum

bandiwth allowed by the Central Data Recording facility.

There are five online systems on the ALICE experiment:

- Trigger: combines information from all triggering detectors and decides whether the data are worth collecting and, if the answer is positive, sends a sequence of triggers for all detectors to acquire synchronously;
- Data Acquisition: realizes the data-flow from the detectors to the data storage and monitors data quality and system performance;
- High-Level Trigger: reduces the volume of data by selection and compression;
- Detector Control System: controls all detector services;
- Experiment Control System: controls the activity of all online systems.

The Data Acquisition (DAQ) and the Experiment Control System (ECS), being responsible for the overall experiment data-flow and control, the detector software, the infrastructure and the data quality monitoring, are the core of the CODAC. These use different software packages to perform their duties and allow operators to monitor the system:

- Lemon, for monitor the status of hardware and software components;
- the electronic logbook, for book-keeping of all operational activities;
- Detector package (DA), which implements a controlled and reliable environment for the execution of detector related tasks;
- AMORE (Automatic MOnitoRing Environment), used by the detectors for monitoring data quality.

These packages share a common software framework to ensure their coherence, called DATE (Data Acquisition and Test Environment).

Chapter 3

CODAC Tools on ISTTOK and Compass

The CODAC in ISTTOK and Compass share several common tools and are thus somewhat similar. These tools were designed to be flexible, modular, independent from operating system and computer architecture. This is achieved by using non-proprietary open-source software, industry standards and cross-platform programming languages. These systems are open-source and their code is stored in a subversion repository. For data collection and system pre-shot configuration a tool developed at IPFN called FireSignal is used. The core of FireSignal is written in Java, but by using CORBA [24], it supports modules written in different languages. Real-time control is done using the MARTe framework, which was originally developed at JET, with drivers and modules developed at IPFN and IPP-Prague for the specific needs of ISTTOK and Compass respectively. Regarding data access, ISTTOK and Compass use different tools due to different databases structures: SDAS, using a Java core and Remote Procedure Calling, and CDB, using a Python core and Python libraries for access.

3.1 FireSignal

FireSignal [23] is the part of the CODAC that allows the operator to configure and monitor the experiments, also retrieving data from diagnostics and storing it in the database.

As a fully modular system FireSignal avoids dependencies of particular technologies. It is composed of five different modules, which provide all the functionality to the system:

- Central Server—works as bridge between all the other components. It is responsible for managing all the connections, commands and data broadcast.

- Database connector—controls the installed database, storing and fetching data when ordered by the Central Server, providing an abstract layer, independent of the database solution.

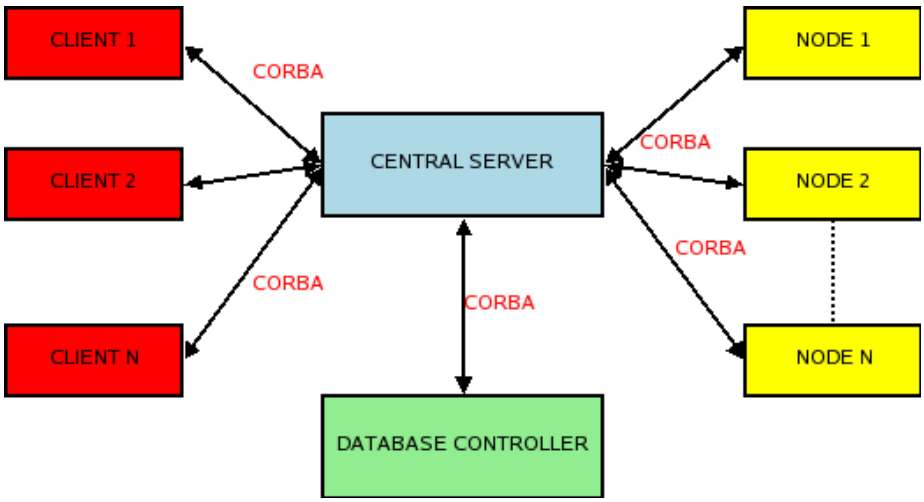

Figure 3.1: Schematic of how the different parts of FireSignal are connected. [23]

- Security Manager—the Central Server queries this component to authenticate users and nodes and to authorize all the operations. The security schema used (LDAP, PAM, etc.) is abstracted by the security manager.
- Hardware Clients—these are responsible for driving the devices and for data readout. Template nodes, with the communication to the Central Server already implemented, are provided in different languages and can be extended to integrate any hardware device.
- User Clients—these are the graphical user-interfaces, which allow the interaction with the system. It is the only optional component as the system can still operate without direct human intervention, e.g. when FireSignal works as a system which receives orders from other data acquisition system.

All the components are connected through CORBA allowing them to run in different operating systems and to be written in different computer languages. Currently FireSignal is using Java and C++ as main languages.

3.1.1 Database connector

All data arrives tagged with two absolute time-stamps: acquisition start time, end time. These can later be used to generate time arrays and correlate data. The data also comes tagged with an event. A large number of what happens during the discharge can be considered an event. Examples of events include the discharge start trigger, external triggers and disruptions. Events are defined by a unique name and number and by an absolute time-stamp. In the case of ISTTOK the event shot is defined by the string '0x0000', where the unique number is the shot number and the time-stamp the time in which there is the formation of plasma. The accuracy of these times stamps isn't guaranted by FireSignal so an external timing system is necessary.

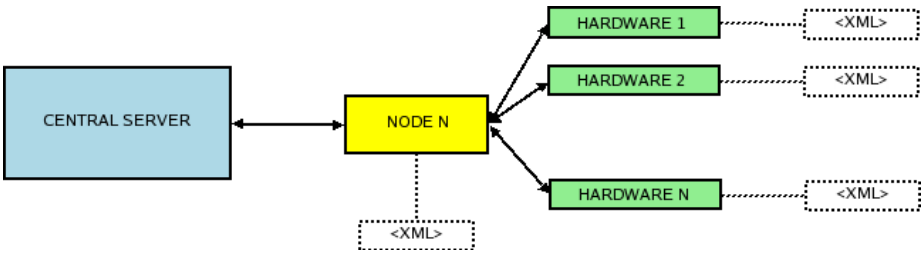

Figure 3.2: Detail on how Nodes connect to other modules. It also shows the different XML files that define their characteristics.

The installer for FireSignal originaly came with three database connector implementations: MySQL, PostgreSQL and PostgreSQL with external files for the binary data. A fourth, supporting Compass Database (CDB), was developed later. This one was also used as the basis for a version using HDF5 (as opposed to plain binary files) files to store data, which isn't currently being used.

The database at ISTTOK is the version using PostgreSQL with raw binary files. The data is stored in a directory structure comprising: a Node directory with directories inside for each Hardware; inside these, there are directories named with a hash code for each acquisition day; finally, inside these there are files with data and configuration XMLs named using the channels unique identifiers and the timestamp for when the acquisition started. For longer acquisition cycles the same channel can be saved in different blocks, each identified by its unique start time. The PostgreSQL database stores the absolute starting time for each shot (the only event in ISTTOK up to now), the absolute starting and ending time for each channel and to which shot the are related, the XMLs describing each Hardware, and information regarding the users and the institutions they are from. Using the unique identifiers of the channels and the absolute start time the system automatically reconstructs the full path to the file.

3.1.2 Hardware Clients

Hardware clients, also called Nodes, provide the interface between the installed devices and the Central Server. Their main responsibilities are the configuration of hardware, tagging data and events with the correct time-stamps and sending them to the server. Nodes are organized as follows:

- Nodes contain Hardware devices.
- Hardware devices contain parameters.
- Parameters contain configuration fields.

This organization is not very different from a typical acquisition system, with crates (Nodes) containing boards (Hardware) with several channels (parameters). By default, each parameter in the same hardware has a different value for the same configuration fields but it's possible to have them share the same value.

Nodes are described in two types of XML files, one for the Node iself and another for each Hardware. The Node file is very simple, only containing the

name and description for the node itself. The Hardware file contains the name and description for the hardware itself and also the list of parameters and their fields. Parameters and fields can be defined to be editable or not, it is possible to force fields be not be empty, have their value shared across the same Hardware and it also defines their type. Parameters have a mime-type that is used to determine how to store and display data (there is a plugin system that allows user defined mime-types). Likewise, there are several standard defined field types and it is possible to have user-defined types, thanks to the plugin system.

3.1.3 CORBA

As with all the FireSignal components, the communication between the Nodes and the Central Server is perform through CORBA. Each Node implements the FireSignal server–node communication standard. The number of functions is quite significant and ranges from hardware configuration details to event broadcasting and data acquisition. FireSignal is bundled with server–node communication standard implementations in two different languages: Java and Python (C++ is now being developed). This allows developers to focus only in the interface between Nodes and hardware without having to learn or concern about the communication protocol with the Central Server.

3.1.4 User Client

The user client is a standalone application, developed in Java and deployed using Java Web Start (JWS) technology, serving as a front-end of the FireSignal system. Distributing through JWS downloads all necessary libraries and if any of these are updated it will also update them automatically. The parameter configuration interface is automatically generated, using the XML Hardware configuration file and the default configuration options can extended by loading plugins. Besides configuration, it also allows the supervision of the states of each Node and Hardware, it allows remote participation, it includes a data viewer and it also has the possibility of performing data analysis using an external tool, but this feature is not currently implemented. The data viewer only supports one dimensional data, but it can be extended with plugins to show more complex data, accordind to the mime-type.

Users can connect to the system anywere in the world, given that they are registered. There is a basic chat room (powered by Jabber) and users are also able to send data references that they find relevant to the discussion to each other. FireSignal recognizes three roles:

- Regular user: can see configurations, and data. Can use the chat and share data;
- Administrator: manages users;
- Operator: can configure the experiment and operate the machine.

It also supports groups with special permissions, for the case were a group of researchers is responsible for the configuration of a susbsystem but are not authorized to operate the machine.

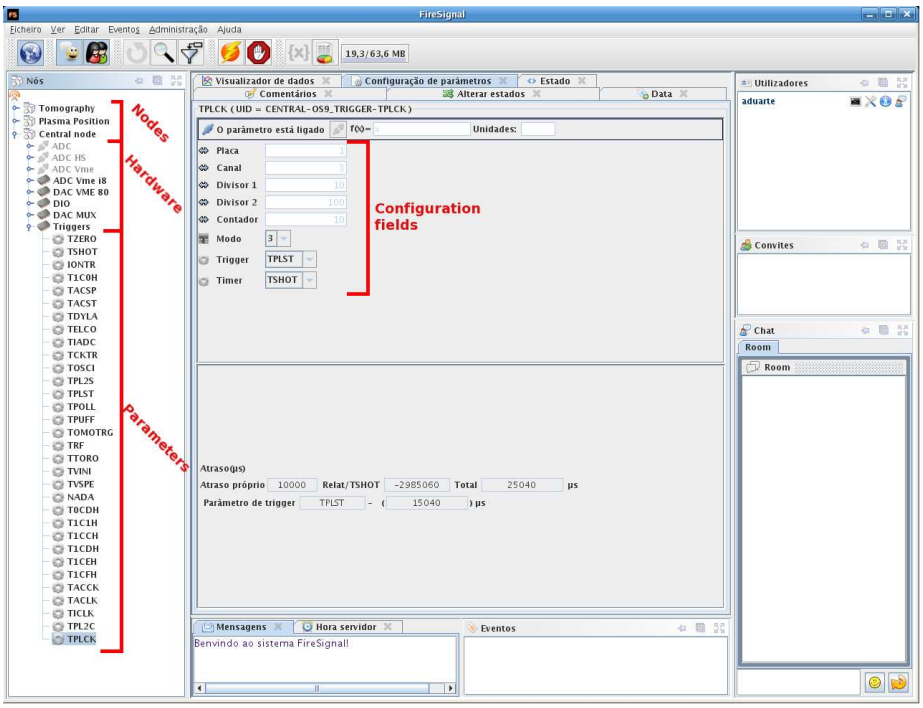

Figure 3.3: The user interface of FireSignal.

3.2 MARTe

Developed at JET, MARTe (Multithreaded Application Real-Time executor) [26] is a modular framework designed for real-time projects, specially real-time control. It is built on a C++ real-time library named BaseLib2, that allows the same code to run in different operating systems. The main features of BaseLib2 are:

- configuration file, used to automatically create and configure objects;
- garbage collector for memory managment;
- HTTP interface vailable for all objects;
- built-in mathematical and algebraic tools;
- configurable logger, including a tool in Java to receive and filter logger messages.

Being a modular design, MARTe itself is made of different components. The basic ones are:

- Generic Application Module (GAM): the main element of MARTe and building blocks. They have three interfaces: one for configuration, one for input and one for output. GAMs that interface with hardware are called IOGAMs. It is also possible to have GAMs that simulate inputs for other GAMs, for debug purposes.

- Dynamic Data Buffer (DDB): optimixed memory bus that GAMs use to transfer data betwen each other. It also ensures the coherance of signals betwen GAMs.

- Real-Time Thread: container of GAMs and responsible for their sequential execution. MARTe must contain at least one real-time thread, but it can handle any number of them, running either concurrently or in parallel.

MARTe is currently being used at JET for the vertical stability control [28], at ISTTOK to control the plasma position and the primary current (for plasma current inversion for AC discharges) [29], and in COMPASS to control the plasma position and shaping [30].

3.3 SDAS

The Shared Data Access System (SDAS) [25] is the part of the CODAC of ISTTOK that is reponsible to find and retrieve data from the database. SDAS is based on Remote Procedure Calls (RPC) specifically XML-RPC, where the calls are made by HTTP using the XML format. SDAS was designed to be used by many different associations, with the objective of users only using a single set of libraries to access any database of the adopting associations. To achieve this it is required for each association to develop a software component which would translate the SDAS calls into something meaningful to their data storage system. There are connectors available for PostgreSQL and MySQL relational databases, with the option of storing data as a binary blob inside the database itself or as a binary file (raw or HDF5).

SDAS design shares some similarities with FireSignal. Data is also indexed by time (start time and end time) and events but a signal unique identifier doesn't necessarily need to follow the node-hardware-structure used by FireSignal. Data is also treated by default as a one-dimensional array of values but it is possible to extend SDAS to support more complex data, like images for example. In the case of varying acquisition rates, it is possible to have the data in slices, each with its own start and end times, and the system will recognize them as being part of the same data set. Data can have linear calibration coeficients applied to them, but on the other hand it is not possible to retrieve the raw data as it is stored: even without coeficients it always returns in double-precision floating-point format.

On the user side, there are libraries in Java, Python and C++ that can be used for programing or be integrated in data analysis software like Matlab, IDL and Octave. There is a simple data viewer in Java that has basic functionalities implemented, allowing users to search for signals and events and view data in either a plot or a table. There is also a Web tool that allows users to search on any web browser for all the data generated by a specific event and to download signals as binary files. SDAS as a basic authentication protocol that allows authorized users to save processed data on the main database. This feature is being used at ISTTOK to store the automated calculations of plasma density and plasma current after each shot.

3.4 CDB

The Compass DataBase (CDB) [31, 32] is the part of the CODAC of Compass that is responsible to find and retrieve data from the main database. Originaly Compass used an implementation of SDAS but due to perceived shortcomings a new solution was designed. These shortcomings were:

- If any changes were made to the Hardware XML configuration file, the version stored in the database would be overwritten. This would cause data previously acquired to appear corrupted or become inaccessible;
- Even though FireSignal allows a channel to be named according to the diagnostic connecting there, there is no consistent way to keep track of what diagnostics were connected before. So it would not be easy for a user to access data from a diagnostic that was connected to different channels in the past. Because of this, in practice, channels are rarely named for their diagnostic;
- Treating the start of the plasma (shot) as any other event in the experiment was considered as adding an unnecessary and redundant abstraction layer, for a pulsed plasma machine like Compass;
- FireSignal (and SDAS) allows data to be stored in blocks, for non-contiguous acquisition or variable acquisition rates. However some diagnostics (p.e. Reflectometry) would generate hundreds of files for a single plasma pulse which can be problematic;
- The design of FireSignal data flow contains a bottleneck because all data must pass through the Central Server. Compass was generating enough data each pulse for this to become an issue.

So, with this in mind, CDB was designed with the following features in mind:

- Nothing stored can be overwritten; instead, revisions are possible as corrective actions;
- These revisions are also used to store the information regarding diagnostics and the physical channels they're connected;
- A relational database is used to store metadata of the numerical data and also physical quantities names, units and scaling;
- Shots (pulse numbers) are treated differently from other possible events (unlike SDAS);
- Timestamps can be stored invidually for each data sample or just for the first and last samples (considering a uniform acquisition rate);
- Data files (HDF5) can be written directly on the main data storage system without having to pass through a central server.

The core application is implemented in Python (pyCDB), with Cython being used to wrap the Python code in a C API. Matlab, IDL etc. clients can then be built using the C API. There is also a Java binding using Jython (Python implementation in Java [33]).

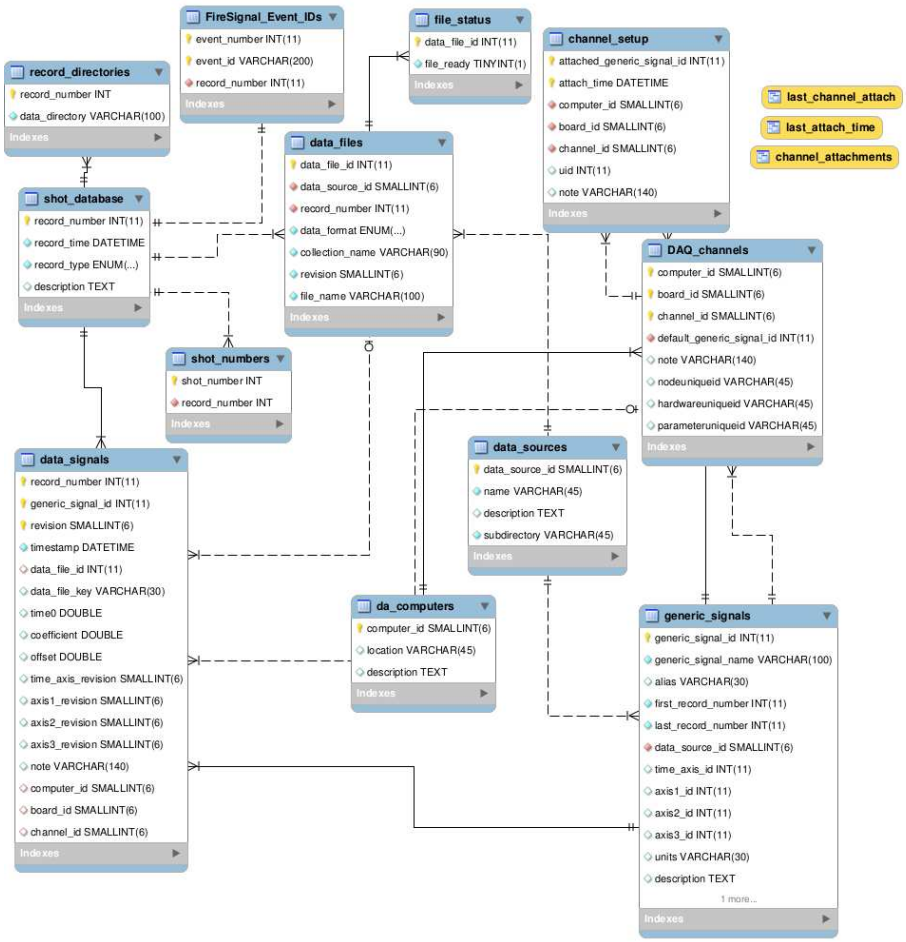

Figure 3.4: CDB database structure. We can see that shot numbers are stored separately from other event numbers (record_number).

3.4.1 Integrating CDB and FireSignal

For CDB and FireSignal integration, two things were required:

1. Design CDB in such a way that FireSignal identifiers for channels and events would still be recognized and usable;

2. Develop a new FireSignal database controller module, for legacy Nodes.

The second point would still cause the issue of bottlenecking that CDB was trying to solve, but it was considered necessary not only because it would allow Compass to operate normally until all nodes had been converted to the new data storage method but it would also allow Nodes developed at IPFN to be tested on Compass before any major changes were applied. The first point was achieved by having functions on the API that convert between FireSignal IDs and CDB IDs and vice-versa. The biggest challenge regarding the second point was the fact that the CDB tools that existed to write the meta-data on the database were only available in Python, so a bridge between Java and Python had to be used. At first Java Embedded Python (JEP) [35] was used as it was more straightforward than Jython, however Jython was easier to install and it was also being used for the Java API bridge. Since it didn't make sense to use two different tools to perform what was basicaly the same task, Jython replaced eventually JEP. The database controller acts like a regular FireSignal one but with some extra steps:

1. Because the two relational databases are being used in parallel, the controller stores meta-data on the FireSignal Database like usual;

2. Following CDB procedure, a request is made for a new signal ID;

3. The data and the FireSignal XML shot configuration file are stored in a single HDF5 file;

4. Meta-data is stored on CDB relational database (MySQL).

Chapter 4

Subsystem Integration

4.1 MARTe - FireSignal Bridge

When MARTe was introduced for real-time control in ISTTOK and COMPASS tokamaks, it was necessary to integrate it with FireSignal [34]. This integration was meant to allow MARTe to receive triggers, send data to the main database and be configurable on the user interface of FireSignal. It was decided that the node would use the standard HTTP server provided with MARTe, since this would allow the node to fulfill the objectives without having to interfere with MARTe core. It was decided to program in Java in order to take advantage of its integrated HTTP and HTML tools.

4.1.1 State Machine

MARTe has its own internal state machine. The states can be cycled manually by the operator either on the console interface or the HTML interface on a web browser. This node completely automates this procedure: once the main event arrives, it will cycle automatically through the states, following closely equivalent states in FireSignal.

Some FireSignal states are mapped to MARTe commands. Although these are not really system states, they tell the operator that the node has sent the related command to MARTe, but the current state hasn't changed yet. There is a thread that constately monitors the MARTe state page and changes the Node status if needed. If necessary, the states can be changed by the operator on the MARTe interface and the Node will acknowledge the change. The system, however, needs to know when to issue the *ejp* command to finish the acquisition.

MARTe	FireSignal
idle	stopped
waiting_for_pre	configured
pulsing	running
post_pulse	running

Table 4.1: MARTe states and matching FireSignal states

MARTe	FireSignal
pulse_setup_completed	configure
pre	run
ejp	N/A
collection_completed	stopping

Table 4.2: MARTe commands and matching FireSignal intermediate states

For this, an extra HTML page is required with the information if there is plasma or not.

4.1.2 Configuration File Parser

A FireSignal Node requires some information on XML files before being built. Because not all the necessary information is available on the HTTP server at start, it was necessary to get this from the configuration file. BaseLib includes a C++ parser, but since it was decided to program the node in Java, making a native parser from scratch was considered a better option than using JNI.

The parser is event-based sequencial access, meaning that while it parses the file it will call methods (events) from a handler object when it finds keywords. Parsing events are sent in the order of the information on the document itself. Besides comments, it defines the following elements of a MARTe configuration file:

- single attributes: elements in the form *att = value*. They are defined with a single value, that can be a string or a numeric value.
- array attributes: elements in the form *att = {value1 value2 ...}*. Single values enclosed between *{...}*.
- blocks: elements in the form *block = { att = ... }*, with attributes and other blocks inside.

4.1.2.1 HTTP Communication

For the communication between MARTe and FireSignal two solutions were available: either developing a GAM implementing the functionalities of a FireSignal Node or developing a Node with HTTP communications implemented. Since the HTTP interface allows access to all configurations and data, the only advantage to have a GAM would be to have it all integrated on MARTe. The HTTP approach offers several other advantages: i) it is possible to develop the MARTe code without FireSignal, ii) the Node is not required to be running on the same machine as the real-time code, iii) any existing MARTe configuration file can be used.

4.2 Fast Acquisition Node

This C++ node was developed for PCIe acquisition boards developed at IPFN [36]. These boards have 32 18-bit ADC channels that can acquire up to 2

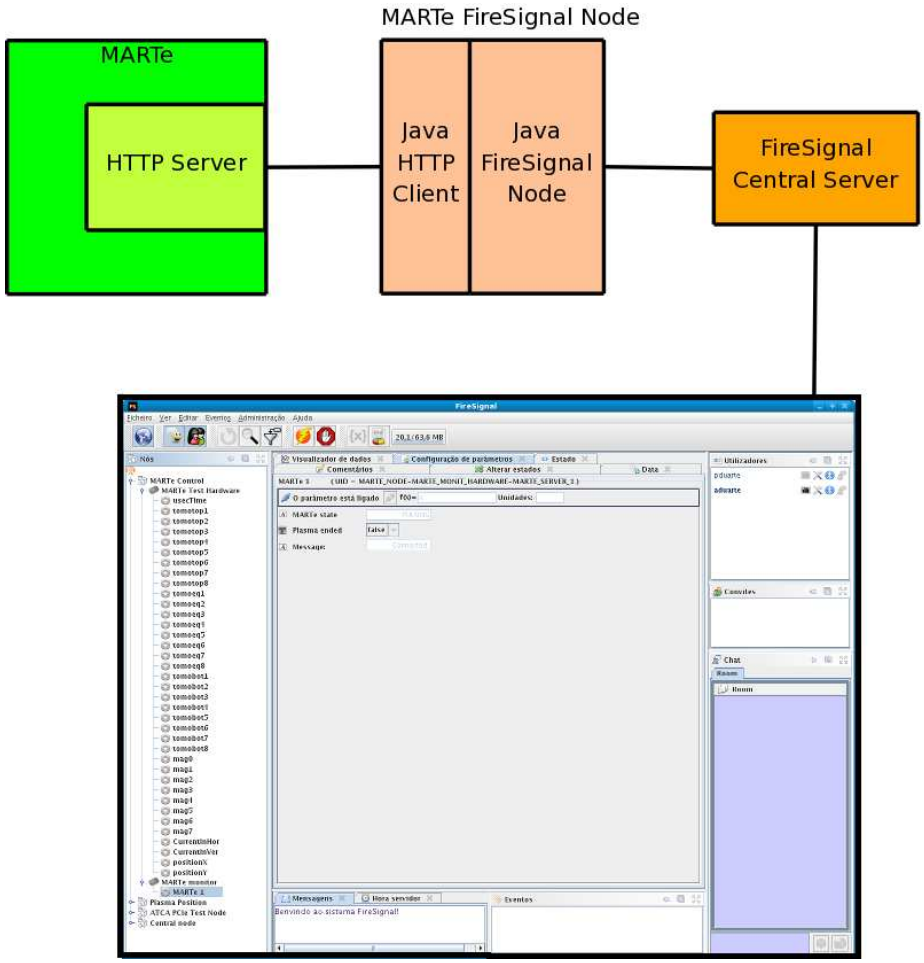

Figure 4.1: Schematic of the MARTe FireSignal Node connects them. It uses HTTP between the Node and MARTe, and uses CORBA for the other connections.

Mega samples per second. The node accesses directly the board through driver commands.

With the node interface it is possible to configure:

- Clock source: choice between shared (common to all boards on the same crate) or local
- Trigger source: choice between shared or local
- Trigger type: choice between software (acquisition begins immediately) or hardware (waits for external trigger)
- Acquisition time:
- Trigger delay: configurable delay, in miliseconds, from the trigger arrival and the acquisition start
- Delay time: delay to be added to the data timestamps, to compensate for any trigger differences with other boards

Besides these fields, there are three other that are not configurable, being used for information and reference:

- Frequency: the acquisition rate is fixed at 2 Msamples/s.
- On board memory size
- DMA byte size

This node defines a main event that usually refers to the plasma start. When a new event arrives the configurations are sent to the boards, which then wait for trigger. In case the software trigger is selected, a call is made to the driver, starting the acquisition.

Acquired data is transfered to the node as a signed 32 bit integer array, with the extra bits padded with 0.

4.2.1 16-bit Node

From the data transmission and storage point of view, using 32 bits to store 18 bit values results in 43.75% of space being wasted. To improve this situation, two solutions were considered: compress the data or only store the 16 most significant bits. Some tests were performed to evaluate these solutions.

Using some acquisitions on ISTTOK, when the shot failed, we analysed the noise level coming from the boards. Assuming it was gaussian noise, we took note of the standard deviation for the entire signal. We then calculated the average for the available channels and took note of the maximum and minimum values, calculating the base 2 logarithm in order to obtain an aproximation on the number of equivalent bits (on a 32 bit integer). We can see some examples on the next table:

The varying number of channels come from the fact that, while some channels were malfunctioning, others were picking up residual signals that were not relevant to our study and because of this were excluded. We can conclude that around 66% of the baseline noise is contained by around 20 bits and is thus

Shot	Average Standard Deviation (log 2)	Min Standard Deviation (log 2)	Max Standard Deviation (log 2)	Number of channels
31652	19.63	18.99	20.19	21
31668	19.63	19.04	20.18	20
31916	19.83	19.14	20.21	27
31942	20.12	19.23	21.82	15
31972	20.05	19.49	21.48	25
32060	19.68	18.98	20.16	20
32116	19.86	19.30	20.33	24
32118	19.85	19.34	20.36	24
32120	19.88	19.44	20.33	24

Table 4.3: Some measurements done to estimate the base noise level at ISTTOK. Not all channels were available for every shot.

safe to remove the 16 less significant bits without any noticeble loss in relevant signal.

The conversion is done in a simple and straightforward manner: a 16 bit pointer is used to run over the data coming from the boards; the less significant part is ignored, while the other is stored in a 16 bit integer buffer that is sent to the central server of FireSignal. This way, without changing the firmware or the drivers, one can choose if he wishes to have the full 24 bits (on a 32 bit integer) or just the 16 more significant.

4.3 Database Growth on ISTTOK

During the years the work laid down in the thesis took, many changes occured on the data acquisition in ISTTOK, specifically an increasing number of channels and the introduction of new technologies. On figure 4.3 we can see this evolution from March 24th 2009) to June 2nd 2016.

March 24th 2009 was the last day before the introduction of the PCIe ATCA boards, when only the TR512 were being used, and also before MARTe. A typical shot then would produce 13.4 MB of data from 94 channels. Almost a month later, the first fast acquisition board replaced some of the old boards so, despite the small decrease in number of channels, the data acquired almost doubled. July saw the introduction of MARTe, with the old control still running. The introduction of another fast acquisition ATCA board in September caused a rise in both channels and data. At the end of November of 2010, the old control was removed from the CODAC and ISTTOK was being fully controled with MARTe; the decrease in data while the number of channels increased is due to the adoption of a shorter acquisition time during this period. It is also important to point out that MARTe can output "virtual" channels that are not directly connected to physical ones, so it is possible to have more channels without adding more boards. This happened between June and September of 2011, where 36 such channels were added without adding extra boards. In January 18th 2012 regular AC discharge operation commenced, which implied longer acquisition times and thus the data acquired almost doubled, despite the number of channels remaining the same. On the 20th more MARTe channels

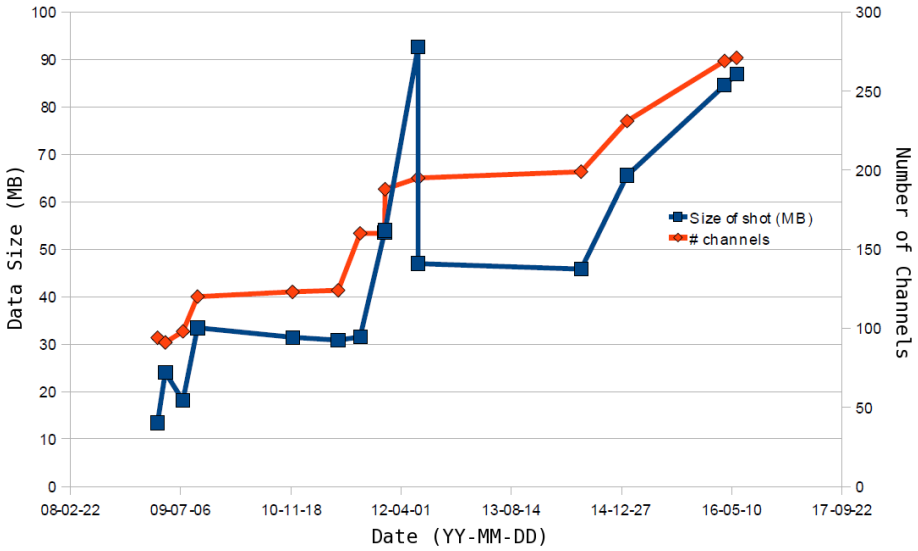

Figure 4.2: The growth of data acquired each shot in ISTTOK from 2009 to 2016, in storage space and number of channels. The choosen dates are related to significant changes, such as: new technologies, new diagnostics and more acquisition channels. The large data size decrease on 2012-06-18 is due to the switch from 32-bit storage to 16-bit, described in subsection 4.2.1.

were added and we can see here that, at this point, the fast acquisition boards were responsible for most of the volume of data, as it barely increased. This volume would peak in June 18th at over 92 MB per shot, but on this day the 16-bit Node replaced the 32-bit version, which caused a dramatic decrease of almost 50% of the volume of data collected, as expected. Two years later no big changes had occured, but in January 22nd 2015 another fast acquisition board was added which caused another noticeable increase in both data and channels. In April 8th 2016 another MARTe board was added and the acquistion time had also increase. The last sample taken for this study, in June 2nd 2016, showed that a typical shot would produce almost 87 MB of data (raw and post-processed) from a total of 271 channels. It is interesting to note that we are aproaching the volume of data acquired when it peak during the time the 32-bit node was being used.

Comparing these results with JET's (Fig.2.3) we can see a similar data growth, although it occured during a smaller time frame. This growth happened for the same reasons: more channels, improved hardware allowing higher acquisition rates, new operation modes. Thus we can conclude that big and small experiments face similar challenges (albeit on a different scale) and it is viable to develop and test designs and technologies on the less expensive, higher availability and less hazardous environment provided by smaller experiements, before aplying them to the large experiments that the study of extreme physics require.

Chapter 5

Future CODAC Design

As we have seen throughout this thesis, CODACs for large, long duration experiments face similar challenges. Many of these were experienced during the work at ISTTOK and COMPASS tokamaks. This chapter discusses how these challenges should be tackled on future designs.

5.1 Guiding Principles

During the work leading to this thesis it became clear that good CODAC design for a large experiment must follow three principles that are somewhat related:

- Scalability
- Adaptability
- Autonomy

5.1.1 Scalability

As we have seen from the case studies and from experience with ISTTOK and COMPASS, long duration experiences have a tendency to grow. New or improved diagnostics are a necessity to improve the scientific output and keep the laboratory relevant. This usually implies more acquisition channels and faster rates. Advancements on electronics will also push the desire for faster acquisition rates and more bits per sample. New techniques (like AC discharges on ISTTOK) can also increase the amount of data generated for each session. Radical changes can be costly, both in time and money, not only in regard to updating the system but also to retrain the staff, so it is desirable that core elements of a CODAC don't change radicaly during the experiment lifetime. This means that the system needs to be ready for this growth from the start.

5.1.2 Adaptability

As new technology becomes available it tends to be less and less compatible with old technology. As such a CODAC must not be too dependent on a given technology, but it should be adaptable to changes that are certain to come, even if they are not obvious. On the other hand, it should be prepared to have

different technologies working side by side. Using fusion resarch as an example, there is a tendency to use cutting-edge technology (as fusion energy itself is cutting-edge), however stable and robust technology is also desirable to protect the investment and the machine itself. That's why space programs still use last century technology, as they can't afford any level of failure.

5.1.3 Autonomy

If a CODAC should not be too dependent on a single technology, it should also not be dependent of a single software or hardware company. Turnkey solutions are apealing, specially when they offer a complete working environment, but can be costly on the long run, as these solutions may not offer the flexibility required by a large experiment. Being able to modify the system without legal restrictions is the easiest way to ensure scalability and adaptability. Of course this is not always possible, but a certain degree of autonomy is desired and thus must be ensured.

5.2 Proposals

5.2.1 General Design

One of the key features that a CODAC should have is undoubtably modularity. This concept is not new, as it was already present on the early designs of JET CODAS, but it is indeed essential if one wants to achieve scalability and adaptability. Idealy, a CODAC should be composed of several different modules, each with its own well-defined function and without overlaps. The connection between modules should be done in a generic and transparent way; each module should not be concerned about the inner workings of other modules. This will also allow modules to be tested outside the main system and be integrated without major changes.

Typically a hierarchical tree-like topology is used, where each component only communicates with the components directly above and below on the hierarchy. Such structure is also found in object oriented programing as it is an intuitive way to implement a transparent module-based structure. However this structue isn't flexible enough and can lead to bottlenecks and excessive complexity. Take for example what happened at COMPASS, where data transfer to the database was hitting a bottleneck at the Central Server of FireSignal, so a bypass between Nodes and Database Manager was implemented. Another example: there has also been some internal discussion on the ISTTOK team regarding a new trigger distribuition hardware and how it should handle events, which has lead to considering a less centralised event distribution. In conclusion, the structure the structure should allow flexibility, with each module being capable of direct communication with more than those directly above and below, in what is called a partially connected mesh topology.

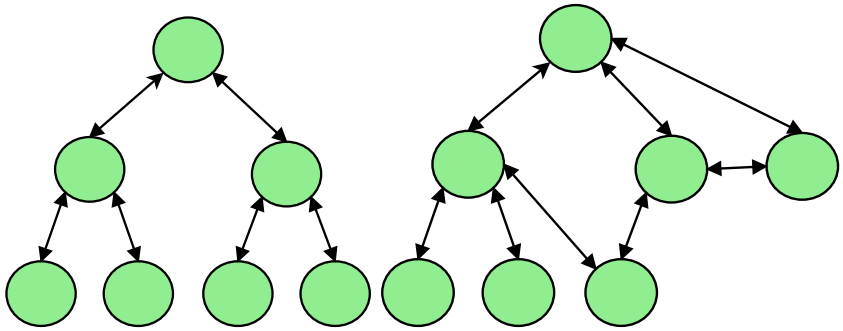

Figure 5.1: Diagrams of example network topologies. On the left, a tree (hierarchical) topology, on the right a Partially connected mesh topology.

5.2.2 Data collection and access

As it was mentioned, experimental data volume increase is a given on these types of experiments so it is important for a CODAC to be prepared for it. This growth has two major implications: i) network bandwidth for data transfer and ii) data storage space. Bandwidth is specially important when one considers that, in experiments like ITER, some of the data must be streamed in real-time. Such data, particularly when concerning critical systems such as machine security, must have a dedicated network, physically separated from the main internal network.

To avoid data transfer bottlenecks, a solution similar to the one adopted at COMPASS, where data is written directly in a database cluster, is one of the best approaches, but it can be made more transparent. The best way to achieve this is to implement a cluster database using different disks on different machines (which could even be the same crate where the acquisition boards are installed) which the Database Connector would treat as a single machine. Systems like this have been implemented in data warehouses where it was proved its large scalling potential, the absence of bottlenecks but also that queries involving joins over large data sets from different machines may take longer than expected. This issue can be handled by implementing a system similar to JET, with a second centralised database where priority data can be stored imediatelly and that can also serve as a backup for the distributed database.

Data should be stored in a standard format, so that researchers may be able to open it and extract the information regardless of their favourite data analisys tool. Naturally, the most universal format should be raw binary but it has some drawbacks, namely the fact that code developed in one archicture is not guaranted to be 100% portable unless the developer takes measurers that only make the code more complicated and harder to read, specially for physicists who may want to use the code but are not familiar with this situation. As there are ISO standards that are widely used and are transparent to end-users, these provide a better alternative to raw storage.

Another measure that can be used to address both large data transfers and storage is compression. There are many algorithms available, both open-source and proprietary, lossless and lossy, some even taylored for fusion data. Lossy algorithms are normaly rejected, as researchers like to have access to as much

		Raw	gzip		bzip2	
shot	FSHardware	size	size	time	size	time
39142	ATCA Board 1	24 MB	9.5 MB	3.005 s	7.1 MB	4.201 s
“	ATCA Board 2	23 MB	12 MB	3.778 s	8.3 MB	3.841 s
“	MARTe	4.6 MB	1.9 MB	0.473 s	1.7 MB	1.457 s
40110	ATCA Board 1	24 MB	11 MB	2.873 s	8.2 MB	4.420 s
“	ATCA Board 2	23 MB	12 MB	3.698 s	8.5 MB	3.915 s
“	MARTe	990 KB	311 KB	0.055 s	292 KB	0.553 s

Table 5.1: An example of data compression at ISTTOK.

information as possible, but can be usefull for the streaming of non-critical data (p.e., video from inside the machine). A downside to compression is that it can sometimes take so much time that any transfer time gain is rendered moot, so this must be taken into consideration when deciding to compress the data and which algorithm to use. A situation where the compression-decompression overhead is not as critical is long term storage since, tipically, the older the data is, the less frequently accessed it is and even basic algorithms can provide more than 50% compression rates in some cases. On table 5.1 we can see a simple example of offline compression of ISTTOK data for two fast acquisition boards and MARTe for two different shots. The data was first archived using the *tar* tool and then compressed with two different open-source tools, *gzip* and *bzip2*, that are standard in every Linux distribuition. A third tool (*pkzip*) was also tested but its results were very similar to *gzip* (<5% difference in compression rate), so they are not shown here. It is clear that, in the case of ISTTOK, even with generic compression algorithms it is possible to save more than half of the disk space. We can also see that compression with bzip2 is more effective, but takes more time.

5.2.3 User tools and interfaces

When designing user tools it is important to account not only what people need but also what they want. A developer must be aware that something that may be easy and intuitive for him may not be for others and some features that may seem pointless to him may be invaluable to ensure that eficiency is maintained. Habits are hard to die so when upgrading a system keeping the same look and feel of the old one will make the transition smoother, like it was done, for example, when JET transitioned from NORD computers to PCs.

Regarding data retrieval in fusion research, there has been some proposals for a shift from the pulse/event number paradigma to pattern recognition. The idea is that, for longer pulses and bigger databases with many signals, it will be easier for researchers to look for signal patterns that match phenomena they wish to study, than relying on lists compiled by other people and having to download data for an entire pulse where only a small portion is interesting (p.e., some plasma instabilities last less than a milisecond while an ITER pulse is expected to be in the order of minutes). However, no matter how advanced the recognition algorithm is, it is still necessary to store data according to shot/event, because it is an intuitive way of labeling data and also because it is what most users are

expecting, as it was mentioned on the previous paragraph.

User interfaces are very important as they are the tools through which users interact with the CODAC. They should be simple and modular, but at the same time provide users with all the tools they need. Having all functionalities on the same application may look efficient, but a single memory heavy application is more prone to crashes and freezes, besides clutering the screen with excessive information. Functionalities such as configuration, monitoring and data visualization should have independent and specialised tools that must, none the less, share a common look and design philosophy.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Designing and implementing a CODAC for a large physics experiment is a very complex and daunting task. It requires rigourous accessment of present challenges and careful planning for future ones.

FireSignal proved its flexibility in integrating new subsystems but it has become apparent that its design philosophy is too rigid, as some changes required by COMPASS were fundamentaly against its original design, namely the hierarchycal message structure. Also, transferring large quantities of data is an issue that is not satisfactorily addressed by FireSignal, because of the inherent bottleneck in its structure. A distributed architecture can help with this, has it was seen in CDB. Related to this issue is the impredictable overheads and timeouts of CORBA communications, which causes delays in data transfer. On the other hand, CORBA has proved to be a good solution for the interoperability of different FireSignal modules, so a similar solution with less and more predictable overhead should be considered. Another issue related to FireSignal, that became apparent during this work, were the drawbacks of Java-based technology, specially regarding memory usage. Also, the security in Java Web Start has increased in every new version and currently all the applications described on this thesis that use this technology require extra configuration[38], which is not in line with the initial objective of being easy to use. Because of this, alternatives are now being studied. Also patent during this work was the fact that the FireSignal/SDAS database lacks flexibility. For example, the acquisition boards used for the reflectometry diagnostic have a mode called "burst" that allows to acquire only when the wave launcher is in operation. This could be implemented in FireSignal as different data slices for each "burst", but it would imply hundreds of files for each channel in a single pulse. In a case like this, it would be more sensible to store the time in a different way.

The event based data labeling, as implement on FireSignal works well as a generic solution, but in the case of fusion research treating the pulse as a special situation still makes sense, even for longer pulses. Experiments are still being programmed per shot basis and will likely continue to be so in the near future, which means that tagging data and events by shot number is important for researchers to find data of interest to them. It would also be useful to classify

events on three categories: 1) pre-programmed events (p.e. pellet injection, probe activation); 2) plasma events automatically detected on real-time; 3) plasma events detected by offline analysis. The latter would be specially useful for researchers looking for specific phenomena on older data.

6.2 Future Work

With the deployment of CBD in COMPASS, an autonomous CODAC team was created and has developed their own forks of IPFN technology, but collaboration is likely to still continue.

Changes to FireSignal in ISTTOK are being considered, namely:

- Replacing the raw binary files with HDF5 files;
- Support for data compression, both in real-time and post-processed;
- Support for signal previews (using decimation);
- Using web technologies to create a more robust yet flexible user interface.

If these are successful, it may be possible to also apply them to COMPASS.

A full migration from VME technology is being prepared. Currently this hardware is being used mostly for triggers, so a new timing board has been built and, in the near future, it will have to be integrated in FireSignal.

Bibliography

[1] J.B. Lister, et al., "The status of the ITER CODAC conceptual design" , Fusion Engineering and Design 83 (2008) 164-169.

[2] J. P. Freidberg, Plasma Physics and Fusion Energy, Cambridge University Press, 2007.

[3] J. A. Wesson, Tokamaks, Clarendon Press, 1997.

[4] D. A. H. Hanaor, et al., "Solution based synthesis of mixed-phase materials in the Li2TiO3-Li4SiO4 system", Journal of Nuclear Materials 456 (2014), 151-161.

[5] "Introduction to Magnetic Fusion", [Online]. http://www-fusion-magnetique.cea.fr/gb/fusion/principes/principes02.htm [Accessed 2016].

[6] "COMPASS tokamak", [Online]. http://www.ipp.cas.cz/vedecka_struktura_ufp/tokamak/tokamak_compass/ [Accessed 2016].

[7] R. Panek, et al., "Reinstallation of the COMPASS-D tokamak in IP-PASCR", Czechoslovak Journal of Physics 56 (2006) 125-137.

[8] "ISTTOK tokamak", [Online]. http://www.ipfn.ist.utl.pt/isttok/. [Accessed 2016].

[9] I. Carvalho, et al., "ISTTOK real-time architecture", Fusion Engineering and Design, 89, (2014), 195-203.

[10] I. Carvalho, Real-time control for long ohmic alternate current discharges, PhD Thesis, Instituto Superior Técnico, Lisbon, 2013.

[11] D.F. Valcárcel, et al., "An ATCA Embedded Data Acquisition and Control System for the Compass tokamak", Fusion Engineering and Design 84 (2009), 1901-1904.

[12] "CERN about", [Online]. Available: https://home.cern/about/. [Accessed 2016].

[13] "JET tokamak", [Online]. Available: https://www.euro-fusion.org/jet/. [Accessed 2016].

[14] J.G. Krom, "The evolution of control and data acquisition at JET", Fusion Engineering and Design 43 (1999) 265-273.

[15] J. Vega, et al., "New developments at JET in diagnostics, real-time control, data acquisition and information retrieval with potential application to ITER", Fusion Engineering and Design 84 (2009) 2136-2144.

[16] "Wendelstein 7-X stellarator", [Online]. Available: https://www.ipp.mpg.de/16931/einfuehrung. [Accessed 2016].

[17] T. Bluhm, et al., "Wendelstein 7-X's CoDaStation: A modular application for scientific data acquisition", Fusion Engineering and Design 89 (2014), 658-662.

[18] H. Laqua, et al., "Test of the steady state W7-X control and data acquisition system at the WEGA stellarator", Fusion Engineering and Design 85 (2010), 520-524

[19] "National Ignition Facility", [Online]. Available: https://lasers.llnl.gov/. [Accessed 2016].

[20] R. Carey, et al., "The National Ignition Facility Data Repository", Proceedings of ICALEPCS2009, Kobe, Japan.

[21] F. Carena, et al., "The ALICE data acquisition system", Nuclear Instruments and Methods in Physics Research A 741 (2014), 130–162

[22] "ALICE information", [Online]. Available: http://aliceinfo.cern.ch/Public/Welcome.html. [Accessed 2016].

[23] A. Neto, et al., "FireSignal—Data acquisition and control system software", Fusion Engineering and Design 82 (2007), 1359-1364.

[24] Object Management Group, CORBA/IIOP Specification, Document — format/04-03-01 (CORBA, v3.0.3), March 2004.

[25] A. Neto, et al., "A standard data access layer for fusion devices R&D programs", Fusion Engineering and Design 82 (2007), 1315-1320.

[26] A. Neto, et al., "MARTe: A Multiplatform Real-Time Framework", IEEE Transactions On Nuclear Science 57 (2010), 479-486.

[27] A. J. N. Batista, et al., "ATCA control system hardware for the plasma vertical stabilization in the JET tokamak".

[28] F. G. Rimini, et al., "First plasma operation of the enhanced JET vertical stabilisation system", Fusion Engineering and Design 86 (2011), 539-543.

[29] I. Carvalho, et al., "Real-time control for long ohmic alternate current discharges", Fusion Engineering and Design 89 (2014), 576-581.

[30] F. Janky, et al., "Upgrade of the COMPASS tokamak real-time control system", Fusion Engineering and Design 89 (2014), 186-194.

[31] J. Urban, et al., "Integrated data acquisition, storage, retrieval and processing using the COMPASS DataBase (CDB)", Fusion Engineering and Design 89 (2014), 712-716.

[32] "Online documentation for CDB", [Online], http://cdb.readthedocs.io/en/latest/. [Accessed 2017].

[33] "Jython Project homepage", [Online], http://www.jython.org/. [Accessed 2016].

[34] D.F. Valcárcel, et al., "Real-Time Software for the COMPASS Tokamak Plasma Control", Fusion Engineering and Design 85 (2010), 470-473.

[35] "JEP homepage" [Online], https://github.com/ninia/jep. [Accessed 2017].

[36] A. J .N. Batista, et al., "ATCA/AXIe compatible board for fast control and data acquisition in nuclear fusion experiments", Fusion Engineering and Design 87 (2012), 2131-2135.

[37] J. A. How, et al., "Trends in computing systems for large fusion experiments", Fusion Engineering and Design 70 (2004), 115-122.

[38] "Security level settings in the Java Control Panel", [Online]. Available: https://www.java.com/en/download/help/jcp_security.xml. [Accessed 2016].