

EVOLUTION OF THE FERMILAB CONTROL SYSTEM

J. Patrick[†], FNAL, Batavia, IL 60510, USA

Abstract

The Fermilab accelerator complex is currently running to simultaneously provide beam for 2 TeV proton-antiproton collider operation including antiproton stacking, 8 GeV miniBooNE operation, and 120 GeV fixed target operation. The current accelerator control system, generally referred to as ACNET, was initially implemented 20 years ago. In recent years, obsolete front-end technologies have been gradually replaced. However the bulk of the mid to high levels of the system are still based on VAX/VMS.

While this serves the complex well, the hardware performance of VAX processors is far below current commodity systems, and there is little support for third-party applications. To address this, a migration of the mid and high-level application programs in the system is in progress. The new system is based on a large cluster of commodity PC and UNIX hardware, with applications primarily written in the Java language. Extensive use is made of web-based technologies to provide easy distributed access to the system. This migration must be accomplished without major interruptions to accelerator operation, and during a time when operational demands on the complex are at the highest level ever.

1 FERMILAB ACCELERATOR COMPLEX

The Fermilab accelerator complex consists of the following machines:

- 400 MeV linear accelerator (“Linac”)
- 8 GeV synchrotron (“Booster”). Provides beam for the fixed target “miniBooNE” experiment .
- 150 GeV synchrotron (“Main Injector”). Provides beam for antiproton production, on-site fixed target experiments, and will provide beam for the MINOS neutrino oscillation experiment in Soudan, Minnesota beginning in late 2004.
- 1000 GeV superconducting synchrotron/storage ring (“Tevatron”). Proton-antiproton collider for CDF and D0 experiments. No longer used for fixed target experiments.
- Antiproton production and accumulation facility (“Antiproton Source”). Consists of a target station, and 8 GeV debuncher and accumulator rings.
- 8 GeV storage ring using permanent magnet technology (“Recycler”). Will be used for anti-proton storage, located in the Main Injector tunnel

The first three machines serve as an injection chain into the Tevatron as well as the dedicated functions listed.

Fermilab Tevatron Accelerator With Main Injector

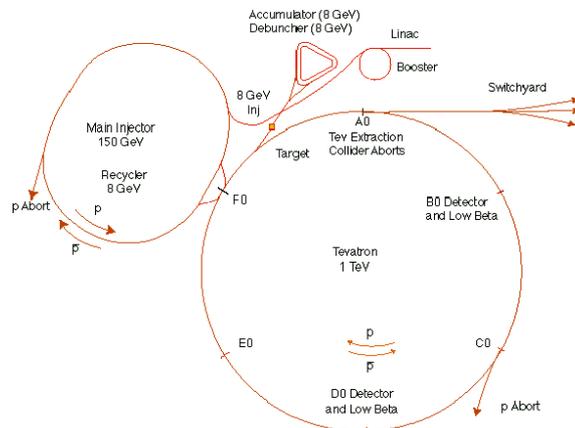


Figure 1: Fermilab Accelerator Complex

2 CONTROL SYSTEM

2.1 Overview

A single unified control system, commonly referred to as “ACNET”, services all machines in the complex. Of order 350 front-end computers interface hardware elements to the control system via a variety of field buses (CAMAC, VME, PCI, Arcnet, GPIB, etc.). These run real-time operating systems, currently VxWorks or pSOS. There are also a small number of Labview based front-ends. Central Services, most of which are referred to as Open Access Clients (OACs) are persistent tasks that run on a high level operating system and have no user interface. Most of these now run on UNIX systems (~90 Sun Netra), some remain on VAX/VMS. Of order 500 console applications provide the primary user interface to the system. Most applications run on VAX/VMS systems (~100 systems available), a migration to Java is underway and will be described below. The primary communication protocol between the various layers is also known as ACNET. Currently this is a custom, connectionless UDP based protocol running over ethernet.

2.2 Key Features

The commercial Sybase relational database is used to store device definitions, application specific data, and data recorded during collider stores. The ACNET communication protocol provides for efficient collection of high rate data up to 1440 Hz by grouping readings into packets returned at 5 Hz. It also supports collection of higher rate but limited duration data known as snapshots. System elements without a hardware clock decoder may take advantage of clock events multicast over ethernet. State transitions may be generated by any element and are multicast to the entire system. An extensive user configurable distributed data logging system provides for

[†]patrick@fnal.gov. On behalf of the Fermilab Controls Department

collection and storage of any device readings in the system at 1 Hz or even greater. Applications can transparently switch between the real accelerator and models or archived data. A sophisticated Sequencer program automates the lengthy process of initiating a store in the Tevatron. A Sequenced Data Acquisition OAC collects and stores data during each of the steps of this process. An Accelerator Control Language (ACL) provides scripting access to the system. Underlying the applications are substantial libraries for data acquisition, graphics, and support of specific hardware.

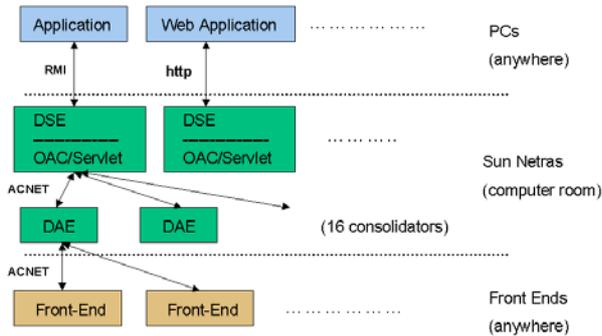


Figure 2: Overview of upgraded portion of control system

3 HISTORY

The basic concepts of the current control system date originally from construction of the Tevatron, which began operation in 1983. This includes the overall architecture including a unified system for all machines, the general application look and feel, and the ACNET communication protocol. Over the past 20 years of course technology has advanced and most components of the original system have become obsolete. However the system has continually evolved to take advantage of new technology, provide for increasing operational demands, and provide for reasonable maintenance.

Front-end systems originally were PDP-11 and Lockheed-Martin MAC16 computers. These gave way to i386 Multibus and 68000 VME based systems running MTOS. And in recent years all of these older systems have been replaced by VME based 68040 or Power PC processors running VxWorks or pSOS.

Console applications originally ran on PDP-11 computers with custom graphics. These gave way to VaxStations with X-window graphics. Currently most VaxStations are centrally located and PCs are used as X displays.

Communication via the ACNET protocol originally used Digital PCL11-B links. The protocol was migrated to IEEE 802.5 token ring links, then raw ethernet, and finally UDP over ethernet

Originally device and application information was stored in DEC Datatrieve and VMS flat files. The Sybase commercial database, originally running on VMS and later transferred to UNIX systems, replaced this.

Programming was originally done in FORTRAN and assembler, then C. Now Java is used for high-level software and there is some C++ in front-end code.

Most of the major upgrades listed above occurred of order 10 years ago or more. Further modernization of the front-ends and field hardware has been done in recent years. All MTOS and token ring based systems have now been retired. The major remaining obsolete technology in the system is VAX/VMS. The processor technology is around 10 years old, much slower than even the most inexpensive current generation PCs. Support for third party applications is becoming progressively more limited, and for new software products non-existent. To address this issue, a project to migrate VAX/VMS code to PC and UNIX is underway.

4 ACCELERATOR PLANS

The Fermilab accelerator complex is expected to operate an average of 40 weeks/year, no lengthy shutdowns are planned. The highest priority activity is colliding beam running, which requires not only the Tevatron but also all other machines in the complex for creating and storing the antiprotons. An extensive series of accelerator improvements are intended to increase the luminosity by a factor of 5 over the next 5 years. In 2009, CDF and D0 are expected to give way to the new BTeV experiment. The miniBooNE experiment currently consumes 80% of the protons accelerated by the Booster. The 120 GeV fixed target program is just now beginning. In late 2004, MINOS operation will place greatly increased demand on Main Injector operation. This simultaneous collider and fixed target running has not been previously attempted at Fermilab. Given the unprecedented operational demands on the accelerator complex, maintaining, modernizing and upgrading the control system to run for another decade will be a significant challenge.

5 MIGRATION STRATEGY

In order to run another 10 years, and accomplish the required accelerator upgrades, it will be necessary to reduce the dependence on VAX/VMS. As there are no lengthy shutdowns planned, this must be performed in a piecemeal manner. New services or applications must often interoperate and thus be plug-compatible with old. Major changes to key features such as the device model, database, and ACNET communication protocol would thus be difficult to accommodate. While this mode of upgrade is constraining, it does provide the opportunity to easily compare the operation of old and new components.

The general architecture of the system remains the same. However the central layer of the system is expanded. Communication between applications and front-ends goes through "Data Acquisition Engines". These perform consolidation of requests for particular devices across all applications running in the control system, thus reducing the load on the front-ends. In addition there is a "Data Server Engine" layer that bridges

the ACNET protocol to other more standard protocols. This is discussed further in section 6.5.

The central and application layers run entirely on commodity hardware. PCs in the control room or elsewhere run console applications. Central services are run on commodity UNIX systems or PCs located in the computer room.

The decision has been made to write most programs in the Java language. This is a modern, object-oriented language with an extensive class library. It is platform independent, allowing use of Windows and UNIX systems equivalently. It is much easier to learn and use than C++, this is a major consideration as physicists, engineers, and other non-computing professionals write much of the machine software. The key disadvantage is speed of execution, however it appears to be fast enough for control system type applications. After all the system is currently run on 25 MHz VAXes.

The strategy is however not to simply port the existing VMS code. Java class libraries replace parts of the application support libraries dealing with graphics, threads, and networking. A more object-oriented API is provided to access data from the control system. Obsolete code will be removed from applications, and it is likely in some cases multiple applications can be consolidated into one.

6 MIGRATION PROGRESS

After some initial investigation, a project to introduce Java into the control system began over 5 years ago. The initial emphasis was on implementation of the ACNET protocol and infrastructure for central services [1]. The first major project to be done in the new framework was SDA, which is an overloaded acronym for both Sequenced Data Acquisition, and Shot Data Analysis. This consists of an OAC to acquire and save data during collider stores, as well as a suite of applications to configure the acquisition and analyze the stored data. As well as exercising data acquisition to a variety of front-ends, it also required development of some substantial applications. It is critical to understanding and improving the operation of the machine. Also, the status display broadcast to the site cable TV network ("Notify") was migrated to Java and substantially enhanced.

The SDA system has been operational for the current collider run, which began over two years ago and has generally worked reliably. Analysis applications have been continually improved according to machine physicists' requests

6.1 Distributed Data Logging

Another example of a major system that has taken advantage of new technology is the distributed data logging. About 70 nodes run an instance of a data logger program that collects data assigned to it and saves it on a local disk. Data are viewed with a plotter application. Groups of nodes are assigned to different machines or departments, who are responsible for what devices are

logged and at what rate. A control application allows easy modification of a logger configuration. When the local disk fills, the wrap around occurs and the oldest data are dropped. The move from VAXes to newer UNIX systems has provided for much larger disk files and thus longer wraparound times. The MySQL database is used to store the data rather than flat files as on VMS. A new "backup" logger runs once per day and consolidates all data from the previous 24 hours into a single large MySQL database. This preserves all logged data indefinitely. New tools to examine the data including a purely web based plotter and simple Java API are provided.

6.2 Application Framework

Console applications written in Java are based on an Application Framework. This framework provides a JFrame subclass containing default menus and a toolbar. Functionality provided by these menus includes the ability to extract images or tables of numbers and save or e-mail for import by other programs. Images can also be posted directly to the electronic logbook. Program messages are captured and saved in a database for external viewing. The default menus may be extended using xml configuration files.

Applications are launched by an Application Index program. Both a Java application and a web-based applet are available. Java Webstart is the mechanism used; it alleviates the need to install any software on a PC other than Java. Webstart automatically caches required code on the local PC, and updates it when a newer version is available. The JNLP description files required by Webstart are dynamically generated from database entries. This greatly simplifies maintenance of these files for the potentially large number of applications.

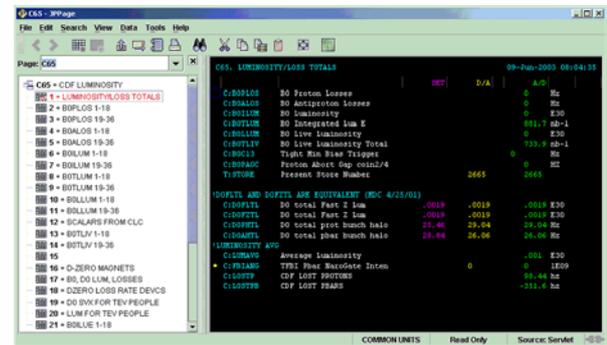


Figure 3: Parameter Page application showing menus and tool bars provided by the Application Framework.

6.3 Web Applications

It is convenient to be able to run some core applications at remote sites where perhaps the proper version of Java is not installed, or one is on a modem line and the time for Webstart to load the code is prohibitive. To accommodate this, purely browser based versions of the device database browser, parameter page, data logger plotter, and synoptic display have been developed. There are primarily web forms with JavaScript used in some cases. Xml encoded requests for device data are made to a Java servlet, which

fetches the data from the control system and returns it to the client. Web services have been investigated for these and similar types of applications, but are not used at this time.

6.4 *Xml-rpc*

There is a need to provide access to accelerator information by outside systems such as the experiments. The method chosen for this is xml-rpc [2]. This is a basic remote procedure call protocol that uses http with xml encoding of data. In addition to remote access, it provides convenient access from languages other than Java. While the update rate capability will be much less than that obtained with the internal protocols, it has proven to be sufficient for current needs. This system is heavily used by the CDF, D0, and miniBooNE experiments. In addition to obtaining accelerator status information, they also can set memory-resident devices with measurements of interest to the machine from their detectors. For example the beam position measured by the CDF silicon detector, and the D0 luminosity measurement are reported this way. For security reasons this mechanism cannot be used to set real hardware devices.

6.5 *Communication*

Although the transport medium has changed, the custom ACNET protocol has evolved only modestly over the years and remains a foundation of the system. While investigations of CORBA and other technologies have been done, it has proven too difficult to implement anything new in some of the older front-ends with limited memory. The Java API emulates some functionality not provided by the protocol, such as larger data offsets, time stamps on returned data, and collection on clock events plus delay. These features are now in the process of being added to the protocol.

The architecture of the new system does however incorporate a "Data Server Engine" layer that is used for protocol bridging. Java applications actually make requests via the Java Remote Method Invocation (RMI) protocol. The DSE translates these into ACNET requests, sends them to the front-end consolidators, and performs RMI callbacks to the application with the data once it is returned. While this increases the latency somewhat, operations performed from applications are not time critical at this level. Also something similar occurs with web and xml-rpc based applications. Here http requests are translated into ACNET requests by servlets or other OACs in the DSE.

Considerable hardware, such as PLCs, scopes, etc. now come with ethernet interfaces and support TCP/IP communication. To minimize the APIs that application programmers must learn, these are usually mapped to the ACNET device model and protocol in one of two ways.

In the past this has mostly been done through front-end systems, which translate ACNET requests into TCP/IP requests to the devices. The newer OAC infrastructure also accomplishes this easily for devices where a real-time operating system is not required.

6.6 *Summary of Migration Status*

As of this time, there are about 70 OACs running in the Java environment, only 6 remain on VMS. In addition to the Application Framework and core applications, a small number of machine specific applications have been written [3]. Notable among these is a program to automatically tune the transport line to the miniBooNE experiment. Also, applications that deal with the Quench Protection system have now completely replaced the VMS versions. A working group is being formed with representatives of the system departments to better plan migration of machine applications. The total size of the code base is about 3000 classes and 500,000 non-comment lines of code.

7 SUMMARY

Although the fundamental architecture is 20 years old, the Fermilab control system has undergone substantial modernization over the years to take advantage of new technology, address operational needs of the accelerators, and ease maintainability. Recent years have seen significant modernization of front-end systems and field hardware. A migration path based on commodity hardware and the Java language has been established for addressing the primary obsolete component of the system, VAX/VMS. Nearly all central services have been migrated. A substantial Application Framework has been developed, and a modest number of core and SDA applications written in that framework. Significant use is made of web technologies. Plans for porting the large number of machine applications are under development.

8 ACKNOWLEDGEMENTS

Thanks go to all my many talented colleagues currently in the Fermilab Controls Department, and to all the past members as well who helped develop the system. The Java migration project, the primary topic discussed above, was initiated by Kevin Cahill.

9 REFERENCES

- [1] D. Nicklaus, "Development of Java Open Access Clients", these proceedings
- [2] <http://www.xmlrpc.org>
- [3] S. Lackey and F. Zhang, "Fermilab Accelerator Application Migration Project", these proceedings.