The Graduate University for Advanced Studies (SOKENDAI)

Doctoral Thesis

Study on Accelerator Operation and Diagnostics in Wide Area Environment

Author:
Akito UCHIYAMA

Supervisor:
Prof. Kazuro Furukawa

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Accelerator Science
School of High Energy Accelerator Science

August 2014

THE GRADUATE UNIVERSITY FOR ADVANCED STUDIES (SOKENDAI)

Abstract

School of High Energy Accelerator Science

Doctor of Philosophy

Study on Accelerator Operation and Diagnostics in Wide Area Environment

by Akito UCHIYAMA

Based on personal experience with accelerator operation and maintenance, the author felt a deep need for remote operation of accelerators in a variety of situations. Currently, enormous projects, such as the International Linear Collider (ILC), will be launched as a result of the collaborative effort of multiple institutes. After the ILC's collaborative commencement, all associates, except for the host country, will need to have methods for remote maintenance, control, and monitoring of associated devices via Wide Area Network (WAN) links. For example, a method has to be provided for connecting to the control system network via WAN links from various collaborating institutions. Such remote operation needs to implement components that have concepts different from those of daily operation, for example, communication, man-machine user interface (UI), and system security. On the other hand, systems for remote operation have been developed by research organizations. Almost all such systems are allocated enough system resources.

As described above, remote operation is necessary in many situations. However, it is not always possible to use environments with rich system resources, such as dedicated networks, for remote operation. Therefore, in this study, methods regarding UI and system security are discussed for remote operation with limited system resources, such as low-band width networks.

Considering operator interface (OPI) for remote operation, the use of standard protocols, such as Hypertext Transfer Protocol (HTTP), has certain advantages because system-dependent protocols are not necessary between remote clients and on-site servers.

On the other hand, Web services that provide a UI component to accelerator control systems have been utilized for various purposes already because such services possess the advantages of being platform-independent systems, providing software maintenance, and can be developed rapidly.

In this study, the author develops a client system based on WebSocket as a next-generation OPI over the Web using Experimental Physics and Industrial Control System (EPICS) Channel Access (CA); WebSocket is a new protocol provided by the Internet Engineering Task Force (IETF) for Web-based systems. In order to construct a WebSocket server as an EPICS CA client, add-on software for Node.js, called Node-CA, was developed by the author in C/C++ using the EPICS CA library, which is included in the EPICS base. As a result of this implementation, WebSocket-based client systems have become available for remote operation using EPICS in wide area environments.

In addition, from a practical application standpoint, the remote operation of an accelerator via WAN is beset by a number of issues. One such issue is that the accelerator has both experimental device and radiation generator characteristics. Therefore, any error in the operation of the remote control system could result in an immediate breakdown. For these reasons, the author proposes the implementation of an operator intervention system for remote accelerator diagnostics and support that can obviate any divergences between the local control room and remote locations.

The purpose of the proposed operator intervention system is to ensure safe remote output control of EPICS via the WebSocket server. To simply monitor the status of the accelerator parameters using the WebSocket-based OPI, permission from the on-site operator is not necessary, provided that authentication has been accomplished through a Secure Sockets Layer (SSL) connection. The WebSocket-based client system is designed such that output control via remote operation always requires the permission of an on-site accelerator operator who can intervene at any time. The main part of the proposed operator intervention system consists of a Process Variable (PV) gateway provided by EPICS collaboration, a MySQL database, and Web applications. After the implementation, the on-site accelerator operator can determine the availability of the equipment, and decide whether to grant operation requests from remote users.

Finally, the system that combines WebSocket-based OPI and an operator intervention system is implemented for the 28 GHz superconducting electron cyclotron resonance ion source (28GHz SC-ECRIS), which is one of the components in the RIKEN Radioisotope (RI) Beam Factory project, for verification of usefulness. Using this system, the author confirmed that the output instructions did not reach the EPICS IOC without the on-site operator's permission. In addition, remote operation was performed satisfactorily with a network latency of 200 ms, which is approximately the same network latency between

Japan and the USA. Therefore, networking performance tolerance should allow this system to be used over networks provided by almost all Japanese Internet service providers. Because the system is implemented with RIKEN 28 GHz SC-ECRIS remote operation, the author confirmed that this new operator intervention system via WebSocket-based OPIs is a useful method for ensuring the physical security of operational errors.

Acknowledgements

The author would like to thank Dr. Masanori Satoh, Dr. Tsuyoshi Suwada, Dr. Fusashi Miyahara, Dr. Norihiko Kamikubota, Dr. Takashi Obina, Dr. Tatsuro Nakamura, Ms. Misaki Komiyama and Mr. Jun-ichi Odagiri, all of whom provided invaluable comments and warm encouragements.

Special thanks to Mr. Tomonori Ohki, Mr. Toshimitsu Aihara, Mr. Hiromoto Yamauchi, Mr. Kazuyuki Oyamada, and Mr. Masahi Tamura, who were colleagues when the author was a member of SHI Accelerator Service, Ltd. Without their support for his shift work, his study would not have materialized by shortness of the development time.

The author's heartfelt appreciation goes to Dr. Yoshihide Higurashi and Dr. Takahide Nakagawa, who provided carefully considered feedback and support for the system implementation in this study.

The author would like to give heartfelt thanks to Mr. Yukito Furukawa at SPring-8 and Dr. Kay Kasemir at SNS, who provided helpful comments and technical suggestions for Web-based systems.

The author would also like to express his gratitude to Dr. Osamu Kamigaito and Dr. Nobuhisa Fukunishi for their moral support.

Finally, the author is deeply grateful to Dr. Kazuro Furukawa and Dr. Noboru Yamamoto, whose comments and suggestions were immeasurably valuable throughout the course of his study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Remote Operation for Large Experimental Physics Facility

Based on an experience with accelerator operation and maintenance, the author felt a deep need for remote operation in a variety of situations. In control system technology, there have been several studies and attempts at implementing such remote operation. Through reference to previous works and his own consideration, the author organized required features in remote operation. Based on the results of a survey, components with features that are essential to remote operation were implemented through in-depth technical considerations.

In large experimental physics facilities, such as accelerator facilities or telescope facilities, control systems are operated using applications such as user interfaces (UIs) from a control room. In general, control rooms are located approximately less than 10 km from control systems; in fact, in some accelerator facilities, the control room is located near the accelerator room. On the other hand, for the maintenance of various equipment, to perform collaborative physics measurements on different machines, and for several other reasons, some users demand the ability to conduct remote operations from distant locations, such as out of state, far from the control room. By utilizing modern control systems with high-performance networks, control system engineers have fulfilled the demands of some of the features of remote operation.

As part of the history of remote operation, previous studies are described in detail in the next section.

1.2 International Linear Collider

The International Linear Collider (ILC), which is the next generation accelerator project, will be launched as a result of the collaborative efforts of multiple institutes from three regions. The ILC required collaborative effort because the size and cost of future large experimental physics projects planned for this collider exceed the resources of a single region. Even if the development of this accelerator will be divided into several parts, and technology development will be pushed forward cooperatively, research organization will bear responsibility for the project, depending on the development.

Once collaboration has begun, all participating organizations, with the exception of the host country, must provide methods for remote maintenance, control, and monitoring of their associated devices, because such will be their responsibility. To realize joint-management of the project from outside the control network, some type of system implementation for remote operation will be required.

For this purpose, implementation of the global accelerator network (GAN) [1] has been planned since March 2000 as a dedicated control system for huge project, such as the ILC control system.

1.2.1 Global Accelerator Network

The idea underlying GAN is the construction of remote accelerator control rooms that, in terms of performance and feature, are virtually equivalent to the local control room, and can be utilized by remote users of the facility. The technical challenge will more than likely be presented by the middle layer software (middleware) that is responsible for the system integration, automation, and presentation of the machine state to the accelerator operator. For GAN study, LabVIEW (produced by National Instruments) was used as an example of the type of high-level graphical programming tool that was available for hardware control and integration at *Deutsches Elektronen-Synchrotron* (German Electron Synchrotron - DESY) in 2001 [2]. To construct a comparatively small-scale control system, LabVIEW is widely used because of its advantage in software usability, and the richness of its supported low-level drivers. In the system employed at DESY, the integration system that uses LabVIEW-based applications can successfully connect to hardware-dependent protocols and to the remote procedure call (RPC) protocol [3].

1.2.2 GANMVL scenario

In May 2005, a system similar to GAN, the Global Accelerator Network Multipurpose Virtual Laboratory (GANMVL), was implemented for the European Design Study towards a Global TeV Linear Collider (EUROTeV) project [4]. GANMVL facilitates remote operation using a client system that utilizes X11 Windowing, virtual network computing (VNC), and JAVA VNC [5]. However, these communication methods can encounter issues such as low bandwidth and high network latency during remote operation over wide area links. Thus, with the limited network resources present in some areas, if the communication methods described in the previous paragraph are used, stable operation and control of the accelerator might prove difficult.

1.3 History of Remote Experiment Systems

The technology for remote monitoring already exists with most modern control systems, and remote operation is generally available with security restrictions from individual institutions. In this section, the author will provide examples of remote operation of accelerator experiments.

1.3.1 Compact Muon Solenoid (CMS) Experiment

Remote operation for this and other experiments that require the Large Hadron Collider (LHC) is achieved through the use of control systems. The researchers in charge of the Compact Muon Solenoid (CMS) experiment built on the LHC located at the European Organization for Nuclear Research (CERN) in Switzerland, and planned to install a virtual control room at the Fermi National Accelerator Laboratory (Fermilab) located just outside Batavia, Illinois [6]. The CMS researchers based at both Fermilab and CERN have been working closely in the areas of data acquisition, triggers, data monitoring, data transfer, software analysis, and database access, in order to commission sub-detectors with the common goal of obtaining efficient and high-quality data for physics study. CMS heavily relies on an online, Web-based monitoring (WBM) system that is composed of the Apache Tomcat [7] Web server and the ROOT [8] data display package. From June through September 2006, the test beam results obtained from the CMS experiment were made available for online monitoring by anyone via the Web and the Internet [9].

1.3.2 A Toroidal LHC ApparatuS (ATLAS) Experiment

For the ATLAS experiment for the LHC, a remote system was implemented, as is the case with the CMS experiment described previously [10]. By using the remote monitoring rooms located at American ATLAS institutions for ATLAS experiment, communication ability among American ATLAS members and scientists at CERN was increased. ATLAS Data Acquisition (DAQ) system can share the information in wide fields, for example of synchronization between processes, error reporting, and operational or physics event monitoring.

1.3.3 Photon Factory (PF) Experiment at KEK

For the Photon Factory (PF) experiment at the High Energy Accelerator Research Organization (KEK) in Japan, a special new beamline equipped with facilities to enable a videoconference system, and in some cases, remote operation, was planned by creating a new "collaboratory" research network [11]. One of the purposes of the collaboratory research network is to establish a system that allows KEK equipment users to collect data from their experiments using remote operation. For this collaborative network, a secure virtual local area network (VLAN) with authentication of public key infrastructures is used to connect the computer network between KEK, Tohoku University, Okazaki National Research Institutes, and Kyoto University [12]. Using Microsoft Windows Remote Assistance, remote users can share graphical UIs (GUIs) for the control system located on site [13].

1.3.4 Remote Experiment System at SPring-8

The wide-area remote experiment system (WRES) was developed for safe experimentation through remote operation at SPring-8, a synchrotron radiation facility located in Japan [14]. Considering issues of safety for remote experiments, WRES must ensure radiation safety, human physical safety, and network access security. To ensure human physical safety during remote operation of the equipment, the physical state of the radiation shield hutch in the experimental equipment is triggered as an "operation-enable signal" of the interlock system. The first experiment that used WRES was successfully performed from the RIKEN Wako campus, which is located 480 km away from SPring-8, in October 2010 [15].

1.4 Motivation

To clarify the features required for remote operation, this study is compared to the systems described in Section 1.2 and 1.3. As a result of such previous works, remote operation needs to implement components with a concept different from that of daily operation. Such components are *network*, *middleware*, *communication*, *man-machine UI*, and *system security*. For implementation of the remote operation in this study, the system design for these components is discussed based on the ideas and reference of previous work. In this section, the components that are required for remote operation are described in more depth.

1.4.1 Network

For implementation of WRES at SPring-8, the Japanese academic Internet backbone with a bandwidth of 40 Gbps was used between the experimental equipment at SPring-8 and remote users at the RIKEN Wako campus. Such Internet bandwidth proved to be sufficient.

As a practical proposition, the GANMVL project and the collaboratory project used the technology of Virtual Private Networks (VPNs) over Wide Area Networks (WANs) to enable the connection between remote users and on-site accelerator control systems, because a dedicated line network would be prohibitively expensive. As a result, VPNs are effectively utilized as standardized application protocols in the remote operation of control systems. Therefore, VPN needs to be used for construction of the system environment in this study as well as other remote operation project. On the other hand, network latency for remote operation from far place, such as between Japan and United States, is a issue to consider in this study.

1.4.2 Middleware

Modern accelerator control systems have been designed with middleware between hardware devices and client systems in order to integrate hardware-dependent protocols as viewed from the client side. It is possible to improve the efficiency of software development by implementing middleware. For example, in the case of the Experimental Physics and Industrial Control System (EPICS – one of the major software systems for the construction of modern control systems), the EPICS input/output controller (IOC), which is a server that functions as middleware, converts hardware-dependent protocols to Channel Access (CA) protocol.

As shown by EPICS, the solution to the technical problem of integrating GAN-like control systems is already available, and can be implemented without much difficulty. For this reason, EPICS is adopted as software for the middleware environment in this study. The middleware features for accelerator control systems are described in Section 2.1.2 in detail.

1.4.3 Human Communication

The sample projects described previously use electronic log notebooks as one of the communication tools employed in the remote operation of control systems. In addition, videoconference systems have been implemented for most remote operation projects in order to facilitate communication between remote users and on-site staff.

In the case of WRES, remote users can view the status of samples on the on-site experiment room using a video-streaming system. As can be intuitively deduced, a communication tool for on-site staff is an essential feature in the remote operation of control systems; therefore, different types of video systems are frequently installed for remote operation projects.

On the other hand, human communication between on-site operators and remote users is also important from a system security point of view because lack of communication can cause accidents. In addition, Voice over Internet Protocol (VoIP) technology, such as Skype [16], may play important role. In general, such VoIP-based communication tools require sufficient network resources.

1.4.4 Man-machine User Interface

For man-machine UIs, any system that interfaces with a GAN project should avoid dependence on specific platforms employed by GAN projects. The design of a user-friendly UI is essential to ensure the smooth operation of a remote accelerator. Considering operator interfaces (OPIs), there are two main types of methods.

In the first method, OPIs are run on the remote client system by sending messages to on-site servers based on dedicated protocols (method A). Such a method is a general desktop application. The second method exports GUIs to the remote computer using systems such as the X Window System, Web technologies, and VNC (method B). In this method, OPI systems use standard protocols to connect the local system (where the collaborating researchers are located) to the remote computer (where the remote accelerator is located), such that the client applications run on the on-site servers. (See Fig. 1.1)

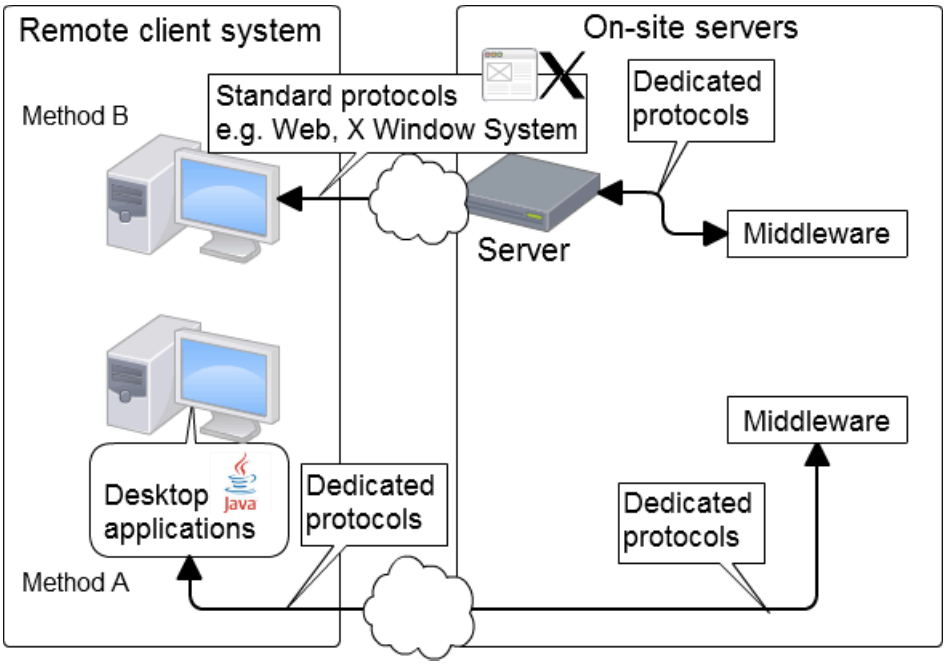

FIGURE 1.1: Comparison of connection methods in client applications during remote operation.

There are arguments in favor of the two methods described in the previous paragraphs, and a mixture of methods is possible. However, in the case of remote operation of control systems, the use of standard protocols has advantages because system-dependent protocols are not necessary between the remote client and the on-site server. In most cases, specialized system-dependent protocols cannot connect through gateways over the Internet.

For the remote operation of GAN projects, the OPI systems are implemented using traditional methods, such as remote desktop technology. However, these traditional systems might not operate well over the Internet with low-bandwidth networks. Because remote desktop technology, such as the X Window System or VNC, is used from the local computer to remote control desktop environments that can include a Window Manager, a large amount of network resources might be required. Consequently, the availability of Web technology for OPIs would be a far more useful method for the remote operation of control systems in low-bandwidth networks, in comparison to remote operation that relies on remote desktop technology.

However, ergonomic interactivity should be ensured for beam operation. In reality, traditional Web applications are not suitable for the operation of the accelerator main parameters, such as Radio Frequency (RF) and magnet power supplies. Thus, WebSocket, a new protocol, is used with EPICS-based control systems in order to utilize Web technology, such as OPIs, in this study.

1.4.5 System Security

For the remote operation of control systems, certain security systems must be considered. From a practical application perspective, the remote operation of an accelerator via WAN is beset by a number of issues. System security, privacy, interruptions, and network policies are important factors that must be considered in any project that requires the remote operation of control systems.

The term "system security" includes computer security, as well as the safety of systems, for example, machine protection, human protection, and human error prevention. Furthermore, communication between on-site operators and remote users is closely related to system safety for the following reasons:

1. An accelerator has both experimental devices and radiation generator characteristics.
2. An operation error during beam tuning can cause the generation of excessive radiation.
3. Radiation generators can legally be operated remotely from an external location; however, there are ethical concerns regarding radiation.
4. Generally, an accelerator control system and a system for radiation safety control are different systems and are separate from each other; therefore, radiation data might not always be shared with the accelerator control system.
5. All information regarding a facility, including radiation information, can be obtained from accelerator operators in their local control room.

In the case of WRES at SPring-8, the physical status of the radiation shield hutch is used as a signal to trigger the interlocking system of the X-ray beam. On the other hand, researchers for the GANMVL project concluded that a safety system was not necessary to monitor their control system via remote access from the outside. Note that a safety mechanism is indispensable when the remote operation of control systems must monitor the parameters of the experiment being performed, and to control the resulting output.

In addition, any errors in the operation of remote control systems could result in immediate equipment breakdown, and might cause serious problems with human protection without enough communication. To prevent operational errors, the accelerator conditions must be completely understood by all project participants, particularly by the on-site operators. Therefore, the author approaches this issue using interposition of on-site accelerator operators, who can confirm safety conditions.

1.5 Tasks of the Study

As shown in Section 1.4, the components to be improved for remote operation are the *man-machine UI, human communication*, and *system security*. In previous studies, remote control systems have been constructed with an environment based on sufficient computer resources. In this thesis, a computer resource means a network-band width and a dedicated client system. On the other hand, this study aims to implement remote operation in environments with limited computer resources. Therefore, VoIP-based communication tools (described in Section 1.4.3) that require more network resources are not covered in this study.

The objective of previous studies and this study for remote operation are summarized in Table 1.1. The tasks for this study include the possibility of controlling a part of the accelerator components and to troubleshoot, even with limited computer resources. In order to manage the tasks, the author believes that use of Web technology is the first candidate.

	For CMS	For ATLAS	Collaboratry	GANMVL	WRES	This Study
Facility	Experiment	Experiment	Experiment	Accelerator	Experiment	Accelerator
System Purpose	Data monitoring, maintenance, and data analysis	Data monitoring and data analysis	Remote Experiment	Remote Accelerator Control Room	Remote Experiment	Troubleshooting and part of accelerator control
Needs of Control Output	No	No	Yes	Yes	Yes	Yes
Required System Resource	Many	Many	Many	Huge	Many	Few
Physical Security Policy	None	None	None	Plan Only	Use of Interlock System for Experimental hutch	Operator Intervention System

TABLE 1.1: To clarify the study's tasks, a comparison between previous studies and this study.

From the perspective of software usability, Web technology will be one of the most useful methods for accelerator control systems. This is because Web-based electronic log notebooks and JAVA VNC, which is run from a Web browser, have already been implemented to improve the operability of previous remote operation projects. If OPIs are coded entirely with HyperText Markup Language (HTML) and JavaScript without the

use of any remote desktop technology, the system that remotely operates the accelerator will also have the advantages of Web technology in the remote operation.

Currently, Asynchronous JavaScript and XML (AJAX) technology is widely used for interactive Web applications, such as Facebook [17] and Google [18]. An AJAX-based OPI has sufficient functions to monitor numerical values using the Internet, and it can be used if fast interactive responses are not required. However, when a project requires remote monitoring of an accelerator in addition to remote operation of said accelerator with a request for output, it is not realistic to replace GUI-based OPI with Web-based systems that use AJAX technology, because AJAX technology lacks interactive performance.

To improve the interactive performance of OPIs using the Web, new technology in accelerator operation must be used to maintain the usability of Web technology. For this reason, and as a next-generation OPI over the Web that uses the EPICS CA protocol, the author proposes a client system based on WebSocket, which is a new protocol provided by the Internet Engineering Task Force (IETF) for Web-based systems. WebSocket is Web technology that provides bi-directional, full-duplex communication channels over a single transmission control protocol (TCP) connection.

Thus, the goals of this study are an examination of the effect of WebSocket-based OPIs and the implementation of WebSocket-based OPIs in accelerator control systems based on EPICS, without losing Web usability and with limited computer resources.

In addition, the security issue must be solved if full-scale use of WebSocket-based OPIs is to become a useful method in the remote operation of control systems. In particular, it is necessary to have exhaustive discussions for system security in remote operation with output controls. In order to implement safe remote operation, smooth communication with on-site accelerator operators and physical intervention are required.

To improve the security of remote operations, on-site accelerator operators must know all the information regarding the devices that are controlled by remote operation. At the same time, on-site accelerator operators must be able to decide the availability of remote operation with output devices. In this study, the security of full-scale remote control of accelerators via the Internet is improved, and the usability of such remote control is evaluated.

1.6 Composition of the Thesis

In this chapter, the history of remote operation in other facilities was described. Then, the background of this study was presented. In addition, the composition of the thesis is presented. In Chapter 2, modern accelerator control systems and the standard model are reviewed as a prerequisite for this study; moreover, the detailed features of the control software are described. In particular, the author further explains the function and benefits of EPICS. In Chapter 3, the detailed effects and advantages of implementing the Web-based system for accelerator operation is described. Because Web systems have many advantages, they have been applied already for various uses in the operation of accelerators. In view of previous work, this chapter also clarifies the impact of Web-based systems on accelerator control systems. In Chapter 4, the detailed architecture of the WebSocket protocol and WebSocket-based system implementation are described. In addition, because the author designed the WebSocket server to interface to the EPICS CA protocol, the system software architecture and the implementation of WebSocket-based OPIs as higher-level application programs for remote operation is described. In this WebSocket-based system, its function and features are also reported. In Chapter 5, system security for remote operation is discussed. For the security system, the software architecture as a technical premise in the EPICS-based system is described. Because physical interposition was necessary as a safety net for remote operation, the required system and the system design are described in detail. In Chapter 6, system implementation of the WebSocket-based OPIs powered by the security system is described. The author implemented both systems for 28 GHz superconducting electron cyclotron resonance ion source (SC-ECRIS) at the RIKEN Radioactive Isotope Beam Factory (RIBF) project. By utilizing a virtual Internet environment using a WAN emulator, this implemented system is tested in a situation similar to remote operation. In Chapter 7, a more general analysis is shown by discussing the methodology of the system design. Several guidelines to design the control system are presented. In Chapter 8, a summary of this thesis is described.

Chapter 2

Modern Accelerator Control System

In the past, systems based on minicomputers were used to manage and operate accelerator control systems. In the last 30 years, Ethernet has evolved greatly and has become extremely popular, not only at home or the office, but also in the industrial field. The standard model, which has a three-layer structure for control systems, is used widely as the hardware and network standard. Once the standard is established, efficient software can be developed. In addition, the operation of particle accelerators requires high-performance electronics for beam diagnostics, data acquisition, and machine control. For all the aforementioned reasons, the following control frameworks were developed based on the standard model: EPICS, Distributed Object Oriented Control System (DOOCS), Message And Database Oriented Control Architecture (MADOCA), and TAco Next Generation Objects (TANGO).

2.1 Standard Model of the Control System

2.1.1 Previous Control System

In general, large accelerator facilities possess long lifespans, and improvements in the performance of accelerators are always demanded, even after accelerators become fully functional. Therefore, control systems must have the flexibility to expand a variety of functions. In the 1980s, the Transposable Ring Intersecting Storage Accelerator in Nippon (TRISTAN) control system, which is an electron-positron colliding beam facility located at KEK, adopted a distributed computer control system using approximately

twenty 16-bit minicomputers [19]. These computers are linked together by optical fiber cables to form an N-to-N, token-passing ring network. (See Fig. 2.1)

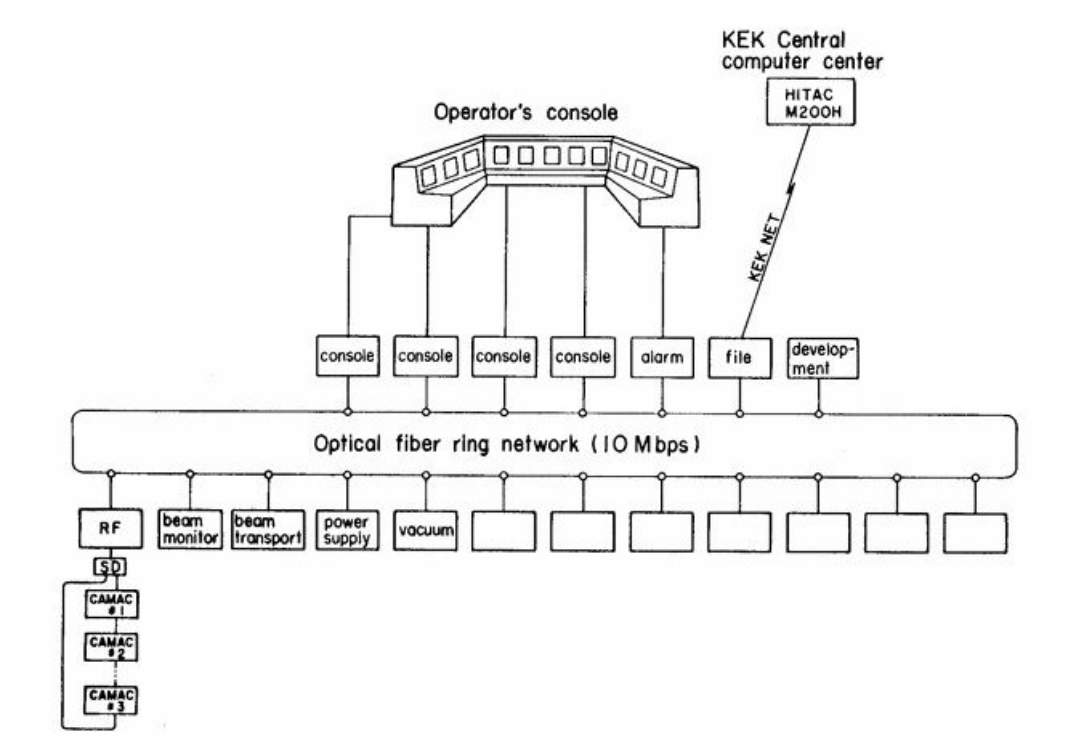

FIGURE 2.1: Schematic layout of the TRISTAN control system. (Ref. Computing in Accelerator Design and Operation Lecture Notes in Physics Volume 215, 1984, pp 367-371, Fig. 1)

Around the same time, the RIKEN Accelerator Research Facility (RARF) control system consisted of three MELCOM 350-60/500s, which is a 32-bit industrial computer made by Mitsubishi Electric Corp. The RARF control system also included Computer Automated Measurement and Control (CAMAC) crate controllers and in-house single board computers [20]. By using optical fibers, a token-ring type computer network that consists of distributed minicomputers was adopted. In the RARF control system, almost all operations are performed using touch panels. The block diagram of the control system is shown in Fig. 2.2.

On the other hand, the control system of the Stanford Linear Collider (SLC) is based on two DEC VAX systems, distributed microprocessor clusters, and CAMAC crate controllers on direct memory access (DMA) that uses a broadband (5—300 MHz) cable TV system [21]. As described in previous examples, minicomputers were the conventional method employed for accelerator control systems 30 years ago; on the other, the unique development of front-end controllers, hardware, and software were also adopted.

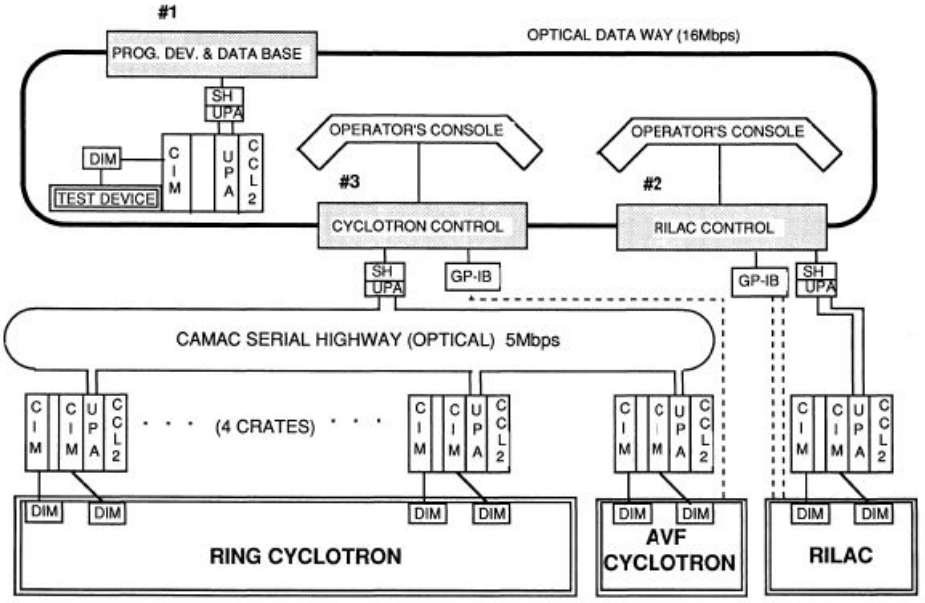

FIGURE 2.2: Block diagram of RARF control system.(Ref. JAPANESE JOURNAL OF APPLIED PHYSICS, vol.30, No.11A, November, 1991, pp.2947–2955, Fig. 2)

In the 1990s, accelerator control systems adopted a combination of standard network (TCP/IP, fiber-distributed data interface, Ethernet) and VME-based front-end controllers because standardized networks and hardware were available. In addition, the X Window System available on UNIX has been widely used as an environment for OPI. On the other hand, the PC-based control system framework with a two-layer structure (one for the human UI and the other for the device interface), called the Component Oriented Accelerator Control Kernel (COACK), was devised by KEK in 1999 [22]. However, the standard model with a three-layer structure is widely used as the modern accelerator control system because of its advantages in software reusability, flexibility, and reliability for distributed control systems.

2.1.2 Developing Control System based on Standard Model

The design concept for control systems was the use of de facto standards such as UNIX, VME, TCP/IP, and the use of optical Ethernet/IP networks for all device controllers without any special fields [21]. The standard model is composed of the following three layers: presentation, equipment control, and device interface. (See Fig. 2.3) At the presentation layer (for example, the man-machine interface), workstations are used to develop and run application programs for accelerator operations. This layer also includes a high-speed reliable network. The Input/Output controllers, which are located

within the equipment control layer, are connected to physical hardware through the device interface layer based on various types of field busses. By establishing the standard model, the research purpose of control systems is shifting from the fundamental control system design to methods that can share the results of software and hardware for the construction of control systems. For this reason, several control system frameworks, such as EPICS, have been developed using the standard model. The goal of the control system frameworks is to provide the accelerator mechanism as a tool with which to control processing with flexibility and higher reliability. The main framework for the construction of accelerator control systems that is currently being used is introduced in the next section.

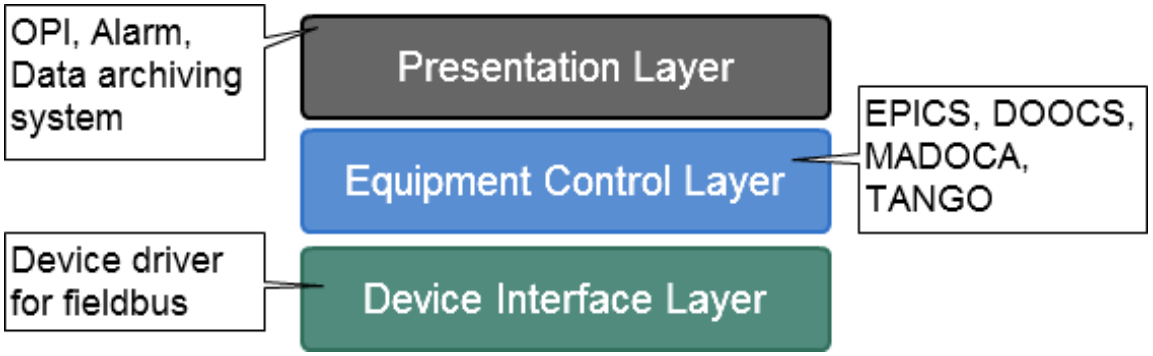

FIGURE 2.3: Outline of standard model for control system.

2.2 Control System Framework

2.2.1 EPICS

EPICS is major control system toolkit that was initially developed by Los Alamos National Laboratory (LANL) and Argonne National Laboratory (ANL) for the construction of control systems [23]. Currently, the development of EPICS is progressing at several research institutions, such as DESY, Oak Ridge National Laboratory (ORNL), and KEK, in addition to ANL and LANL, as an international joint development project called EPICS collaboration. In order to integrate hardware-dependent protocols in the EPICS-based control system, the EPICS Input/Output Controller (IOC), which is a server middleware that functions as the equipment control layer in the standard model, is implemented.

Given that the subject of the research in this thesis is an EPICS-based control system, the EPICS architecture will be described in detail in Section 2.4.

2.2.2 DOOCS

DOOCS is a distributed control system that was developed for the Hadron Electron Ring Facility (HERA) and the Tera Electron Volt Energy Superconducting Liner Collider (TESLA) Test Facility applications at DESY [24]. DOOCS is an object-oriented system designed from the device server for integration of front-end controllers to operator consoles. The Equipment Name Server (ENS) consists of a central database that contains all the names and protocols in the system. In addition, ENS is consulted by client programs before initial data transfer to a device . The DOOCS system architecture is shown in Fig. 2.4. Communication separates client programs completely from device servers.

2.2.3 MADOCA

MADOCA was developed at SPring-8 in 1995, and has been successfully utilized at SPring-8 and other facilities, such as HiSOR, NewSUBARU, and SACLA for the control of accelerator, beam line, and data acquisition system in experiments [25]. The overview of the MADOCA-based control system is shown in Fig. 2.5. The control command to the device from the high-level layer application is utilized by sending a message that is formatted in a human-readable format called SVOC. Currently, MADOCA2, the successor system to the MADOCA system, has been developed [26].

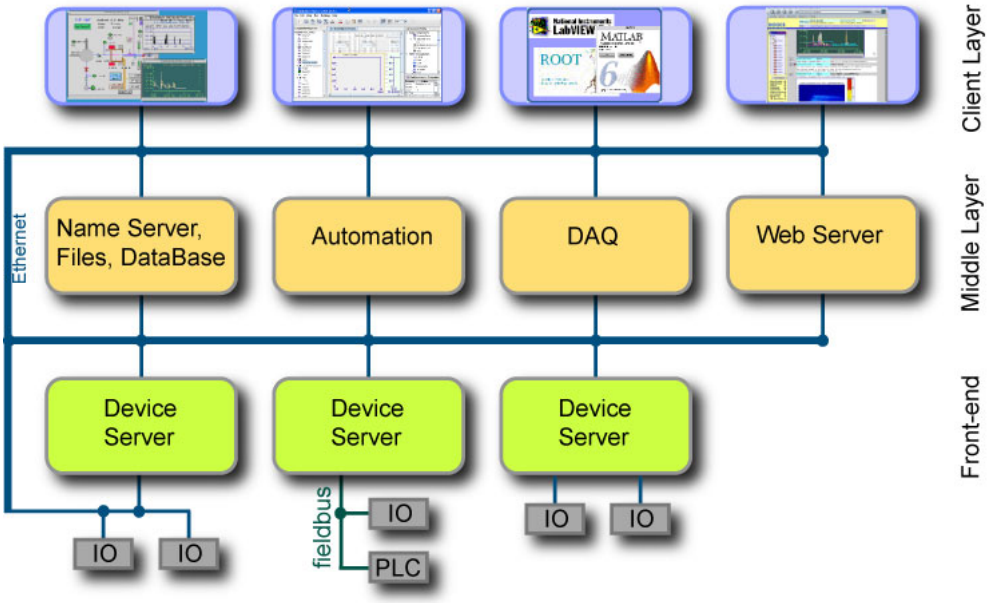

FIGURE 2.4: Outline of the control system using DOOCS.
(Ref. http://tesla.desy.de/doocs)

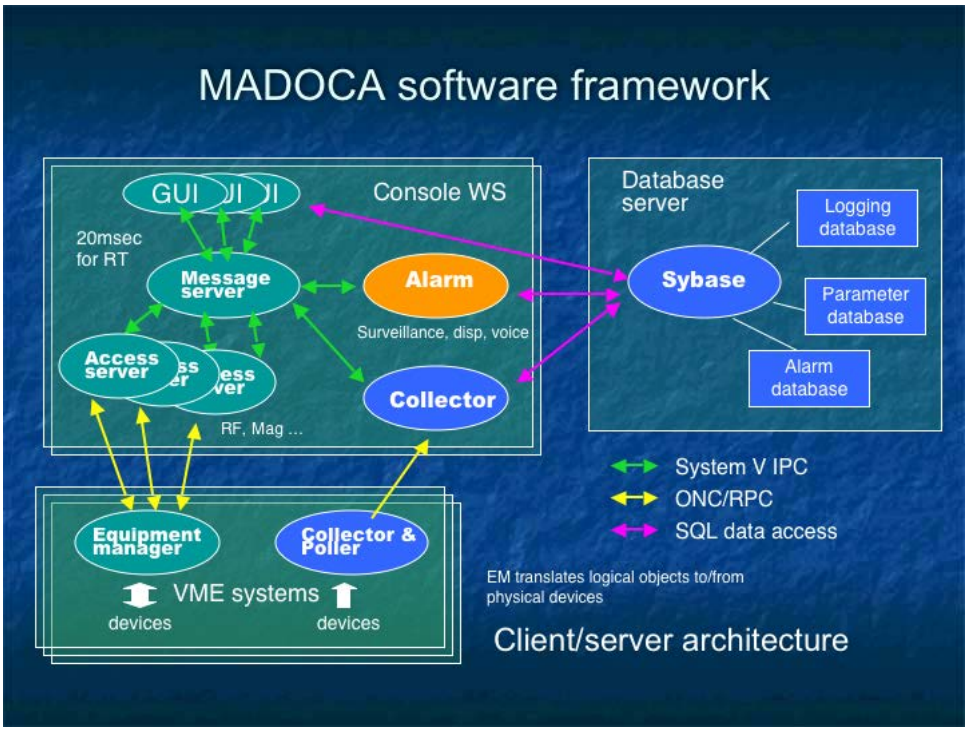

FIGURE 2.5: Overview of the MADOCA-based control system at SPring-8.
(Ref. http://www.spring8.or.jp/ja/facilities/control_system/madoca)

2.2.4 TANGO

TANGO is a control system based on the Common Object Request Broker Architecture (CORBA), and it was developed by the European Synchrotron Radiation Facility (ESRF) and Soleil synchrotrons in France [27, 28]. The TANGO system is used for the construction of a computing tool dedicated to the implementation of distributed, heterogeneous, and control/command oriented systems (See Fig. 2.6). CORBA is a definition for writing object request brokers (ORB). By using TANGO based on CORBA, a given device is connected as an object via a software bus, and the control actions are executed by invoking methods on objects over ORB. In recent years, a feature to bridge EPICS and TANGO has been implemented [29].

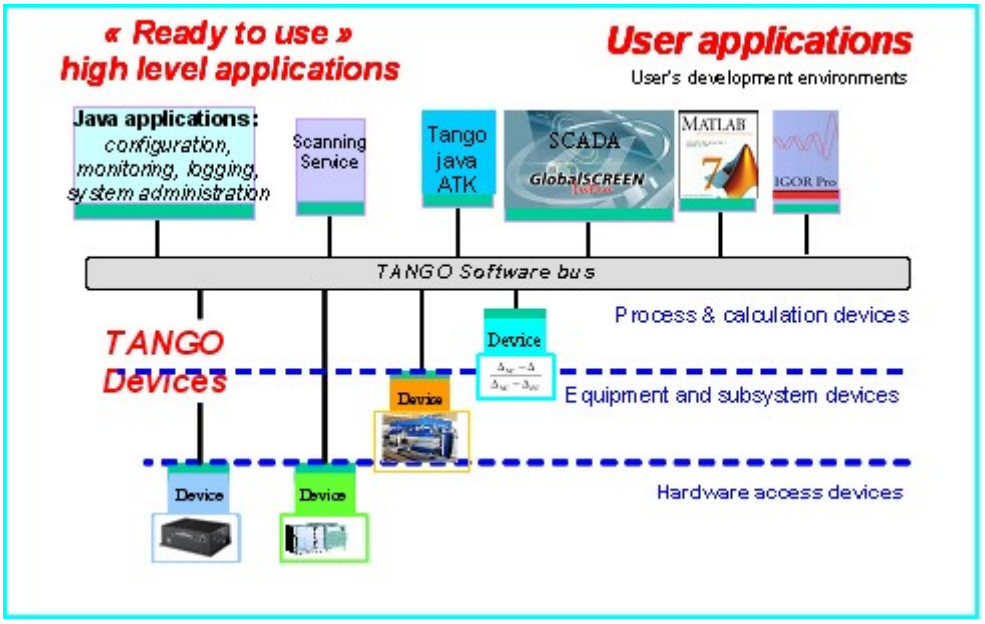

FIGURE 2.6: Overview of TANGO-based control system at Soleil synchrotrons. (Ref. http://www.synchrotron-soleil.fr/)

2.3 Front-end Controller and Device Interface

Computer controls that connect to devices and are located within proximate distance to the device are called front-end controllers. Front-end controllers communicate with control system frameworks via the device interface. Recently, various types of network-based controller that use TCP/IP as the device interface are widely used for control systems [30]. In general, asynchronous access is required for device interface software in this case, because TCP/IP access is a slow line compared with field busses such as VMEbus. For this purpose, the device support software with asynchronous access has been provided in the case of EPICS-based systems [31, 32].

2.3.1 VME

VMEbus is a type of computer architecture. The term was first coined in 1980 by the group of manufacturers who defined it. Data transfer between modules and the CPU occurs via the VMEbus. One of the features of the VME-based controller is acceptable real-time performance. Almost all VME CPU modules support vxWorks as the operating system (OS). On the other hand, Windows and Linux are also available to run on the CPU in several cases, for example, the Intel-based (x86) VME CPU module. In the case of the RIBF control system, VME-based controllers are used to control the power supply in electric magnets [33]. (See Fig. 2.7)

2.3.2 cPCI

Compact Peripheral Component Interconnected (cPCI) is a standard bus specification for next-generation industrial computers that has been standardized by the PCI Industrial Computer Manufacturers Group (PICMG). cPCIs are dedicated for fast communication busses and high-performance applications. For example, cPCI is used successfully for low-level radio frequency control (LLRF) for the Linear Accelerator (LINAC) in the J-PARC Main Ring injector [34, 35] and for the Korea Superconducting Tokamak Advanced Research (KSTAR) control system [36].

2.3.3 PLC

A Programmable Logic Controller (PLC) (Fig. 2.8) is a digital computer used for the automation of electromechanical processes, such as the control of machinery on factory assembly lines, or the elevator. PLCs are used in many industries and machines. Unlike

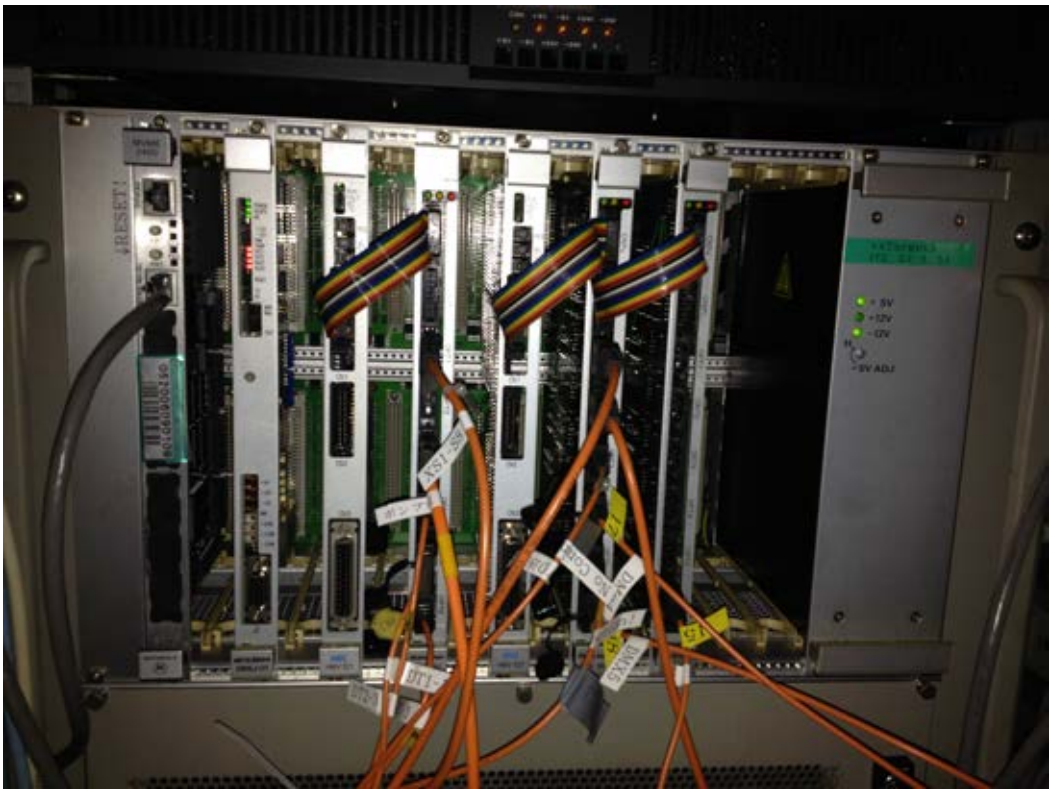

FIGURE 2.7: VME-based front-end controller for magnet power supplies at RIKEN RIBF. The module installed on the left side is VME CPU.

general-purpose computers, PLCs are designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Generally, a development environment called the ladder language is utilized for the development of sequence controls with PLC. PLCs for KEKB, the Japan Proton Accelerator Research Complex (J-PARC), and the RIBF control system, especially the FA-M3 PLC produced by the Yokogawa electronic corporation, are adopted as front-end controllers in many situations [37]. Use of the Embedded EPICS technology, in which EPICS IOC is run inside a PLC CPU module without ladder programming, is widespread in many cases [38–44].

2.3.4 microTCA

Micro Telecommunications Computing Architecture (microTCA) and Advanced TCA provide a means for the telecommunications equipment market to take advantage of standardized systems. Recently, microTCA was adopted by DESY and the SuperKEKB as the next project for the KEKB LLRF control system, as well as for the telecommunications system [45, 46]. In order to achieve higher stability for better beam quality for Super KEKB, fast digital feedback or feed-forward mechanisms are utilized on LLRF

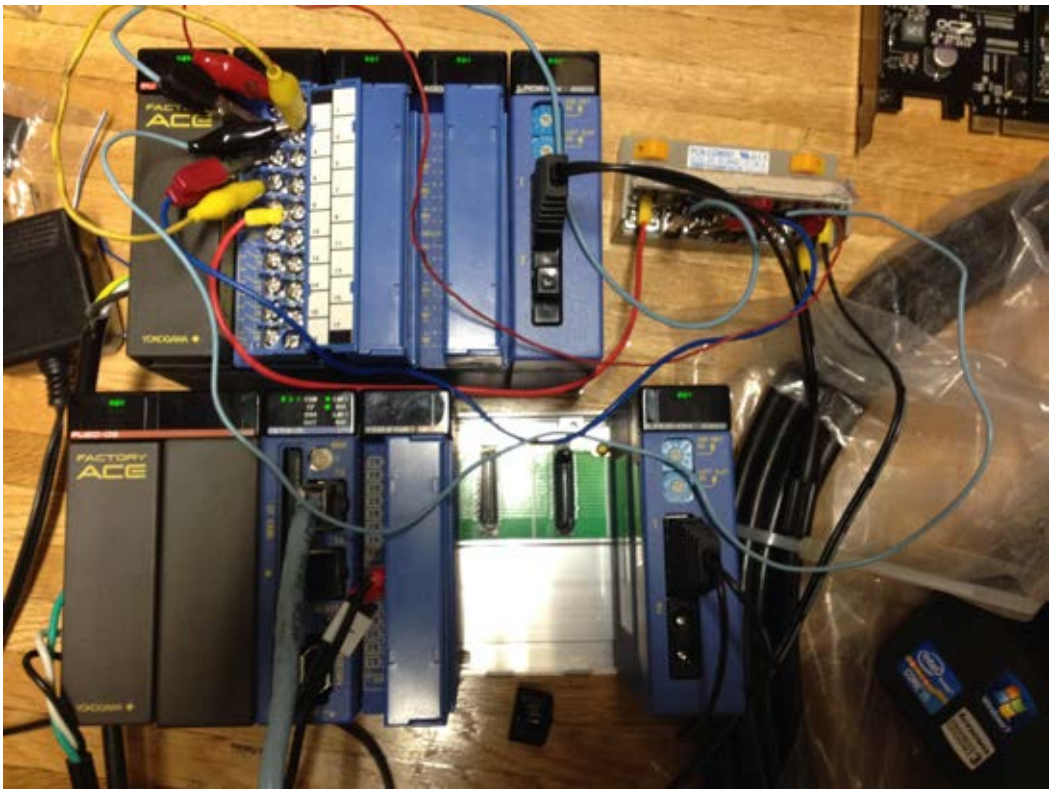

FIGURE 2.8: PLC based on Yokogawa FA-M3 under construction at RIKEN Nishina Center. Two units are connected to each other via FA LINK modules.

through a microTCA module that uses field-programmable gate arrays (FPGAs). There are several advantages for combining microTCA and EPICS. The EPICS IOC can be run inside microTCA platforms as an Embedded EPICS technology; that is, the CA protocol is communicated on the microTCA backplane as a hardware bus. Therefore, system-dependent communication can be managed effectively in a unified manner with the CA protocol [47]. The picture of microTCA, which is installed for the LLRF of the Super KEKB project, is shown in Fig. 2.9.

2.4 EPICS Architecture

EPICS software is an excellent representative of modern accelerator control software. Some of the improvements found in EPICS are described in this work; however, compared to other available control software, EPICS is reliable and powerful, and it provides a significant degree of flexibility and performance. EPICS software construction consists of the following:

EPICS base, a library set for the application programming interface (API) and software building systems for EPICS application.

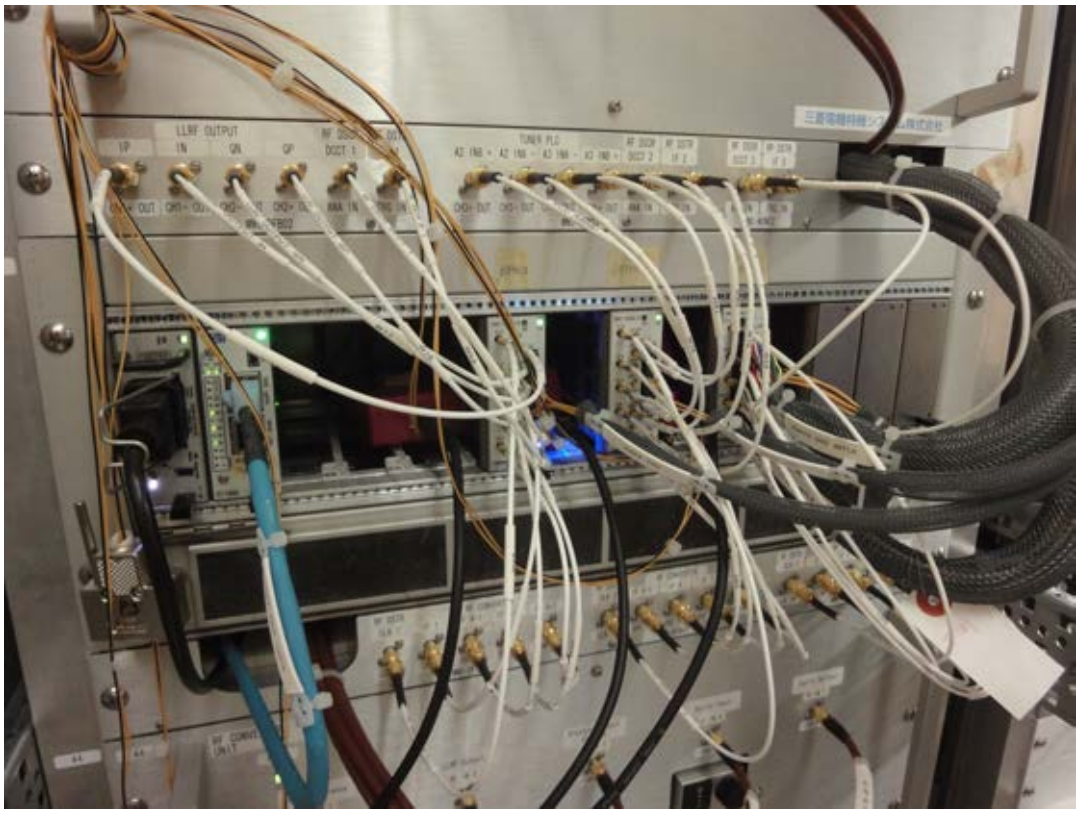

FIGURE 2.9: microTCA-based LLRF controller installed in Super KEKB.

EPICS application, which includes the EPICS runtime database, EPICS IOC, and tools.

EPICS extensions, which are extension applications not included in EPICS base, such as the EPICS sequencer, AutoSave, and the Motif Editor and Display Manager/Extensible Display Manager (MEDM/EDM).

2.4.1 EPICS Runtime Database

EPICS has adopted a client server model. The system configuration diagram for the EPICS IOC, the runtime database, and the CA client is shown in Fig. 2.10. By reading and writing the values in a database called the EPICS Runtime DB through the EPICS IOC, a control that uses EPICS becomes available [48]. In the EPICS system, a reading I/O device is described as "Get", and the action of outputting to I/O devices is described as "Put". Unlike relational databases such as PostgreSQL and MySQL, EPICS DB uses a hierarchical database model. In this model, the type of record is described within the record itself. For example, when a record works simply by reading analog devices, there is a line that indicates "ai record" in the database file. The types of records that are prepared by default in the EPICS system are summarized in Table. 2.1. On the other

hand, when a record has fields, the behavior of the record is defined by coding the fields. The following code is an example of an EPICS database file [49].

EPICS record names	Types
aai	Array Analog Input
aao	Array Analog Output
ai	Analog Input
ao	Analog Output
aSub	Array Subroutine
bi	Binary Input
bo	Binary Output
calc	Calculation
calcout	Calculation Output
compress	Compression
dfanout	Data Fanout
event	Event
fanout	Fanout
histogram	Histogram
longin	Long Input
longout	Long Output
mbbi	Multi-Bit Binary Input
mbbiDirect	Multi-Bit Binary Input Direct
mbbo	Multi-Bit Binary Output
mbboDirect	Multi-Bit Binary Output Direct
permissive	Permissive
sel	Select
seq	Sequence
state	State
stringin	String Input
stringout	String Output
subArray	Sub-Array
sub	Subroutine
waveform	Waveform

TABLE 2.1: Type of records in EPICS runtime database by default.

```
record(ai, "$(user):aiExample")
{
        field(DESC,"Analog input")
        field(INP,"$(user):calcExample.VAL NPP NMS")
        field(EGUF,"10")
        field(EGU,"Counts")
        field(HOPR,"10")
        field(LOPR,"0")
        field(HIHI,"8")
        field(HIGH,"6")
        field(LOW,"4")
        field(LOLO,"2")
        field(HHSV,"MAJOR")
        field(HSV,MINOR")
        field(LSV,"MINOR")
        field(LLSV,"MAJOR")
```

```
17 }
```

LISTING 2.1: Example of ai record coded for EPICS database

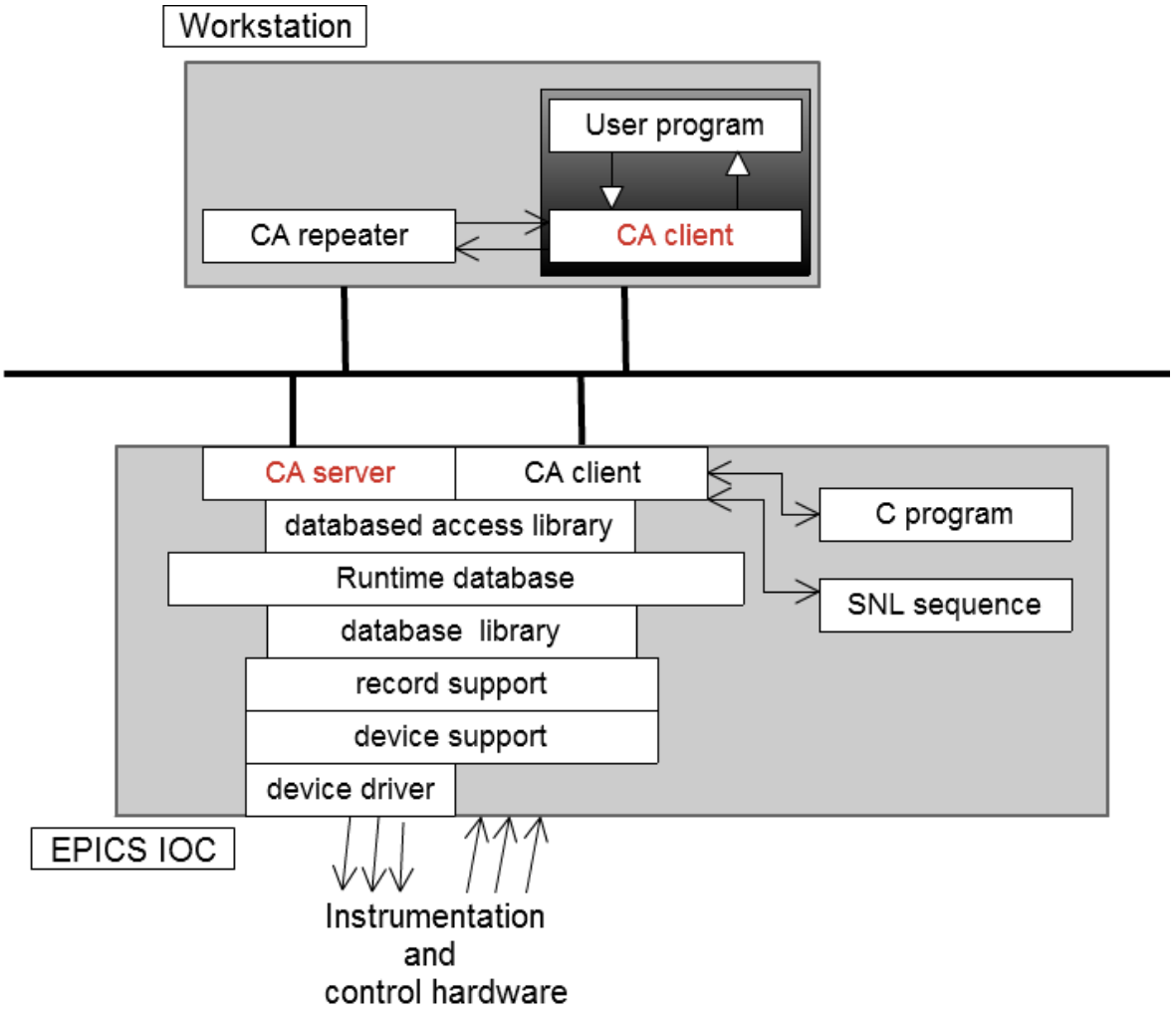

FIGURE 2.10: The system configuration diagram for the EPICS IOC, the runtime database, and CA client

2.4.2 EPICS CA Protocol

The EPICS CA protocol is implemented in order to enable the communication between the EPICS server and an EPICS client [50]. A comparison of an EPICS-based system and a non-EPICS-based system is shown in Fig. 2.11. The CA protocol represents the presentation layer of the standard model. Through software development in the specification of the CA protocol, various types of hardware-dependent protocols can be unified in the client system. The behavior of the CA protocol is shown in Fig. 2.12.

First, the EPICS client searches the records stored in the EPICS server, such as EPICS IOCs, by sending a user datagram protocol (UDP) broadcast to the subnet. Second, the EPICS server replies with a UDP packet to the client if the record is found in the database of the EPICS server. After receiving the UDP packet, the CA client connects to the EPICS server that sent the UDP packet through TCP. A simple example of a CA program, which obtains the numerical value from the EPICS IOC and prints the value, is shown in Listing [49].

```
/*caSimpleExample.c*/
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "cadef.h"
main( int argc, char** argv){
   double data;
   chid mychid;
   if( argc!= 2) {
      fprintf(stderr,"usage:caExample pvname\n");
      exit(1);
   }
   SEVCHK(ca_task_initialize(),"ca_task_initialize");
   SEVCHK(ca_search(argv[1],&mychid),"ca_searchfailure");
   SEVCHK(ca_pend_io(5.0),"ca_pend_io failure");
   SEVCHK(ca_get(DBR_DOUBLE,mychid,(void*)&data),"ca_get failure");
   SEVCHK(ca_pend_io(5.0),"ca_pend_io failure");
   printf("%s %f\n", argv[1],data);
   return(0);
}
```

Listing 2.2: Simple example of CA client program coded in C

2.4.3 OPI as Presentation Layer

OPI is the software interface between a computer and a human user. In general, OPI is a GUI. In the standard model, OPI is implemented in the presentation layer. Given that

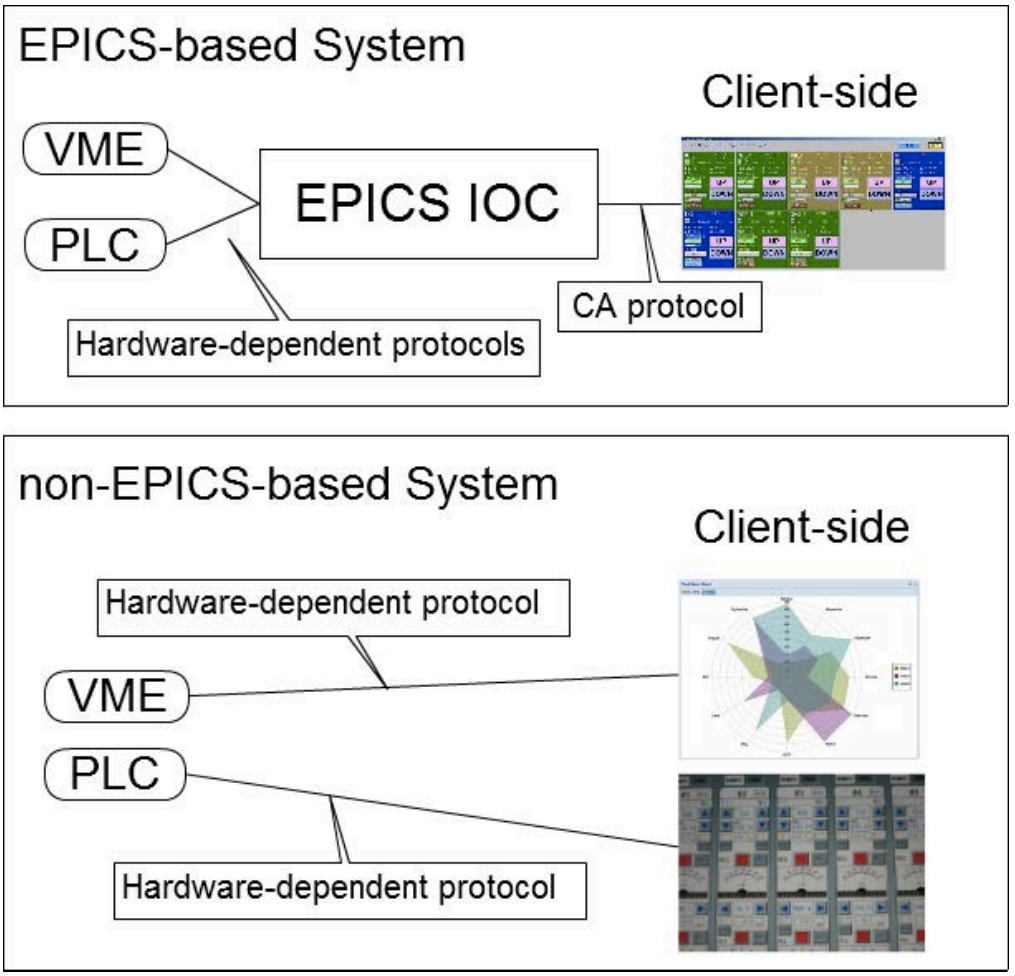

FIGURE 2.11: A comparison of an EPICS-based system and a non-EPICS-based system. The hardware-dependent protocols are converted by CA protocol in EPICS-based systems.

GUI development can be costly, the EPICS collaboration project allows several tools for easy GUI construction .

2.4.3.1 MEDM/EDM

MEDM is a Motif GUI used for designing and implementing control screens [51]. EDM [52], which is another tool for MEDM, also provides the ability to create and edit display content in the EPICS-based system. All GUI components developed in MEDM/EDM are displayed on the X Window server. An example of an EDM screen is shown in Fig. 2.13. Just as drawing a picture is possible using MEDM/EDM, the GUI can be constructed by arranging GUI components. Currently, the Control System Studio/Best OPI Yet (CSS/BOY) described in the next section is becoming a standard GUI building tool for EPICS.

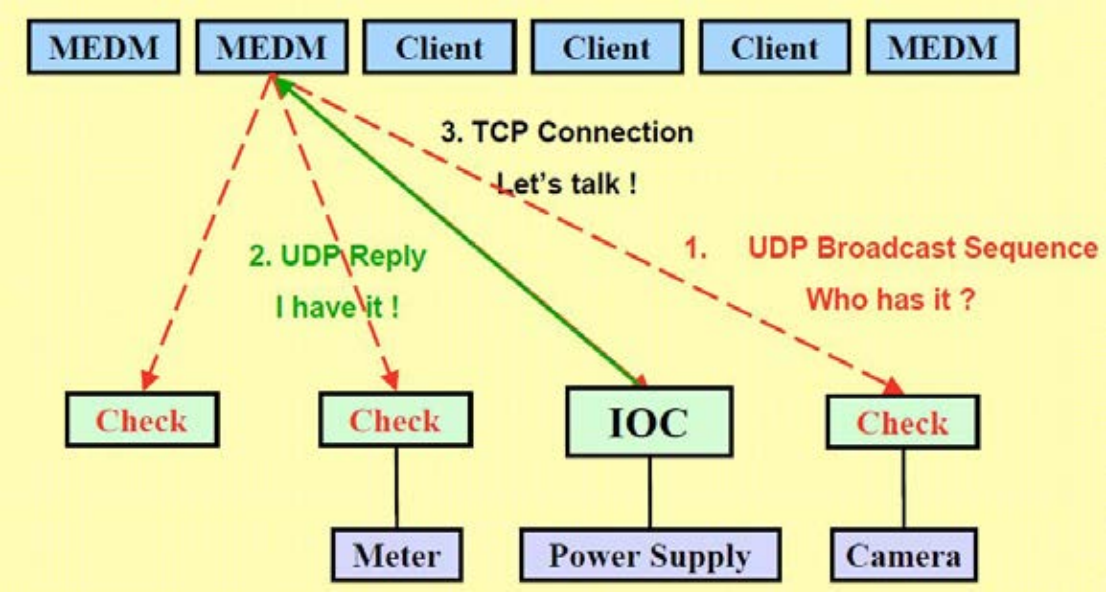

FIGURE 2.12: Behavior of the CA protocol. (Ref. Part of the EPICS "Getting Started" Lecture Series in EPICS Workshop 2006, VECC, India. Original slide is "Introduction to Channel Access Clients" by Kenneth Evans, Jr. Semptember 16, 2004)

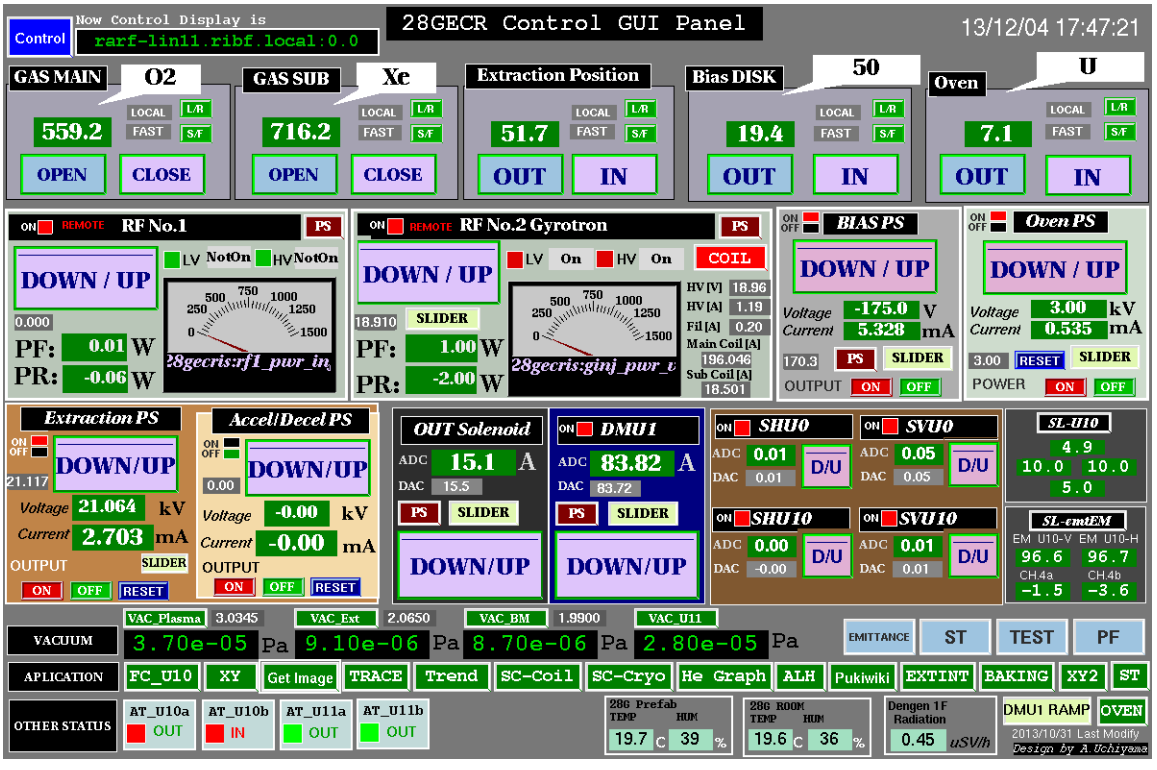

FIGURE 2.13: GUI using EDM for RIKEN 28 GHz super conducting ECR ion source control in RIBF.

2.4.3.2 CSS/BOY

In order to integrate the development environment, CSS, which is an Eclipse-based collection of tools that use JAVA, is provided by the EPICS collaboration project [53]. On the other hand, BOY is implemented as one of the tools included in the CSS for the construction of OPI [54]. An example of a CSS/BOY screen is shown in Fig. 2.14. WebOPI (to be described in a later chapter) is also one of the features found in the CSS.

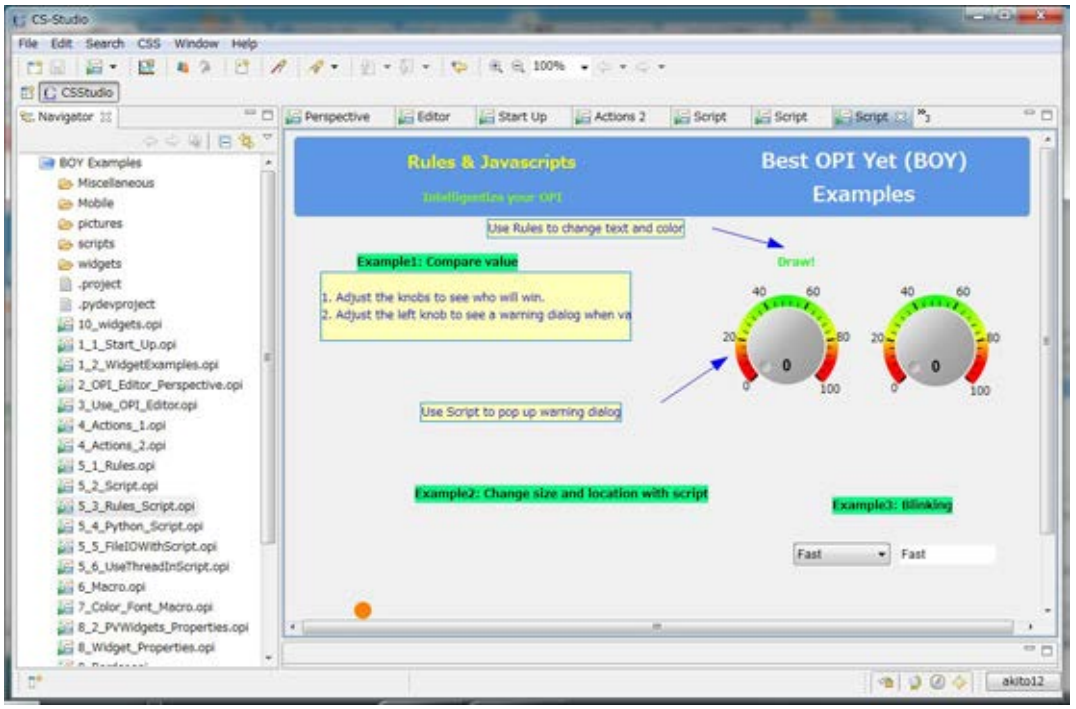

FIGURE 2.14: Example of a CSS/BOY screen.

2.4.3.3 caQtDM

Instead of the traditional MEDM/EDM based on Motif, a similar function is implemented by caQtDM using Qt [54]. Because Qt is an application UI framework that is coded in C++, it can be used for developing modern GUIs, such as desktop environments for major Linux distributions.

2.4.4 Channel Archiver

Data storage is a primary issue in any research facility. Data archiving means moving data that is no longer actively used to a separate data storage device for long-term accelerator operation. In the standard model, the archive engine is implemented on the presentation layer, as well as OPI. For data archiving in EPICS, the Channel Archiver

can be used as the standard software [55]. On the other hand, the data archiving function that uses a relational database, such as Oracle database, is implemented in CSS. Moreover, several data archiving systems developed in-house can also be utilized by each facility [56–58]. For example, the proprietary Channel Archiver that uses hadoop, which is a distributed framework, is developed in-house at J-PARC as a method for effectively managing big data.

2.4.5 Alarm

The Alarm Handler is provided as standard OPI by EPICS collaboration. By setting the upper and lower limit in EPICS records, the Alarm Handler sounds an alarm for the accelerator operator when the set value is over an actual value. In addition to the Channel Archiver, several in-house alarm systems for EPICS-based control systems are implemented by each facility [59].

Chapter 3

Web-based Systems for Accelerator Operation

Currently, the Web is widely used as the most common Internet protocol. Web services that provide a UI component for accelerator control systems already have been utilized for various purposes because such services have the advantages of being platform-independent systems, providing software maintenance, and can be developed rapidly . In previous works, Web technology has been utilized to view archived data as an electronic log notebook, and as a monitoring system without fast response. Further, Web technology is useful as a means of widely informing about the status of accelerators and beams. For this purpose, WebOPI was implemented by Spartan Neutron Source (SNS) as a Web-based system that uses AJAX with EPICS. By improving Web interactive performance compared to AJAX, a wide variety of applications, such as those used for troubleshooting and communication, could become available for the remote operation of accelerators because of their advantages. The effect of utilizing Web-based applications in accelerator operations is described in this chapter.

3.1 Advantage of Web-based System

Accelerator control systems that use Web services were already devised in 1995 during development of Internet technologies [60]. Web-based systems have many advantages, especially in non-large-scale systems. When computers connect to the Internet, such computers become available to any type of information via the Web HTTP is widely used as the standard protocol through gateways with a firewall built into the network. In contrast, dedicated protocols cannot easily reach internal hosts because such protocols cannot access the gateway. In addition, most UI requirements might be satisfied

using "Web service" techniques owing to the rapid development of the Internet and other information networks. Compared to custom-built applications developed using GUI building toolkits such as JAVA Swing, Web applications have advantages for the following reasons: platform-independence, rapid system development, and ease of system maintenance.

In traditional systems, the client system is dependent on UNIX and Linux; such systems also have X Window System servers. In addition, where Microsoft Windows is the platform, LabVIEW, Visual C, and Visual Basic have been adopted in accelerator control systems. Then, because of the advent of JAVA, which can run on Mac OS X, Windows, and Linux, platform-dependency has been resolved to a certain extent. However, JAVA has not solved completely the issue of platform-independent systems. In the development of mobile devices that utilize Android and iOS as the OS, the need to resolve platform-dependency has increasingly become an important issue in recent years. These devices have provided us with many candidates for an operator console, as well as different styles for the operation of accelerators and for physical experiments. Using a mobile device, accelerator operators can also access other equipment, and can easily troubleshoot the devices, if required.

JavaScript and HTML provide a simplified, rapid method for creating dynamic Web content, such as AJAX technology. On the other hand, building GUIs, which are not an essential feature of applications, incurs costs in many cases of desktop applications that use common programing languages, such as JAVA, C, and C++. For simple and efficient development of GUI applications, a combination of HTML and JavaScript enables rapid development of Web-based applications that are server and platform-independent.

In order to update to the latest version of an application, it is costly and inefficient to provide a method where a desktop application is required to be installed on many PCs. From the perspective of software maintenance, the implementation of a Web application for control systems might result in significantly reduced costs.

In addition, Web-based remote access has possibilities for saving the bandwidth of network resources compared with traditional technologies, such as VNC, remote desktop, and the X Window System. Based on these advantages, there have been many attempts to introduce Web-based applications for accelerator and experimental control.

3.2 Implementation of Web-based System for Static Access

From the 1990s to around 2007, almost all Web-based systems have been used for static access in control systems, such as applications for archive viewers and log notebooks, where fast response is not required. For comparison, in this section, the author introduces previous studies that are implementations of Web-based systems for static access.

3.2.1 Hardware Maintenance for GAN

In the GAN project, Web services were discussed to be used for diagnostic access [2]. According to the article, hardware maintenance personnel often have many different requirements from machine operators; consequently, the Web server interface must allow the former to perform much of their own software development and maintenance work.

3.2.2 Video Sharing System at KEKB

By using Comet technology, the KEKB control group has constructed a sharing system in order to view GUI panels and video images using a Web browser [61]. Comet is a Web application model in which a long-held HTTP request allows a Web server to push data to a browser. The main feature is to deliver the GUIs, which appear on the X Window System, to a large number of users using Web technology. By converting the GUI panel to an image file and providing the image file periodically, the system realizes a display similar to an interactive action on the Web browser.

3.2.3 MyDAQ2 at SPring-8

At JASRI/SPring-8, a simple data logging and display system, called MyDAQ2, was developed utilizing a MySQL-based relational database and the Apache HTTP server [62]. For this system, the logged data is displayed and plotted as charts using gnuplot, and provided by a Comma-Separated Value (CSV) file via a Web browser (See Fig. 3.1).

3.2.4 Electric Log Notebook

Instead of recording information on paper, an electronic log notebook is widely used as a Web service in accelerator operations [63–66]. Using a Web application for the

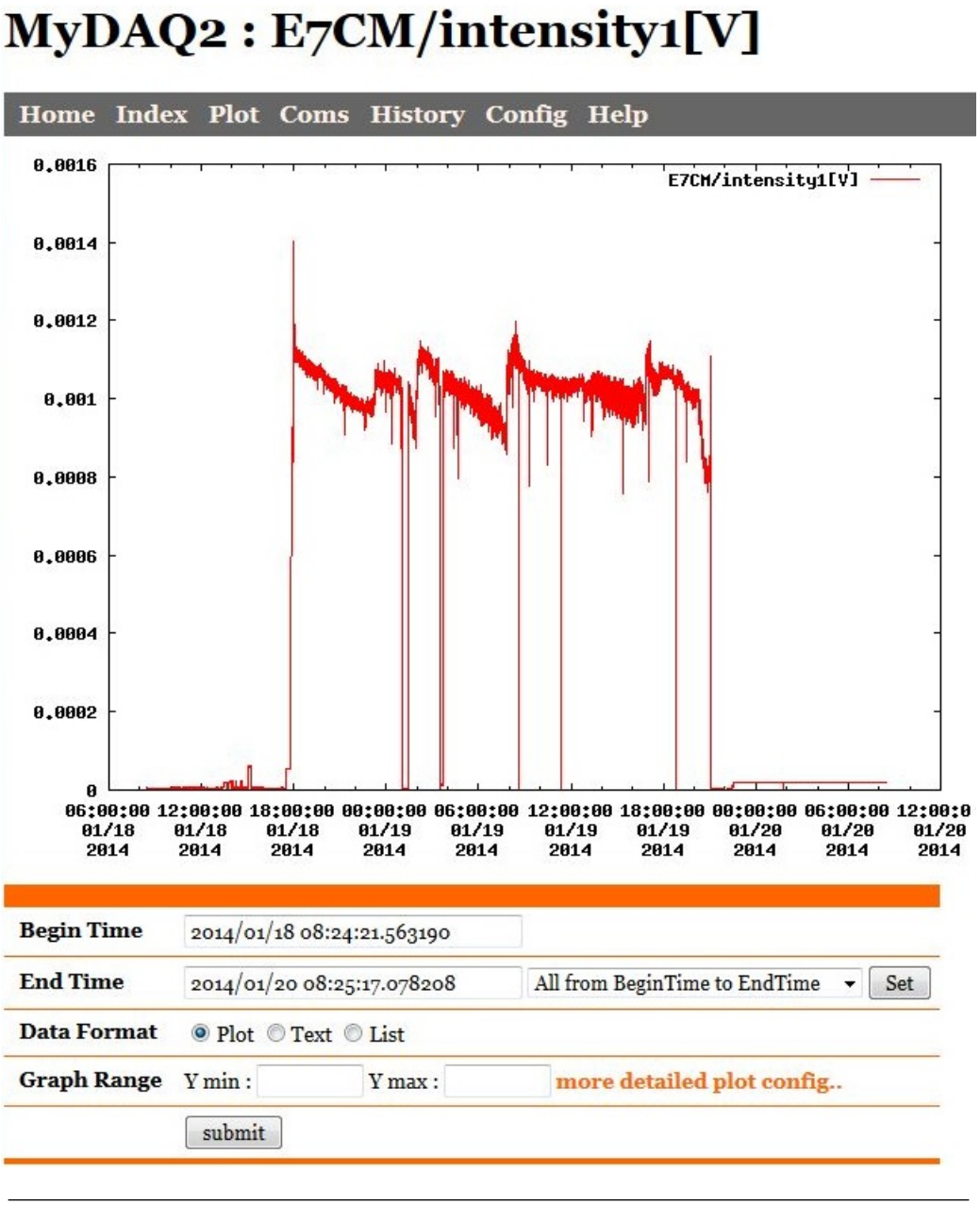

FIGURE 3.1: Screenshot of MyDAQ2.

electronic log notebook is a suitable method, considering their increased flexibility and lower maintenance costs when compared to desktop application electronic notebooks. The screenshot of the Zope-based electronic log notebook (Zlog), which has been adopted for KEKB, J-PARC, and RIBF, is shown in Fig. 3.2.

3.2.5 EPICS Channel Archive Viewer

For the KEKB injector linear particle accelerator (LINAC), a Web-based data viewer that displays and plots data on a chart has been implemented and developed using the

FIGURE 3.2: Screenshot of Zope-based electric log notebook (Zlog).

JAVA Swing GUI library and a commercial charting library [67].

3.2.6 NAGIOS Alarm

At the Argonne National Laboratory's Advanced Photon Source, an Infrastructure Monitoring System (IMS) has been developed to provide immediate notification to the Control Group on-call staff in the event of failure of critical controls hardware components or software processes [68]. In this system, NAGIOS , an open-source host, service, and network monitoring program, has been interfaced to EPICS CA monitoring tools to provide historical tracking of control infrastructure events, email, and pager notifications to on-call staff, as well as Internet-accessible status Web displays.

3.3 Implementation of Web-based System for Dynamic Access

After work on Web-based systems with static access, Web-based systems have been used for monitoring the system of accelerator parameters with dynamic access such as OPI in accelerator control systems. In this section, the implementation of Web-based systems with dynamic access by previous studies is discussed because this study also aims to improve the interactive performance of Web-based EPICS OPI for use with remote operation.

3.3.1 CAML and WebCA

As is clear from previous sections, Web applications have been used conventionally for static work only, such as viewer applications for archived data and electronic notebooks. On the other hand, the use of Web applications for static work, in addition to OPIs with dynamic access, has been implemented since 2008 [69]. In the case of EPICS-based distributed control systems, SNS has planned and implemented the CA Markup Language (CAML) to call the CA API from an XML file. The behavior of the CA connection in CAML is based on WebCA, which is the Firefox/Safari plug-in module developed by Cosylab. Therefore, a Web browser with the WebCA plugin connects to EPICS IOC using CA directly. The behavior of WebCA, CAML, and a Web browser is shown in Fig. 3.3, and the screenshot of the CAML-based client system is shown in Fig. 3.4.

The main feature of CAML is that the Web browser connects to the CA server directly via the WebCA plug-in module. The values obtained from the CA server can be managed in the XML format, and a JavaScript chart library is used for plotting the data.

In CAML-based systems, it is possible to display the GUI using a Web browser. However, there is no use for an HTTP method between the web browser and the CA server, and the CA protocol is used as dedicated protocol. Though a Web browser is utilized to display OPIs, it is not a suitable method for use with remote operation because this system is similar to a general desktop application as described in Section 1.4.5. Given that CAML utilizes the function of the CA protocol in the layer between the Web browser and EPICS IOC, the sending of a request might not be reached in the case of remote operation via a WAN. In addition, CAML depends on the type of Web browser because it is necessary to use the plug-in for Firefox/Safari.

3.3.2 Web Services Interface to EPICS CA

The standard model of modern accelerator control systems consists of three layers: the presentation layer, equipment control layer, and device interface layer. In previous studies [70], a presentation layer on the standard model that is a multi-purpose Web interface layer has been proposed for the development of higher EPICS CA application using cross-platform and cross-development methods.

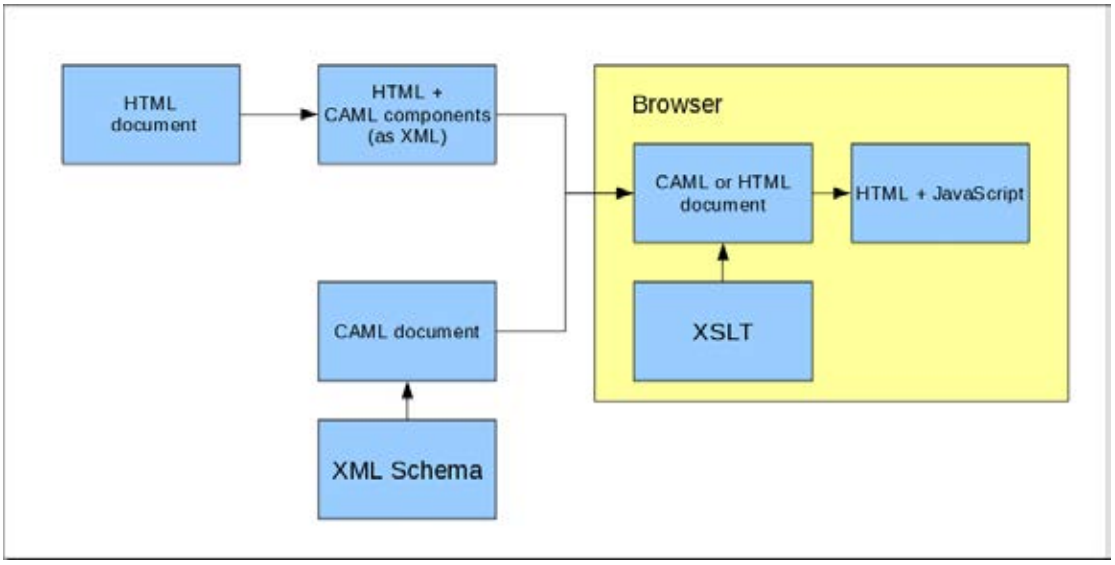

FIGURE 3.3: The system chart of CAML.

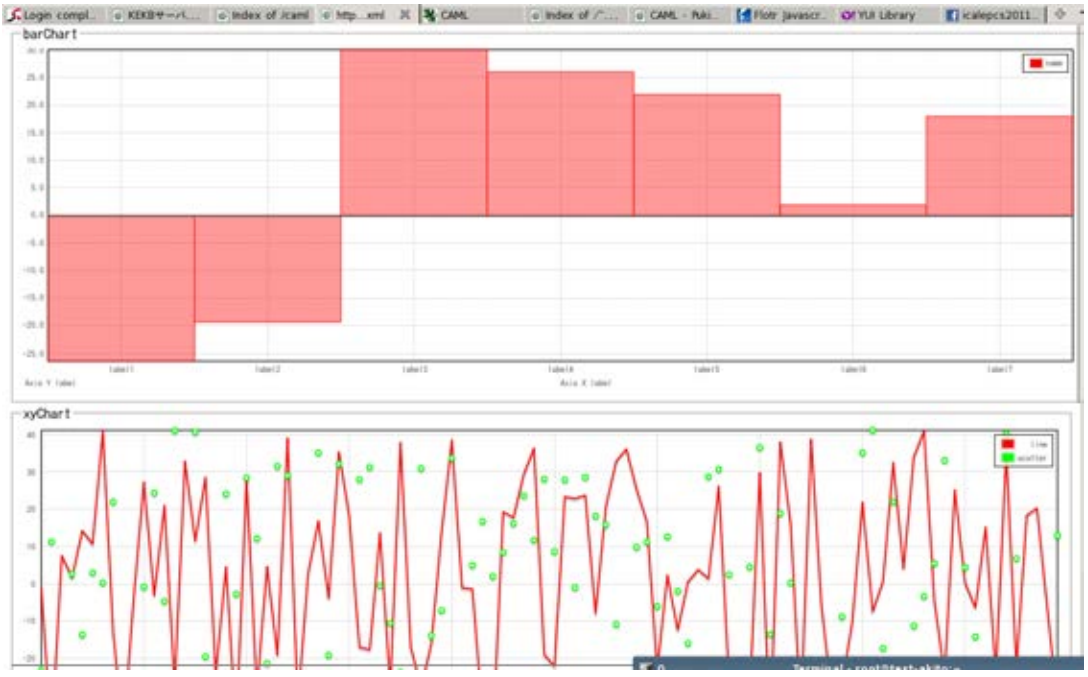

FIGURE 3.4: The example of CAML-based GUI panel displayed on Web browser.

3.3.3 Flex-based Web Interface

At Beijing Electron-Positron Collider II (BAPCII), EPICS data that includes beam energy, beam current, lifetime, and luminosity are displayed in a Web-based real-time chart developed by Adobe Flex [71]. This Flex-based system is utilized by a Web server installed CAJ, which is a module to interface to JAVA and EPICS CA. On the other hand, a data acquisition method in this system utilizes polling access to Web servers.

3.3.4 WebOPI

At the International Conference on Accelerator and Large Experimental Physics (ICALEPCS) 2011 in Grenoble, France, the SNS engineers published WebOPI as one of the features for CSS [72]. WWebOPI aims to provide Web access to EPICS-based OPIs that were created in CSS/BOY, without modification. In WebOPI, the function for displaying a numerical value on the Web in real-time is implemented using AJAX technology. To publicize information via the Internet, it is only necessary to install Tomcat as the Web server with JAVA servlet container, and it is not necessary to modify the OPI file created by CSS/BOY. The screenshot of WebOPI is shown in Fig. 3.5.

3.4 Communication in Remote Operation

It is often necessary to control an accelerator from various locations, in addition to the central control room, during beam operation and maintenance. On the other hand, accelerator control systems have many types of users: accelerator operators, accelerator engineers, accelerator physicists, and experiment users. During experiments using the accelerator, many users will need to share standardized displays and be able to communicate with each other.

For communication between on-site operators and remote users, it is also necessary to implement a remote control system that comprises collaboration technologies, such as text-based chat tools, videoconference tools, and electronic log notebooks. Various types of communication tools have been developed already, and electronic notebooks are widely used at standard accelerator control systems.

In the case of GANMVL, video/audio communication tools might be invaluable in allowing experts to work together on accelerator operation at the global level. This is because such communication tools help mitigate communication apprehension caused by differences in the native languages of local staff and some of the global experts. According to this model, the expert staff from each laboratory remains based at their home

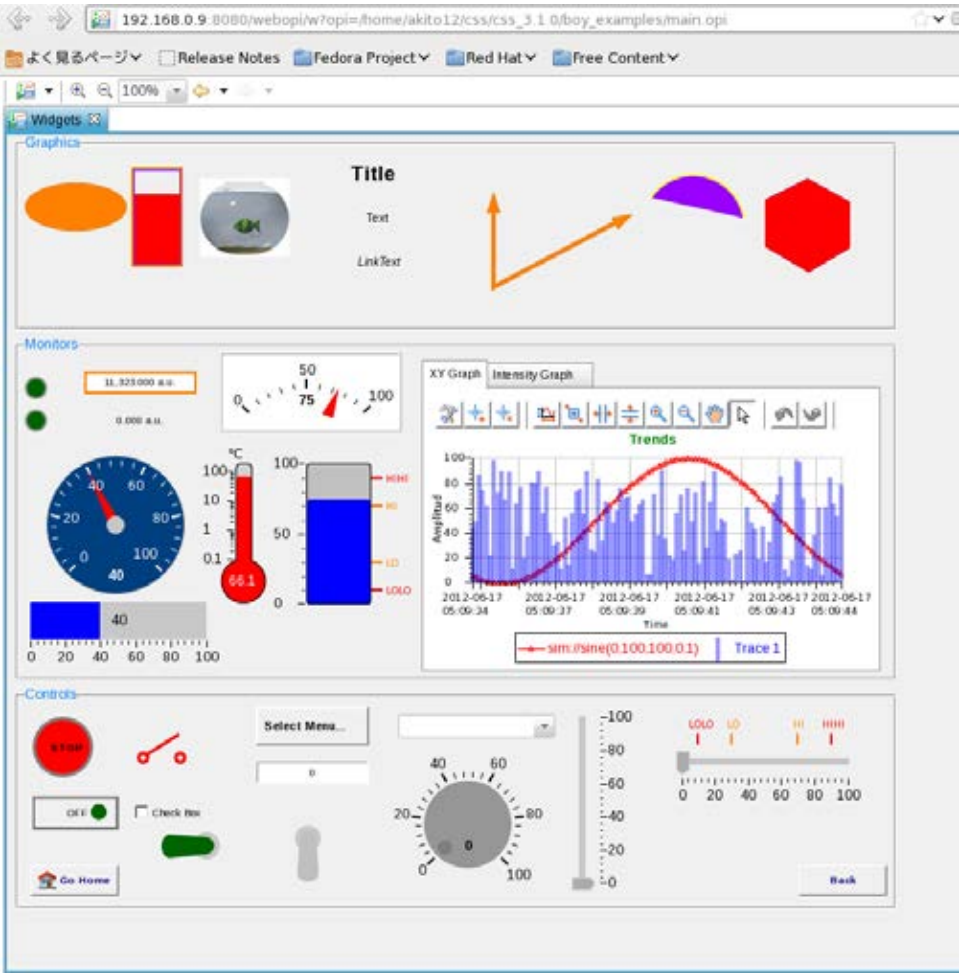

FIGURE 3.5: The example of WebOPI-based GUI panel displayed on Web browser.

institution, and continues to participate in the machine operation after construction. For this reason, there is significant utility in implementing such a feature as communication tools using the Web.

3.5 Remote Operation for Troubleshooting

In general, an accelerator facility consists of many components, such as RF, the magnet's power supply, vacuum, beam diagnostic instruments, water cooling, and the control system. Given that the staff/developers of the accelerator components have a responsibility for resolving machine problems, such personnel should verify conditions as soon as possible if there is a warning sign.

In some rare situations, such as accelerator problems and change of beam conditions (beam emittance, etc.), on-site accelerator operators might require seeking direction from other engineers or scientists over the telephone when said engineers or scientists are

away from the laboratory. In such scenarios, the ability to provide the exact directions required by accelerator operators is contingent on the amount of information required for remote troubleshooting. However, even in cases where the native languages are not in conflict, it is difficult to correctly provide detailed information about accelerator conditions via telephone and voice communication, because insufficient information is obtained by listening compared to observing. For these reasons, there is a significant need to monitor and maintain devices from anywhere via the Internet and the Web.

3.6 Cost-effective Accelerator Operation

Future accelerator projects will grow larger, resulting in higher electricity costs. Even if there is only one problem with the components, it might not be possible to provide beam service to experiment users. In Japan, which is one of the sites proposed for ILC, electricity costs are increasing because of the Fukushima nuclear accident that occurred after the Great East Japan Earthquake of 2011. In order to save electricity costs, possible errors caused by the operation of remote users must be prevented during troubleshooting tasks performed via remote operation.

For such cases, it is important to reduce beam tuning time and accelerator standby time to the maximum. Therefore, in this situation, communication between on-site operators and remote users is considered extremely important, and all information about the activities that occur in the local control room should be communicated to remote users.

As described in the previous sections, Web technology is used for applications with dynamic access such as OPI; however, interactive performance is not sufficient. In addition, given that general accelerator control systems are implemented with dedicated local networks, the control systems must also improve the safety of remote operation and network security when remote users gain access from the Internet.

Chapter 4

WebSocket Interface for EPICS

During beam operation and maintenance, it is often necessary to control accelerators from different locations and from the central control room. However, it is not practical to replace the GUI-based OPI with a Web-based system using AJAX technology because of interactive performance issues. Therefore, as a next-generation OPI over the Web using the EPICS CA protocol, the author developed a client system based on WebSocket, which is a new protocol provided by IETF for Web-based systems. WebSocket is a web technology that provides bidirectional, full-duplex communication channels over a single TCP connection. By utilizing Node.js and the WebSocket access library called Socket.IO, WebSocket servers have been implemented for use in accelerator control systems. Node.js is a server-side JavaScript language built on the Google V8 JavaScript Engine. In order to construct a WebSocket server as an EPICS CA client, an add-on for Node.js was developed by the author in C/C++ using the EPICS CA library, which is included in the EPICS base. As a result, for accelerator operation, Web-based client systems have become available with various types of equipment as well as in central control rooms.

4.1 Development Background

One of the advantages of an EPICS-based system is the unified communication protocol between a front-end controller and the client systems provided by the CA protocol. Thus, even when various types of controllers are used as control systems, all client systems, such as OPIs, can be developed based on the CA protocol without hardware dependencies. In general, a method that uses the CA libraries or the display manager supported by EPICS collaboration, such as MEDM/EDM (see Section 2.4.3.1) and CSS/BOY (see Section 2.4.3.2) has been adopted for GUI development in the EPICS-based system. On the other hand, the engineers at SPring-8 proposed development methods for the

main OPI using WebSocket and implemented a prototype [73]. The prototype system was constructed using a MADOCA control system framework that was developed at SPring-8.

As described in Chapter 3, Web-based systems have many advantages for accelerator control systems. By utilizing WebSocket technology, it is possible to solve the issues of Web-based systems, such as those described in Section 3.4, because of dramatic improvements in interactive performance compared with AJAX-based Web applications. That is, for OPIs, the communication method, and the troubleshooting method, vast improvements are possible through the Web when remotely operating accelerator equipment. For the EPICS-based system, traditional Web applications lack the interactivity required to operate some of the hardware for accelerator control systems.

However, in the case of SPring-8, the control system framework is MADOCA, which is described in Section 2.2.3. Given that real-time Web applications have many advantages, the author developed a corresponding WebSocket server for the EPICS CA, and implemented Web applications using WebSocket for the EPICS-based control system.

4.2 WebSocket Architecture

WebSocket is a new protocol for achieving bidirectional communication between a WebSocket server and a Web browser. Originally, WebSocket was part of HTML5. Through draft versioning, WebSocket was formulated as an RFC6455 by ITEF in December 2011 [74]. WebSocket can be used for real-time applications because of low protocol overhead. The WebSocket API specifies how to create and access WebSocket within a Web browser. A WebSocket connection is established via JavaScript, and then, it becomes available for sending/receiving messages and executing error handling through a connection.

4.2.1 Overview of the Protocol

The WebSocket protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request. An overview of the communication for the WebSocket protocol is shown in Fig. 4.1. The WebSocket specification is as follows:

1. The initial handshake starts at the HTTP level. A Web browser sends a handshake request to the HTTP server to connect to WebSocket. The following is an example of a handshake request from the client [74].

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

2. The server sends a handshake response after approving the request from the Web browser. The following is an example of a handshake response from the server.

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

3. After establishing the handshake, the protocol switches to WebSocket, and then a bidirectional communication occurs between the WebSocket server and the Web browser.

A more detailed view of the WebSocket frame structure is provided in Fig. 4.2. As shown in the figure, WebSocket allows interactive responses, thus compensating for disadvantages that cannot be eliminated with traditional HTML.

4.2.2 WebSocket API

To establish a WebSocket connection with a Web server, it is necessary to send data from the Web browser using the *send( )* method to the client-side JavaScript. The following line of code is the API that creates a new WebSocket object.

```
var Socket = new WebSocket(url, [protocol] );
```

where the first argument, url, specifies the uniform source locator (URL) to which to connect. In general URL naming convention, the WebSocket protocol defines the *ws://* and the *wss://* prefixes as indicators for a WebSocket and a WebSocket Secure connection, respectively. The second attribute, protocol, is optional and if present, it specifies a sub-protocol that the server must support for the connection to be successful. Table

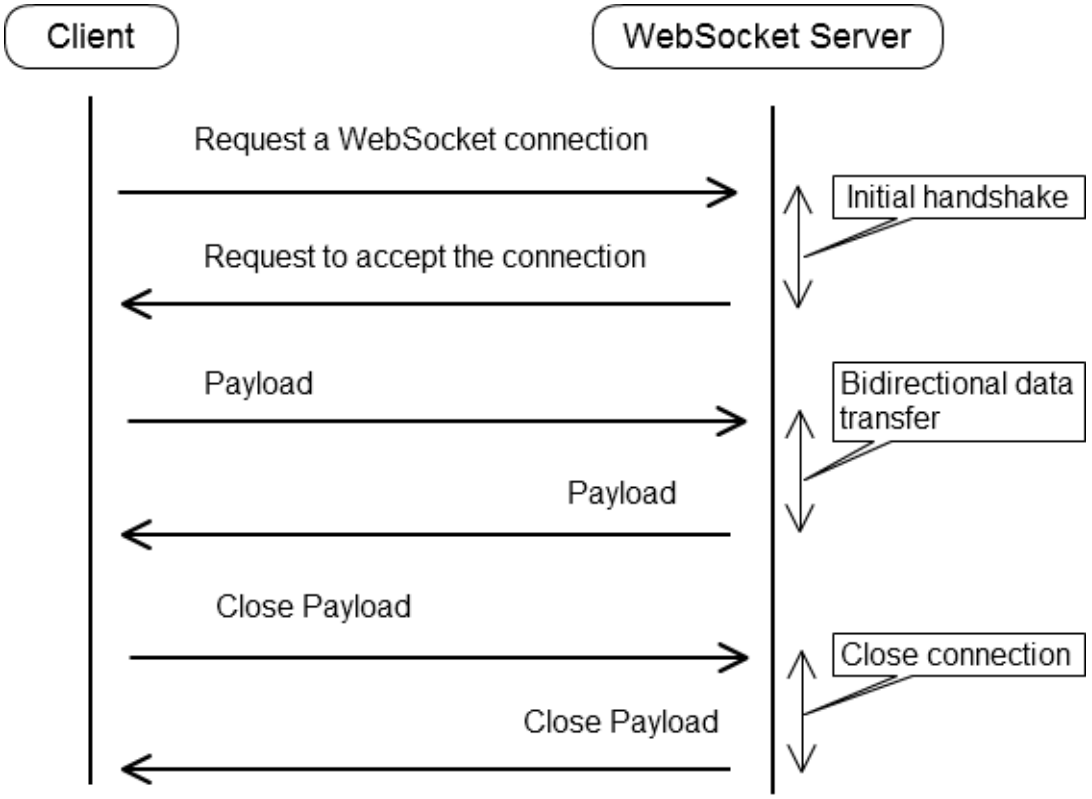

FIGURE 4.1: Overview of the communication with the WebSocket protocol.

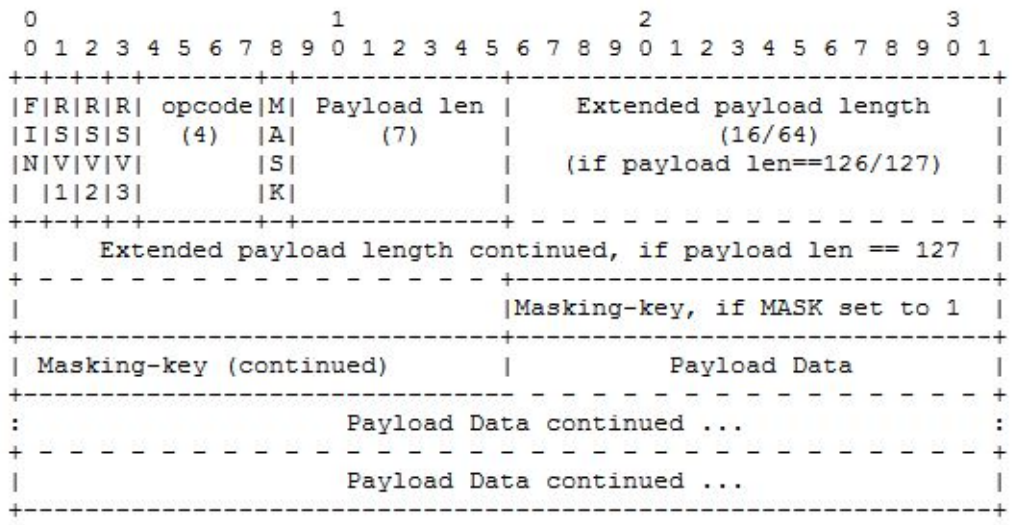

FIGURE 4.2: Detailed WebSocket frame structure provided by RFC6455. (Ref. https://tools.ietf.org/html/rfc6455)

Event	Event Handler	Purpose
open	Socket.onopen	Socket connection is established.
message	Socket.onmessage	Client receives data from server.
error	Socket.onerror	Error has occurred with communication.
close	Socket.onclose	Socket connection is closed.

TABLE 4.1: Types of WebSocket events.

4.1 lists the events associated with a WebSocket object. In a WebSocket connection, data are sent as events from the WebSocket server to the client.

Table 4.2 lists the methods associated with a WebSocket object.

Method	Purpose
Socket.send()	send(data) method transmits data via the connection.
Socket.close()	close() method is called before any existing connection can terminate.

TABLE 4.2: Types of WebSocket methods.

4.2.3 Example of WebSocket code

In this section, the author shows an example of WebSocket code for client-side JavaScript (see Section 4.1). After the HTTP handshake, the WebSocket starts to connect with the bidirectional TCP socket between the Web browser and the WebSocket server. This sample JavaScript code uses the free echo server (*http://www.websocket.org/echo.html*) to establish the WebSocket connection.

```
1  var Socket = new WebSocket('ws://echo.websocket.org');
2  Socket.onopen = function () {
3        console.log("Connection is successful");
4        Socket.send("Hello world");
5  };
6  Socket.onmessage = function (rs) {
7        var receivedData = rs.data;
8        console.log("Message exists.");
9        console.log(receivedData);   // Hello world
10 };
```

LISTING 4.1: Example of clent-side JavaScript code for echo server.

4.3 WebSocket for EPICS-based System

Thus far, various types of Web services in EPICS-based systems have been implemented, for example, data archive viewer, electric log notebook, and so on (see Section 3.2). However, such services have been implemented only in Web-based systems, where a fast response and monitoring are unnecessary, such as for archive viewers and electric-log systems. This is because it is difficult for a Web-based OPI to realize a real-time response similar to native applications, such as EDM/MEDM/CSS, even if AJAX technology is utilized. Given that bidirectional transmission between a server and clients is now possible, the aforementioned problem can be solved with WebSocket. In practice, the author has confirmed a sufficient interactive response in a 100-ms cycle using WebSocket on a Web browser (Mozilla Firefox), and the performance is similar to that of native EPICS applications. Furthermore, given that periodic polling is unnecessary, unlike AJAX and Comet, network traffic can be reduced.

OOn the other hand, WebSocket cannot work with Internet Explorer (IE) 9, which is one of the main browsers in existence today, because a WebSocket connection is unavailable in IE 9. However, this problem was resolved in IE 10, because Microsoft intended for IE to be compliant with WebSocket [75]. Currently, IE 10 and higher support WebSocket.

In order to implement a WebSocket-based system with EPICS, the former must consist of a WebSocket server that corresponds with the EPICS CA protocol as the server-side system, and a WebSocket client that is coded based on the document object model (DOM)[76]. The outline of the EPICS-based OPI using WebSocket is shown in Fig. 4.3.

4.4 Server-Side System

Because the development of an entirely original EPICS-based WebSocket server is expensive, the author decided to use a library that includes a WebSocket server function. Initially, the author made use of libwebsockets [77] and Jetty [78], which also provides a WebSocket server feature, for the development of WebSocket support for EPICS. However, after considering the difficulty of developing such a WebSocket, the author utilized Node.js [79] and the WebSocket library Socket.IO [80] to aid in the development of the WebSocket server. Node.js is one of the server-side JavaScript languages developed based on the Google V8 JavaScript Engine [81].

One of the main features of Node.js is that it works asynchronously with single-thread processing. A significant number of human resources might be required for the development of an entirely original asynchronous WebSocket server; however, Socket.IO, which

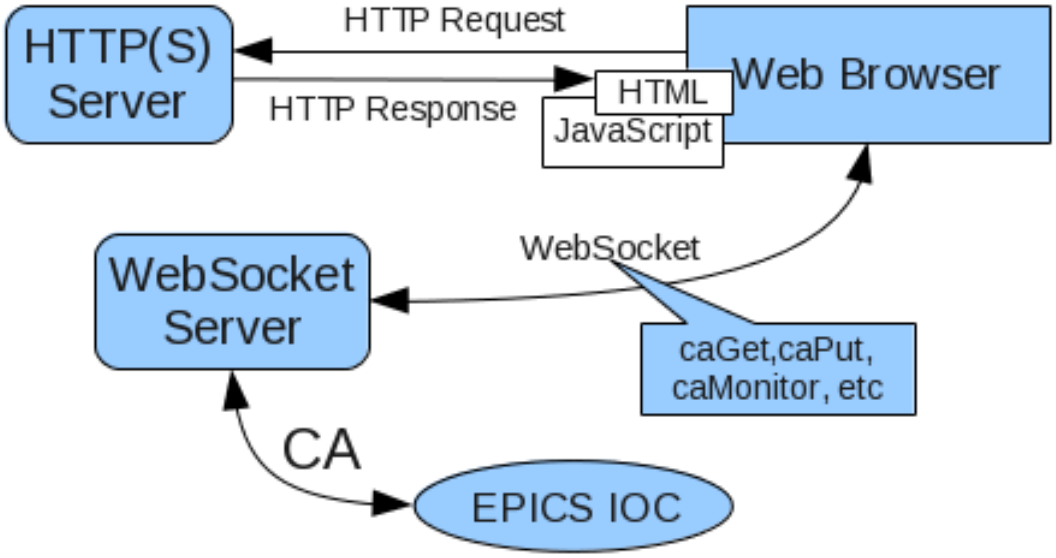

FIGURE 4.3: Outline of EPICS-based OPI using WebSocket

is a library for the construction of WebSocket servers, is used because it contains all the functions required for the development of an asynchronous WebSocket server. Therefore, Node.js 0.6.18 and Socket.IO 0.9.6, which are the stable versions of these tools, were utilized for the development of the WebSocket server.

4.5 Channel Access for Node.js

In order to implement the WebSocket server with a CA connection using Node.js, Node.js must call the CA API. Therefore, the author developed the add-on software called NodeCA using C++ in order to interface with the CA from Node.js. Given that Node.js can call NodeCA, NodeCA utilized the CA library provided by the EPICS base. To call CA from Node.js, caGet, caPut, and caMonitor, which are basic functions in EPICS, were created.

In general, an event-driven caMonitor requires a non-blocking algorithm in the program that uses it, but the single-threading Node.js can cause the thread to be blocked because of negligent implementation. Consequently, in order to prevent thread blocking, NodeCA waits for the CA event in another thread; when the CA event arrives at that thread, NodeCA sends the message queue to the main thread. Fig. 4.4 shows an overview of NodeCA. In Fig. 4.4, the software developed by this research is shown in the grey area. Examples that use NodeCA with caGet, caPut, and caMonitor are provided in this section. Note that at the start of each example, it is necessary to set *require("nodeca")* as the add-on software.

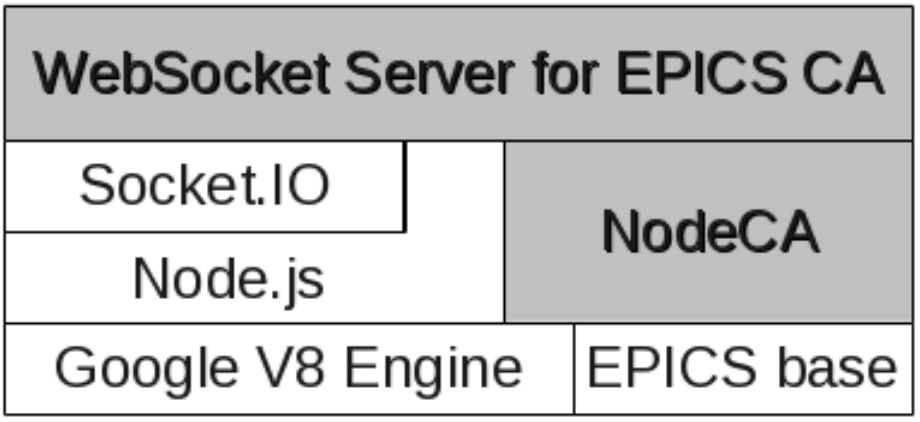

FIGURE 4.4: Overview of NodeCA.

4.5.1 Sever-side caGet Example

This program fetches a value as a *TEST:calc* record from the EPICS IOC in intervals of 100 ms using caGet, and outputs the numerical value using *console.log( )*.

Based on a comparison with a simple CA client written in the C language, which is described in Section 2.4.2, it is evident that coding the CA client using NodeCA can be performed with ease.

```
1  var pv = require("../build/Release/nodeca")
2  function cagets(){
3      console.log(pv.caget("TEST:calc"));
4  }
5  setInterval(cagets,100);
```

Listing 4.2: Example of server-side JavaScript code for caGet function.

4.5.2 Sever-side caPut Example

This program changes the value that is stored in the *TEST:subExample* record of EPICS IOC to five via the WebSocket server.

```
1  var pv = require("../build/Release/nodeca")
2  pv.caput("TEST:subExample",5);
```

Listing 4.3: Example of Server-side JavaScript code for caPut function.

4.5.3 Server-side caMonitor Example

This program outputs the numerical value of the *TEST:ai* record through an event-driven behavior. (When the *TEST:ai* record is changed, the value is sent to the client automatically.) In order to receive the value, the pv.callback method is required before running the pv.camonitor method, because Node.js is implemented through an asynchronous algorithm.

```
1  var pv = require('../build/Release/nodeca');
2  function mon(){
3      pv.callback=function(v){
4          console.log('Value: ' + v);
5      }
6      pv.camonitor("TEST:ai",0);
7  }
8  setTimeout(function(){
9      mon();
10 },1000);
```

Listing 4.4: Example of server-side JavaScript code for caMonitor function.

4.6 WebSocket Server with EPICS CA

The WebSocket server that corresponds to the EPICS CA protocol was developed utilizing NodeCA as described in Section 4.3.2. Given that the WebSocket server was written with Node.js, both the Web server and the Web client are developed using the JavaScript language. (Recall that Node.js is a server-side JavaScript language.) The system chart for the WebSocket server is shown in Fig. 4.5. The corresponding algorithm is described as follows:

1. In the code of the client-side JavaScript, custom events (caGet, caPut, and caMonitor) are sent to the server using the WebSocket protocol.
2. In the case of a caGet/caPut event, the main process makes threads (caGet, caPut), and the threads connect to EPICS IOC using the CA protocol.
3. The information between the main process and the thread is communicated using a message queue as the interprocess communication (IPC).
4. In the case of a caMonitor event, the main process makes a child process and the child process makes a thread (caMonitor). The thread continues to connect to EPICS IOC using the CA protocol.
5. Similar to the case of caGet, in the case of caMonitor, the information between the main process and the thread is communicated by message queue, and the main process and the child process communicate using the IPC (*child.fork( )*) provided by Node.js.

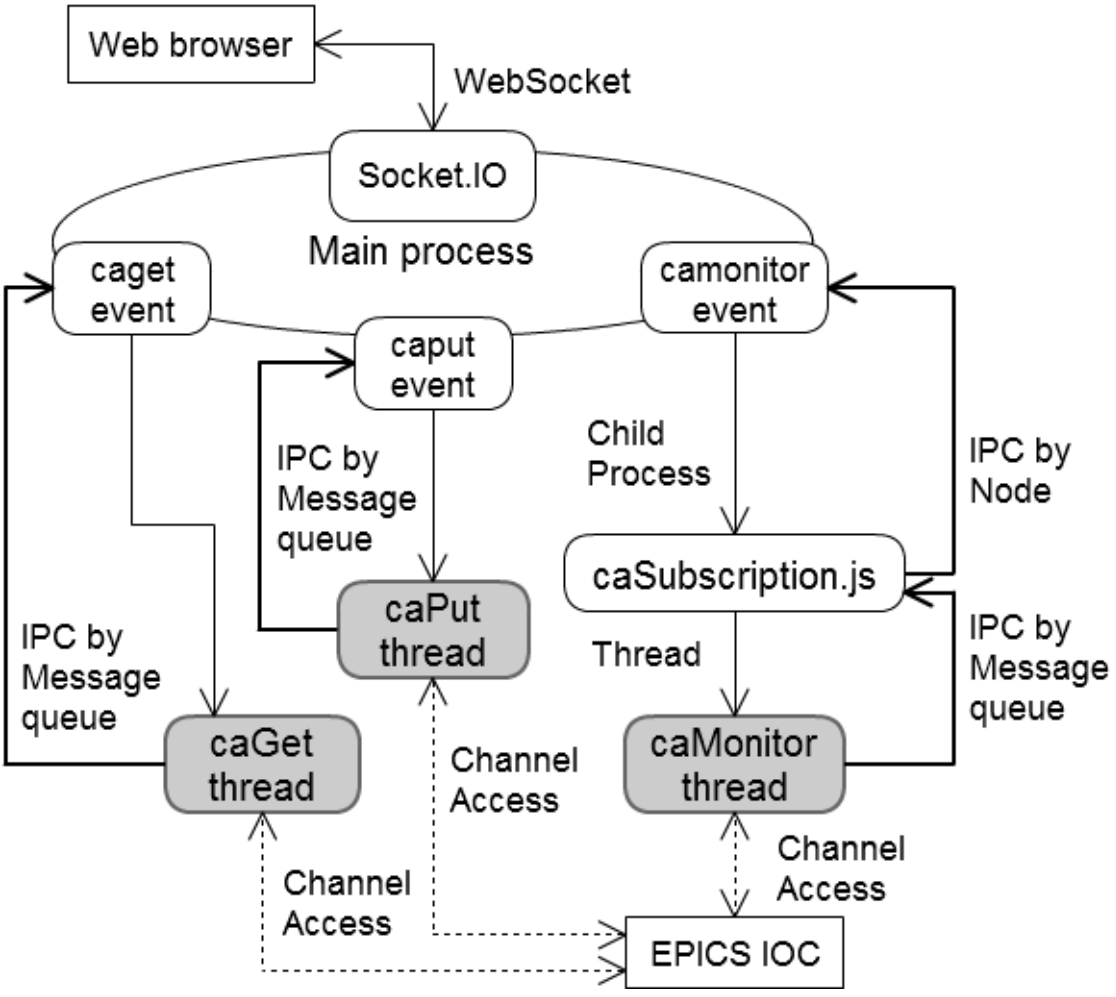

FIGURE 4.5: System chart for WebSocket server.

4.7 Client-Side System

An example of coding in the client-side JavaScript is as follows:

```
1  socket.emit('caMonitor',{
2      pv: 'Example:ai1',
3      data_name: 'TEST2',
4  });
5  socket.on('ca', function(data){
6    document.getElementById("TEST_record1").innerHTML = data.TEST2;
7  });
```

LISTING 4.5: Example of caMonitor for client-side JavaScript.

The behavior of this example is as follows:

1. The client-side JavaScript obtains a value from the *Example:ai1* record by sending a message of the event-driven caMonitor in WebSocket.
2. The value is stored in *data.TEST2*.
3. Through DOM operation, the value is updated on the HTML with respect to each CA event.

It is possible to realize a text update, such as EDM, by coding DOM with JavaScript on the Web. DOM is a platform and language-neutral interface that allows programs and scripts to access and update content dynamically, and to structure and style documents [82]. The following is a concrete example of HTML, where a DOM element, such as the tag, is updated on a Web browser through JavaScript code.

```
1  <div id="hoge">hogehoge</div>
2  <script type="text/javascript">
3    document.getElementById("hoge").innerHTML = "hello world";
4  </script>
```

LISTING 4.6: Example of conding DOM

In this example, the code replaces *"hogehoge"*, which is surrounded by a tag, with *"hello world"*. This method is used for WebSocket-based systems and for traditional HTML technology, such as AJAX. The difference between AJAX or Comet and WebSocket is based on the communication method used to update values.

Other libraries used in this WebSocket-based system are flot [83] and jsgause [84], which are jQuery-based JavaScript libraries; these libraries are used in this system for the visualization of accelerator parameters. Many JavaScript libraries, other than those mentioned at the start of this paragraph, are also available for the visualization of

information, such as strip charts. As a result, the client system has the advantage of low development costs because money can be saved in multiple steps.

A verification system was constructed that is useful for remote operation because a response comparable to native EPICS applications is confirmed. In addition to PC-based Web browsers, the use of OPIs on Web browsers for iPhone4S and Android mobile is available. A screenshot of WebSocket-based OPIs is shown in Fig. 4.6, 4.7 and the Web browsers supported by this system as of July 2012 are listed in Table 4.3. Chapter 6, which describes the system implementation, reports such systems in detail.

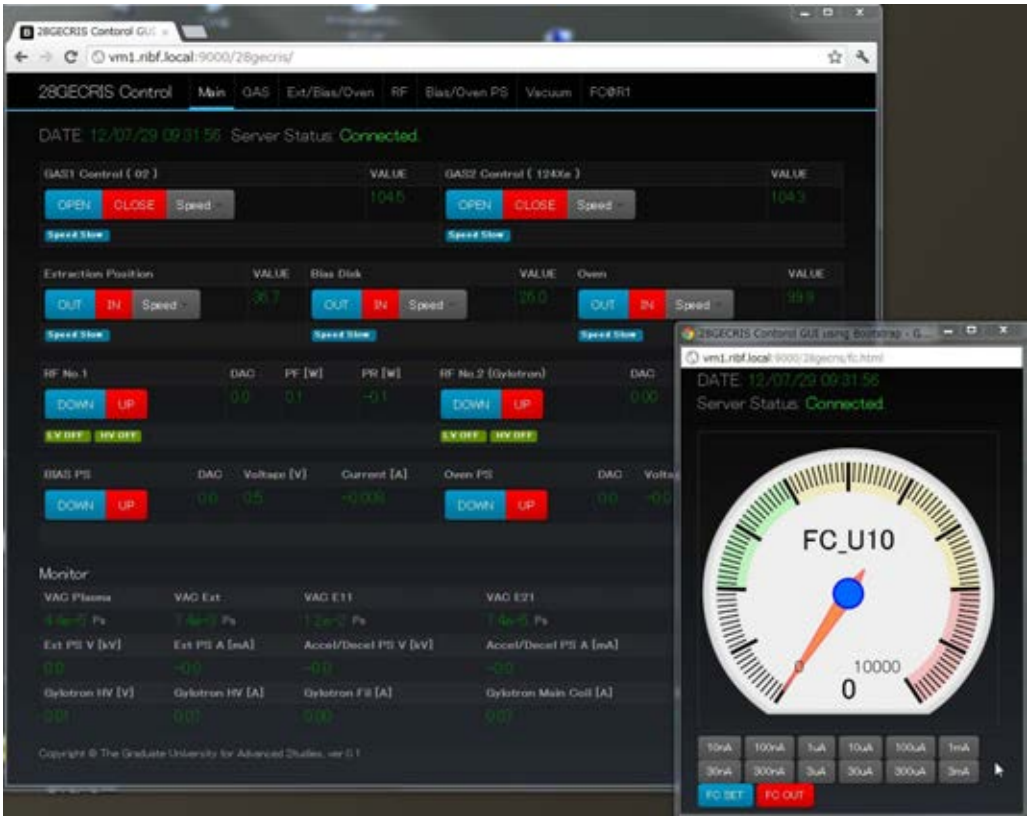

FIGURE 4.6: Screen shot of sample OPI displayed on PC-based Web browser (Google Chrome).

Windows/Linux	Google Chrome 20, Mozilla Firefox 14, Opera 11
Android 4.0	Google Chrome 18, Mozilla Firefox 14
iOS	Safari 5, Google Chrome 19, Mercury

TABLE 4.3: Web browsers supported by the WebSocket-based OPI as of July, 2012.

FIGURE 4.7: Screen shot of example OPI displayed on mobile Web browsers. Left figure is for android mobile and right figure is for iPhone4S.

4.8 Example of Use of Node-CA

As well as this study, Node-CA was used for construction of WebSocket-based system in KEKB Linac control system [85]. In the KEKB injector Linac, information regarding accelerator operation has been provided through HTTP services and CGI since 1994. In order to provide such information in real time, WebSocket-based OPI was developed using Node-CA. As a result, detailed information on the accelerator operation was successfully provided for the entire KEK campus using a new GUI panel, which appears on the Web (See Fig. 4.8).

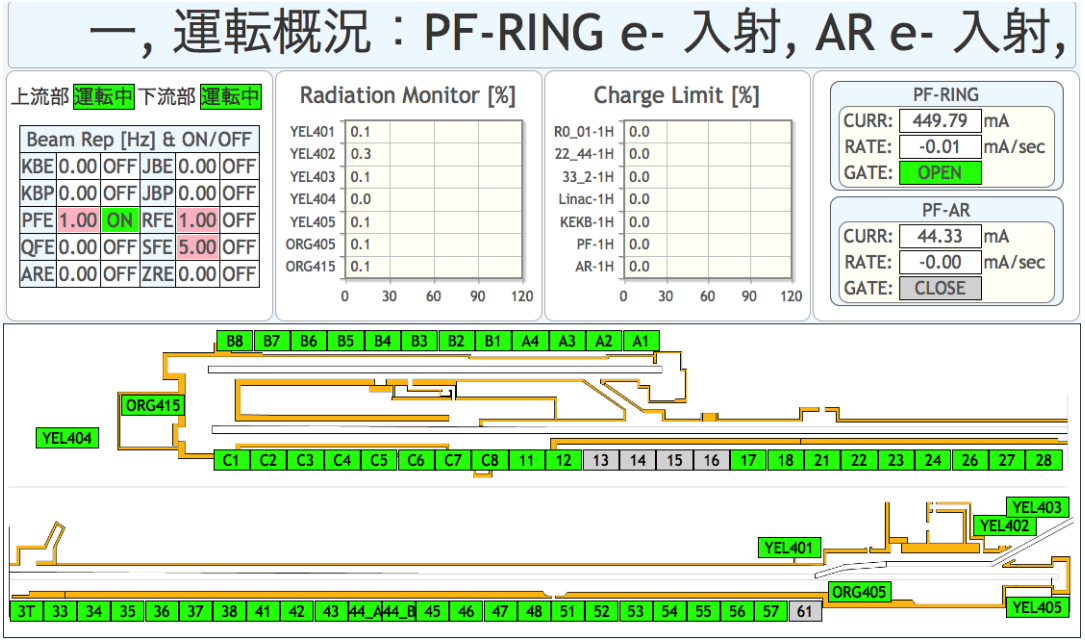

FIGURE 4.8: Screen shot of operational panel using Node-CA for KEKB injector Linac.

Chapter 5

Operator Intervention System

Accelerator facility management through international collaboration often requires methods for remote maintenance, control, and monitoring of associated devices. For example, a method needs to be provided to connect to the control system network via WAN links from various collaborating institutions. However, from a practical application standpoint, the remote operation of an accelerator via WAN is beset by a number of concerns, including security concerns.

A particular concern is that the accelerator has both experimental devices and radiation generator characteristics. Additionally, any error in the operation of remote control systems could result in an immediate breakdown. For these reasons, the author proposes the implementation of an operator intervening system for remote accelerator diagnostics and support that can obviate any differences between the local control room and remote locations. After the implementation, the on-site accelerator operator can determine the availability of the equipment, and decide whether to grant operation requests from remote users.

5.1 Development Background

As described in Section 1.3, there are several cases of cooperation between experts and remote operators across WAN links for accelerator operations. In situations such as accelerator problems and change-of-beam conditions (for example, beam emittance from ion sources), on-site accelerator operators require directions over the telephone from other engineers or scientists with regard to accelerator components, such as RF, vacuum, beam diagnostic, control system, and ion source. In such scenarios, the ability to provide the exact directions that are required by accelerator operators is contingent on

the amount of information required for remote troubleshooting. Even in cases where the native languages are not in conflict, it is difficult to correctly provide detailed information about accelerator conditions via the telephone and through voice communication because not enough information can be obtained by listening compared with seeing.

On the other hand, and as described in Chapter 4, WebSocket-based operator interfaces (OPIs) are available to resolve problems via WAN links because of the development of a global Internet society. In addition to monitoring accelerator parameters from locations distant to the accelerator control room, remote users can operate accelerator devices that provide outputs, even if there are no concerns such as radiation safety, human physical safety, and network access security.

However, accelerators should be controlled with utmost care because accelerators have both experimental devices and radiation generator characteristics. From past experience with the operation of accelerators, the author believes that the atmosphere of the accelerator control room is an important factor during beam tuning. For example, because almost all accelerator information is integrated into the accelerator control room, the implementation of joint monitoring with multiple operators to observe the condition of the accelerator is a useful method for avoiding any problems that might arise.

In order to prevent operational mistakes, the accelerator conditions need to be understood in their entirety. Therefore, it is necessary to consider a safety system with on-site accelerator operator intervention in order to control output via remote access. Note that a safety mechanism is indispensable when the remote operation needs to monitor parameters and control output in order to prevent accelerator problems caused by operational mistakes.

To address safety issues and improve the usability of remote operations, the author developed an operator intervention system for WebSocket-based OPIs for the EPICS [86].

5.2 Security Policy

First, the author considered the following in order to protect accelerator networks from outside intrusion. In the case of accessing accelerator networks from the WAN, the author ensured the security of WebSocket access using VPN and authentication via a Secure Sockets Layer (SSL). In addition, accelerator operators have to completely understand users attempting to access the network (login ID and so on) and access route (full domain of Internet providers and so on) before WebSocket control is allowed. Therefore, all accessing logs for WebSocket control should be open to accelerator operators.

Next, the author considered instructions for accelerator operation to EPICS IOC from the Web browser via WebSocket. For caPut, which provides instructions to each device or component during accelerator operation, accelerator operators need to know the action and behavior perfectly. On the other hand, it would not matter if such operators did not understand some simple instructions in detail, such as caGet and caMonitor, which are used to monitor or obtain numerical values for accelerator parameters. If some accelerator parameters are changed using the Web application from outside internal networks without considering beam tuning, the accelerator condition might be complicated in many cases. Thus, when using caPut, a policy is required between accelerator operators in the control room and remote users exerting control remotely over the Web. The following are proposed for this policy.

1. Before remote users exercise remote control over the Web, they have to communicate with accelerator operators using VoIP system such as Skype.
2. On-site accelerator operators check the user ID and the domain or IP address of the Internet provider for remote users.
3. Remote users send a request to accelerator operators when sending caPut instructions to the EPICS record.
4. In response to requests from remote users, accelerator operators make judgment decisions regarding whether components in the EPICS record can be controlled using WebSocket. A response is returned to remote users regarding whether it is possible to operate said components in the EPICS record.
5. After receiving this response, remote users can not only monitor, but also operate components via WebSocket.
6. If accelerator operators make judgment decisions to interrupt operation, accelerator operators discontinue WebSocket operations as soon as possible. In addition, if a certain period passes, permission for WebSocket operation will be cancelled through timeout.

5.3 System Concept of Operator Intervention System

The purpose of the proposed operator intervention system is to ensure safe remote output control of EPICS via the WebSocket server. To simply monitor the status of the accelerator parameters using the WebSocket-based OPI, permission from the on-site operator is not necessary, provided that authentication has been accomplished through an SSL connection. The WebSocket-based client system is designed such that output control via remote operation always requires the permission of an on-site accelerator operator who can intervene at any time.

The main part of the proposed operator intervention system consists of a Process Variable (PV) gateway provided by EPICS collaboration [87], a MySQL database, and AJAX web applications. Unlike OPI, real-time interactive response is not required for sending output control permission, or for viewing access logs in the operator intervention system. Consequently, in consideration of the difficulty of system development, the author utilized a Linux, Apache, MySQL, PHP (LAMP) environment for the operator intervention system without using WebSocket. The procedure for the implementation of the proposed operator intervention system is as follows (see Fig. 5.1 for an illustration.). The detailed system design is described in Section 5.5.

1. First, on-site accelerator operators grant permission to every output of PVs that corresponds to the EPICS hardware devices.
2. If PVs have not been granted permission, the operator intervention system always rejects the output command in the CA protocol layer.
3. On-site operators can decide the upper and lower limits of remote control operation before sending permission commands.

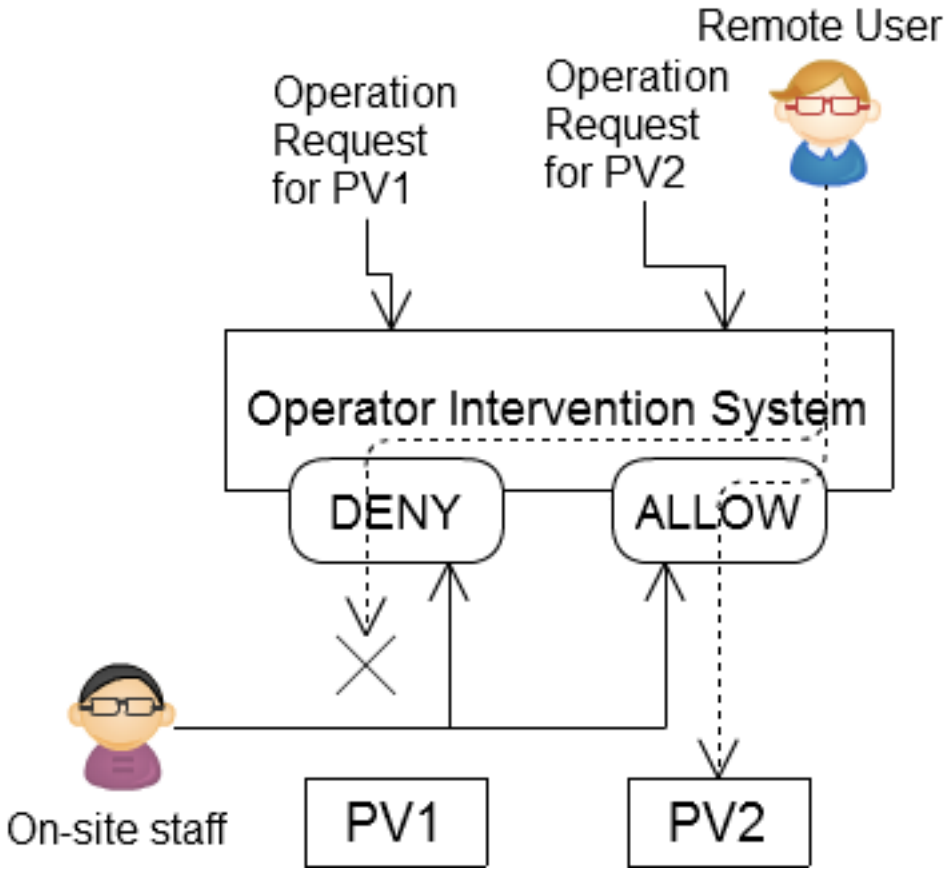

FIGURE 5.1: Procedure for implementation of proposed operator intervention system.

5.4 Basic Technology for Operator Intervention System

In this section, the underlying systems used by the proposed operator intervention system are described in detail.

5.4.1 Virtualization Technology

Virtualization technology allows the sharing of hardware resources through cloud computing. Some examples of such virtualization technology are KVM, Xen, Hyper-V, and VMware, and such software is widely used in many situations. Server virtualization is a technology where the OS of several virtual guests can be run in one physical server (see Fig. 5.2). In order to enhance the efficient operation of server hardware resources, the software listed previously can be used also to construct accelerator control systems [88–93]. On the other hand, such an operator intervention system needs to consist of several servers with low server load. Therefore, VMware vSphere Hypervisor, which is free virtualization software produced by VMware Inc., has been adopted considering system security and ease of system construction.

5.4.2 EPICS Process Variable Gateway

In the EPICS-based system, the PV gateway is used for its feature as a proxy system for the CA connection. In general, the PV gateway is installed for the purpose of relaying the CA servers, such as EPICS IOC, and CA clients, to reduce the network load in the EPICS-based system (see Fig. 5.3). In addition, access restriction, namely the EPICS Access Security Group (ASG), is implemented in the PV gateway as well as the EPICS IOC. ASG is described in the next section. In the proposed operator intervention system, the WebSocket server always connects to the EPICS CA server via the PV gateway.

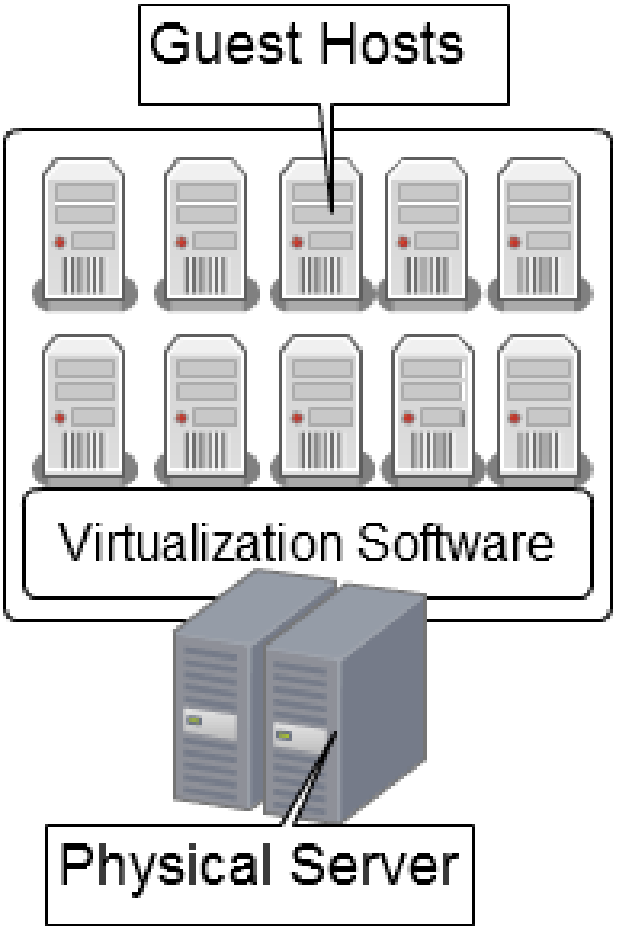

FIGURE 5.2: Outline of virtualization technology.

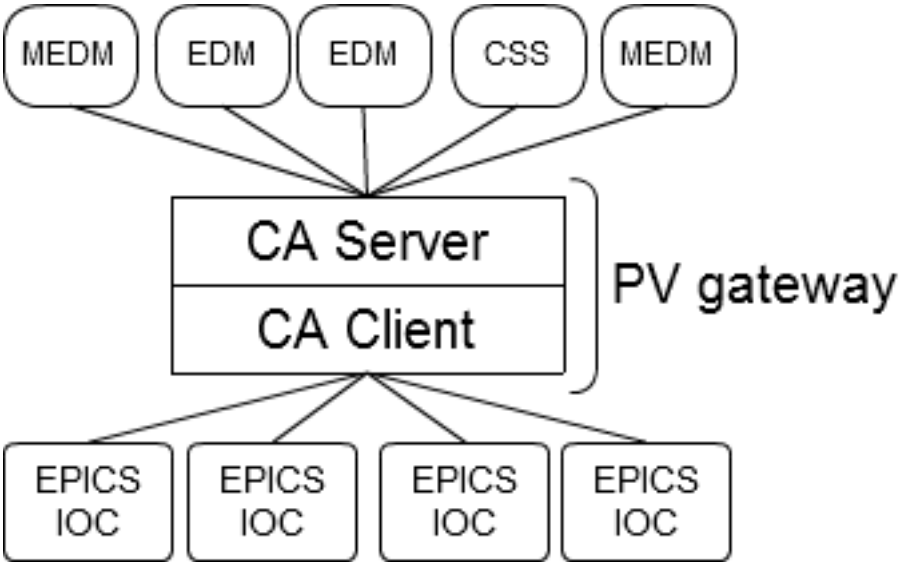

FIGURE 5.3: Outline of PV gateway.

5.4.3 EPICS Access Security Group

EPICS ASG is a security function for implementing restrictions to access the EPICS IOC and PV gateway through the use of a configuration file [50]. Using ASG, it is possible to realize access permission and read/write access through a host name and a login name. An example of the configuration file is as follows.

```
UAG(uag) {user1,user2}
HAG(hag) {host1,host2}
ASG(DEFAULT) {
  RULE(1,READ)
  RULE(1,WRITE) {
    UAG(uag)
    HAG(hag)
  }
}
```

Listing 5.1: Example of the configuration file for ASG.

The previous sample file configures all hosts to be allowed to read PV values. In addition, the sample file configures host1/host2 as the hostname, user1/user2 as the login name, and grants the hostname and login name permission to write PV values. By utilizing the login name and hostname used by the client PC, ASG can restrict access to the CA server. However, it is not possible to use ASG as a complete security mechanism because the login name and hostname depend on system configuration, and malicious impersonation is possible.

5.5 System Policy

The operator intervention system can control the availability of remote operations by sending system flag values to the MySQL database. The access control mechanism for the CA protocol is utilized by the access control security system of the PV gateway. In particular, a daemon polls the database and updates the security access file (GATEWAY.pvlist) with information that corresponds to the flag values dynamically. A flowchart that illustrates system operation for remote connections is depicted in Fig. 5.4 and described as follows:

1. Remote users send EPICS PV requests to the operator intervention system to control the output, such as caPut command, via WebSocket (Phase flag 0).

2. Then, on-site operators ascertain whether the requested PVs are available for control (Phase flag 1).

3. If control is possible, a command to accept is sent by on-site operators (Phase flag 2); otherwise, a refuse command is sent (Phase flag 3).

4. On-site operators can terminate remote operation at any time, for example, if problems with the accelerator arise (Phase flag 3).

5. A process that manages the time duration of the remote operation sends a timeout command after a certain period has elapsed (Phase flag 4).

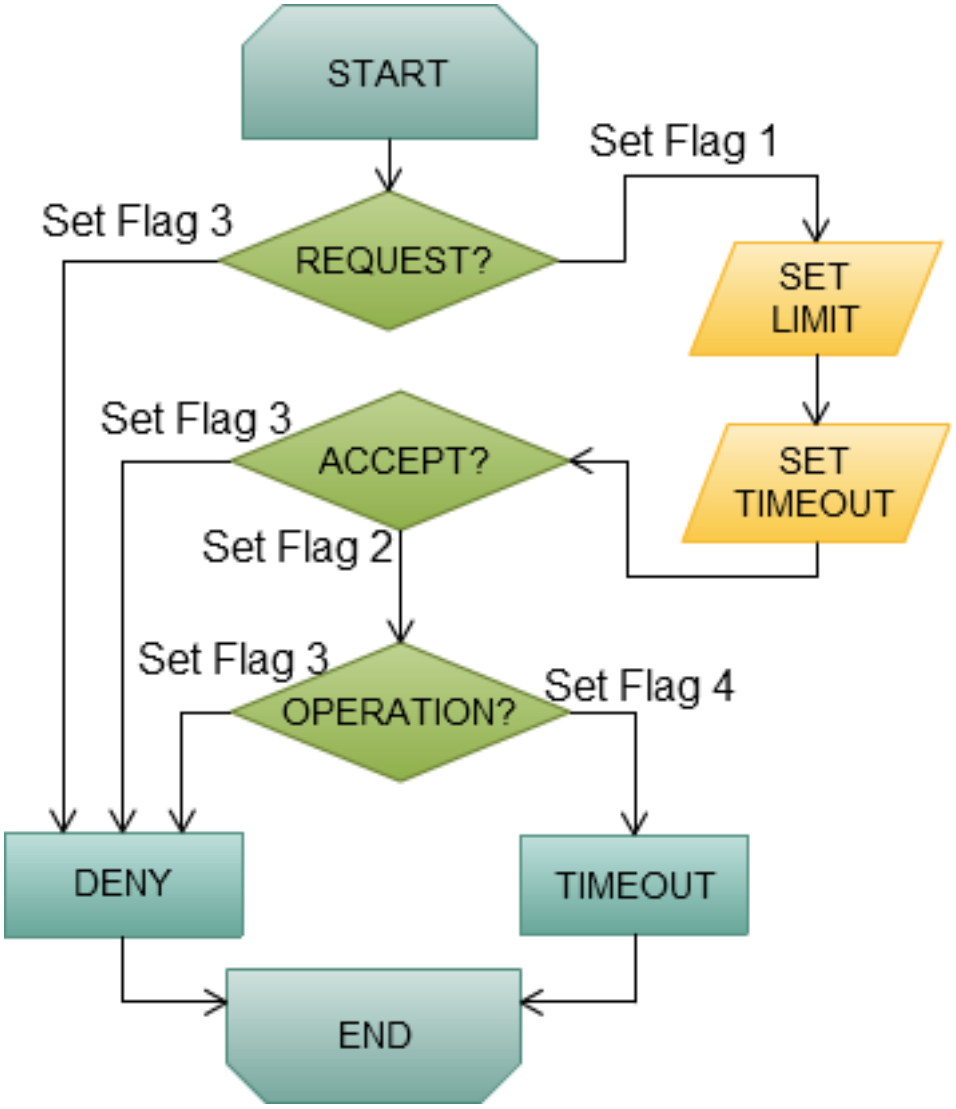

FIGURE 5.4: Flowchart illustrating operation of system for remote connection.

5.6 System Design

The proposed operator intervention system was developed utilizing virtualization technology and the EPICS PV gateway described in Section 5.3.2. In this section, the system design for the operator intervention system is described.

5.6.1 Required Services

The operator intervention system comprises important services that run on several distributed hosts. A chart of the system is depicted in Fig. 5.5. The MySQL service is necessary to store the flag value via an internal network. The Apache web server and network file system (NFS) is run on an application server. NFS is used to share log files among the WebSocket server, the Apache web server, and the PV gateway via the internal network, as well as the MySQL service. WebSocket server programs developed using Node.js run on a separate host that connects to the MySQL database and the EPICS IOCs via the PV gateway.

5.6.2 Preventing Impersonation

In order to make a test installation of multi-layer network structures, virtualization technology is utilized in order to prevent malicious impersonation. The access control security system does not comprehensively cover access from the Internet because EPICS is not able to completely prevent impersonation (see Section 5.3.3). In general, a defense-in-depth network system with a multi-layer network structure is one of the most useful methods for improving network security. In this system, the author implemented a multi-layer network structure that consists of servers running in a virtual environment on one physical server. In the system, network attacks and unlawful access are thwarted using a defense-in-depth network to prevent access from a physical network (see Fig. 5.6).

The author implemented a virtual environment using VMware vSphere Hypervisor 5, which can be used free of cost. The specifications for the physical server are listed in Table 5.1. The author used CentOS 5.9 as the operating system for the virtual machines.

On the other hand, the defense-in-depth network will be composed of multiple physical servers in the real implementation.

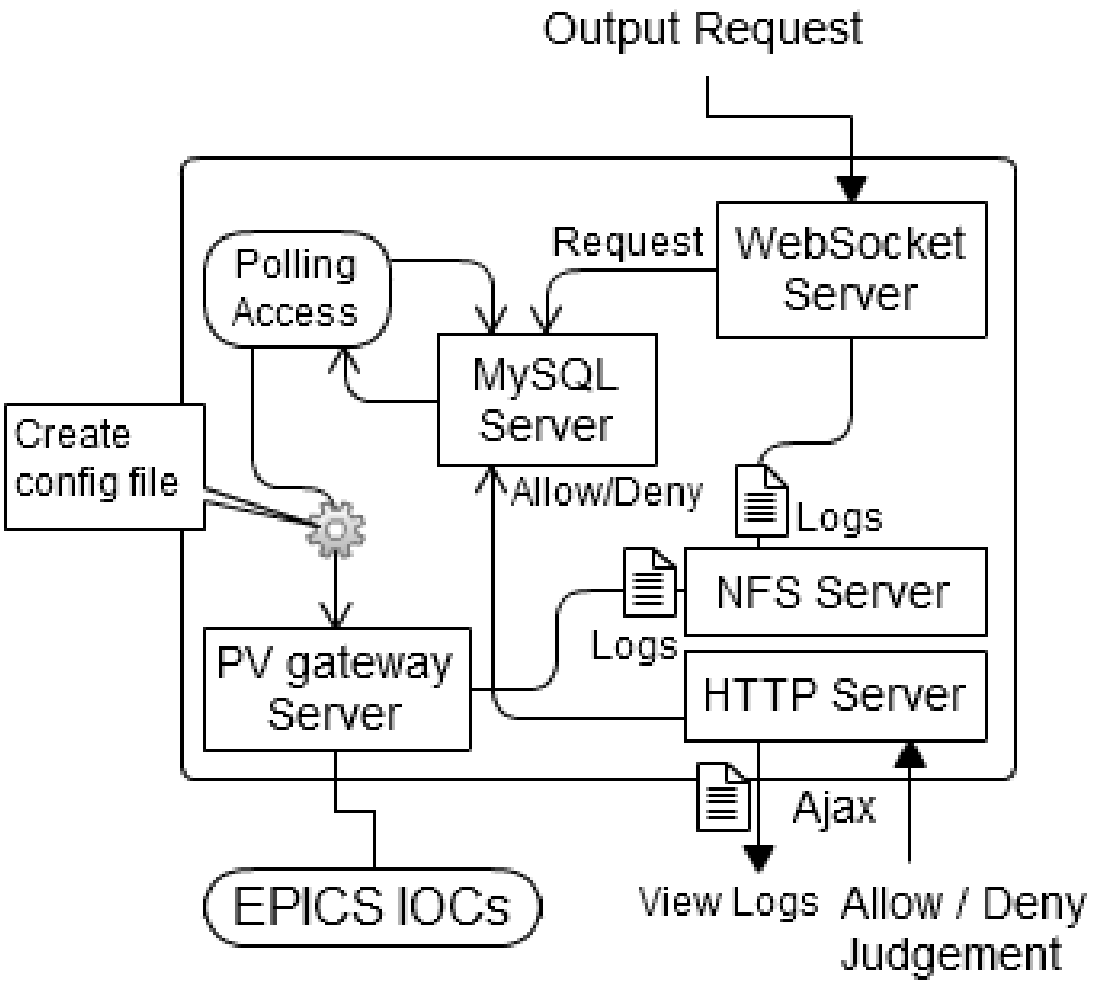

FIGURE 5.5: System chart showing operator intervention system, EPICS IOCs, and WebSocket server.

CPU	Intel Core i7-3770
Memory	16G Byte
Storage	SATA
Ethernet	Dual-network connection

TABLE 5.1: Physical server specifications.

5.6.3 Setting Upper/Lower Limits

In general, EPICS utilizes DRVH/DRVL fields to set the upper and lower limits of the analog output (ao) records [50]. In this case, however, utilizing the DRVF/DRVL fields of the EPICS IOCs to control the system network is not wise because it might cause problems, such as lack of parameter settings, in normal operation with remote users.

As a result, the author implemented the function by modifying the source code of the PV gateway. Consequently, the customized PV gateway has a GATEWAY.limit file to

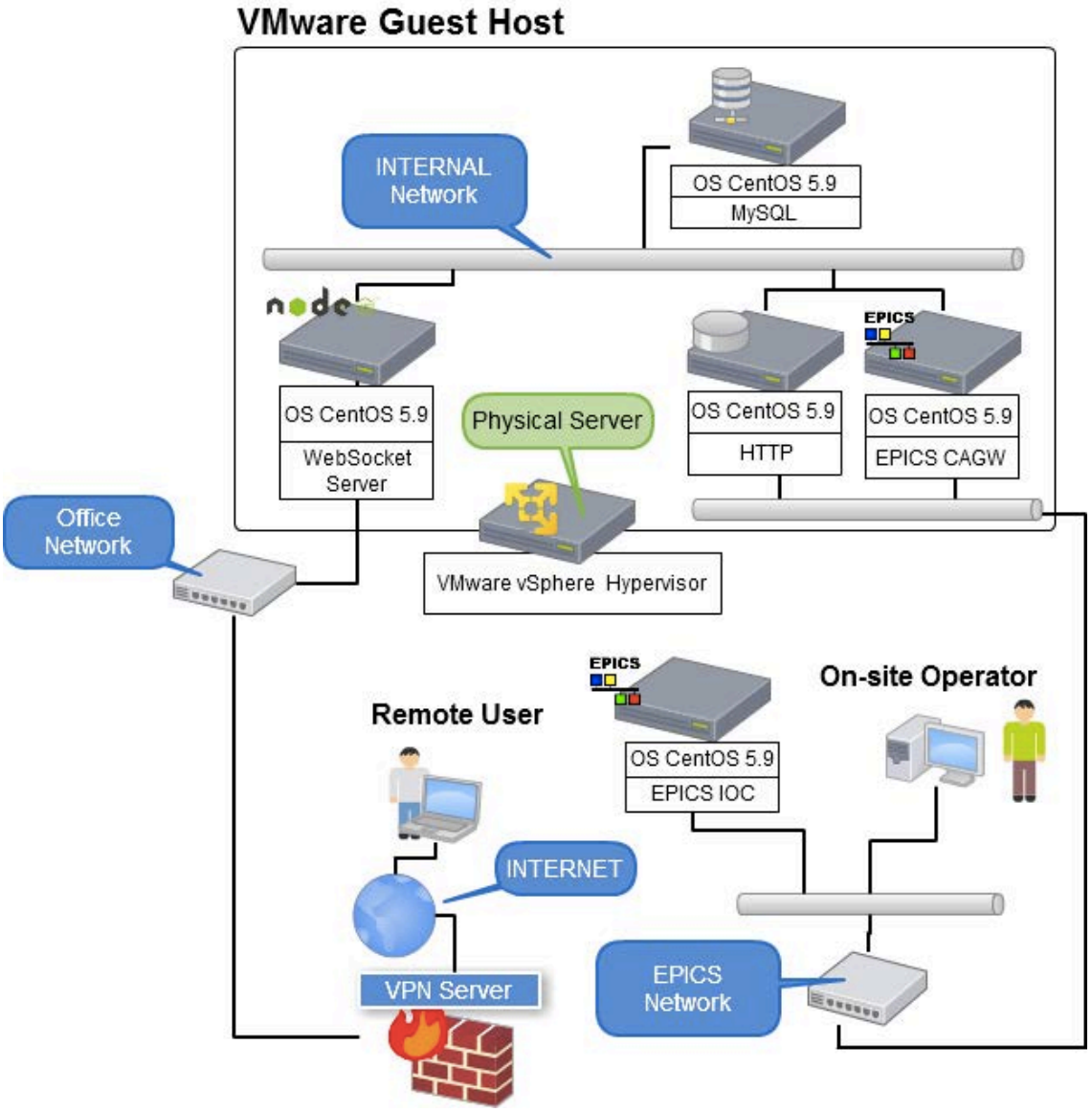

FIGURE 5.6: Network chart of system in VMware virtualized environment.

set the upper and lower limits, as well as GATEWAY.pvlist and GATEWAY.access files for access control.

5.6.4 UIs

The author developed the interface of the operator intervention system as a web application using AJAX because such interface is not required to be as responsive as the WebSocket-based OPI. The UI of the operator intervention system is shown in Fig. 5.7. On-site operators can verify the requested EPICS PVs, completed PVs, and in-operation PVs in one view. Similarly, all the access logs and caPut logs are available for verification by the on-site operator at any time.

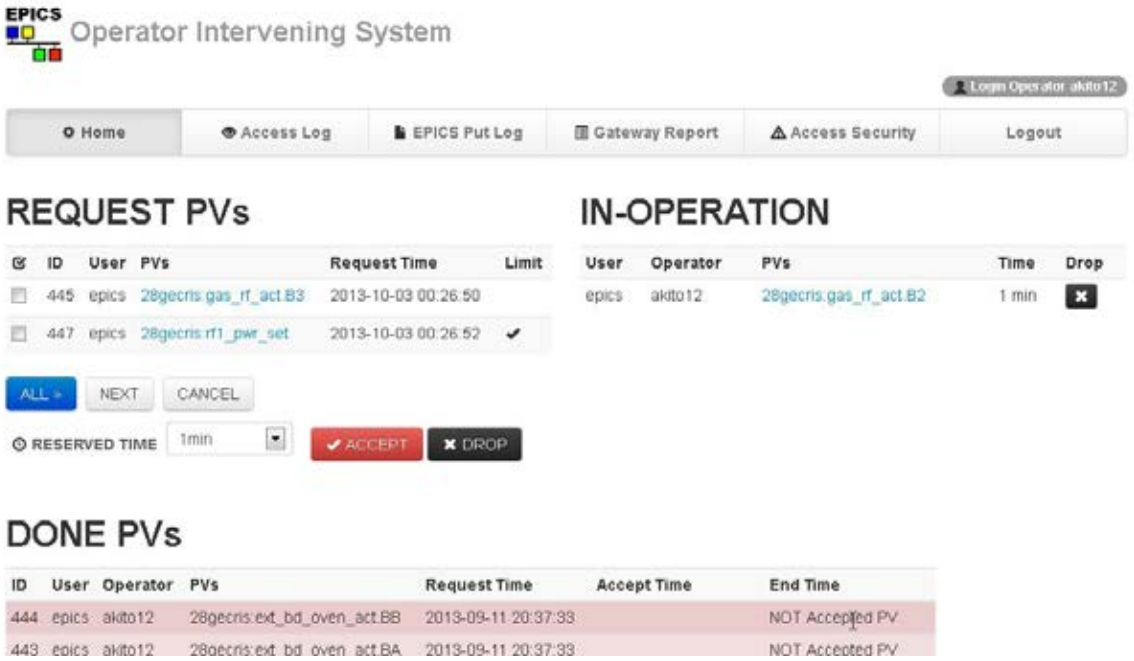

FIGURE 5.7: User interface for operator intervention system. On-site operator can control remote operation through EPICS PVs with this UI.

5.6.5 Security for WebSocket-side

The author ensures secure WebSocket access to the accelerator network from the WAN using a VPN and authentication with SSL. In addition, accelerator operators have to comprehensively verify a user (via login ID) that seeks access to the network and access route (e.g., full domain of Internet providers, etc.) before WebSocket control is allowed.

5.7 Similar System

A new wide area remote control system (WARCS) with a concept similar to the proposed operator intervention system was implemented at SPring-8 in 2012 [94]. At SPring-8, an old version of WARCS that uses Zebedee, a tunneling tool via VPN, was implemented for remote maintenance via WAN [95]. However, the system has the following limitations.

1. In some cases of network routing, WARCS cannot ensure the network port used by Zebedee.
2. To use a Zebedee-based VPN for WARCS, a dedicated client terminal is necessary.

To solve these limitations, WARCS was upgraded using SSL-VPN, which is one of the standard cryptographic technologies. In addition to being used as a terminal for typical PCs, the main characteristic of WARCS is that the on-site staff responsible for the operation of the accelerator issue SSL-VPN accounts. Because intervention of on-site staff is necessary for security, such a characteristic of the new WARCS is a concept similar to the operator intervention system described in this chapter. A comparison of the features of the new WARCS and the proposed operator intervention system are summarized in Table 5.2.

	Operator Intervention System	New WARCS
Object controlled by on-site staff.	Control I/Os via WebSocket	VPN accounts
UI	Web	Web
Purpose	Security for remote operation in EPICS-based control system.	Security for trouble shooting in control system at SPring-8.

TABLE 5.2: Comparison of features between new WARCS and proposed operator intervention system.

Chapter 6

System Implementation

The control system for the RIKEN 28 GHz SC-ECRIS consists of a distributed control system based on EPICS and RIBF. To maintain beam quality for extended beam-service time at RIBF, beam tuning is required in order to prevent subtle changes in 28 GHz SC-ECRIS conditions. Once this is achieved, it should be possible to check conditions and operate the ion source remotely at any time. The author has designed WebSocket-based OPIs to control the ion source remotely; however, for access and control from outside the facility, suitable access security, policies, and methods are required. Therefore, the author implemented an operator intervention system that allows access to the network safely from external sites with permission from on-site accelerator operators in the control room.

6.1 Target Component for the Implementation

In this section, an ion source, which is a component of a heavy ion accelerator facility such as RIBF, is described for implementation in this study.

6.1.1 RI-Beam Factory Project

To provide the world's most intense radioisotope (RI) beams over the entire range of atomic masses of heavy ions, the RIBF project was started at RIKEN [96, 97]. This accelerator facility consists of four ring cyclotrons that include a superconducting ring cyclotron, and two linear accelerators with Azimuthally Varying Field (AVF) cyclotrons as injectors. In order to increase the energy of the heavy ion beams from the RIKEN Ring Cyclotron (RRC: K 540 MeV) in the RARF project, three new ring cyclotrons-a fixed-frequency ring cyclotron (fRC: K 570 MeV), an intermediate stage ring cyclotron

(IRC: K 980 MeV), and a superconducting ring cyclotron (SRC: K2600 MeV) have been updated in the RIBF project [98]. The schematic layout of the RIKEN RIBF is shown in Fig. 6.1.

In the RIBF, variable heavy ion beams, from helium to uranium, have been provided for experiments. However, the capability of the ion source has a decisive influence on accelerator facility performance, such as beam intensity. Therefore, the ion source is one of the most essential components that constitute the heavy ion accelerator facility.

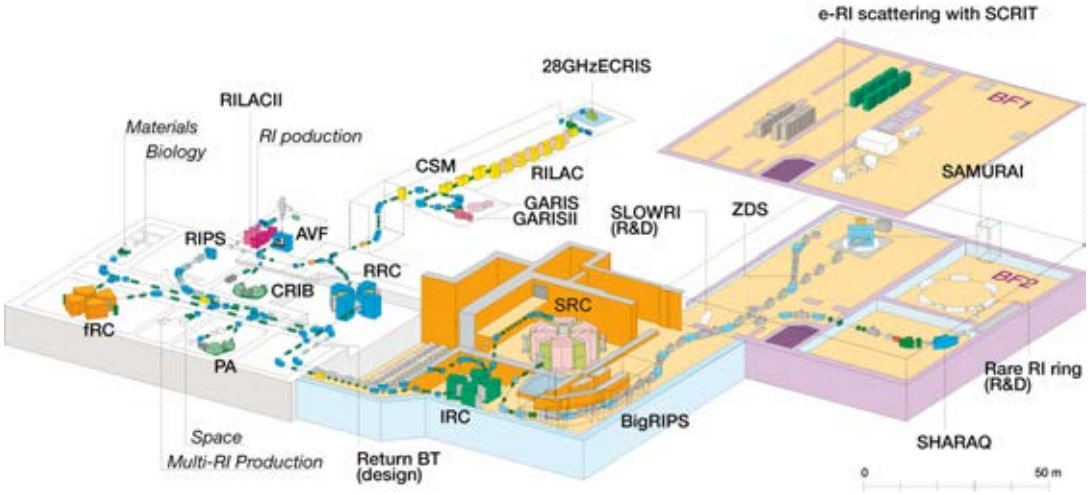

FIGURE 6.1: Schematic layout for RIKEN RIBF. 28 GHz SC-ECRIS was constructed in the injection line for RIBF at RIKEN.

6.1.2 Electron Cyclotron Resonance Ion Source

The kinetic energy per nucleon E of heavy ions accelerated by the cyclotron is represented by the following formula:

$$E = K(q/M)^2 \tag{6.1}$$

where K is an accelerator-specific value that is determined by the strength of the magnetic field and the size of the cyclotron, q is the valence, and M refers to the mass number. In the above formula, in order to meet the demands of heavy ions with larger kinetic energy, ion beams for which 'q/M' is larger (i.e., highly charged beams) are required.

Currently, ECRIS is the best equipment that can provide stable direct current highly charged beams for heavy ion accelerators [99]. The main components required for ECRIS control are as follows.

1. Material supply (gas, rod, oven, etc)
2. Ionization (plasma production)
3. Extraction

Heavy-ions are generated by introducing material supply in plasma using electron cyclotron resonance, and the heavy-ions are extracted using an electric field (See Fig. 6.2). In RIBF, EPICS is used for these controls; the number of control points for 28 GHz SC-ECRIS is approximately 1,000 points EPICS records. In the next section, the 28 GHz SC-ECRIS, which is one of the ECRIS used in RIBF, is described.

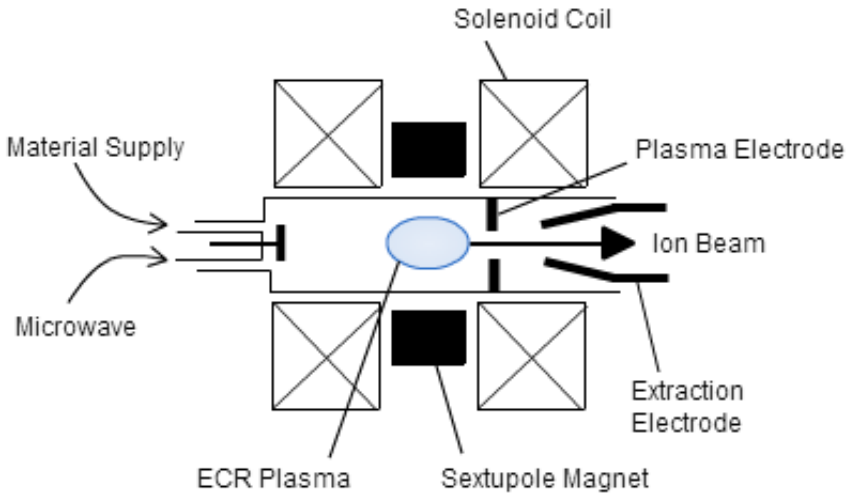

FIGURE 6.2: Outline of the ECRIS.

6.1.3 28GHz SC-ECRIS

The performance of the ion source is an important factor in determining the overall performance of a heavy ion accelerator. At the RIKEN RIBF, the 28 GHz SC-ECRIS (see Fig. 6.3, 6.4) was constructed in 2009 in an effort to increase the uranium-ion beam intensity [100, 101]. In reality, the stability of the beam is important in terms of beam service availability for experiments at RIBF, and periodic fine-tuning of parameters such as the RF power and gas pressure is required to ensure a high-intensity beam with long-term stability. Therefore, it is common for ion source developers to telephone on-site accelerator operators to check the status of the 28 GHz-SC-ECRIS when not present at the facility. However, ion source developers could obtain more detailed information regarding the status of the source, relay operational instructions, and maintain beam quality if it were possible to access the control system from outside the facility.

In addition, because the ion source includes all components (e.g., RF, vacuum, magnet, and beam diagnostics) that compose the accelerator, it is suitable as a target for test implementation of the operator intervention system and WebSocket-based OPIs via a WAN link. Thereby, the author implemented WebSocket-based OPIs and an operator intervention system for the 28 GHz SC-ECRIS control system.

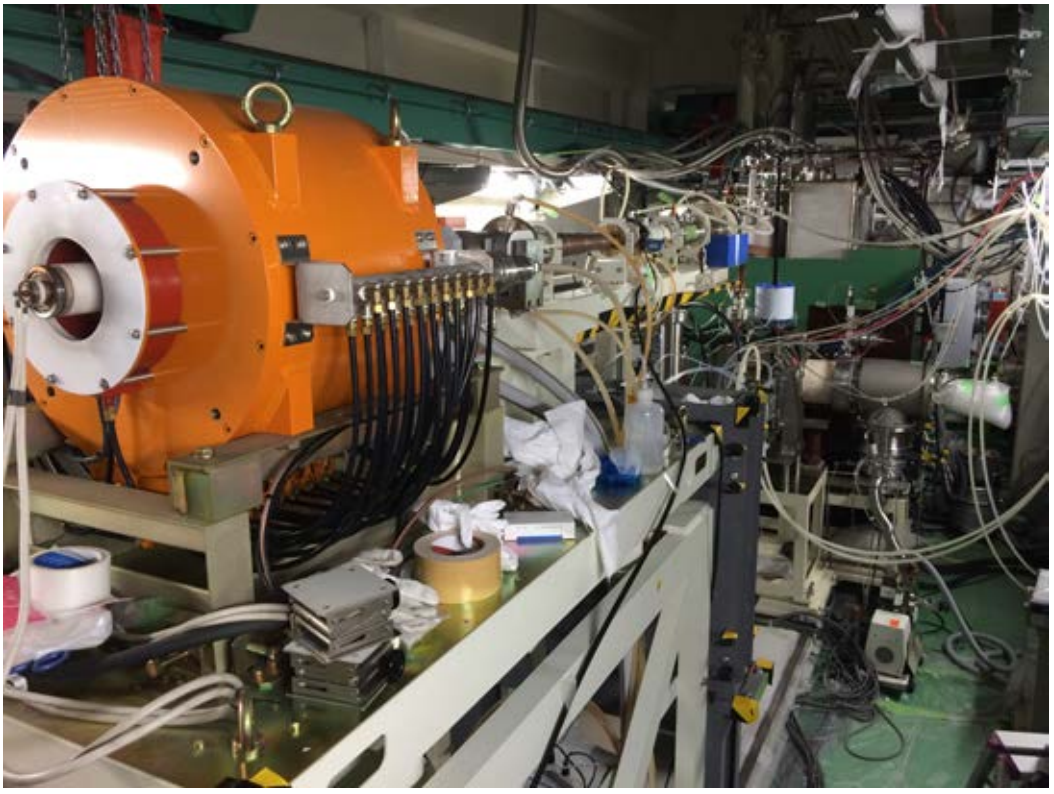

FIGURE 6.3: Photograph of the RIKEN 28GHz SC-ECRIS.

6.2 Implementation Background

As described in Chapter 4, traditional Web applications that use AJAX lack the interactivity required to act as OPI for some accelerator hardware. However, the WebSocket protocol can be implemented to improve interactive performance. This protocol achieves bidirectional communication between a Web server and a browser, unlike the periodic polling access of AJAX. Bidirectional communication can be utilized to develop an OPI that can both monitor and control ion-source devices. Based on the description in Section 4.6, the author has successfully developed a WebSocket-based client system to monitor the real-time performance of the EPICS-based system via a Web browser.

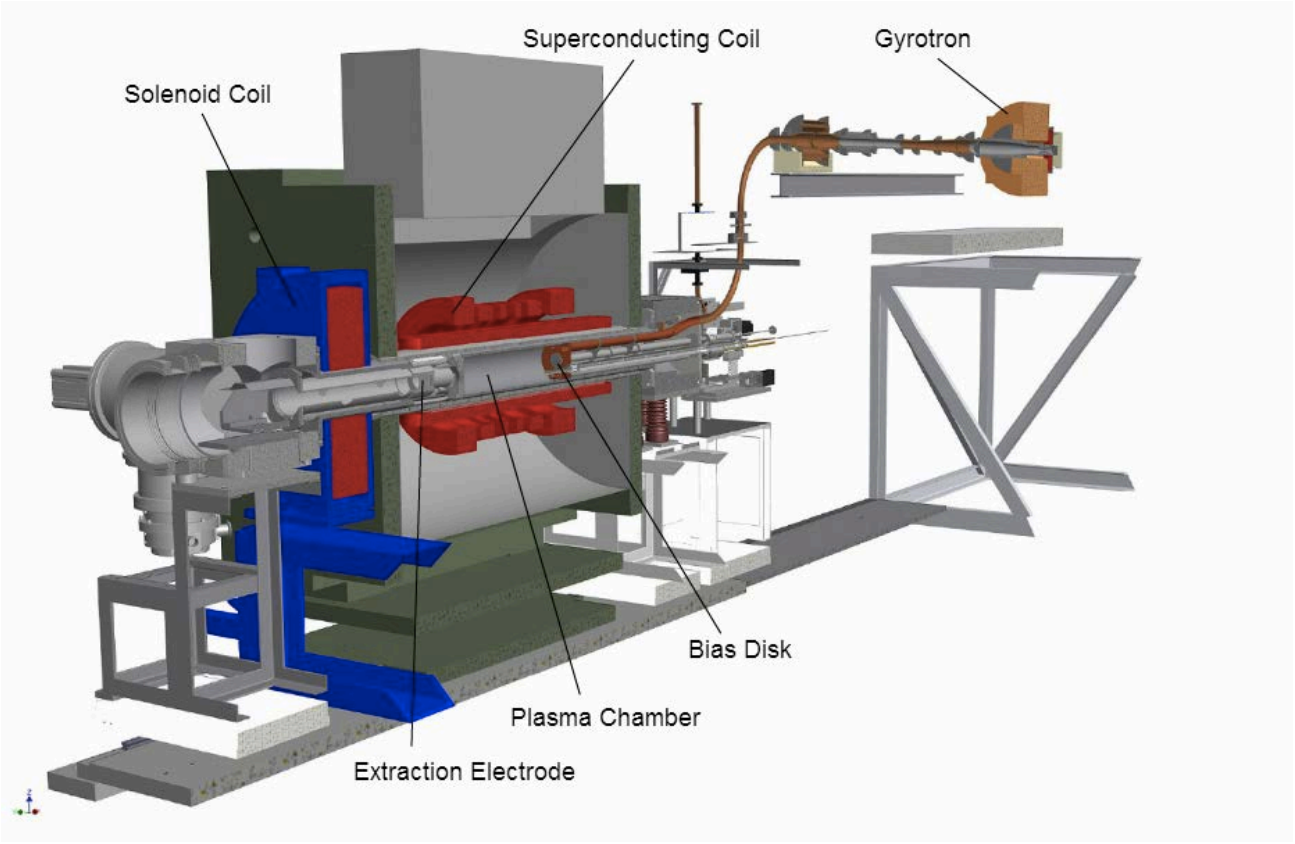

FIGURE 6.4: Section View of RIKEN 28GHz SC-ECRIS.

There are clear advantages to the developed client system; however, there are still some problems that need to be addressed, namely, hazards associated with access via WebSocket and operator errors. Focusing on the latter, operator errors can arise because it is difficult to provide a complete picture of the system status verbally over the telephone (seeing RIBF parameters on control-room PC displays is of much greater benefit) and because operating the 28 GHz SC-ECRIS without real-time response is difficult, even if remote operators can access snapshots of the GUI screens as image files. To prevent operator errors from occurring, accelerator conditions that include the 28 GHz SC-ECRIS need to be understood in their entirety. Because accelerator operators in the control room have the best understanding of the RIBF accelerator conditions, their judgment is indispensable to ensure that devices are controlled safely. On-site accelerator operators must then be allowed to intervene in remote operation. The WebSocket-based client system is designed such that output control by the remote ion source developer always requires the permission of an on-site accelerator operator who can intervene at any time.

6.3 28GHz SC-ECRIS Control System

In this section, the control system of the 28 GHz SC-ECRIS in normal operation is described for comparison with remote operation.

6.3.1 Outline of Control System for 28 GHz SC-ECRIS

Because the 28 GHz SC-ECRIS control system is based on EPICS as part of the RIBF control system, the same network system is used as that for the RIBF control system. To control the main devices, such as gas valves and rod position, in the 28 GHz SC-ECRIS, a combination of a Yokogawa F3RP61-2L module that acts as an EPICS IOC and FA-M3 PLCs (Yokogawa Electric Corp.) are adopted with the embedded EPICS technology (see Section 2.3.3). A vacuum control system and a beam diagnostic system are constructed by EPICS using Linux-based IOCs connected with N-DIMs via Ethernet [102]. The bending magnet control system comprises a VME-based IOC that communicates with NIO [33]. The outline of the 28 GHz SC-ECRIS control system is shown in Fig. 6.5.

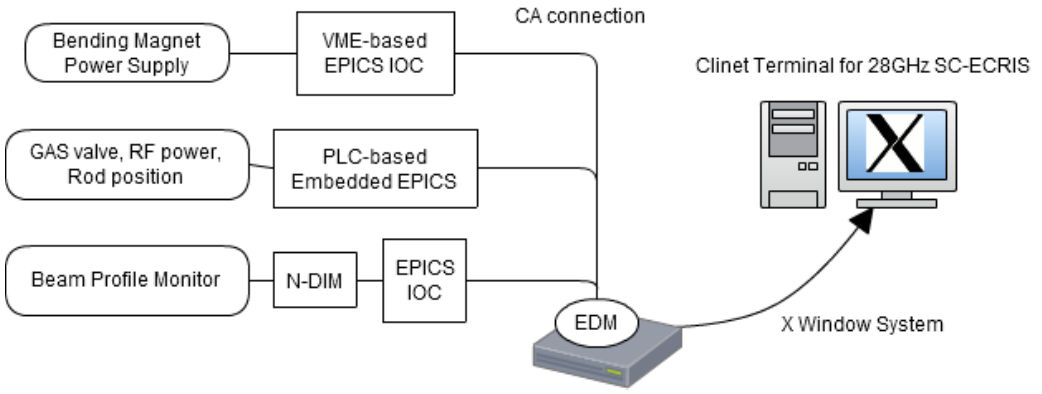

FIGURE 6.5: Outline of the 28GHz SC-ECRIS control system. Hardware dependent protocols are unified by CA protocol.

6.3.2 Client System

To facilitate information exchange among different types of controllers, the protocol for controllers such as FA-M3, N-DIMs, and NIOs is unified through a CA protocol from the client's perspective. A constructed x-y plotter chart that uses EDM as described in Section 2.4.3.1 is shown as a client in Fig. 6.6. Because there are numerous parameters related to the operation of the 28 GHz SC-ECRIS, maintaining records in a paper notebook is not feasible. To save the 28 GHz SC-ECRIS parameters in an effective manner, a paperless log system that uses wiki (see Fig. 6.7) is adopted. Wiki is a

system that can edit and create hypertext documents on a Web server through a Web browser; knowledge of HTML is not required in this case.

Using MyDAQ2 described in Section 3.2.3 as the DAQ system for the 28 GHz SC-ECRIS, EPICS values such as vacuum, extraction current, and RF power are recorded in the MySQL database every 10 s. Because graphics and text can be easily displayed in Web browsers, the Web applications have a user-friendly GUI and a Wiki-based system. DAQ based on MyDAQ2 can be used to reduce the turnaround time during system construction.

To manage differences between terminals used in the accelerator control room and remote locations, a cross-platform software environment should be adopted (for example, less common software such as the X Window System might not necessarily be installed on a remote computer, as it is on the control-room Linux PCs). In this regard, Web-based technology has the advantage that any device with a browser, e.g., a PC, mobile phone, tablet, or TV, can be used, and Web-based technology has been shown to be useful for conveying status information from the 28 GHz SC-ECRIS control system.

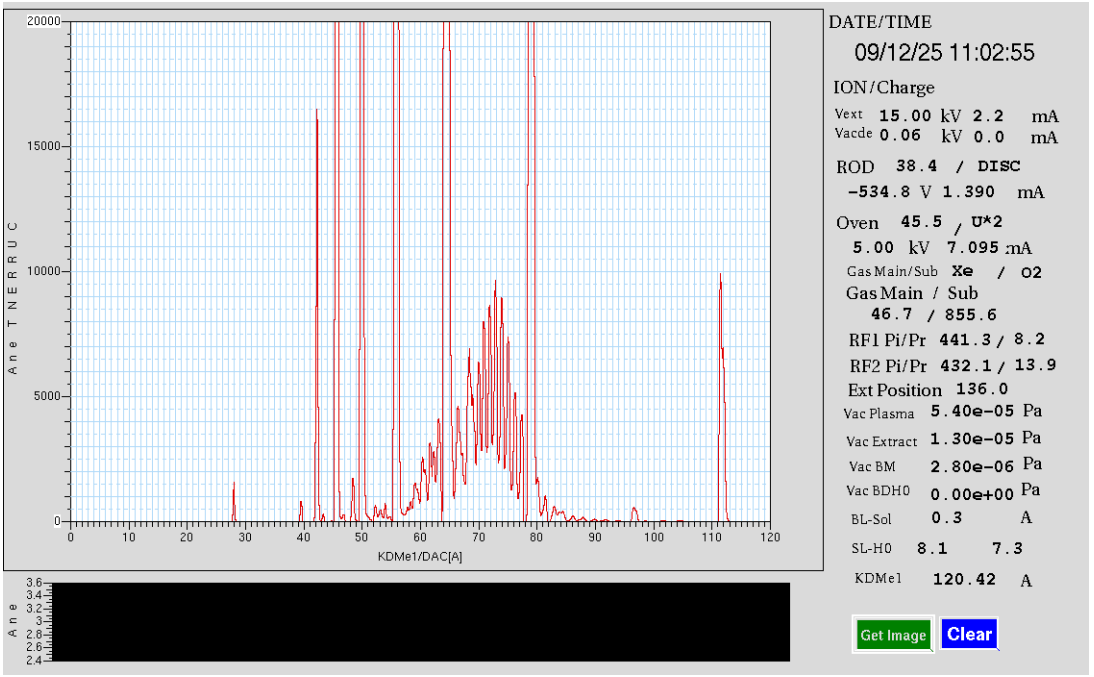

FIGURE 6.6: An x-y plotter chart application constructed by EDM.

6.3.3 Network System

A closed network is more reliable for accelerator control from the perspective of information security. The RIBF control system is also constructed on a closed network that

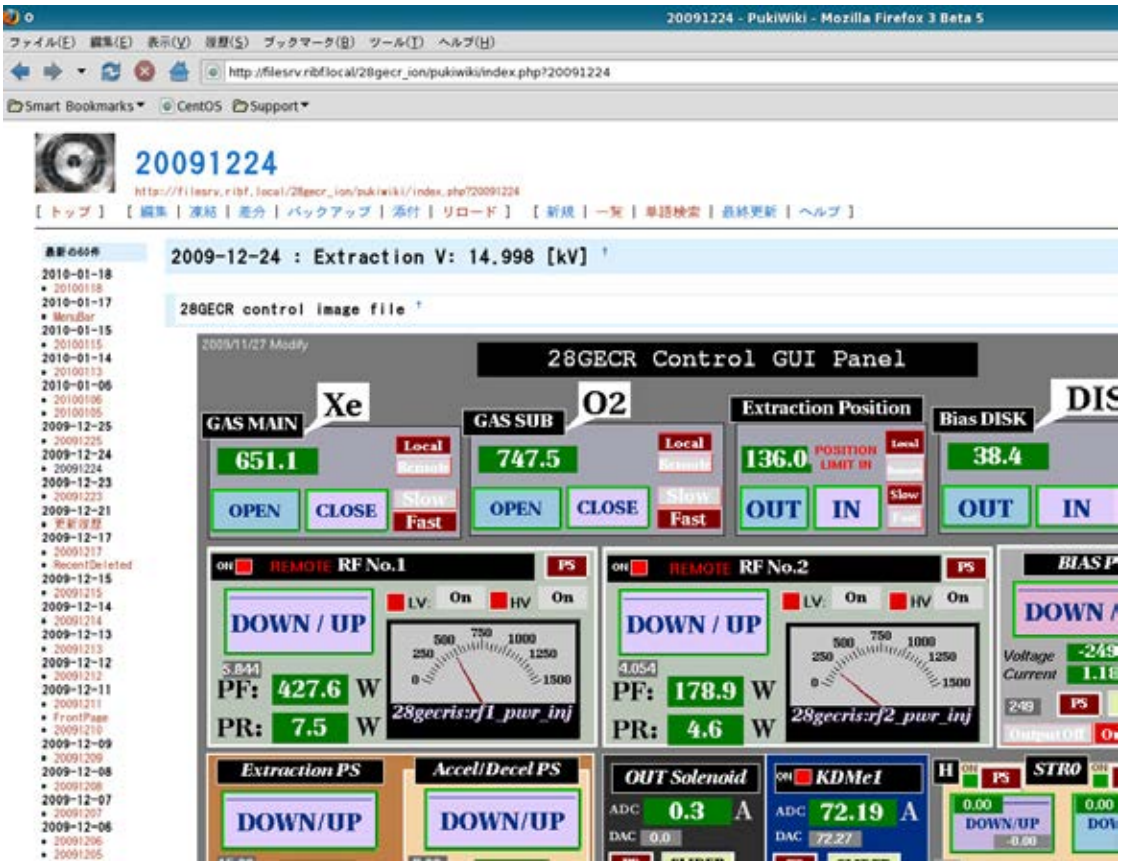

FIGURE 6.7: Wiki page with a screen capture from the 28GHz SC-ECRIS main control GUI panel.

is completely disconnected from all other networks, including laboratory Intranets used for daily research activities [103]. In this network system, it is difficult for staff to obtain parametric data, such as from the 28 GHz SC-ECRIS, from their offices because access ports to the network for EPICS are not installed in the member's offices. However, from the perspective of system management and ease of system construction, Web communication is suitable for providing information as mentioned in Section 6.2.2. To meet this requirement, a combined system with reverse proxy servers for Web communication and a firewall was constructed for providing accelerator information to the office network while ensuring secured access. Installation of reverse proxy servers in front of real Web servers is a widely used prescription for security and caching.

In order to implement the system, reverse proxy servers and firewalls were constructed. The outline of the system is shown in Fig. 6.8. The primary feature of the system is that access from the office network are masked from the Web servers used for Wiki and MyDAQ2 behind the reverse proxy servers.

Because it is possible to analyze data such as vacuum and gas pressure efficiently using Web services from offices as well as the control room, the system has allowed more

effective use of working time for 28 GHz SC-ECRIS operation [104]. An outline of the routine work in 28 GHz SC-ECRIS operation and the relationship with Web services are shown in Fig. 6.9.

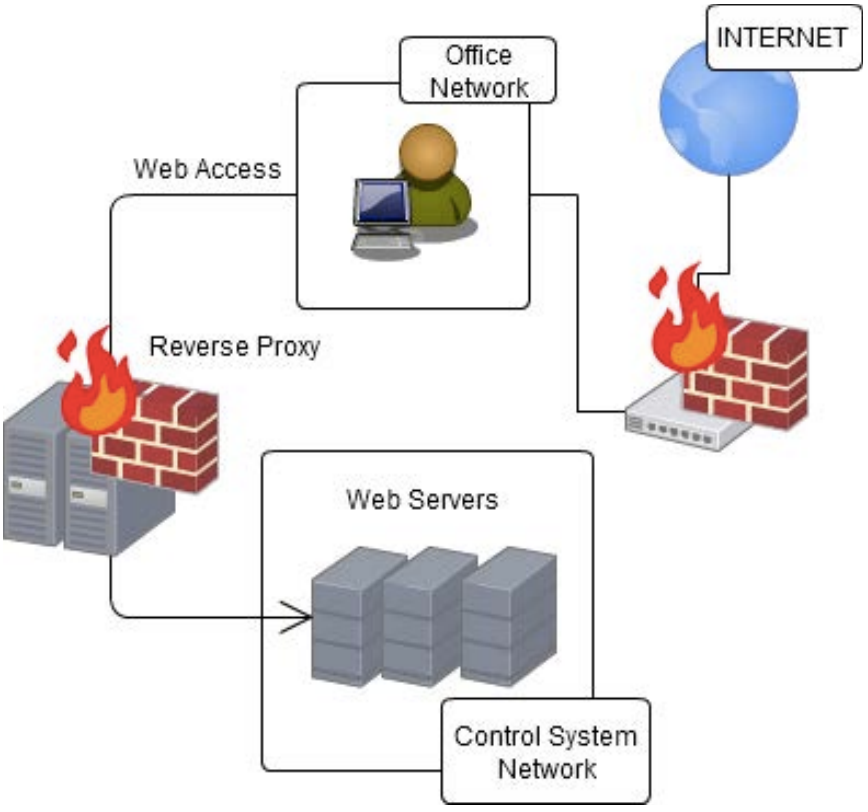

FIGURE 6.8: Network diagram of Web communication via a reverse proxy server between office network and control system network.

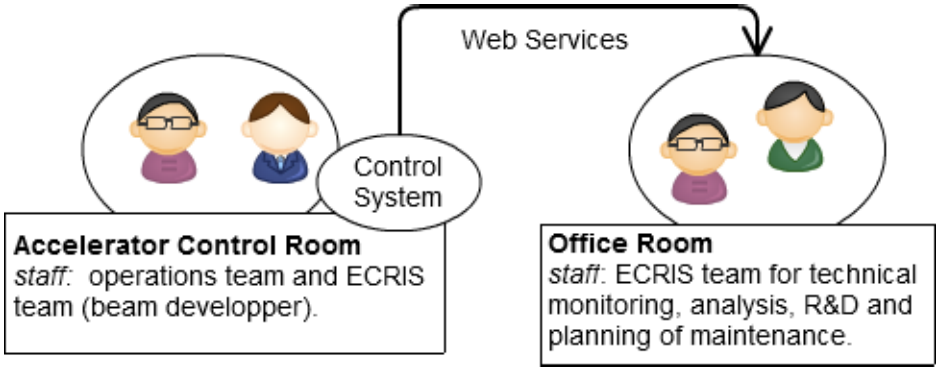

FIGURE 6.9: After operation of the 28GHz SC-ECRIS in the control room, the data are downloaded to office room via Web services for analysis and R&D.

6.4 WebSocket-based client system for 28GHz SC-ECRIS Control System

As described in Chapter 4, WebSocket enables interactive response using bidirectional communication, thus eliminating the disadvantage of traditional HTML. Web browsers such as Firefox, Chrome, and Internet Explorer can be used as a cross-platform environment with the implementation of a WebSocket-based OPI. This allows the user to access OPI not only from a PC-based Web browser, but also from other devices, such as Android and Apple tablets and mobile phones. After satisfying the conditions for interactive access to the EPICS-based system, the author implemented the WebSocket server that connects to the EPICS IOC via CA protocol as a Web-based OPI for 28 GHz SC-ECRIS operation. Figure 6.10 shows a chart of the whole system. The WebSocket-based client system is comprised of an HTTP server, HTML and JavaScript files, and a WebSocket server. The WebSocket server obtains the parameters as PVs from EPICS IOC by calling the CA API.

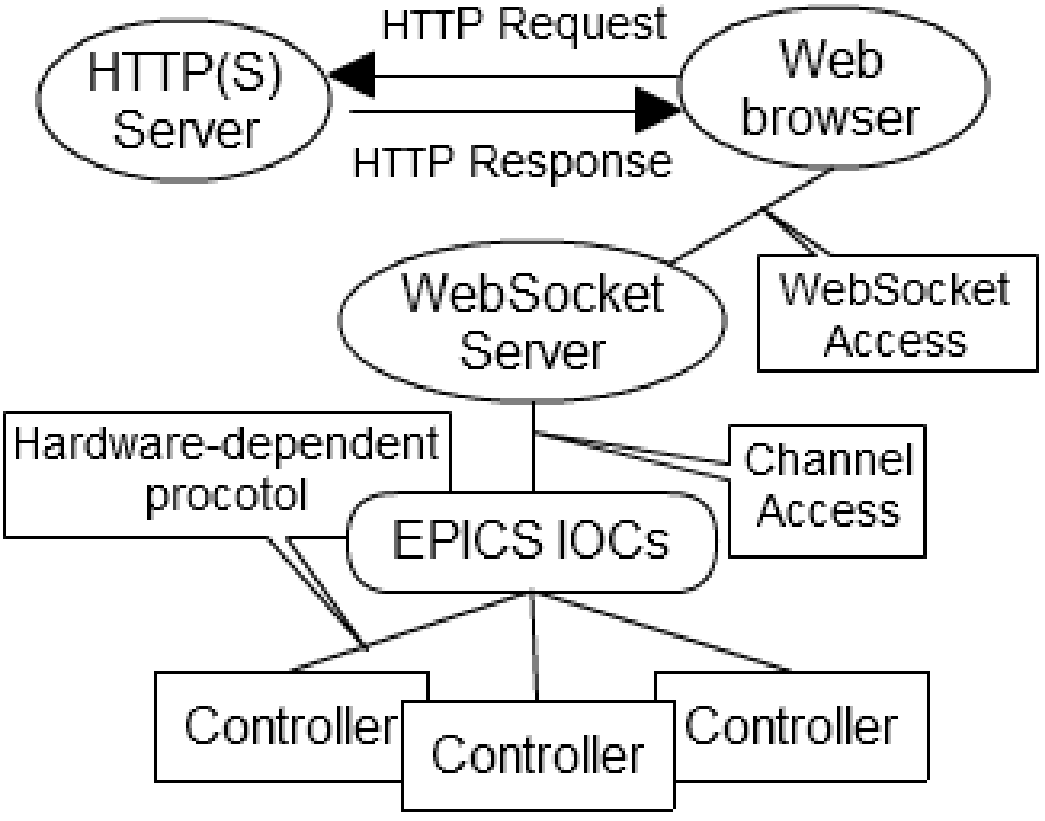

FIGURE 6.10: System diagram of WebSocket-based OPI in EPICS-based control system.

6.5 Implementation Results

As described in Chapter 5, the operator intervention system consists mainly of a PV gateway provided by EPICS collaboration, a database, and Web applications. An outline of the implemented system is as follows. First, on-site accelerator operators grant permission to each PV output that corresponds to the hardware of the 28 GHz SC-ECRIS control system. If PVs have not been granted permission, the operator intervention system always rejects the output command at the CA protocol layer. GUI for the WebSocket-based OPI is shown in Fig. 6.11. Operation requests to on-site operators are sent by clicking the buttons that correspond to the EPICS PVs. In addition, on-site accelerator operators can check access logs for all remote operations with output devices and monitor numerical values for the EPICS PVs.

Remote operation of the 28 GHz SC-ECRIS control system was tested using a WAN emulator with remote users and on-site operators in the same control room, but accessing different network environments [105] (see Fig. 6.12). The WAN emulator creates another network with variable performance. As a result, the ion source developer was successfully able to operate part of the 28 GHz SC-ECRIS (gas valves, RF power, position of an electrode, and so on) remotely without serious problems.

Using this system, the author confirmed that output instructions did not reach the EPICS IOC without permission from the on-site operator. In addition, remote operation was performed satisfactorily with a network latency of 200 ms, which is approximately the same network latency between Japan and the USA. Therefore, networking performance tolerance should allow this system to be used over networks provided by almost all Japanese Internet service providers [106].

The author has to note that "debugging" HTML and JavaScript programs proves to be more important for preventing operational errors than other general Web applications. In the case of EPICS native OPI coded in the C language, the compiler should verify incomplete syntax of source codes. However, even when the HTML or JavaScript code has a "bug", the code can be run in a Web browser in some situations.

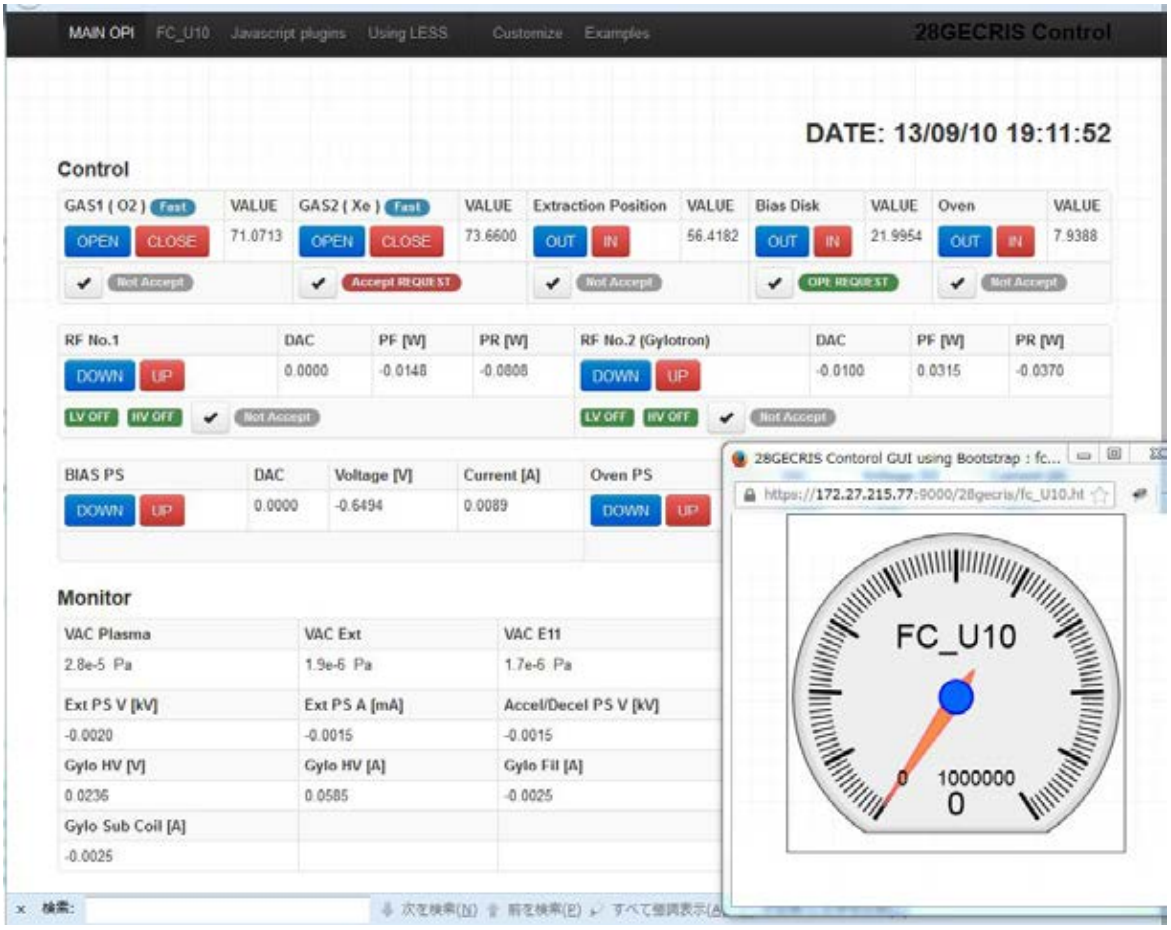

FIGURE 6.11: User interface of WebSocket-based OPI. By clicking on buttons, operation requests are sent to the on-site operators.

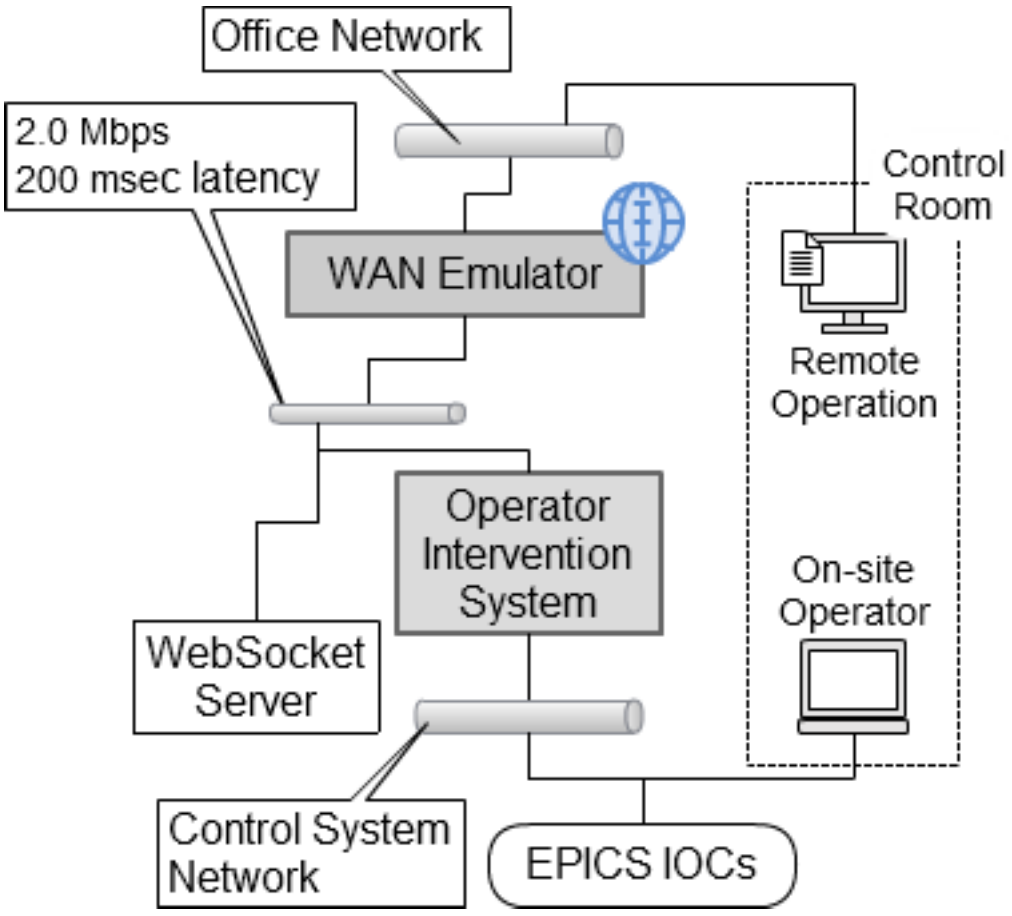

FIGURE 6.12: Test environment with another network-emulated WAN. Gray area indicates implemented system.

Chapter 7

Discussion

7.1 Performance Measurement of WebSocket-based OPI

For comparison of performance in remote operation, network bandwidth was measured for WebSocket-based OPI and AJAX-based OPI. In order to measure the network bandwidth, OPIs for the 28 GHz SC-ECRIS operation that is described in Chapter 6 were used. A feature of the WebSocket server developed with the Socket.IO library is that the system behaves as AJAX access when a WebSocket-unsupported browser connects to the WebSocket server. Utilizing the feature of Socket.IO, the difference in network bandwidth obtained from accessing the system in two different ways, AJAX and WebSocket, were compared with the same OPI. In OPI, 30 numerical values, such as gas valves positions, rod position, and extraction power supply, were monitored at intervals of 0.1 s. Then, four numerical values for vacuum were monitored at intervals of 4 s. In this paper, an example of a response through WebSocket and AJAX access is described below.

```
1 5::/28gecris/index:{"name":"ca","args":[{"gref_pwr_v":"-0.0325"}]}
```

Listing 7.1: Example of responce data in the WebSocket server.

The same data frame is received by the Web browser regardless of the difference in access methods. First, the author measured latency in the case of AJAX access to OPI using Wireshark [107], which is a network packet analyzer software. In the case of UI for the remote operation of 28 GHz SC-ECRIS, the result of the AJAX-polling measurement through LAN is a latency of 0.5 s that was observed on average upon one access.

In order to test in a network compatible with the network used for implementation of the operator intervention system described in Chapter 5, a WAN emulator was used as a measuring instrument.

7.1.1 Network Bandwidth Measurement for WebSocket

In this study, to measure network bandwidth, a WAN emulator (Linktropy Mini2) produced by Apposite Technologies, Inc. [108] was used. Across the WAN emulator, LAN A shows the network of the virtual server side; LAN B shows the office network (see Fig. 7.1). The author measured the bandwidth of WebSocket-based OPIs through LAN using Google Chrome version 34 as the Web browser. The graph in 7.2 shows the network bandwidth when a single client is connected to the WebSocket server to monitor accelerator parameters.

The graph in Fig. 7.3 shows that the bandwidth requires 84.39 kbps with an average of 10 min from LAN A to LAN B in the case of a single client. On the other hand, network bandwidth to LAN B from LAN A requires 22.42 kbps only with an average of 10 min.

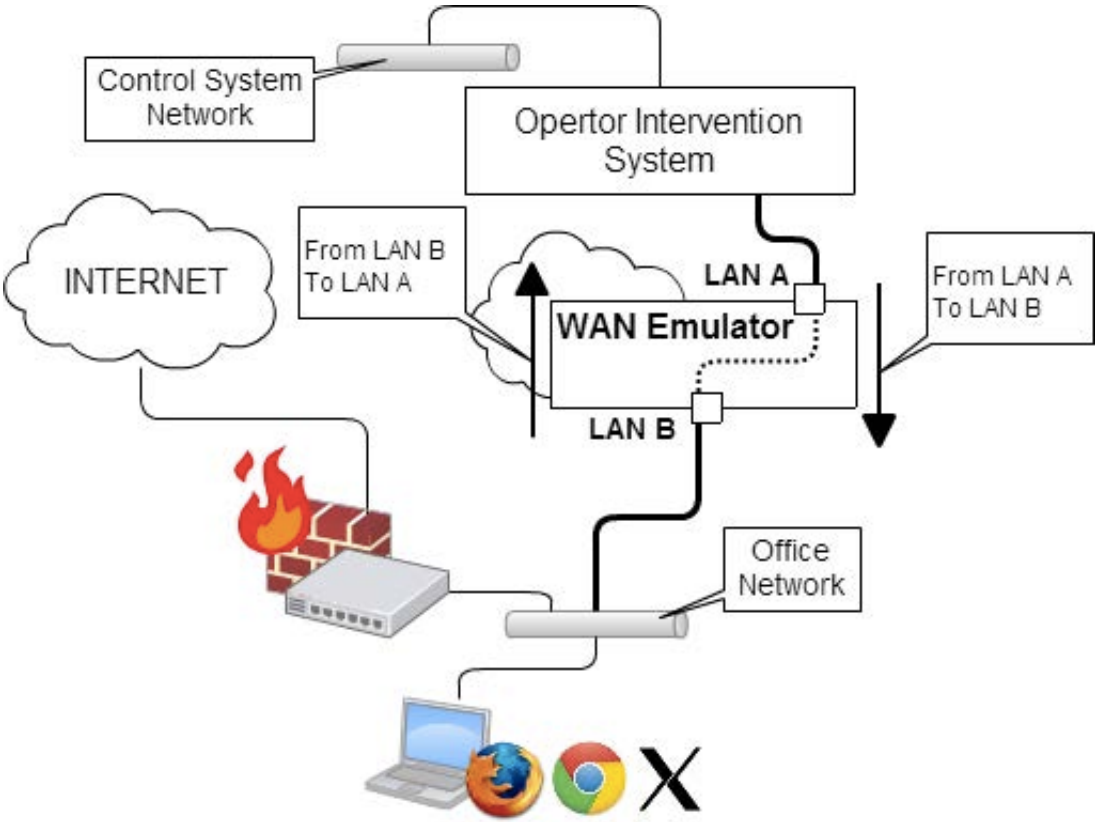

FIGURE 7.1: Network chart for measuring network bandwidth.

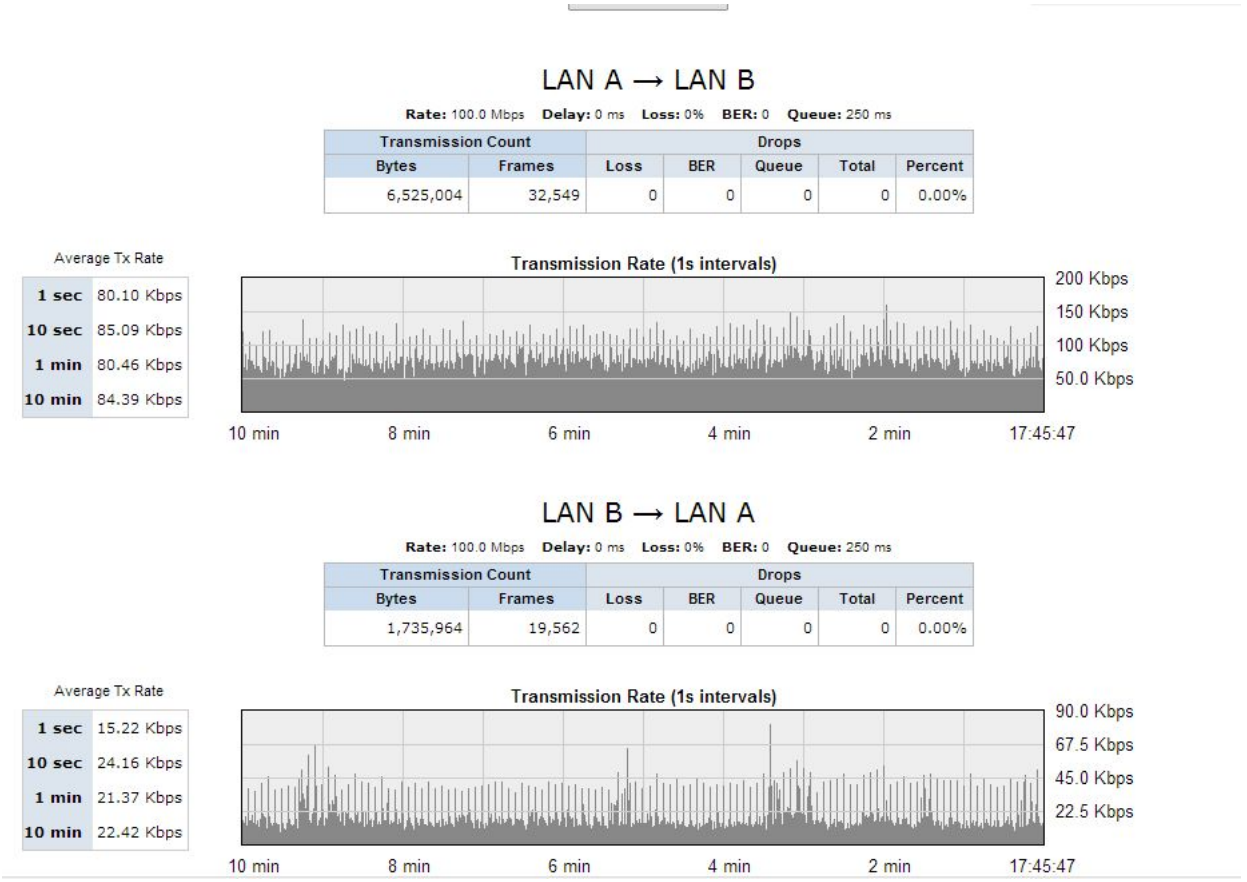

FIGURE 7.2: Graph of network traffic for WebSocket-based OPI.

7.1.2 Network Bandwidth Measurement for AJAX

In the case of AJAX-based OPI, the network bandwidth was measured in terms comparable to the WebSocket-based OPI described in Section 6.5. Mozilla Firefox version 3.6.13 for Linux was adopted as the Web browser. The graph in Fig. 7.3 shows the network bandwidth when a single client is connected to the WebSocket server with AJAX to monitor accelerator parameters.

As shown in the graph, the bandwidth requires 42.0 kbps with an average of 10 min from LAN A to LAN B in the case of AJAX access from a single client. On the other hand, network bandwidth to LAN B from LAN A requires 83.51 kbps with an average of 10 min.

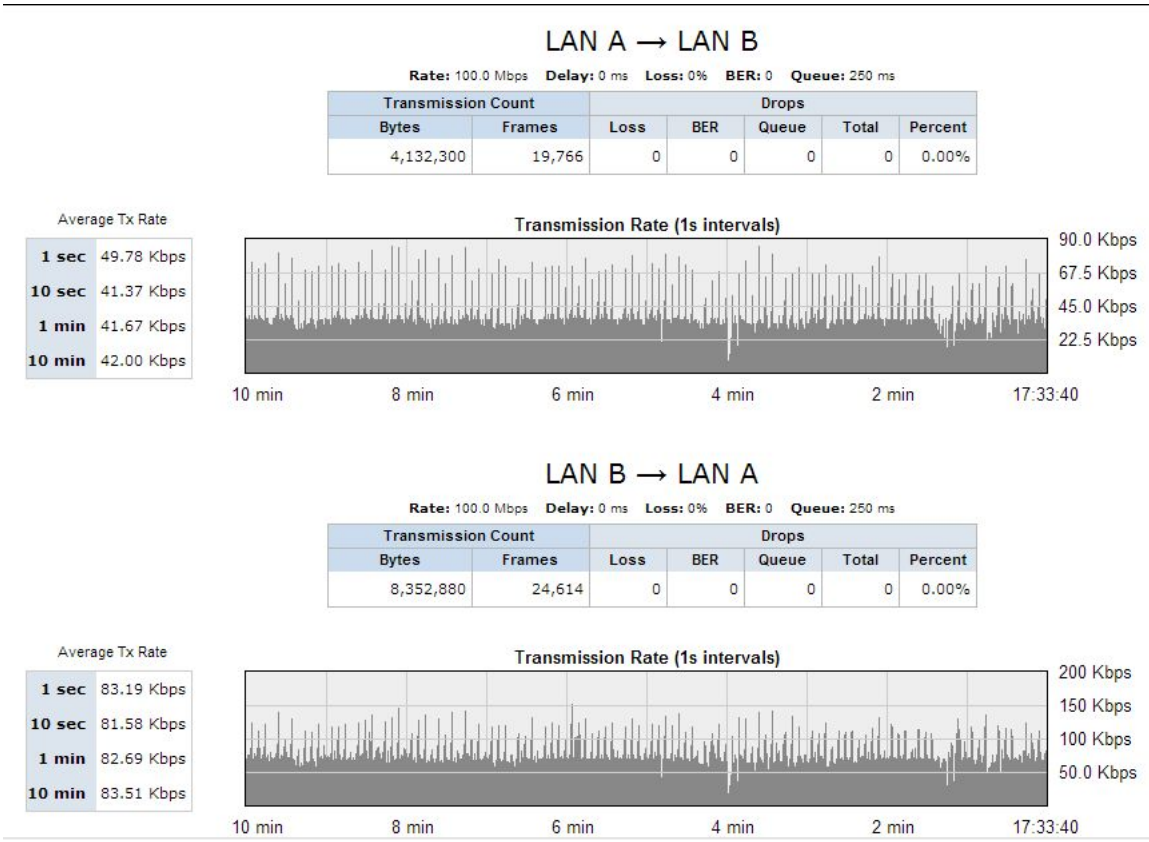

FIGURE 7.3: Graph of network traffic for AJAX-based OPI.

7.1.3 Network Bandwidth Measurement for X Window System

As described in Section 1.3.4, the X Window System is widely used for OPIs by exporting GUIs to a remote computer in the case of Web technology. In reality, the main GUI panel for the 28 GHz SC-ECRIS control is constructed with EDM provided by EPICS collaboration (see Fig. 2.13). Because the specifications of the protocol are completely different from each other, it is difficult to compare them directly. However, to confirm the advantage of WebSocket-based OPIs, network bandwidth for EDM-based OPIs, which are implemented for the 28 GHz SC-ECRIS daily operation, were measured. For the X server, Cygwin-X that runs on Microsoft Windows was adopted, and the X client (used by EDM installed in a virtualization server) is based on CentOS 5.9. This EDM-based OPI has approximately the same number of monitored parameters compared with WebSocket-based OPI for 28 GHz SC-ECRIS.

The graph in Fig. 7.4 shows the network bandwidth in the case of a single EDM-based OPI. This graph shows that more than approximately 140 kbps is necessary as network bandwidth to display the 28 GHz SC-ECRIS main GUI panel via the X Window System.

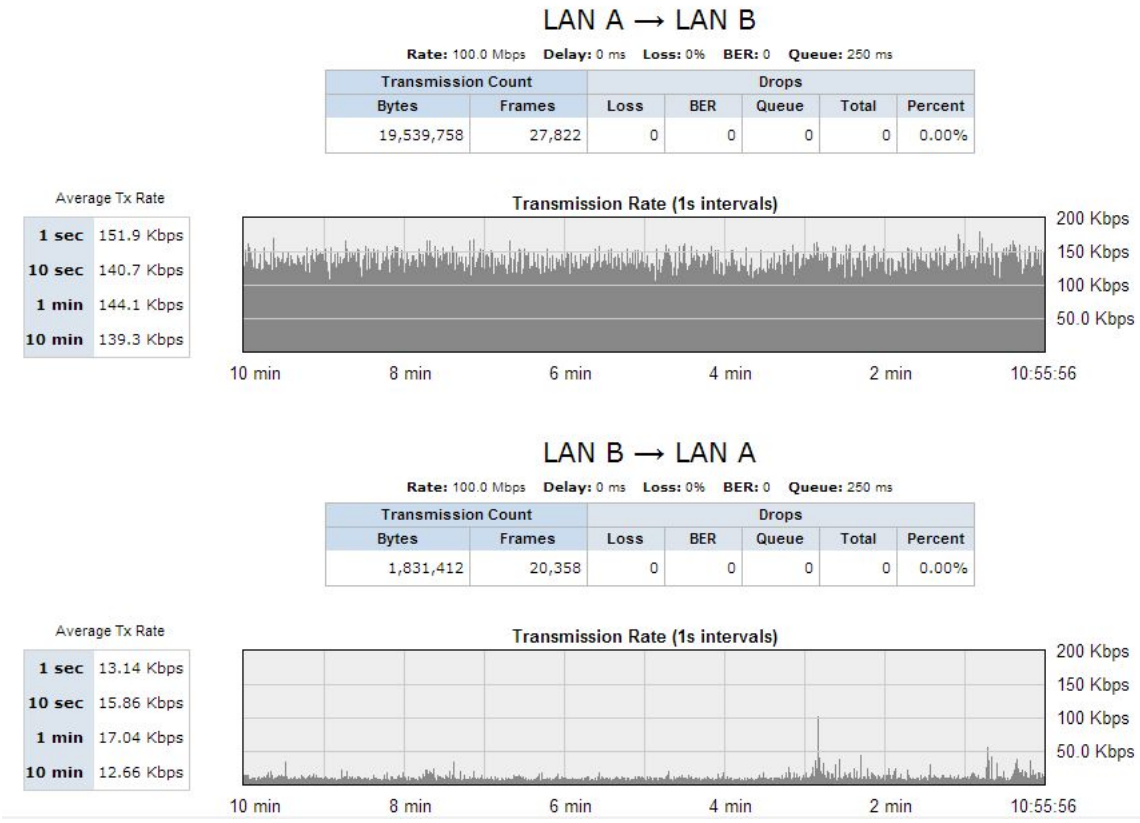

FIGURE 7.4: Graph of network traffic for OPI constructed with EDM based on X Window System.

7.1.4 Measurement Results

The measurement results from requests to the WebSocket server show that AJAX-based OPI requires eight times more network bandwidth compared to WebSocket-based OPI. However, AJAX-based OPI can receive only half the data volume that WebSocket-based OPI receives. On the other hand, almost all the values to be monitored by EPICS IOC are changed at intervals of 0.1 s (SCAN 0.1 second), and one AJAX communication requires approximately 0.5 s to receive data in LAN after transmitting data requests. Therefore, this means that a 0.5 s delay occurs regardless of LAN-links, and AJAX-based OPI can receive only half of the data volume because timing of the AJAX transmission is too slow compared with timing of the SCAN in EPICS IOCs.

Moreover, if this AJAX-based OPI is used via WAN-links, it is clear that the delay increases. For this reason, for AJAX-based systems, although a large amount of network bandwidth is necessary for transmitting requests, a few amount of data is received. Thereby, AJAX-based OPI is an inefficient method compared to WebSocket-based OPI.

On the other hand, EDM-based network traffic shows that the network bandwidth for data requests requires only approximately 13 kbps; however, a relatively large 140 kbps is necessary to receive data. This is because an event for the X Window System is not generated if the operator does not move a window or the mouse cursor, or clicks the mouse. On the other hand, the X client sends packets to the X server to request drawables. The network bandwidth for sending packets (from LAN A to LAN B) is increased by the increase of CA events because requests for drawables are sent to the X server by changing the record value on EPICS IOC.

As a result, AJAX-based OPI is not suitable for fast responses, such as “field(SCAN,“.1 second”),” in EPICS Runtime DB, and it is difficult to use EDM-based OPI using the X Window System from WAN-links because data received for the X server is a large frame. Thereby, the author considers that WebSocket-based OPI should be implemented for the most efficient remote operation.

7.2 Future Improvements of WebSocket-based OPI

As described in Section 4.5, the author developed Node-CA, which is interfaced between CA and Node.js by calling CA API. However, the current version Node-CA implements caMonitor, caGet, and caPut, which are the main functions of CA API. In particular, functions related to arrays, such as the caGet array or the caPut array, were not implemented in this study because these functions have no need for 28 GHz SC-ECRIS control. In the future, it is necessary to update Node-CA for full implementation of other EPICS-based applications coded in Node.js.

Moreover, existing GUI managers, such as CSS/BOY and MEDM, have strong points in EPICS-based control systems, such as software productivity, because there is no need to write code such as drawing software; therefore, it is possible to construct GUI applications for EPICS-based control systems. On the other hand, compared with the development of X applications and Java Swing-based applications, there is a small amount, but WebSocket-based OPI is necessary for coding in HTML in the case of this study. In order to broaden the base of users who employ the WebSocket-based OPI, the author would like to develop a display manager that supports WebSocket-based OPI.

The software that is a result of this study is expected to be released as open source software in the near future.

7.3 Challenges for the future

For remote operation of accelerators, the author confirmed the effectiveness of WebSocket-based OPI in Section 6.4. On the other hand, data archive systems are widely used for daily accelerator operation in local control rooms. As described in Section 3.2.5, Web-based systems are used for data archive viewers. In order to determine the cause of accelerator conditions by comparing different types of archived data, data archive viewers are one of the frequently used applications in accelerator control systems.

However, there is an issue for using data archive viewers via remote operation because the amount of data transfer increases if the search scope is spread. For example, a considerable amount of time might be required to search and display a year's worth of vacuum data via WAN-links. In this case, the author considers that using machine learning is an effective method for searching and transferring data via WAN-links.

In many cases, operators utilize data archivers to determine the time when data changed and the amount by which such data changed, for example, vacuum values.

Thus, if operators can determine the time when such data changed, it is possible to reduce the amount of data transferred.

For this purpose, utilizing the framework of machine learning that has a function for anomaly detection, such as Jubatus [109], will be a key point for constructing an effective system.

Chapter 8

Conclusion

From the need for operation and maintenance, some accelerator research facilities have studied wide area environments for remote operation since 2000. Such facilities adopted a mechanism that requires considerable system resources, for example, dedicated networks and special-purpose terminals, for this remote operation. The purpose of this study is to perform the implementation of remote operation in control systems using EPICS as middleware, even in environments with limited resources.

On the other hand, Web-based systems have multi-purpose potential in local area networks; therefore, some projects have attempted to develop OPI using traditional Web technology, although their main purpose is to monitor accelerator parameters through static access. However, Web-based OPI that uses traditional technology, such as AJAX, is not always available for use in wide area environments because previous systems have disadvantages in interactive performance. Thereby, the author proposed a hypothesis that Web-based systems might be strong candidate methods for remote operation if the shortcomings of traditional Web-based OPIs can compensated through this study. For this reason, WebSocket-based OPIs for EPICS-based control systems and an operator intervention system, which is a new approach compared with previous works, was developed in this study; moreover, the proposed WebSocket-based OPIs increase software portability and reusability dramatically without damaging the advantage of Web-based systems.

In addition, local control rooms have aggregated information; therefore, the judgment of on-site operators is extremely important from the perspective of safety systems. To solve this issue in remote operation via WebSocket, the author discussed the development and implementation of an operator intervention system for remote operation using WebSocket-based OPIs. This intervention system ensures that operational errors are minimized and that the accelerator is operated safely during remote operation. Because

the system was implemented for RIKEN 28 GHz SC-ECRIS remote operation, the author confirmed that this new operator intervention system via WebSocket-based OPIs is a useful method for ensuring the physical security of operational errors.

On the other hand, remote participation according to a multi triangle-model and the GAN operational model, which is an extensive remote operation of the facility, were discussed for the GAN project. The GAN operational model means a relationship between accelerator control rooms, accelerator technical infrastructure, and home laboratories for collaboration; such a relationship is required for the effective direction of significantly large physics projects, such as ILC.

- In virtual accelerator control rooms, multiple operators from the operation team are deployed.
- For repair and maintenance, the accelerator technical infrastructure located near to the accelerator is used by a small core team.
- Experienced staff (or teams) work in technical monitoring, fault analysis, R&D,and planning of repair and maintenance in home laboratories.

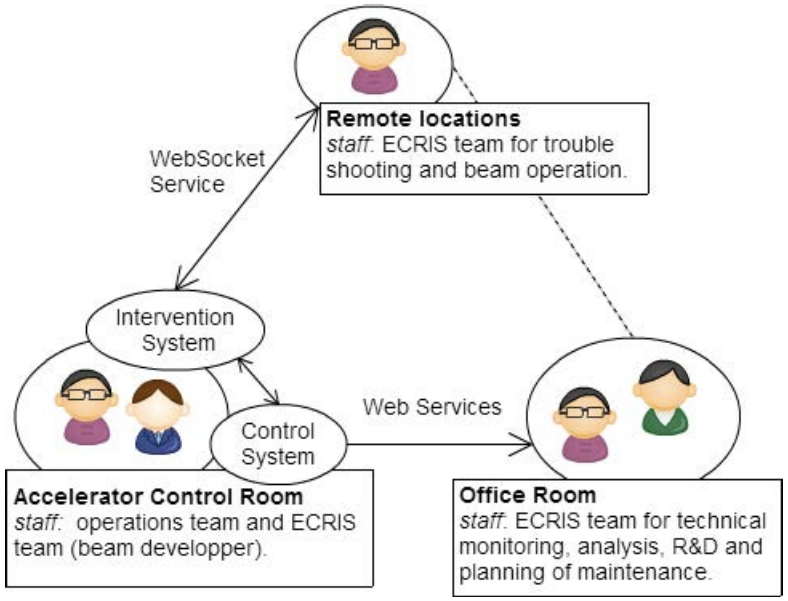

FIGURE 8.1: Remote operation model using WebSocket and operator intervention system in this study.

One of the continued efforts of the GAN project, efficient communication protocols such as remote method invocation and messaging services, was discussed. On the other hand, WebSocket-based OPIs and the operator intervention system were successfully implemented for the remote operation of 28 GHz SC-ECRIS control system in this

study. As shown in Fig. 8.1, this system consists of a counterpart system based on the GAN operational model, although the system scale is much different between the GAN model and the 28 GHz SC-ECRIS remote operation.

Therefore, the proposed method, which combines HTTP and WebSocket, is a primary candidate as the communication layer used on the Internet for remote operation. Because of advantages in traditional implementation and new implementation with WebSocket, remote operation needs to be realized through a suitable approach that uses both the HTTP and WebSocket protocols. For example, for implementation ease, traditional Web applications should be adapted if a fast response is not required for remote operation. For these reasons, a combination of HTTP and WebSocket, and the on-site operator intervention are essential methods for remote operation.

Finally, the author hopes that this system concept and system design become part of the absolutely necessary features of standard models for control systems in future accelerator projects.

Appendix A

Achievement

A.1 International Conference

- Akito UCHIYAMA, Kazuro Furukawa, and Yoshihide Higurashi, "Operator Intervention System for Remote Accelerator Diagnostics and Support" Proc. of ICALEPCS2013, San Francisco, CA, USA, Oct. 2013, pp.832-835, TUPPC110. (Poster)
- Akito Uchiyama, Kazuro Furukawa, Yoshihide Higurashi, and Takahide Nakagawa, "Implementation of an Operator Intervention System for Remote Control of the RIKEN 28GHz Superconducting ECR Ion Source", Proc. of ICIS2013, Chiba, Japan, Sep. 2013, (Poster) (To be published in Review of Scientific Instruments)
- Akito Uchiyama, Kazuro Furukawa, and Yoshihide Higurashi, "EPICS Channel Access Using WebSocket", Proc. of PCaPAC2012, Kolkata, India, Dec. 2012, pp.7-9, (Oral)
- Akito Uchiyama, Kazuro Furukawa, Yoshihide Higurashi, Kazutaka Ozeki, Misaki Komiyama, and Takahide Nakagawa, "Design of Web-based Interface to RIKEN 28 GHz Super-conducting ECR Ion Source and the Future Plan", Proc. of ECRIS2012, Sydney, Australia, Sep. 2012, (Poster)

A.2 Domestic Conference

- Akito Uchiyama, Kazuro Furukawa, Yoshihide Higurashi, Takahide Nakagawa, "Development of Operator Intervening System for remote Accelerator diagnostics and support", Proc. of Particle Accelerator Society of Japan 2013, Nagoya, Japan, Aug. 2013, (Poster)

- Akito Uchiyama, Kazuro Furukawa, Yoshihide Higurashi, "Development and Implementation of EPICS Channel Access Client with Real-time Web using WebSocket", Proc. of Particle Accelerator Society of Japan 2012, Toyonaka, Japan, Aug. 2012, pp.218-221, (Oral)

A.3 Award

- Oral Presentation Award (Particle Accelerator Society of Japan, 2013).

Appendix B

Source Code

B.1 Channel Access for Node.js

B.1.1 nodeca.cc

```
1  //=====================================================================//
2  //  2012-04-03 [ A. UCHIYAMA ]          //
3  //  File: nodeca.cc ver.0.3.2         //
4  //  The Graduate University for Advanced Studies (KEK)    //
5  //  mailto:akito12@gmail.com          //
6  //  Required node.js version is v0.6.11 or later.     //
7  //=====================================================================//
8  #include <queue>
9
10 // node headers
11 #include <v8.h>
12 #include <node.h>
13 #include <uv-private/ev.h>
14 #include <pthread.h>
15 #include <unistd.h>
16
17 #include <stddef.h>
18 #include <stdlib.h>
19 #include <stdio.h>
20 #include <string.h>
21 #include <cstring>
22
23 //EPICS
24 #include "cadef.h"
25 #define MAX_BUF_LEN 512
26
27 using namespace node;
28 using namespace v8;
29
30
31 // handles required for callback messages
```

```
32 static pthread_t thread_id;
33 static ev_async eio_tcamonitor_notifier;
34
35 Persistent<String> callback_symbol;
36 Persistent<Object> module_handle;
37
38 std::queue<char*> msg_queue = std::queue<char*>();
39 pthread_mutex_t queue_mutex = PTHREAD_MUTEX_INITIALIZER;
40
41 struct thread_data{
42   const char *_time;
43   const char *_pv;
44   chid _channelId;
45 };
46
47 const char* ToCString(v8::String::Utf8Value& value) {
48   return *value ? *value : "<string conversion failed>";
49 }
50
51 //------------------------//
52 //    caGet method        //
53 //------------------------//
54 Handle<Value> caGet (const Arguments& args) {
55   HandleScope scope;
56   const char *usage = "usage: caget(record_name)";
57   if( args.Length() != 1 || !args[0]->IsString() ){
58
59     printf ("node-ca: %s \n",usage);
60           return ThrowException(Exception::Error(String::New("Node-ca: Bad
     argument.")));
61   }
62   v8::String::Utf8Value str(args[0]);
63
64   //EPICS Channel Access using C lang
65   double dData;
66   int iData;
67   char sData[128];
68   double *adData;
69   chid node_chid;
70   int status, result;
71   SEVCHK(ca_context_create (ca_disable_preemptive_callback),"ca_context_create
     failure");
72   SEVCHK(ca_create_channel (*str, NULL, NULL, 10, &node_chid),"ca_create_channel
     failure");
73   status = ca_pend_io (5.0);
74   if (status != ECA_NORMAL){
75     return ThrowException(Exception::Error(String::New("Node-ca: Channel connect
     timed out.")));
76   }
77
78   switch (ca_field_type (node_chid)){
79     case DBR_ENUM:
80     case DBR_CHAR:
81     case DBR_STRING:
82     result = ca_get (DBR_STRING, node_chid, (void *)&sData);
```

```
83      status = ca_pend_io (.5);
84      if (status != ECA_NORMAL){
85        return ThrowException(Exception::Error(String::New("Node-ca: Channel
        connect timed out.")));
86      }
87      return String::New(sData);
88            break;
89
90          case DBR_DOUBLE:
91      case DBR_FLOAT:
92      result=ca_get(DBR_DOUBLE,node_chid,(void *)&dData);
93      if(result != ECA_NORMAL ){
94        return ThrowException(Exception::Error(String::New("Node-ca: ca_get Failure
        .")));
95            }
96            status = ca_pend_io (.5);
97            if (status != ECA_NORMAL){
98          free (adData);
99        return ThrowException(Exception::Error(String::New("Node-ca: Channel
        connect timed out.")));
100           }
101     return Number::New((double)dData);
102           break;
103
104         case DBR_SHORT:
105         case DBR_LONG:
106           SEVCHK(ca_get (DBR_LONG, node_chid, (void *) &iData),"ca_get failure");
107           status = ca_pend_io (.5);
108           if (status != ECA_NORMAL){
109       return ThrowException(Exception::Error(String::New("Node-ca: Channel
        connect timed out.")));
110     }
111     return Number::New((long)iData);
112           break;
113
114     case TYPENOTCONN:
115     return ThrowException(Exception::Error(String::New("Node-ca: Channel is not
        connecting (TYPENOTCONN).")));
116           break;
117
118     default:
119     return ThrowException(Exception::Error(String::New("Node-ca: undefined type."
        )));
120       }
121     return String::New("OK");
122 }
123
124 //-----------------------//
125 //    caPut method       //
126 //-----------------------//
127 Handle<Value> caPut (const Arguments& args) {
128   HandleScope scope;
129   const char *usage = "usage: caput(record_name,val)";
130   if( args.Length() != 2 || !args[0]->IsString() ){
131
```

```
132     printf ("node-ca: %s \n",usage);
133           return ThrowException(Exception::Error(String::New("Node-ca: Bad
        argument.")));
134   }
135
136   v8::String::Utf8Value str1(args[0]);
137   v8::String::Utf8Value str2(args[1]);
138
139   char  *wdata = *str2;
140   chid  node_chid;
141   int status;
142         int success_val;
143
144   SEVCHK(ca_context_create(ca_disable_preemptive_callback),"ca_context_create
        failure");
145   SEVCHK(ca_create_channel(*str1,NULL,NULL,10,&node_chid),"ca_create_channel
        failure");
146   status = ca_pend_io(0.5);
147
148   if (status != ECA_NORMAL){
149     return ThrowException(Exception::Error(String::New("Node-ca: Channel connect
        timed out.")));
150   }
151
152         success_val=ca_put(DBR_STRING,node_chid,(void *)wdata);
153         if (success_val != 1){
154                 ca_context_destroy();
155     return ThrowException(Exception::Error(String::New("Node-ca: ca_put failure."
        )));
156   }
157         status = ca_pend_io(0.5);
158   if (status != ECA_NORMAL){
159     return ThrowException(Exception::Error(String::New("Node-ca: Channel connect
        timed out.")));
160   }
161   ca_context_destroy();
162   return String::New("OK");
163 }
164
165 //---------------------------//
166 //    camonitor method       //
167 //---------------------------//
168
169 void monitorFunc( struct event_handler_args args ){
170   chid node_chid = args.chid;
171     char *pname = strdup(ca_name(node_chid));
172     char data[MAX_BUF_LEN];
173
174   if(args.status!=ECA_NORMAL) {
175     printf ("Node: Channel connect timed out \'%s\' not found.\n", ca_name(
        node_chid));
176   } else {
177     char *pdata = (char *)args.dbr;
178           sprintf(data,"%s,%s",pname,pdata);
179     pthread_mutex_lock(&queue_mutex);
```

```
180     msg_queue.push(data);
181     pthread_mutex_unlock(&queue_mutex);
182     ev_async_send(EV_DEFAULT_UC_ &eio_tcamonitor_notifier);
183   }
184 }
185
186 // The background thread
187 static void* camonThread(void* p){
188     chid channelId;
189     struct thread_data *_pdata = (struct thread_data*)p;
190     const char *pv = _pdata->_pv;
191     double timeout = atof(strdup(_pdata->_time));
192     channelId = _pdata->_channelId;
193
194     ca_context_create (ca_disable_preemptive_callback);
195     ca_create_channel (pv, NULL, NULL, 10, &channelId);
196     ca_pend_io(.05);
197     ca_add_event( DBR_STRING, channelId, monitorFunc, NULL, NULL );
198     ca_pend_event(timeout);
199     free(_pdata); // free malloc
200     // thread detach
201     pthread_detach(pthread_self());
202     printf("Monitor record is finished.\n");
203     pthread_exit(0);
204     return NULL;
205 }
206
207 // callback that runs the javascript in main thread
208 static void Callback(EV_P_ ev_async *watcher, int revents)
209 {
210     HandleScope scope;
211     assert(watcher == &eio_tcamonitor_notifier);
212     assert(revents == EV_ASYNC);
213
214     Local<Value> callback_v = module_handle->Get(callback_symbol);
215     if (!callback_v->IsFunction()) {
216    printf("Node-ca: Error not set Callback function\n");
217          return;
218     }
219     Local<Function> callback = Local<Function>::Cast(callback_v);
220
221     // dequeue callback message
222     pthread_mutex_lock(&queue_mutex);
223     char *number = msg_queue.front();
224     msg_queue.pop();
225     pthread_mutex_unlock(&queue_mutex);
226
227     TryCatch try_catch;
228
229     // prepare arguments for the callback
230     Local<Value> argv[1];
231     argv[0] = Local<Value>::New(String::New(number));
232
233     // call the callback and handle possible exception
234     callback->Call(module_handle, 1, argv);
```

```
235
236     if (try_catch.HasCaught()) {
237         FatalException(try_catch);
238     }
239 }
240
241
242 // Start the background thread
243 Handle<Value> start_camon(const Arguments &args)
244 {
245     HandleScope scope;
246     struct thread_data *d;
247     if((d = (thread_data*)malloc(sizeof(*d)))==NULL){
248   return v8::ThrowException(v8::String::New("Node-ca: Malloc thread error"));
249     }
250     const char *usage = "usage: camonitor(record_name,val)";
251     v8::String::Utf8Value str(args[0]);
252     v8::String::Utf8Value cNum(args[1]);
253
254     if(*str == NULL){
255   printf ("node-ca: %s \n",usage);
256   return v8::ThrowException(v8::String::New("Node-ca: Bad parameters"));
257     }
258
259     // EPICS record name , timeout
260     const char *pv = ToCString(str);
261     const char *time_out = ToCString(cNum);
262     d->_time = strdup(time_out);
263     d->_pv = strdup(pv);
264     ev_async_init(&eio_tcamonitor_notifier, Callback);
265     ev_async_start(EV_DEFAULT_UC_ &eio_tcamonitor_notifier);
266     ev_unref(EV_DEFAULT_UC);
267     // make thread
268     if(pthread_create(&thread_id, NULL, camonThread,(void*)d) != 0){
269       perror("pthread_create");
270     }
271
272     return v8::Undefined();
273 }
274
275 void Initialize(Handle<Object> target)
276 {
277     HandleScope scope;
278     NODE_SET_METHOD(target, "camonitor", start_camon);
279     callback_symbol = NODE_PSYMBOL("callback");
280     // store handle for callback context
281     module_handle = Persistent<Object>::New(target);
282     NODE_SET_METHOD(target, "caget", caGet);
283     NODE_SET_METHOD(target, "caput", caPut);
284 }
285
286 extern "C" {
287     static void Init(Handle<Object> target){
288         Initialize(target);
289     }
```

```
290     NODE_MODULE(nodeca, Init);
291 }
```

Listing B.1: Channel Access for Node.js.

B.1.2 wscript

```
1  srcdir = '.'
2  blddir = 'build'
3  VERSION = '0.3.5'
4
5  def set_options(opt):
6    opt.tool_options('compiler_cxx')
7
8  def configure(conf):
9    conf.check_tool('compiler_cxx')
10   conf.check_tool('node_addon')
11
12 # Library for EPICS
13   conf.env.CPPPATH = ['/usr/local/epics/base/include','/usr/local/epics/base/
       include/os/Linux']
14   conf.check_cxx(lib='cas', libpath = '/usr/local/epics/base/lib/linux-x86')
15
16 def build(bld):
17   obj = bld.new_task_gen('cxx', 'shlib', 'node_addon')
18   obj.cxxflags = ["-g", "-D_FILE_OFFSET_BITS=64",
19                   "-D_LARGEFILE_SOURCE", "-Wall"]
20   obj.target = 'nodeca'
21   obj.source = "nodeca.cc"
22   obj.libpath='/usr/local/epics/base/lib/linux-x86'
23   obj.lib = 'cas'
```

Listing B.2: Makefile to build as Node.js add-on.

B.1.3 ca_subscription.js

```
1  /*!
2   * ca_subscription.js
3   * Copyright(c) 2012 A.UCHIYAMA in Sokendai University (KEK) <akito12@gmail.com>
4   * MIT Licensed
5   */
6  // Required Node.js v0.6.11 or later
7
8  var pv = require('../build/Release/nodeca');
9
10 var opts = require('opts');
11 opts.parse([
12     {
13         'short': 'p',
14         'long': 'pv',
```

```
        'description': 'EPICS record name',
        'value': true,
        'required': true
    },
    {
        'short': 't',
        'long': 'time',
        'description': 'Monitor timeout',
        'value': true,
        'required': false
    }
]);

var record_name = opts.get('pv')
var timeout = opts.get('time')||0

pv.callback=function(v){
    val=v.split(",");
  process.send(val[1]);
}
pv.camonitor(record_name,timeout);
```

Listing B.3: Use for event-driven function (caMonitor) using child process.

B.2 WebSocket Server for EPICS CA

B.2.1 app.js

```
/*!
 * nodeca_app.js
 * Copyright(c) 2012 A. UCHIYAMA in Sokendai University (KEK) <akito12@gmail.com>
 * MIT Licensed
 */
// Required Node.js v0.6.11 or later

/**
* Module dependencies.
*/
var express = require('express');
var socketio = require('socket.io');
var fs = require('fs');
var cp = require('child_process');
var opts = require('opts');
var log4js = require('log4js');

var use_SSL;
opts.parse([
    {
        'short'    : 's',
        'long'     : 'ssl',
        'description' : 'use SSL connection',
```

```
24          'value'    : false,
25          'required'  : false
26      },
27      {
28          'short'    : 'p',
29          'long'     : 'port',
30          'description' : 'port number',
31          'value'    : true,
32          'required'  : false
33      }
34  ]);
35
36
37  /**
38  * EPICS CA
39  */
40  var pv = require("./build/Release/nodeca");
41
42  /**
43  * SSL key
44  */
45  var SSL_KEY = 'privatekey.pem';
46  var SSL_CERT= 'certificate.pem';
47
48  /**
49   * Set Express
50   */
51  var PORT = opts.get('port') || 9000;
52  var HOME_DIR = '/public';
53
54  var use_SSL = opts.get('ssl') || 0;
55  if(use_SSL == 1){
56    var app = express.createServer({
57      key: fs.readFileSync(SSL_KEY).toString(),
58          cert : fs.readFileSync(SSL_CERT).toString()
59    }
60      , express.basicAuth(authorize)
61      , express.logger()
62      , express.static(__dirname + HOME_DIR));
63    console.log("Info : SSL is used.");
64  }else{
65    var app = express.createServer(
66        express.logger()
67      , express.static(__dirname + HOME_DIR));
68  };
69
70  /**
71   * Output access log
72   */
73
74  app.configure(function(){
75      log4js.configure({
76          'appenders': [
77              { 'type': 'console' },
78              {
```

```
79                  'type': 'file',
80                  'filename': '/mnt/websocketlogs/access.logs',
81                  'maxLogSize': 1024 * 1024,
82                  'backups': 5,
83                  'category': [ 'project-name', 'console' ],
84              },
85          ],
86          'replaceConsole': true
87      });
88
89      var logger = log4js.getLogger('project-name');
90      app.use(log4js.connectLogger(logger, {
91          'level': log4js.levels.DEBUG,
92          'nolog': [ '\\css', '\\.js', '\\.gif' ],
93          'format': ':remote-addr - - ":method :url HTTP/:http-version" :status :
    content-length ":referrer" ":user-agent"'
94      }));
95      app.use(app.router);
96  });
97
98
99  /**
100 * Basic authentication
101 */
102 function authorize(username,password){
103   return 'admin' === username & 'admin' === password;
104 }
105
106 // Open server port
107 app.listen(PORT);
108
109 var io = socketio.listen(app, {
110   // log level in socket.io
111   // 0 : Error
112   // 1 : warn
113   // 2 : info
114   // 3 : debug
115   'log level' : 1,
116   key : fs.readFileSync(SSL_KEY).toString(),
117   cert : fs.readFileSync(SSL_CERT).toString()
118 });
119
120 var sublist = [];          // for caSubscription
121 var intlist = [];          // for caGet_interval
122 var interval_Idlist = []; // for intervalId of caGet_interval
123 var client_info = [];    // for IPaddr:port of clients
124 var sock_names = [];
125
126 var caSubscription_count = 0;
127 var caGet_interval_count = 0;
128 var total_connection_count = 0;
129
130 var connection_address;
131 //var disconnect_address;
132
```

```
133 fs.readdirSync(__dirname + HOME_DIR).forEach(function(filename){
134   function stat(filename){
135     var stats = fs.statSync(__dirname + HOME_DIR + '/' + filename);
136     return stats;
137   }
138
139   stats = stat(filename);
140
141     // Directory check & Files (.js, .html, .htm) check
142     extension1 = '.js';
143     extension2 = '.html';
144     extension3 = '.htm';
145
146     function check_extension(filename){
147         if(filename.lastIndexOf(extension1) == -1  &&
148             filename.lastIndexOf(extension2) == -1 &&
149             filename.lastIndexOf(extension3) == -1
150         ){
151             return -1
152         }else{
153             if(filename.lastIndexOf('jquery') == -1){
154                 return 1
155             }else{
156                 return -1
157             }
158         }
159     }
160
161   if(stats.isDirectory()){
162     dirname = filename;
163         sock_names.push(dirname);
164     stats = stat(dirname);
165
166     if(stats.isDirectory()){
167             fs.readdirSync(__dirname + HOME_DIR + '/'+ dirname).forEach(function(
    filename){
168                 if(check_extension(filename) == 1){
169                 path_name = dirname;
170                     path_name += "/";
171                 path_name += filename.substr(0, filename.lastIndexOf('.'));
172                 sock_names.push(path_name);
173                 }
174         });
175     }else{
176             if(check_extension(filename) == 1){
177                 sock_names.push( filename.substr(0, filename.lastIndexOf('.')) );
178             }
179     }
180   }else{
181         if(check_extension(filename) == 1){
182           sock_names.push( filename.substr(0, filename.lastIndexOf('.')) );
183         }
184   }
185 });
186 sock_names = unique(sock_names)
```

```
187
188 /**
189 * Set socket_main function
190 */
191 for (i = 0; i < sock_names.length; i++){
192     sock_name = sock_names[i];
193     io.of('/' + sock_names[i]).on('connection', socket_main );
194     console.log( "Set NameSpace " + i  + " : " + sock_name);
195 }
196 io.of('').on('connection', socket_main);
197
198 function socket_main(socket,data){
199   /**
200    * Store the information of connected clients
201    */
202   connection_address = socket.handshake.address;
203
204   client_info[total_connection_count] = "";
205   client_info[total_connection_count] += connection_address.address;
206   client_info[total_connection_count] += ":";
207   client_info[total_connection_count] += connection_address.port;
208   client_info[total_connection_count] += ",";
209   client_info[total_connection_count] += socket.id;
210   client_info[total_connection_count] += ",";
211   client_info[total_connection_count] += socket.namespace.name;
212
213   console.log("New connection from " + client_info[total_connection_count]);
214   total_connection_count++;
215
216     socket.on('caPut', function(data) {
217         caPut(socket,data);
218     });
219
220     socket.on('caGet', function(data) {
221       process.nextTick(
222         function(){ caGet(socket,data) }
223       );
224     });
225
226     socket.on('caGet_interval', function(data) {
227       process.nextTick(
228         function(){ caGet_interval(socket,data) }
229       );
230     });
231
232     socket.on('caMonitor',function(data) {
233       process.nextTick(
234       function(){ caMonitor(socket,data) }
235       );
236     });
237
238 /*****************************************************/
239 /* Enable under here if use Operator Intervening System */
240 /*****************************************************/
241     socket.on('RequestOpe',function(data) {
```

```
242     var n = cp.fork(
243         __dirname + '/module/request',
244         ['-r',data.pv,'-u',process.env.USER],
245         {env:process.env,timeout:"1"}
246     );
247     });
248 
249     socket.on('OpeInfo',function(data) {
250   var n = cp.fork(
251         __dirname + '/module/ope_info',
252         ['-r',data.pv,'-u',process.env.USER, '-s',data.st],
253         {env:process.env,timeout:"1"}
254     );
255 
256   if(!data.event){
257         data.event = "ca";
258     }
259 
260   n.on('message',function(m){
261     var obj = {};
262         obj[data.data_name] = m;
263         socket.volatile.emit(data.event,obj);
264         socket.volatile.broadcast.emit(data.event,obj);
265     });
266 
267   setTimeout(function (){
268     process.kill(n.pid, 'SIGHUP');
269   },1000);
270      });
271 /*********************************************************/
272 
273     socket.on('caPut_counter', function(data) {
274         caPut_counter(socket,data);
275     });
276 
277     socket.on('disconnect', function(){
278       var disconnect_sid = socket.id;
279       console.log("Disconnect:" + disconnect_sid);
280       process.nextTick(
281       function(){ do_disconnect(disconnect_sid) }
282       );
283     });
284 }
285 
286 function caGet(socket,data){
287   try{
288     v = pv.caget(data.pv);
289   }catch(e){
290     v = "undefined";
291     console.log("caGet : " + data.pv + " : " + e );
292   }
293   if(!data.event){
294     data.event = "ca";
295   }
296   if(typeof obj === "undefined"){
```

```
297       var obj = {};
298     }
299     obj[data.data_name] = v;
300     socket.volatile.emit(data.event,obj);
301     socket.volatile.broadcast.emit(data.event,obj);
302   }
303
304   function caPutCallback(socket,data){
305     try{
306       v = pv.caget(data.callback_pv);
307     }catch(e){
308       v = "undefined";
309       console.log("caGet Error :" + e);
310     }
311     if(!data.event){
312       data.event = "ca";
313     }
314     if(typeof obj === "undefined"){
315       var obj = {};
316     }
317     obj[data.callback_obj] = v;
318     socket.volatile.emit(data.event,obj);
319     socket.volatile.broadcast.emit(data.event,obj);
320   }
321
322   function caFailCallback(socket,data){
323     if(!data.event){
324       data.event = "ca";
325     }
326     v = "caPut_failuer";
327     if(typeof obj === "undefined"){
328       var obj = {};
329     }
330     obj[data.callback_obj] = v;
331     socket.volatile.emit(data.event,obj);
332     socket.volatile.broadcast.emit(data.event,obj);
333   }
334
335   function caSuccessCallback(socket,data){
336     if(!data.event){
337       data.event = "ca";
338     }
339     v = "caPut_Success";
340     if(typeof obj === "undefined"){
341       var obj = {};
342     }
343     obj[data.callback_obj] = v;
344     socket.volatile.emit(data.event,obj);
345     socket.volatile.broadcast.emit(data.event,obj);
346   }
347
348   function caPut(socket,data){
349     process.nextTick(function(){
350       try{
351           pv.caput(data.pv,data.val);
```

```
352
353         if(data.callback_pv){
354             caPutCallback(socket,data);
355         }
356         if(data.data_name){
357             setTimeout(caGet(socket,data),1000);
358         }
359         if(data.callback_obj && !data.callback_pv){
360             caSuccessCallback(socket,data);
361         }
362     }catch(e){
363         if(data.callback_obj){
364             caFailCallback(socket,data);
365         }
366         console.log("caPut : " + data.pv + " : " + e );
367     }
368   });
369 }
370
371 function caMonitor(socket, data){
372   /**
373    * Check the sublist array for caSubscription function
374    * run_flag = 0; Run caSubscription
375    * run_flag = 1; Already run
376    */
377   if(!data.timeout){
378     data.timeout="0";
379   }
380   if(!data.event){
381     data.event = "ca";
382   }
383
384   if(data.timeout == "0"){
385       str_sublist = "";
386       str_sublist += data.event;
387       str_sublist += ",";
388       str_sublist += data.pv;
389       str_sublist += ",";
390       str_sublist += data.timeout;
391       str_sublist += ",";
392       str_sublist += data.data_name;
393       str_sublist += ",";
394       str_sublist += socket.namespace.name;
395   }else{
396     var PID_id = caSubscription(socket,data);
397   }
398
399   run_flag = 0;
400   for (j = 0; j < sublist.length; j++){
401     if(sublist[j].indexOf(str_sublist) !== -1){
402       run_flag = 1;
403       sublist[j] += ",";
404       sublist[j] += socket.id;
405       console.log("Already running: caMonitor:" + j + ":" + sublist[j]);
406       break;
```

```
407       }
408     }
409
410     if (run_flag == 0){
411       var PID_id = caSubscription(socket,data);
412       // Concatennate  1. pv, 2. timeout, 3. data_name, 4. name_space, 5. PID_id
413       str_sublist += ",";
414       str_sublist += PID_id;
415       str_sublist += "|";
416       str_sublist += socket.id;
417       sublist[caSubscription_count] = str_sublist;
418       console.log("caMonitor:" + caSubscription_count + ":" + sublist[
          caSubscription_count]);
419       caSubscription_count++;
420     }
421   }
422
423   function caSubscription(socket,data){
424     var n = cp.fork(
425       __dirname + '/module/ca_subscription',
426       ['-p',data.pv,'-t',data.timeout],
427       {env:process.env,timeout:data.timeout}
428     );
429
430     if( data.timeout != "0" ){
431         setTimeout(
432           function(){
433               stop_caMonitor(n.pid);
434           },
435           data.timeout * 1000
436         );
437     }
438
439     if(!data.event){
440       data.event = "pv";
441     }
442     var sid = socket.id;
443     n.on('message',function(m){
444       var obj = {};
445       obj[data.data_name] = m;
446       socket.volatile.emit(data.event,obj);
447       socket.volatile.broadcast.emit(data.event,obj);
448     });
449     return n.pid;
450   }
451
452   function caGet_interval(socket,data){
453
454     if(!data.int_sec){
455       data.int_sec = 1;
456       var interval_time=1000;
457     }else{
458       var interval_time=data.int_sec * 1000;
459     }
460
```

```
461   if(!data.event){
462     data.event = "ca";
463   }
464
465   str_intlist = "";
466   str_intlist += data.event;
467   str_intlist += ","
468   str_intlist += data.pv;
469   str_intlist += ",";
470   str_intlist += data.int_sec;
471   str_intlist += ",";
472   str_intlist += data.data_name;
473   str_intlist += ",";
474   str_intlist += socket.namespace.name;
475
476   var event = data.event;
477
478   function caget_int(){
479     try{
480       v = pv.caget(data.pv);
481     }catch(e){
482       console.log("caGet : " + data.pv + " : " + e );
483       v = 'undefined';
484   }
485     var obj = {};
486     obj[data.data_name] = v;
487     socket.volatile.emit(data.event,obj);
488     socket.volatile.broadcast.emit(data.event,obj);
489   }
490
491   run_flag = 0;
492   for (j = 0; j < intlist.length; j++){
493
494     if(intlist[j].indexOf(str_intlist) !== -1){ // Not include == -1
495       run_flag = 1;
496       intlist[j] += ",";
497       intlist[j] += socket.id;
498       console.log("Already running: caGet_interval:" + j + ":" + intlist[j]);
499       break;
500     }
501
502   }
503
504   if (run_flag == 0){
505     interval_Id = setInterval(caget_int, interval_time);
506   interval_Idlist[caGet_interval_count] = interval_Id;
507     str_intlist += ",";
508     str_intlist += "|";
509     str_intlist += socket.id;
510     intlist[caGet_interval_count] = str_intlist;
511     console.log("caGet_interval:" + caGet_interval_count + ":" + intlist[
      caGet_interval_count]);
512     caGet_interval_count++;
513   }
514   return interval_Id;
```

```
515 }
516
517 function do_disconnect(disconnect_sid){
518
519    client_sid = [];
520    for (j = 0; j < sublist.length; j++){
521       str_sid = "";
522       client_sids = sublist[j].split("|");
523       if(client_sids[1]){
524     client_sid = client_sids[1].split(",");
525       }
526       for (m = 0; m < client_sid.length; m++){
527     if(client_sid[m] !== disconnect_sid && client_sid[m] !== ""){
528         str_sid += client_sid[m];
529         str_sid += ",";
530     }
531       }
532
533       if(sublist[j]){
534     client_sids = sublist[j].split("|");
535     sublist[j] = client_sids[0];
536     sublist[j] += "|";
537     sublist[j] += str_sid;
538
539     if(!str_sid){
540         // pv_data : 0. event, 1.pv, 2.timeout, 3.data_name, 4.PID_id
541       pv_data=client_sids[0].split(",");
542       stop_caMonitor(pv_data[5]);
543         sublist[j] = "";
544     }
545       }
546    }
547
548    str_sid = "";
549    client_sid = [];
550    for (j = 0; j < intlist.length; j++){
551       client_sids = intlist[j].split("|");
552       if(client_sids[1]){
553     client_sid = client_sids[1].split(",");
554       }
555    }
556
557    for (j = 0; j < client_sid.length; j++){
558       if(client_sid[j] !== disconnect_sid && client_sid[j] !== ""){
559         str_sid += client_sid[j];
560         str_sid += ",";
561       }
562    }
563
564    for (j = 0; j < intlist.length; j++){
565      if(intlist[j]){
566       client_sids = intlist[j].split("|");
567       intlist[j] = client_sids[0];
568       intlist[j] += "|";
569       intlist[j] += str_sid;
```

```
570
571     if(!str_sid){
572         // pv_data : 0.event, 1.pv, 2.timeout, 3.data_name, 4.intervalId
573         stop_caGet_interval(interval_Idlist[j]);
574         intlist[j] = "";
575     }
576      }
577    }
578 }
579
580 function stop_caMonitor(PID_id){
581   process.kill(PID_id, 'SIGHUP');
582   console.log("LOG: child_process.kill : " + PID_id);
583   return 0;
584 }
585
586 function stop_caGet_interval(Id_obj){
587   clearInterval(Id_obj);
588   console.log("LOG: clearInterval ");
589   return 0;
590 }
591
592 function caPut_counter(socket,data){
593   try{
594     v = pv.caget(data.pv);
595   }catch(e){
596     console.log("caGet : " + data.pv + " : " + e );
597     v = 'undefined';
598     return 0
599   }
600   try{
601      if(v !== null && v !== undefined && !isNaN(v)){
602        pv.caput(data.pv,v+data.val);
603      }
604            if(data.callback_pv){
605              caPutCallback(socket,data);
606            }
607   }catch(e){
608            if(data.callback_pv){
609                caFailCallback(socket,data);
610            }
611      console.log("caPut_counter : " + data.pv + " : " + e );
612   }
613 }
614
615 function unique(array) {
616     var storage = {};
617     var uniqueArray = [];
618     var i,value;
619     for ( i=0; i<array.length; i++) {
620         value = array[i];
621         if (!(value in storage)) {
622             storage[value] = true;
623             uniqueArray.push(value);
624         }
```

```
625     }
626     return uniqueArray;
627 }
```

LISTING B.4: WebSocket Server coded in Node.js and Socket.IO (server-side javascript).

B.3 Example of WebSocket Client for EPICS CA

B.3.1 index.html

```
1  <html>
2    <head>
3      <title>ca-WebSocket example page</title>
4      <link rel="stylesheet" type="text/css" href="index.css" />
5      <script type="text/javascript" src="/socket.io/socket.io.js"></script>
6      <script type="text/javascript" src="/flot/jquery.js"></script>
7      <script type="text/javascript" src="./index.js"></script>
8    </head>
9    <body>
10     <h4>Record information with realtime update (camonitor written in Node.js)</
       h4>
11     <div id="EPICS2">
12       <p>PV: akito12Host:ai1 (caMonitor): <span id="TEST_record1"></p>
13       <p>PV: akito12Host:aiExample (caMonitor): <span id="TEST_record2"></p>
14       <p>PV: akito12Host:aiExample1 (caGet): <span id="TEST_record3"></p>
15     </div>
16
17     <div id="EPICS">
18       <p>PV: akito12Host:xxxExample <span id="TEST_record4"></p>
19       <li><a id="caput-update1" href="#" onclick="caput_update1()">caPut
       akito12Host:xxxExample 1</a>
20       <br>
21       <li><a id="caput-update2" href="#" onclick="caput_update2()">caPut
       akito12Host:xxxExample 0</a>
22     </div>
23     <div id="CALLBACK">
24       <p>CALLBACK: <span id="LOG"></p>
25   </div>
26   </body>
27 </html>
```

LISTING B.5: Example of WebSocket Client for EPICS Channel Access (html file)

B.3.2 index.js

```
1 var socket = io.connect('/index');
2 function callback_func(obj,msg){
```

```
  $(obj).html(msg);
}

socket.emit('caMonitor',{pv: 'rootHost:aiExample1', data_name: 'TEST_record1'});
socket.emit('caMonitor',{pv: 'rootHost:aiExample1', data_name: 'TEST_record2'});
socket.emit('caGet',{pv: 'rootHost:aiExample1', data_name: 'TEST_record3'});
caput_update1 = function() {
  socket.emit('caPut',{
    pv: 'rootHost:xxxExample',
    val: 199,
    data_name: 'TEST_record4',
    callback_pv: 'rootHost:aiExample2',
    callback_obj: 'TEST_callback1'
  },callback_func('#LOG','<font color=red> Data set 1 !!</font>'));
}

caput_update2 = function() {
  socket.emit('caPut',{
    pv: 'rootHost:xxxExample',
    val: 0,
    data_name: 'TEST_record4',
    //callback_pv: 'rootHost:aiExample2',
    callback_obj: 'TEST_callback1'
  },callback_func('#LOG','<font color=green> Data set 0 ..</font>'));
}


socket.on('ca', function(data){
  $('#TEST_record1').html(data.TEST_record1);
  $('#TEST_record2').html(data.TEST_record2);
  $('#TEST_record3').html(data.TEST_record3);
  $('#TEST_record4').html(data.TEST_record4);
    if(data.TEST_callback1 !== undefined) console.log("!!" + data.TEST_callback1
    + "!!");
});
```

Listing B.6: Example of WebSocket Client for EPICS Channel Access (client-side javascript)

B.4 Operator Intervention System

B.4.1 accept.js

```
/*!
 * accept.js
 * Copyright(c) 2013 A. UCHIYAMA in Sokendai University (KEK) <akito12@gmail.com>
 * MIT Licensed
 *
 */

/**
```

```
9  * Module dependencies.
10 */
11 var mysql = require('mysql');
12 var conf = require('./conf');
13 var opts = require('opts');
14
15 opts.parse([
16     {
17   'short'       : 'v',
18         'long'        : 'version',
19         'description' : 'Show version and exit',
20   'callback'    : function () { console.log('v1.0'); process.exit(1); }
21     },
22     {
23         'short'         : 'i',
24         'long'          : 'id',
25         'description'   : 'record id for MySQL table',
26   'value'         : true,
27         'required'      : true
28     },
29     {
30         'short'         : 'o',
31         'long'          : 'operator',
32         'description'   : 'Set operator name',
33   'value'         : true,
34         'required'      : true
35     },
36 /*
37     {
38   'short'         : 'r',
39         'long'          : 'pv',
40         'description'   : 'request EPICS record',
41         'value'         : true,
42         'required'      : false
43     },
44 */
45     {
46         'short'         : 't',
47         'long'          : 'time',
48         'description'   : 'reserved time',
49   'value'         : true,
50         'required'      : false
51     }
52 ]);
53
54 var connection = mysql.createConnection({
55   host     : conf.MYSQL_HOST,
56   user     : conf.MYSQL_USER,
57   password : conf.MYSQL_PASS,
58   database : conf.DATABASE,
59   insecureAuth: true
60 });
61
62 var TABLE = 'ope_request';
63 var _id = opts.get('id');
```

```
64 var _ope = opts.get('operator');
65
66 if (!opts.get('time')){
67   var _time = 120;
68 }else{
69   var _time = opts.get('time');
70 }
71
72 console.log("Record id:" + _id);
73 console.log("Operator:" + _ope);
74 console.log("Reserved time:" + _time);
75
76 var nowdate = new Date();
77
78 var post = {
79   ope_name: _ope,
80   ope_allowed_time: nowdate,
81   reserved_sec: _time,
82   status: 1
83 }
84 connection.query(
85   'UPDATE ' + TABLE + ' SET ? where id = ' + _id + ' and status = 0',
86   post,
87   function(err, result) {
88     if (err) {
89       console.log('database update error');
90       console.log(err);
91       connection.destroy();
92     }else{
93       console.log(result);
94       connection.end();
95     }
96   }
97 );
```

Listing B.7: Insert accept command into MySQL-based database for operation with output.

B.4.2 conf.js

```
1 module.exports = {
2   MYSQL_HOST: 'RDB-srv',
3   DATABASE: 'access_data',
4   TABLE: 'ope_request',
5   MYSQL_USER: 'xxx',
6   MYSQL_PASS: 'xxxxxx'
7 };
```

B.4.3 end.js

```
/*!
 * end.js
 * Copyright(c) 2013 A. UCHIYAMA in Sokendai University (KEK) <akito12@gmail.com>
 * MIT Licensed
 *
 */

/**
* Module dependencies.
*/
var mysql = require('mysql');
var conf = require('./conf');
var opts = require('opts');

opts.parse([
    {
  'short'       : 'v',
        'long'        : 'version',
        'description' : 'Show version and exit',
  'callback'    : function () { console.log('v1.0'); process.exit(1); }
    },
    {
        'short'         : 'i',
        'long'          : 'id',
        'description'   : 'record id for MySQL table',
  'value'         : true,
        'required'      : true
    },
    {
        'short'         : 'o',
        'long'          : 'operator',
        'description'   : 'Set operator name',
  'value'         : true,
        'required'      : true
    },
/*
    {
  'short'         : 'r',
        'long'          : 'pv',
        'description'   : 'request EPICS record',
        'value'         : true,
        'required'      : false
    },
*/
    {
        'short'         : 't',
        'long'          : 'time',
        'description'   : 'reserved time',
  'value'         : true,
        'required'      : false
    }
]);

var connection = mysql.createConnection({
```

```
55    host     : conf.MYSQL_HOST,
56    user     : conf.MYSQL_USER,
57    password : conf.MYSQL_PASS,
58    database : conf.DATABASE,
59    insecureAuth: true
60  });
61
62  var TABLE = 'ope_request';
63  var _id = opts.get('id');
64  var _ope = opts.get('operator');
65
66  if (!opts.get('time')){
67    var _time = 120;
68  }else{
69    var _time = opts.get('time');
70  }
71
72  console.log("Record id:" + _id);
73  console.log("Operator:" + _ope);
74
75  var nowdate = new Date();
76
77  var post = {
78    ope_endtime: nowdate,
79    status: 2
80  }
81
82  connection.query(
83    'UPDATE ' + TABLE + ' SET ? where id = ' + _id + ' and status = 1 and ope_name
        = \''+ _ope + '\'',
84    post,
85    function(err, result) {
86      if (err) {
87        console.log('database update error');
88        console.log(err);
89        connection.destroy();
90      }else{
91        console.log(result);
92        connection.end();
93      }
94    }
95  );
```

Listing B.8: Insert end command into MySQL-based database for timeout.

B.4.4 request.js

```
1  /*!
2   * request.js
3   * Copyright(c) 2013 A. UCHIYAMA in Sokendai University (KEK) <akito12@gmail.com>
4   * MIT Licensed
5   *
6   */
```

```
7
8  /**
9  * Module dependencies.
10 */
11 var mysql = require('mysql');
12 var conf = require('./conf');
13 var opts = require('opts');
14
15 opts.parse([
16     {
17         'short'         : 'v',
18         'long'          : 'version',
19         'description'   : 'Show version and exit',
20         'callback'      : function () { console.log('v1.0'); process.exit(1); }
21     },
22     {
23         'short'         : 'u',
24         'long'          : 'user',
25         'description'   : 'Set user name',
26   'value'         : true,
27         'required'      : true
28     },
29     {
30         'short'         : 'r',
31         'long'          : 'pv',
32         'description'   : 'request EPICS record',
33   'value'         : true,
34         'required'      : true
35     }
36 ]);
37
38 var connection = mysql.createConnection({
39   host     : conf.MYSQL_HOST,
40   user     : conf.MYSQL_USER,
41   password : conf.MYSQL_PASS,
42   database : conf.DATABASE,
43   insecureAuth: true
44 });
45
46 var TABLE = 'ope_request';
47 var _user = opts.get('user');
48 var _pv = opts.get('pv');
49
50 console.log("User:" + _user);
51 console.log("PV:" + _pv);
52 var nowdate = new Date();
53
54 var post = {
55   user_name: _user,
56   request_record: _pv,
57   request_time: nowdate,
58   status: 0
59 }
60
61 connection.query(
```

```
62    'INSERT INTO ' + TABLE + ' SET ?',
63    post,
64    function(err, result) {
65      if (err) {
66        console.log('database insert error');
67        console.log(err);
68        connection.destroy();
69      }else{
70        console.log(result);
71        connection.end();
72      }
73    }
74  );
```

LISTING B.9: Insert request command into MySQL-based database for output operation request.

B.4.5 pooling.js

```
1  var mysql = require('mysql');
2  var conf = require('./conf');
3
4  var pool = mysql.createPool({
5    host     : conf.MYSQL_HOST,
6    user     : conf.MYSQL_USER,
7    password : conf.MYSQL_PASS,
8    database : conf.DATABASE,
9    insecureAuth: true
10 });
11 var TABLE = 'ope_request';
12
13 pool.getConnection(function(err,connection){
14   connection.query('select * from ' + TABLE, function(err, result, fields) {
15         if (err) throw err;
16         else {
17               console.log('-----------------------------------');
18             for (var i in result) {
19                 var val = result[i];
20                   console.log(
21         val.id +': '
22         + val.user_name + ':'
23         + val.request_time + ':'
24         + val.status
25         );
26             }
27         }
28     connection.end();
29   });
30 });
```

LISTING B.10: Check the operation flag in MySQL-based database.

B.4.6 main.php

```
<?php
 /*  2013-07-09 [A.UCHIYAMA] */
 /*  file:   main.php        */
 require("./session_check.php");
?>

<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title> Operator Intervening System &middot; Powered by Bootstrap</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="A.UCHIYAMA">

    <!-- Le styles -->
    <link href="../docs/assets/css/bootstrap.css" rel="stylesheet">
    <style type="text/css">
      body {
        padding-top: 20px;
        padding-bottom: 60px;
      }

      /* Custom container */
      .container {
        margin: 0 auto;
        max-width: 1000px;
      }
      .container > hr {
        margin: 60px 0;
      }

      /* Main marketing message and sign up button */
      .jumbotron {
        margin: 80px 0;
        text-align: center;
      }
      .jumbotron h1 {
        font-size: 100px;
        line-height: 1;
      }
      .jumbotron .lead {
        font-size: 24px;
        line-height: 1.25;
      }
      .jumbotron .btn {
        font-size: 21px;
        padding: 14px 24px;
      }

      /* Supporting marketing content */
      .marketing {
        margin: 60px 0;
```

```
54        }
55        .marketing p + h4 {
56          margin-top: 28px;
57        }
58
59
60        /* Customize the navbar links to be fill the entire space of the .navbar */
61        .navbar .navbar-inner {
62          padding: 0;
63        }
64        .navbar .nav {
65          margin: 0;
66          display: table;
67          width: 100%;
68        }
69        .navbar .nav li {
70          display: table-cell;
71          width: 1%;
72          float: none;
73        }
74        .navbar .nav li a {
75          font-weight: bold;
76          text-align: center;
77          border-left: 1px solid rgba(255,255,255,.75);
78          border-right: 1px solid rgba(0,0,0,.1);
79        }
80        .navbar .nav li:first-child a {
81          border-left: 0;
82          border-radius: 3px 0 0 3px;
83        }
84        .navbar .nav li:last-child a {
85          border-right: 0;
86          border-radius: 0 3px 3px 0;
87        }
88      </style>
89      <link href="../docs/assets/css/bootstrap-responsive.css" rel="stylesheet">
90
91      <!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
92      <!--[if lt IE 9]>
93        <script src="../docs/assets/js/html5shiv.js"></script>
94      <![endif]-->
95
96      <!-- Fav and touch icons -->
97      <link rel="apple-touch-icon-precomposed" sizes="144x144" href="../docs/assets
        /ico/apple-touch-icon-144-precomposed.png">
98      <link rel="apple-touch-icon-precomposed" sizes="114x114" href="../docs/assets
        /ico/apple-touch-icon-114-precomposed.png">
99        <link rel="apple-touch-icon-precomposed" sizes="72x72" href="../docs/assets
        /ico/apple-touch-icon-72-precomposed.png">
100                     <link rel="apple-touch-icon-precomposed" href="../docs/assets
        /ico/apple-touch-icon-57-precomposed.png">
101                                   <link rel="shortcut icon" href="../docs/assets
        /ico/favicon.png">
102   </head>
103
```

```
104   <body>
105
106     <div class="container">
107
108       <div class="masthead">
109         <h3 class="muted"><img src="../img/logo101.gif" width="50" height="50">
      Operator Intervening System</h3>
110         <p class="text-right"><span class="badge"><i class="icon-user"></i> Login
       Operator <strong><?php echo $user; ?></strong></span></p>
111         <div class="navbar">
112           <div class="navbar-inner">
113             <div class="container">
114               <ul class="nav">
115                 <li class="active"><a href="./main.php"><i class="icon-cog"></i>
      Home</a></li>
116                 <li><a href="./accesslog.php"><i class="icon-eye-open"></i>
      Access Log</a></li>
117                 <li><a href="./putlog.php"><i class="icon-file"></i> EPICS Put
      Log</a></li>
118                 <li><a href="./gwreport.php"><i class="icon-list-alt"></i>
      Gateway Report</a></li>
119       <li><a href="./asg.php"><i class="icon-warning-sign"></i> Access Security</a
      ></li>
120                 <li><a href="./logout.php">Logout</a></li>
121               </ul>
122             </div>
123           </div>
124         </div><!-- /.navbar -->
125       </div>
126       <!-- Example row of columns -->
127       <div class="row-fluid">
128         <div class="span6">
129           <h2>REQUEST PVs</h2>
130           <div id="request"></div>
131           <p><a class="btn btn-primary" onClick="On_all_check()">ALL &raquo;</a>&
      nbsp;
132             <a class="btn" onClick="open_box('<?php echo $user; ?>')">NEXT</a>&
      nbsp;
133             <a class="btn" onClick="close_box()">CANCEL</a>
134
135             <div id="reserved1"></div></p>
136         </div>
137
138         <div class="span6">
139           <h2>IN-OPERATION</h2>
140           <div id="operation"></div>
141           <p></p>
142         </div>
143       </div>
144 <br>
145       <div class="row-fluid">
146         <div class="span10">
147           <h2>DONE PVs</h2>
148           <div id="done"></div>
149           <p></p>
```

```
150         </div>
151       </div>
152
153       <hr>
154
155 <!-- Modal -->
156 <div aria-hidden="true" aria-labelledby="myModalLabel" class="modal hide fade" id
      ="myModal" role="dialog" tabindex="-1">
157   <div id="modal"></div>
158 </div>
159
160       <div class="footer">
161         <p>&copy; RIKEN Nishina Center Beam Dynamics & Diagnostics Team. and KEK
      </p>
162       </div>
163
164     </div> <!-- /container -->
165
166     <!-- Le javascript
167     ================================================== -->
168     <!-- Placed at the end of the document so the pages load faster -->
169     <script src="../docs/assets/js/jquery.js"></script>
170     <script src="../docs/assets/js/bootstrap-transition.js"></script>
171     <script src="../docs/assets/js/bootstrap-alert.js"></script>
172     <script src="../docs/assets/js/bootstrap-modal.js"></script>
173     <script src="../docs/assets/js/bootstrap-dropdown.js"></script>
174     <script src="../docs/assets/js/bootstrap-scrollspy.js"></script>
175     <script src="../docs/assets/js/bootstrap-tab.js"></script>
176     <script src="../docs/assets/js/bootstrap-tooltip.js"></script>
177     <script src="../docs/assets/js/bootstrap-popover.js"></script>
178     <script src="../docs/assets/js/bootstrap-button.js"></script>
179     <script src="../docs/assets/js/bootstrap-collapse.js"></script>
180     <script src="../docs/assets/js/bootstrap-carousel.js"></script>
181     <script src="../docs/assets/js/bootstrap-typeahead.js"></script>
182     <script src="../js/main.js"></script>
183   </body>
184 </html>
```

Listing B.11: Web interface for operator intervention system.

Bibliography

[1] International Committee for Future Accelerators. A global accelerator network: Icfa task force reports, December 2001. URL http://www.fnal.gov/directorate/icfa/icfa_tforce_reports.html.

[2] Reinhard Bacher, Philip Duval, and Stephen Herb. What are the controls requirements for the global accelerator network? *Proceedings of 8th International Conference on Accelerator & Large Experimental Physics Control Systems*, pages 588–590, 2001. URL http://www.slac.stanford.edu/econf/C011127/THAP056.pdf.

[3] Deborah Agarwal, Gary Olson, and Judy Olson. Collaboration tools for the global accelerator network: Workshop report. pages 1–15, September 2002. URL http://cds.cern.ch/record/747060/files/34081372.pdf.

[4] Silvia Gabrielli, Markus Hodapp, and Roberto Ranon. Designing a multipurpose virtual laboratory to support communities of practice in physics. In *e-Science and Grid Computing, 2006. e-Science'06. Second IEEE International Conference on*, pages 139–139. IEEE, 2006.

[5] R Pugliese, A Busato, D Favretto, F Bonaccorso, M Prica, and A Curri. Virtually there: The control room of the future. *Proceedings of ICALEPCS07 Knoxville Tennessee USA*, pages 418–420, 2007. URL http://accelconf.web.cern.ch/accelconf/ica07/PAPERS/WPPB10.PDF.

[6] Lucas Taylor, Erik Gottschalk, Kaori Maeshima, and Patricia McBride. Cms centres for control, monitoring, offline operations and prompt analysis. In *Journal of Physics: Conference Series*, volume 119, page 072029. IOP Publishing, 2008. URL http://iopscience.iop.org/1742-6596/119/7/072029.

[7] Apache tomcat. URL http://tomcat.apache.org/.

[8] Root. URL http://root.cern.ch/drupal/.

[9] Alan L Stone. Cms remote monitoring at fermilab. In *Real-Time Conference, 2007 15th IEEE-NPSS*, pages 1–5. IEEE, 2007.

[10] I Alexandrov, A Amorim, E Badescu, M Barczyk, D Burckhart-Chromek, M Caprini, J Da Silva Conceicao, J Flammer, B Di Girolamo, M Dobson, et al. Online software for the atlas test beam data acquisition system. *Nuclear Science, IEEE Transactions on*, 51(3):578–584, 2004.

[11] High Energy Accelerator Research Organization. Summary of activities of the photon factory for the period 2000-2005, 2005. URL http://pfwww.kek.jp/hyoka05/.

[12] N Yamamoto. Recent activities toward worldwide remote operations of accelerators. *Proceedings of ICALEPCS2003 Gyeongju Korea*, pages 603–605, 2003. URL http://accelconf.web.cern.ch/Accelconf/ica03/PAPERS/FR101.PDF.

[13] H. Sawa. Collaboratory in photon factory. In *International Symposium on Frontier in Materials Design, Synthesis and Measurements, Hyogo, JAPAN*, May 2005.

[14] Yukito Furukawa, Kazuya Hasegawa, Daisuke Maeda, and Go Ueno. Development of remote experiment system. *Proceedings of ICALEPCS2009 Kobe Japan*, page 615, 2009. URL http://accelconf.web.cern.ch/AccelConf/icalepcs2009/papers/wed002.pdf.

[15] Yukito Furukawa, Kazuya Hasegawa, and Go Ueno. First operation of the wide-area remote experiment system. *Proceedings of ICALEPCS2011 Grenoble France*, pages 1193–1195, 2011. URL http://accelconf.web.cern.ch/accelconf/icalepcs2011/papers/thbhaust05.pdf.

[16] Skype web page, 2014. URL http://skype.com.

[17] Facebook web page, 2014. URL http://facebook.com.

[18] Google web page, 2014. URL http://google.com.

[19] A Akiyama, K Ishii, E Kadokura, T Katoh, E Kikutani, Y Kimura, I Komada, K Kudo, S Kurokawa, K Oide, et al. Computer control system of tristan. *Nuclear Science, IEEE Transactions on*, 28(3):2359–2361, 1984.

[20] Takeshi Wada, Tadashi Kambara, Ichiro Yokoyama, Kazuo Shimizu, Hideki Takebe, Makoto Nagase, and Hiromichi Kamitsubo. Control system of riken heavy ion accelerator complex. *Japanese Journal of Applied Physics*, 30(part 1):2947–2955, 1991.

[21] RE Mielen. Design and performance of the stanford linear collider control system. *Nuclear Science, IEEE Transactions on*, 32(1):230–237, 1985.

[22] Isamu Abe, Hitoshi Kobayashi, and Masahiko Tanaka. Pc based control system using activex in the kek e-/e linac. *Proceedings of PCaPAC99, Tsukuba, Japan*.

[23] Leo R Dalesio, Jeffrey O Hill, Martin Kraimer, Stephen Lewis, Douglas Murray, Stephan Hunt, William Watson, Matthias Clausen, and John Dalesio. The experimental physics and industrial control system architecture: past, present, and future. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 352(1):179–184, 1994.

[24] Piotr Pucyk, Tomasz Jezynski, Waldemar Koprek, Tomasz Czarski, Krzysztof T Pozniak, and Ryszard S Romaniuk. Doocs server and client application for fpga-based tesla cavity controller and simulator. In *Wilga-DL Tentative*, pages 52–60. International Society for Optics and Photonics, 2005.

[25] R Tanaka, T Fukui, K Kobayashi, T Masuda, A Taketani, T Wada, and A Yamashita. The first operation of control system at the spring-8 storage ring. *Proceedings of ICALEPCS97*, 1997.

[26] Takahiro Matsumoto, Yukito Furukawa, and Miho Ishii. Development of new control framework madoca ii at spring-8. In *Proceedings of Particle Accelerator Society Meeting 2013, Nagoya, Japan*, 2013.

[27] JM Chaize, A Götz, WD Klotz, J Meyer, M Perez, E Taurel, and P Verdier. The esrf tango control system status. *8th International Conference on Accelerator & Large Experimental Physics Control Systems, 2001, San Jose, California*, 2001.

[28] P Betinelli-Deck, A Buteau, D Corruble, B Gagey, N Leclercq, M Ounsy, and JP Ricaud. Status of the soleil control system. *10th ICALEPCS Int. Conf. on Accelerator & Large Expt. Physics Control Systems. Geneva*, 2005.

[29] JM Chaize. Tango control system status, 2006. URL http://www.jlab.org/conferences/PCaPAC/talks/Chaize.pdf.

[30] A Uchiyama, H Yamauchi, J Odagiri, M Komiyama, and M Fujimaki. Implementation of pic-based embedded i/o controller with epics for rilac control system. *Proceedings of ICALEPCS2009 Kobe Japan*, .

[31] J Odagiri, J Chiba, K Furukawa, N Kamikubota, T Katoh, H Nakagawa, N Yamamoto, M Komiyama, I Yokoyama, H Song, et al. Epics device/driver support modules for network-based intelligent controllers. *Proceedings of ICALEPCS2003, Gyeongju, Korea*, 2003.

[32] Martin R Kraimer, Mark Rivers, and Eric Norum. Epics: Asynchronous driver support. *ICALEPCS05, Geneva*, 2005.

[33] Toshiya Tanabe, Toshikatsu Masuoka, Koichi Yoshida, Keiko Kumagai, Misaki Komiyama, and Takashi Emoto. Current status of the control system development at riken ri-beam factory. *Proceedings of ICALEPCS2003, Gyeongju, Korea*, pages 597–599, 2003.

[34] S Michizono, S Anami, M Kawamura, S Yamaguchi, and T Kobayashi. Digital rf control system for 400—mev proton linac of jaeri/kek joint project. *Proceedings of the 2002 Linac Conference, Gyeongju, Korea*, 2002.

[35] S Michizono, S Anami, S Yamaguchi, and T Kobayashi. Digital feedback system for j-parc linac rf source. *Proceedings of the 22nd International Linear Accelerator Conference, Luebeck, Germany*, 2004.

[36] M Kwon, IS Choi, JW Choi, JS Hong, MC Keum, KH Kim, MG Kim, MK Park, SH Seo, S Baek, et al. The control system of kstar. *Fusion engineering and design*, 71(1):17–21, 2004.

[37] J Odagiri, S Araki, K Furukawa, N Kamikubota, A Kiyomichi, H Nakagawa, and TT Nakamura. Application of epics on f3rp61 to accelerator control. *Proceedings of ICALEPCS2009, Kobe, Japan*, pages 916–918, 2009.

[38] A Uchiyama, K Furukawa, N Kamikubota, H Nakagawa, TT Nakamura, J Odagiri, M Tomizawa, N Yamamoto, K Kameda, T Natsui, et al. Development of embedded epics on f3rp61-2l. *Proceedings of PCaPAC08, Ljubljana, Slovenia*, page 145, 2008.

[39] Takuya Nakamura, Kazuro Furukawa, Tatsuro Nakamura, and Jun-Ichi Odagiri. Upgrading the control system of the movable masks for kekb. In *Proceedings of Particle Accelerator Society Meeting 2009, JAEA, Tokai, Naka-gun, Ibaraki, Japan*, .

[40] M Takagi, N Kamikubota, A Kiyomichi, S Murasugi, R Muto, H Nakagawa, J Odagiri, K Okamura, and N Nagura. Control of the j-parc slow extraction line based on embedded epics. *Proceedings of ICALEPCS2009, Kobe, Japan*, pages 549–551, 2009.

[41] M Komiyama, M Fujimaki, N Fukunishi, RIKEN Nishina Center, A Uchiyama, and J Odagiri. Upgrading the control system of riken ri beam factory for new injector. *Proceedings of ICALEPCS2009, Kobe, Japan*, pages 275–277, 2009.

[42] Jian Zhuang, Kejun Zhu, Yuanping Chu, Jiajie Li, Lei Hu, and Dapeng Jin. The performance test of f3rp61 and its applications in csns experimental control system. *Proceedings of ICALEPCS2011, Grenoble, France*, pages 763–766, 2011.

[43] CY Wu, CS Fann, Jenny Chen, YS Cheng, SY Hsu, Demi Lee, KH Hu, CH Kuo, KT Hsu, KK Lin, et al. Control of the pulse magnet power supply by epics ioc embedded plc. *Proceedings of IPAC'10, Kyoto, Japan*, pages 2731–2733, 2010.

[44] A. UCHIYAMA, M. Kobayashi-Komiyama, E. Ikezawa, M. Fujimaki, T. Ohki, H. Yamauchi, and N. Fukunishi. Replacement of aging controller gmacs with f3rp61-2l for power supplies of quadrupole magnets in rilac. *RIKEN Accelerator Progress Report*, 45, 2012.

[45] K Rehlich. Status of the control system for the european xfel. *Proceedings of ICALEPCS2011, Grenoble, France*, pages 597–599, 2011. URL http://accelconf.web.cern.ch/Accelconf/icalepcs2011/papers/tudaust04.pdf.

[46] J Odagiri, K Akai, K Furukawa, S Michizono, T Miura, TT Nakamura, H Deguchi, K Hayashi, J Mizuno, and M Ryoshi. Fully embedded epics-based control of low level rf system for superkekb. *Proceedings of IPAC'10, Kyoto, Japan*, pages 2686–2688, 2010.

[47] K Furukawa, K Akai, A Akiyama, T Kobayashi, S Michizono, T Miura, K Nakanishi, H Deguchi, K Hayashi, and M Ryoshi. Embedded llrf controller with channel access on microtca backplane interconnect. *Proceedings of ICALEPCS2011, Grenoble, France*, pages 1274–1276, 2011.

[48] Leo R Dalesio, MR Kraimer, and AJ Kozubal. Epics architecture. In *ICALEPCS*, volume 91, pages 92–15, 1991.

[49] Epics-base r3.14.12.1 source code. URL http://www.aps.anl.gov/epics/download/base/index.php.

[50] Martin R. Kraimer, Janet B. Anderson, Andrew N. Johnson, W. Eric Norum, Jeffrey O. Hill, Ralph Lange, Benjamin Franksen, and Peter Denison. Epics application developer's guide. URL http://www.aps.anl.gov/epics/base/R3-14/11-docs/AppDevGuide.pdf.

[51] Kenneth Evans Jr. An overview of medm. In *Proc. ICALEPCS 1999 Conference, Trieste, Italy*, pages 466–468, 1999.

[52] R Keitel. Edlbuild–display generation for the epics edm display manager'. *ICALEPCS05, Geneva*, 2005.

[53] Jan Hatje, M Clausen, Ch Gerke, M Moeller, and H Rickens. Control system studio (css). *Proceedings of ICALEPCS07, Knoxville, Tennessee, USA*, 2007.

[54] Xihui Chen and Kay Kasemir. Boy, a modern graphical operator interface editor and runtime. *Proceedings of 2011 Particle Accelerator Conference, New York, NY, USA*, 2011.

[55] Kay-Uwe Kasemir and LR Dalesio. Overview of the experimental physics and industrial control system (epics) channel archiver. *Proceedings of 8th International Conference on Accelerator & Large Experimental Physics Control Systems, 2001, San Jose, California*, 2001. URL http://accelconf.web.cern.ch/AccelConf/ica01/papers/THAP019.pdf.

[56] Misaki KOMIYAMA, Nobuhisa FUKUNISHI, and Akito UCHIYAMA. Construction of new data archive system in riken ri beam factory. *Proceedings of ICALEPCS2011, Grenoble, France*, pages 90–93, 2011.

[57] TT Nakamura, N Yamamoto, and T Katoh. Data archiving system in kekb accelerators control system. *Proceedings of the 10th ICALEPCS, Geneva*, 2005.

[58] Akinobu Yoshii and Nobuhiro Kikuzawa. Examination of applying hadoop for j-parc driving data archive. In *Proceedings of the 8th Annual Meeting of Particle Accelerator Society of Japan*.

[59] Takuya Nakamura, Kazuro Furukawa, Takashi Obina, and Kay Kasemir. Upgrade of alarm system at pf-ar accelerator control. In *Proceedings of the 9th Annual Meeting of Particle Accelerator Society of Japan*, . URL http://www.pasj.jp/web_publish/pasj9/proceedings/PDF/WEPS/WEPS108.pdf.

[60] F Momal and C Pinto-Pereira. Using world-wide-web for control systems. In *Proceedings 1995 International Conference on Accelerator and Large Experimental Physics Control Systems, Chicago, IL, Oct*, volume 3, 1995. URL http://www-bd.fnal.gov/icalepcs/abstracts/PDF/m3ac.pdf.

[61] Takuya Nakamura, Kenji Yoshii, Tomohiro Aoyama, Kazuro Furukawa, Tatsuro Nakamura, and Takashi Obina. Delivery of accelerator operation panel active images and video image using web server. In *Proceedings of the 5th Annual Meeting of Particle Accelerator Society of Japan Hiroshima*, 2008. URL http://www.pasj.jp/web_publish/pasj5_lam33/contents/PDF/TP/TP011.pdf.

[62] T Hirono, T Matsushita, T Ohata, and A Yamashita. Development of data logging and display system, mydaq2.

[63] Kenzi Yoshii, T Nakamura, K Furukawa, TT Nakamura, T Obina, M Satoh, and N Yamamoto. Web-based electronic operation log system-zlog system. *Proceedings of ICALEPCS07 Knoxville Tennessee USA*, 2007. URL https://accelconf.web.cern.ch/accelconf/ica07/PAPERS/WOAB04.PDF.

[64] Akito Uchiyama, Kazuro Furukawa, Yoshihide Higurashi, Misaki Komiyama, Takahide Nakagawa, and Kazutaka Ozeki. Design of web-based interface to riken 28 ghz super-conducting ecr ion source and the future plan (to be published). *Proceedings of ECRIS2012 Sydney Australia*, 2012.

[65] Takuya Kudou, Shiro Kusano, Kazuro Furukawa, and Masanori Satoh. Upgrade of electronic logbook system at kek injector linac. In *Proceedings of the 8th Annual Meeting of Particle Accelerator Society of Japan Tsukuba*, 2011. URL http://www.pasj.jp/web_publish/pasj8/proceedings/poster/MOPS094.pdf.

[66] N Kamikubota, N Yamamoto, H Nakagawa, S Yamada, KC Sato, T Katoh, J Odagiri, N Kikuzawa, Y Kato, M Kawase, et al. J-parc control toward future reliable operation. *Proceedings of ICALEPCS2011 Grenoble France*, pages 378–381, 2011. URL http://epaper.kek.jp/icalepcs2011/papers/mopms026.pdf.

[67] Kazuro Furukawa, Masanori Satoh, Igor Mejuev, and Keisuke Nakao. A java-based epics archive viewer with soap interface for data retrieval. *Proceedings of ICALEPCS2003 Gyeongju Korea*, pages 223–225, 2003. URL http://accelconf.web.cern.ch/AccelConf/ica03/PAPERS/MP707.PDF.

[68] D Quock, N Arnold, and A Johnson. Infrastructure monitoring system for the advanced photon source control system. *Proceedings of PCaPAC08 Ljubljana Slovenia*, pages 19–21, 2008. URL https://accelconf.web.cern.ch/AccelConf/pc08/papers/moz01.pdf.

[69] Thomas Pelaia II and Matthew Boyes. Introducing caml ii. *Proceedings of ICALEPCS2009 Kobe Japan*, pages 922–924, 2009. URL http://accelconf.web.cern.ch/AccelConf/icalepcs2009/papers/fra001.pdf.

[70] Lei DUAN and Liren SHEN. Web services interface to epics channel access. *Nuclear Science and Techniques*, 19(2):74–78, 2008.

[71] LF Li and CH Wang. A web based realtime monitor on epics data. *Proceedings of ICALEPCS2011 Grenoble France*, pages 121–123, 2011. URL https://accelconf.web.cern.ch/accelconf/icalepcs2011/papers/mopkn014.pdf.

[72] Kay-Uwe Kasemir, Xihui Chen, John Purcell, and Katia Danilova. Sns online display technologies for epics. *Proceedings of ICALEPCS2011 Grenoble France*, pages 1178–1181, 2011. URL http://accelconf.web.cern.ch/AccelConf/icalepcs2011/papers/thbhaust01.pdf.

[73] Y Furukawa. Web-based control application using websocket. *Proceedings of ICALEPCS2011, Grenoble, France*, pages 673–675, 2011.

[74] Ian Fette and Alexey Melnikov. The websocket protocol, 2011. URL http://tools.ietf.org/html/rfc6455.

[75] Microsoft web page, 2012. URL http://msdn.microsoft.com/ja-jp/library/ie/hh673567%28v=vs.85%29.aspx.

[76] A Uchiyama, K Furukawa, and Y Higurashi. Epics channel access using websocket. *Proceedings of PCaPAC2012, Kolkata, India*, . URL https://accelconf.web.cern.ch/accelconf/pcapac2012/papers/wecc02.pdf.

[77] URL http://libwebsockets.org/.

[78] URL http://www.eclipse.org/jetty/.

[79] URL http://nodejs.org.

[80] URL http://socket.io/.

[81] URL http://code.google.com/p/v8/.

[82] Document object model (dom), 2009. URL http://www.w3.org/DOM/.

[83] URL http://www.flotcharts.org/.

[84] URL http://code.google.com/p/jsgauge/.

[85] Takuya Kudou, Shiro Kusano, Masanori Satoh, and Kazuro Furukawa. Development of operation information display panel using websocket at kek injector linac. In *Proceedings of the 10th Annual Meeting of Particle Accelerator Society of Japan, Nagoya*, 2013.

[86] Akito UCHIYAMA, Kazuro Furukawa, and Yoshihide Higurashi. Operator intervention system for remote accelerator diagnostics and support. *Proceedings of ICALEPCS2013, San Francisco, CA, USA*. URL http://accelconf.web.cern.ch/AccelConf/ICALEPCS2013/papers/tuppc110.pdf.

[87] Kenneth Evans and Martin Smith. Experience with the epics pv gateway at the aps. In *Particle Accelerator Conference, 2005. PAC 2005. Proceedings of the*, pages 3621–3623. IEEE, 2005. URL http://www.aps.anl.gov/epics/EpicsDocumentation/ExtensionsManuals/Gateway/FPAT064.pdf.

[88] N Kamikubota, S Yamada, N Yamamoto, T Iitsuka, S Motohashi, M Takagi, S Yoshida, and H Nemoto. Virtual io controllers at j-parc mr using xen. *Proc. of ICALEPCS11, Grenoble, France*, 2011. URL http://accelconf.web.cern.ch/Accelconf/icalepcs2011/papers/wepmu039.pdf.

[89] G Liu, C Li, K Xuan, J Wang, and X Bao. First experience with vmware servers at hls. *Proc. of ICALEPCS11, Grenoble, France*, 2011. URL http://accelconf.web.cern.ch/accelconf/icalepcs2011/papers/mopms004.pdf.

[90] O Khalid and A Shaikh. Optimizing infrastructure for software testing using virtualization. *Proc. of ICALEPCS11, Grenoble, France*, 2011. URL http://accelconf.web.cern.ch/Accelconf/icalepcs2011/papers/webhaust02.pdf.

[91] E Matias, D Beauregard, R Berg, G Black, MJ Boots, W Dolton, D Hunter, R Igarashi, D Liu D Maxwell, CD Miller, et al. Phase ii and iii the next generation of cls beamline control and data acquisition systems.

[92] Sangil Lee, Jinseop Park, Jaesic Hong, Mikyung Park, and Sangwon Yun. Management tools for distributed control system in kstar. *Proc. of ICALEPCS11, Grenoble, France*, 2011. URL https://accelconf.web.cern.ch/accelconf/icalepcs2011/papers/wemmu006.pdf.

[93] Akito Uchiyama, Misaki Komiyama, and Nobuhisa Fukunishi. System design and implementation using virtualization technology for ribf control system. In *Proceedings of the 10th Annual Meeting of Particle Accelerator Society of Japan, Nagoya*, 2013.

[94] Tatsuaki Sakamoto and Takashi Sugimoto. Upgrade of wide area remote control system (warcs) at spring-8 and sacla. In *Proceedings of the 10th Annual Meeting of Particle Accelerator Society of Japan, Nagoya*, 2013.

[95] A Yamashita and Y Furukawa. Warcs: Wide area remote control system in spring-8. *Proc. of 10th ICALEPCS Int. Conf. on Accelerator & Large Expt. Physics Control Systems*, 2005. URL http://accelconf.web.cern.ch/accelconf/ica05/proceedings/pdf/P1_068.pdf.

[96] Yasushige Yano. The riken ri beam factory project: A status report. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 261(1):1009–1013, 2007.

[97] Yasushige Yano, A Goto, M Kase, and T Katayama. Ri beam factory project at riken. In *AIP CONFERENCE PROCEEDINGS*, pages 161–166. IOP INSTITUTE OF PHYSICS PUBLISHING LTD, 2001. URL http://accelconf.web.cern.ch/AccelConf/c01/cyc2001/paper/C-1.pdf.

[98] H Okuno, N Fukunishi, and O Kamigaito. Progress of ribf accelerators. *Progress of Theoretical and Experimental Physics*, 2012(1):03C002, 2012.

[99] Yoshihide Higurashi. Doctoral thesis in graduate school of science, rikkyo university, 2001.

[100] T Nakagawa, Y Higurashi, J Ohnishi, T Aihara, M Tamura, A Uchiyama, H Okuno, K Kusaka, M Kidera, E Ikezawa, et al. First results from the new riken superconducting electron cyclotron resonance ion source (invited). *Review of Scientific Instruments*, 81(2):02A320, 2010.

[101] Y Higurashi, J Ohnishi, T Nakagawa, H Haba, M Tamura, T Aihara, M Fujimaki, M Komiyama, A Uchiyama, and O Kamigaito. Results of riken superconducting electron cyclotron resonance ion source with 28 ghz. *Review of Scientific Instruments*, 83(2):02A308, 2012.

[102] M Komiyama, M Fujimaki, M Kase, and A Uchiyama. Status of control system for riken ri-beam factory. In *Proceedings of the 2007 International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2007), Knoxville, Tennessee, USA*, 2007. URL http://accelconf.web.cern.ch/AccelConf/ica07/PAPERS/TPPB11.PDF.

[103] A Uchiyama, R Koyama, M Komiyama, M Fujimaki, and N Fukunishi. Network security system and method for ribf control system. *Proceedings of ICALEPCS2011, Grenoble, France*, . URL http://accelconf.web.cern.ch/AccelConf/icalepcs2011/papers/wepmu038.pdf.

[104] A Uchiyama, K Furukawa, Y Higurashi, K Ozeki, M Komiyama, and T Nakagawa. Design of web-based interface to riken 28 ghz super-conducting ecr ion source and the future plan. *Proceedings of ECRIS2012, Sydney, Australia*, . URL http://accelconf.web.cern.ch/AccelConf/ecris2012/papers/tupp12.pdf.

[105] A Uchiyama, K Furukawa, Y Higurashi, and T Nakagawa. Implementation of an operator intervention system for remote control of the riken 28 ghz superconducting electron cyclotron resonance ion source. *Review of Scientific Instruments*, 85 (2):02A904, 2013.

[106] Akamai Technologies. The state of the internet. *Akamai Report*, 6(1), 2013. URL http://www.akamai.com/stateoftheinternet/.

[107] A Žagar, K Žagar, K Furukawa, and R Rechenmacher. Network analyser for the epics channel access protocol. URL http://accelconf.web.cern.ch/accelconf/pc08/papers/tup008.pdf.

[108] Webpage of apposite technologies, inc., 2014. URL http://www.apposite-tech.com/products/mini2.html.

[109] Jubatus web page, 2014. URL http://jubat.us/ja/.