NEXT-GENERATION MADOCA FOR THE SPRING-8 CONTROL FRAMEWORK

T. Matsumoto#, Y. Furukawa, M. Ishii, JASRI/SPring-8, Hyogo, Japan

Abstract

Message and Database Oriented Control Architecture (MADOCA) was originally developed at SPring-8; since 1997, it has been successfully utilized for control systems at SPring-8 and other accelerator facilities. Despite the successful implementation of MADOCA, several problems arose during the course of its application. For example, we need to treat data, including image data obtained by beam profile monitoring, and control specific devices that can be only managed by Windows drivers; these requirements cannot be fully met by MADOCA. Hence, we developed a next-generation MADOCA control framework, called MADOCA II, which is also based on a message-oriented control scheme as in of MADOCA. However, the core part of the messaging scheme is completely rewritten using ZeroMQ socket library, thereby greatly improving the feasibilities. Here, we report on MADOCA II from the viewpoint of messaging. We confirmed the stability of MADOCA II at BL36XU beamline after one-year operation. Several new applications with MADOCA II are being implemented at SPring-8.

INTRODUCTION

The Message and Database Oriented Control Architecture (MADOCA) control framework was originally developed at SPring-8 for the control systems of SPring-8 accelerators and beamlines and has been utilized since 1997 [1]. MADOCA is also adopted for control systems in other accelerator facilities such as HiSOR, NewSUBARU, and SACLA. At SACLA, MADOCA is applied to experimental station controls as well as accelerator and beamline controls.

However, MADOCA had a few shortcomings, and hence, we developed a new framework, next-generation MADOCA (MADOCA II), with the perspective of accommodating future upgrades in our control systems. The software framework of MADOCA II comprises messaging and data logging. In this paper, we describe MADOCA II from the perspective of messaging. The data logging aspects of MADOCA II is described in another paper [2].

The following functionalities were newly implemented in MADOCA II for improving messaging flexibilities.

1. Messaging data with variable lengths.
2. Controls in a Windows environment.
3. Asynchronous communication between operator workstations and front-end computers.

These functionalities are important to realize flexible control systems. For example, messaging data of variable lengths is necessary since data of various formats such as waveform and image data are usually used for beam monitoring. Further, since we sometimes need to treat specific equipment that can be only controlled by Windows drivers, it is necessary to implement controls in a Windows environment. Asynchronous communication is important for handling multiple controls simultaneously; however, it is not fully implemented in current MADOCA. Therefore, improvements in this regard are required.

MADOCA AND ITS SHORTCOMINGS

MADOCA is based on client/server control architecture, as seen in Figure 1, and messages are communicated between operator workstations and front-end computers. The framework consists of the Message Server (MS), Access Server (AS), and Equipment Manager (EM). The middleware of messaging utilizes System V Inter Process Communication (IPC) and Open Network Computing Remote Procedure Call (ONC/RPC).

Messages in MADOCA are based on text and are composed of a character string with S/V/O/C syntax—for example,
“123_matumot_oprgui_opcon01/get/sr_mag_ps_b/current .” S/V/O/C stands for subject (S), verb (V), object (O) and complement (C). In the above example, “get” (V) represents the control action; “sr_mag_ps_B” (O) represents the equipment to be controlled; “current” (C) represents the value to describe the contents of the action. The subject (S) is composed of the process number, application name, account name, and the hostname, and is assigned by the MADOCA framework itself. The message is sent from a Graphical User Interface (GUI); the response is obtained from EM, and reads as “sr_mag_ps_b/get/123_matumot_oprgui_opcon01/123.45 A” (i.e., S and O get interchanged in the response).

The main advantages of MADOCA are summarized as follows.

1. Control messages are constructed as S/V/O/C messages, which are abstracted and are easy to understand.
2. Users can control equipment with a unified method using MADOCA and do not need to know the details of controls for each of the many devices.
3. The system can be easily applied at large scales with the same architecture.

#matumot@spring8.or.jp

ISBN 978-3-95450-139-7

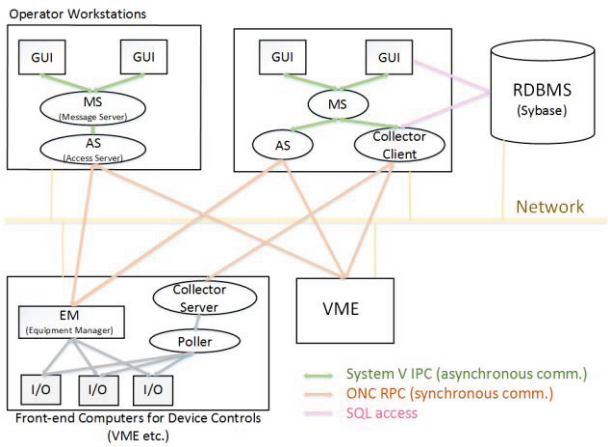

Figure 1: The software structure of MADOCA.

Although these advantages play a crucial role in our control systems, we experienced several problems during long-term operation with MADOCA; these included the difficulties arising because of the restriction in message length, time-consuming synchronous communication, the lack of controls applicable in a Windows environment, limitations imposed by the control architecture, and the improvements required in object management.

Restriction in S/V/O/C Message Length

The message length in MADOCA is restricted to 255 characters. Therefore, it is difficult to transmit data of variable lengths directly in a message. Currently, these data are managed using an intermediate file by specifying the file name in the message. However, this procedure is time consuming and makes the control scheme unclear.

Synchronous Communication in ONC/RPC

Operator workstations and front-end computers communicate via a remote procedure call (ONC/RPC), wherein control messages are processed sequentially after receiving the response for the previous message. Although this procedure enables the easy management of messages, it is time consuming. Further, it does not allow for distributed controls for multiple procedures because we cannot send control messages while we wait for the reply to another message.

Lack of Controls in a Windows Environment

MADOCA is only supported on UNIX-like systems since the MS utilizes System V IPC for interprocess communication. Therefore, we cannot apply MADOCA for controls in a native Windows environment. To use a Windows computer for the MADOCA control system, we need to use Cygwin or external computers to communicate with a socket server on the Windows computer.

Limitations on Communication due to Client-Server Control Architecture

As shown in Figure 1, in the MADOCA framework, the software structure at the client side in the operator workstation differs from that at the server side in the front-end computers. It allows communication between clients and servers. However, we cannot achieve communication with hosts in the same category. For example, communication between operator workstations is not possible because both are registered as clients.

Object Management

To deliver S/V/O/C messages to the appropriate destinations, we utilize object names as the key to determine the rules. When the AS distributes messages to the appropriate EM on the remote hosts, we use Relational Database Management System (RDBMS) to refer to the remote hosts associated with the objects. We use the configuration file to determine the AS that should send the message from the MS. This is necessary since the AS manages specific objects for each group (magnets, vacuum, etc.), and there are usually several such servers on the same host.

To determine the rules for message routing, we need to set object-related information in the RDBMS and configuration in advance. Therefore, we need elaborate object management.

MADOCA II MESSAGING

To overcome the problems described in the previous section, further flexibilities are required for messaging in MADOCA. Upon investigating, we found that the replacement of MADOCA control framework with ZeroMQ socket library [3] and reconfiguration of the messaging scheme can be a solution to these problems.

Figure 2 shows the software structure of MADOCA II messaging. As you can see, an MS2 (Message Server for MADOCA II) is set for each host. Two hosts can communicate when the MS2 connections are established. However, we do not allow message routing over three hosts. Further, there is no AS in MADOCA II because the function of the AS is implemented in the MS2 itself.

Messaging Data of Variable Lengths

ZeroMQ has a function for transmitting a number of messages sequentially. We use this function to manage data with variable lengths in messaging. Usually, we transmit the data with a simple text message; however, when required, we also transmit data with variable lengths. Thus, data such as image data can be directly transmitted with this messaging scheme.

The data are serialized using the MessagePack library [4], since MessagePack can serialize data of many formats (arrays, maps, etc.). MessagePack also has several advantages. For example, this format does not depend on language or byte order. Therefore, messages can be exchanged easily among different computing

ISBN 978-3-95450-139-7

environments. Further, data size is efficiently compacted during the serialization, thus enabling rapid data exchange.

Asynchronous Communications Between an Operator Workstation and a Front-end Computer

ZeroMQ enables asynchronous communications, thus removing the restrictions imposed by sequential controls in MADOCA. Parallel processing of multiple controls can be realized for fast processing.

Further, we can set multiple EMs on a front-end computer, as shown in Figure 2 so that an EM is set for each device. This enables efficient distributed processing of multiple controls. It also makes the system robust; in case of any problems with the EM for a specific device, we can continue the operation without that particular EM since other EMs are separated from it.

Asynchronous communications allows users to send multiple messages simultaneously and receive a target message when they want. To realize such communication, we need to set rules for sending and receiving messages. Therefore, we tagged messages with unique message identification numbers when transmitting them. The users can receive the message by specifying the message identification number.

Controls in a Windows Environment

Since ZeroMQ and MessagePack can be run on multiple OS, MADOCA II supports controls in a Windows environment as well as in Linux/Solaris environments. The codes are written in C++ and can be built using Visual C++. The codes are also unified between Windows and Linux/Solaris. We can apply MADOCA II for native Windows environment and can control specific devices that can be only managed with Windows drivers.

Message Communications Between Operator Workstations and Between Front-end Computers

As shown in Figure 2, MS2s are set on each host and GUI, and the EM can be easily connected to an MS2 on demand. This means that each host can behave as a client or server or both. Such flexibilities enable communications between operation workstations and between front-end computers. Figure 2 shows an example of communication between MS2s at two operator workstations.

Message Routing and Automated Object Management

In MADOCA II, message communication is realized by using the XREP/XREQ pattern in ZeroMQ for message routing. To determine the rules for message routing, the following procedures are performed in advance.

1. When an application such as a GUI or an EM is started, S (composed of process number, application name, account name, and the hostname) is registered in the MS2 in the same host.
2. MS2s used for controls are connected to each other from the client side to server side.
3. Objects names (O) managed by the EMs are registered in the MS2 in the same host; this information is also sent to the connected MS2s. Since the object information is also tagged with S, we can identify the destination of the message from the O in the S/V/O/C message.

After these procedures, the GUI can send a message to the EM by using the object information in the MS2s. The response from the EM can be returned to the GUI by using the S information in MS2. Thus, object management is automated, and RDBMS is not required, thereby facilitating easy maintenance.

We still use the configuration file for MADOCA II, but only for access controls.

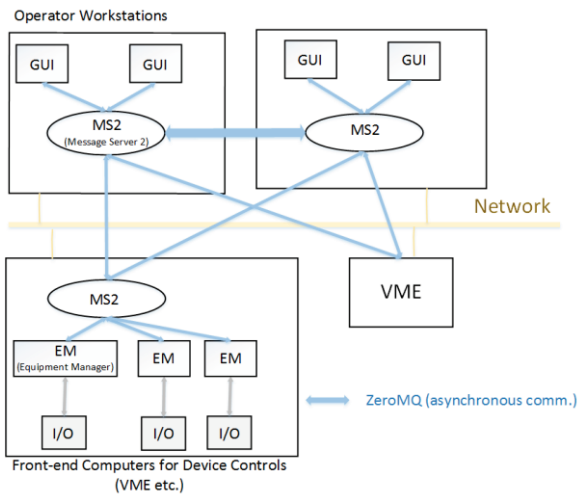

Figure 2: The software structure of MADOCA II messaging.

INTRODUCTION OF MADOCA II AT SPRING-8

We have introduced MADOCA II in the control systems at SPring-8. Since MADOCA II was designed as an upgrade of MADOCA, the API was designed to have backward compatibilities. In other words, users can upgrade applications from MADOCA to MADOCA II by recompiling and relinking the programs.

To introduce MADOCA II into our control systems, we first tested it at BL36XU beamline. Figure 3 shows the implementation of MADOCA II messaging at BL36XU; although MADOCA II was introduced at BL36XU, the other systems were still controlled with MADOCA. Thus, we prepared control interfaces to connect MADOCA with MADOCA II. AS2 (Access Server for MADOCA II) is used to control the MADOCA server from the MADOCA II client. RPC_MS2 (the interface between the AS of MADOCA and MS2) is used to control the MADOCA II server from the MADOCA client. Both interfaces are set to the hosts of MADOCA II. We tested MADOCA II over a one-year period (since September 2012) to confirm the

ISBN 978-3-95450-139-7

stability of the system and to establish the procedures for troubleshooting.

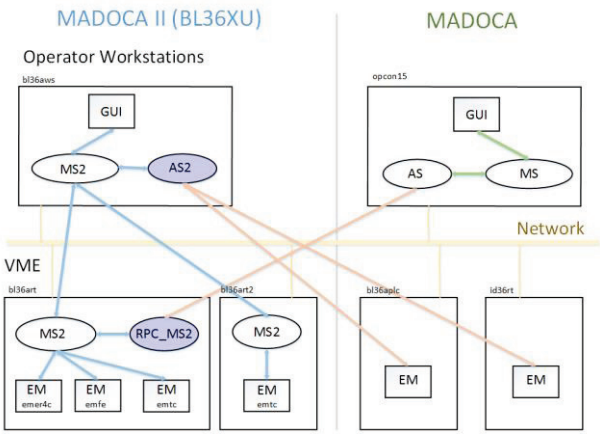

Figure 3: An example of the implementation of MADOCA II messaging at BL36XU. AS2 and RPC_MS2 interfaces are used to connect controls between MADOCA and MADOCA II.

Starting this autumn, several new applications that employ other features of MADOCA II will be introduced, as reported in these proceedings. In these applications, waveform data are utilized in the Beam Position Monitoring (BPM) [5], and image data are utilized in the two-dimensional synchrotron radiation interferometer [6]. Data of variable lengths, such as waveform and image data, are directly transmitted with the messages. For the BPM, MADOCA II is applied under Windows environment to control NI's PXI-5922 digitizers [5]; we also developed the MADOCA II-LabVIEW interface to control the LabVIEW system with MADOCA II.

Thus, MADOCA II can be used for various purposes with high flexibilities. The performance of MADOCA II with multi-core processors is described in the literature [7].

PLAN

As described in the previous section, MADOCA II has been introduced for the control systems at SPring-8, and some new applications are also being implemented.

We are preparing to replace our control system with MADOCA II in the near future. However, several issues are yet to be solved. For example, we need to confirm MADOCA II availability in many distributions such as Solaris, Linux, and ARM, especially for front-end computers. For Solaris 10, we needed to prepare own development environment because we could not use MessagePack with the default settings owing to the old version of gcc. The interface for messaging with the data logging system [2] is currently under development.

Although the basic features of MADOCA II messaging have been developed, some utilities such as messaging log viewer and object discovery service are yet to be developed.

SUMMARY

We developed MADOCA II as the next-generation MADOCA control framework. Although MADOCA II is based on a message-oriented control scheme as in MADOCA, the former has improved feasibilities realized by replacing the software in MADOCA with the ZeroMQ library. New features such as messaging data with variable lengths and controls in a Windows environment have already been implemented in several new control applications at SPring-8. Owing to its high flexibility, we expect MADOCA II to be satisfactorily applied for upgrading the SPring-8 control systems and control applications in other facilities.

ACKNOWLEDGMENT

We would like to thank Dr. Ryotaro Tanaka, Dr. Akihiro Yamashita, Masahiro Kago, and other colleagues in JASRI controls and computing division for useful discussions and suggestions.

REFERENCES

[1] R. Tanaka et al., "Control System of the SPring-8 Storage Ring", Proceedings of ICALEPCS'95, Chicago, p.201 (1995), R. Tanaka et al., "The first operation of control system at the SPring-8 storage ring", Proceedings of ICALEPCS'97, Beijing, China, p.1 (1997) .

[2] M. Kago et al., "Development of a Scalable and Flexible Data Logging System Using NoSQL Databases", TUPPC012, ICALEPCS 2013, to be published.

[3] http://www.zeromq.org/

[4] http://msgpack.org/

[5] Y. Furukawa et al., "MADOCA II Interface for LabVIEW", MOPPC129, ICALEPCS 2013, to be published.

[6] A. Kiyomichi et al., "Development of MicroTCA-based Image Processing System at SPring-8", TUPPC088, ICALEPCS 2013, to be published.

[7] M.Ishii et al., "Real-Time Process Control on Multi-Core Processors", MOPPC128, ICALEPCS 2013, to be published.

ISBN 978-3-95450-139-7