EPICS CHANNEL ACCESS USING WEBSOCKET

A. Uchiyama#, The Graduate University for Advanced Study (SOKENDAI), Tsukuba, Japan
K. Furukawa, High Energy Accelerator Research Organization (KEK), Tsukuba, Japan
Y. Higurashi, RIKEN Nishina Center, Wako, Japan

Abstract

Web technology is useful as a means of widely disseminating accelerator and beam status information. For this purpose, WebOPI was implemented by SNS as a web-based system using Ajax (asynchronous JavaScript and XML) with EPICS [1]. On the other hand, it is often necessary to control the accelerator from different locations as well as the central control room during beam operation and maintenance. However, it is not realistic to replace the GUI-based operator interface (OPI) with a Web-based system using Ajax technology because of interactive performance issue. Therefore, as a next-generation OPI over the web using EPICS Channel Access (CA), we developed 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 a web technology that provides bi-directional, full-duplex communication channels over a single TCP connection. [2] By utilizing Node.js and the WebSocket access library called Socket.IO, a WebSocket server was implemented. Node.js is a server-side JavaScript language built on the Google V8 JavaScript Engine. [3] In order to construct the WebSocket server as an EPICS CA client, an add-on for Node.js was developed 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 became available not only in the central control room but also with various types of equipment.

INTRODUCTION

One of the advantages of an EPICS-based system is the unified communication protocol between the front-end controller and client systems provided by the CA (Channel Access) protocol. Thus, even when various types of controllers were used as control systems, all the client systems such as the OPI (operator interface) could be developed on the basis of the CA protocol without hardware dependencies. In general, a method that used the CA libraries or display manager supported by EPICS collaboration, such as CAJ/JCA, CA-Python, MEDM/EDM, and CSS (Control System Studio) [4], was adopted for the development of the GUI in the EPICS-based system. On the other hand, the engineers at SPring-8 proposed the development methods for the main OPI using WebSocket and conducted a prototype implementation [5]. The prototype system was constructed using a MADOCA (message and database oriented control architecture)-based system that was developed at SPring-8. It was different from the EPICS-based system. Since the web application with real-time have many advantages, we started to develop WebSocket server corresponding to EPICS CA, and implemented Web applications using WebSocket for the EPICS-based control system. The traditional Web application lacked the interactivity needed to operate some of the hardware for the accelerator control system. By utilizing WebSocket technology, it is possible to solve the problems of Web-based OPI, such as the interactive response.

#a-uchi@riken.jp

WEBSOCKET PROTOCOL

WebSocket is a new protocol for achieving the bidirectional communication between a Web server and Web browser. In the beginning, WebSocket was part of HTML5. It was formulated as an RFC6455 by ITEF in Dec. 2011. An overview of the protocol is as follows.

1. A Web browser sends a handshake request to the server for connecting to the WebSocket.
2. The server returns the handshake response after approval.
3. After the establishment of the handshake, the protocol switches to the WebSocket, and then the bi-directional communication occurs between the Web server and Web browser.

WebSocket makes interactive response realizable and thus compensates for the disadvantage that could not be eliminated in traditional HTML. Thus far, various types of Web services in EPICS-based systems have been implemented. However, there were only implemented in Web-based systems, where a quick response and monitoring are unnecessary such as in 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. Since it becomes bi-directional transmission between a server and clients has become possible, the aforementioned problem is solved by WebSocket. In practice, we confirmed sufficient interactive response in a 100 ms cycle using WebSocket on the Web browser, and it had a performance similar to that of native EPICS applications such as EDM. Furthermore, since periodic polling is not necessary like Ajax, it becomes possible to reduce network traffic. WebSocket does have a problem with Internet Explorer 9 (IE9), which is one of the main browsers, because a WebSocket connection is unavailable. However, this problem may be resolved in IE10, which is the next version, because Microsoft planned it to be compliant with WebSocket. [6]

SERVER SIDE SYSTEM

We utilized Node.js and Socket.IO [7] as a WebSocket library to develop the WebSocket server. Node.js is one of the server-side JavaScript languages developed based on the Google V8 JavaScript Engine. As a main feature, Node.js works asynchronously with single-thread processing. Although many human resources may be required for the development of an asynchronous WebSocket server from scratch, Socket.IO solve the aforementioned problem. Node.js 0.6.18 and Socket.IO 0.9.6, which were stable versions of the languages, were utilized for the development. In order to implement the WebSocket server with a CA connection using Node.js, Node.js has to call the CA API. Therefore, we developed add-on software to interface CA from Node.js, that is, NodeCA. Since Node.js can call the function developed by C++, 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 prepared. In general, an event-driven caMonitor needs a non-blocking algorithm in the program, although single-threading Node.js causes the thread to be blocked by the implementation negligently. Consequently, in order to prevent the thread blocking, NodeCA is in waiting the CA event into another thread, and then it send the message queue to the main thread. Figure 1 shows an overview of NodeCA. The software developed by this research is shown in the gray area.

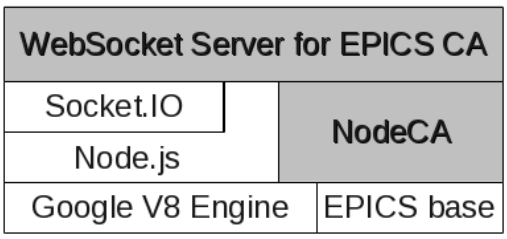

Figure1: Overview of the System

CLIENT SIDE SYSTEM

It is possible to realize a text update like EDM by coding DOM (document object model) with JavaScript on the Web. In addition, flot [8] and jsgause [9], which are jQuery-based JavaScript libraries, were used for the visualization of the accelerator parameters in our system. Many JavaScript libraries, other than those mentioned above, are also available for the visualization of information, such as strip chart. As a result, the client system has the advantage of low development costs since money can be saved multiple steps. Figure 2 shows the whole system chart.

1. In the code of client side JavaScript, custom events (caGet, caPut, caMonitor) are sent to the server by using WebSocket protocol
2. WebSocket server connect to EPICS IOC via NodeCA.
3. It is available to get the accelerator parameters from EPICS IOC.

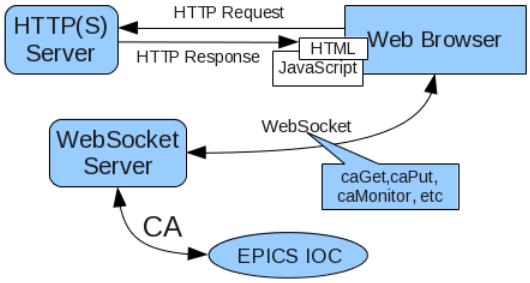

Figure 2: Whole System Chart

IMPLEMENTATION

After satisfying the need for interactive access for WebSocket using an EPICS-based system, we attempted to implement NodeCA and the WebSocket server as a Web-based OPI. As a result, the OPI was found to have adequate ability for accelerator operation compared to the traditional EPICS-based OPI. Moreover, we confirmed that almost all the browsers for the iPhone4S/Android are capable of obtaining and displaying numerical values from EPICS IOC via WebSocket. The present software support for browsers and OS is shown in Table 1. In the case of iPhone, it is difficult to install self-produced software without the approval of Apple Store. On the other hand, approval of Apple Store is unnecessary to install the self-produced Web application on an iPhone, although a Web server is required. We consider that the Web application using the WebSocket server and NodeCA have satisfactory performance as an OPI compared with the native applications of the iPhone.

At present, we are implementing it for controlling the 28GHz-ECRIS at RIKEN RIBF on a trial basis. [10] We already verified that we had a good response when monitoring the vacuum of the plasma chamber, controlling the gas-valve, and so on.

Table 1: The Present Software Support

OS	Browsers
Windows/Linux	Google Chrome 20, Firefox 14, Opera 11
Android 4.0	Google Chrome 18, Firefox 14
iOS (iPhone)	Safari 5, Google Chrome 19, Mercury

DISCUSSION

In 2000, GAN (global area network) was proposed by ICFA (International Committee for Future Accelerators) as a network for ILC accelerator control. [11] Although GAN aimed at an accelerator-only network with international collaboration, it has not been realized thus far. Currently, a WAN with has sufficient high-speed data communication is already in place around the world for the development of Internet technology. On the other

hand, it is difficult to gain direct access from other networks using this WAN with the CA protocol because firewalls are implemented between the accelerator network and gateways. For this reason, we considered the possibility of accelerator control using HTTP technology, which is a standard protocol for the Internet, and WebSocket.

First, we considered the following in order to protect the accelerator networks from outside intrusion. In the case of accessing the accelerator network from the WAN, we ensure the security of WebSocket access by using VPN and authentication with SSL. Additionally, accelerator operators have to completely understand the user attempting to access the networks (login ID and so on) and access route (full domain of Internet providers and so on) before WebSocket control is allowed. Therefore, all of the accessing logs for WebSocket control should be open to the accelerator operators.

Next, we 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 behaviour perfectly. On the other hand, it would not matter if they did not understand some simple instructions in detail such as caGet and caMonitor, which are used to monitor or obtain the numerical values for the accelerator parameters. If some accelerator parameters are changed using the Web application from outside the internal networks without considering the beam tuning, the accelerator condition will be complicated in many cases. Thus, when using caPut, a policy is needed for between the accelerator operators in the control room and a maintainer exerting control remotely over the Web. The following are proposed for this policy.

1. Before the maintainer exercises remote control over the Web, they have to call the accelerator operator or send an E-mail.
2. The maintainer authenticates users (and passwords) using SSL on the Web. After the authentication, it is possible to monitor the accelerator parameters via WebSocket.
3. The accelerator operators check the user ID and the domain or IP address of the Internet provider for the maintainer.
4. The maintainer sends a request to the accelerator operators when sending caPut instruction to the EPICS record.
5. In response to the request from the maintainer, the accelerator operators make a judgement decision about whether the component in the EPICS record may be controlled using WebSocket. A response is returned to the maintainer about whether it is possible to operate the component in the EPICS record.
6. After receiving this response, the maintainer can not only monitor, but also operate the component via WebSocket.
7. If the accelerator operators make a judgement decision to interrupt the operation, the WebSocket operation is discontinued as soon as possible by the accelerator operator. In addition, if a certain period of time passes, the permission for the WebSocket operation will be cancelled by a timeout.

CONCLUSION

We developed a server that makes it possible to connect with EPICS CA by WebSocket. It enables the display of information on the accelerator conditions and make it possible to control it from the Web browser in real time. It also makes it possible to call CA API from Node.js by NodeCA as an add-on to the interface for the CA protocol. We confirmed that we could control and monitor one part of the accelerator parameters as well as the traditional EPICS-based application. Additionally, the developed Web-based OPI runs not only on the main PC-based browsers, but also on almost all of the browsers for Android and iPhone4S. We believe that this technology will be useful as a means of accelerator operation using Internet access, even if it has some problems of security. In order to resolve these security problem for the operation from a WAN, it is necessary to lay everything out on the table to develop an access policy. In the future, we will develop the NodeCA and WebSocket server for the latest version of Node.js, which will be released as open-source software.

REFERENCES

[1] K. U. Kasemir, et al., Proceedings of ICALPECS 2011, Grenoble, France, 2011, THBHAUST01.
[2] I. Fette and A. Melnikov, The WebSocket Protocol, IETF HyBi Working Group. 2011.
[3] http://nodejs.org/
[4] X. Chen, et al., Proceedings of 2011 Particle Accelerator Conference, New York, NY, USA, 2011, WEOBN3 .
[5] Y. Furukawa, et al., Proceedings of ICALPECS 2011, Grenoble, France, 2011, WEMAU010.
[6] http://msdn.microsoft.com/en-us/library/ie/
[7] http://socket.io/
[8] http://code.google.com/p/flot/
[9] http://code.google.com/p/jsgauge/
[10] A. Uchiyama, et al., Proceedings of ECRIS 2012, Sydney, Australia, 2012, TUPP12
[11] "A Global Accelerator Network: ICFA Task Force Reports", http://www.fnal.gov/directorate/icfa/icfa_tforce_reports.html, Dec. 2001