WEB-BASED CONTROL APPLICATION USING WEBSOCKET

Y. Furukawa
SPring-8/JASRI, Kouto, Sayo-cho Hyogo, 679-5198, Japan.

Abstract

The WebSocket [1] allows asynchronous full-duplex communication between a Web-based (i.e. JavaScript-based) application and a Web-server. WebSocket started as a part of HTML5 standardization but has now been separated from HTML5 and has been developed independently. Using WebSocket, it becomes easy to develop platform independent presentation layer applications for accelerator and beamline control software. In addition, a Web browser is the only application program that needs to be installed on client computor. The WebSocket-based applications communicate with the WebSocket server using simple text-based messages, so WebSocket is applicable message-based control system like MADOCA, which was developed for the SPring-8 control system. A simple WebSocket server for the MADOCA control system and a simple motor control application were successfully made as a first trial of the WebSocket control application. Using Google-Chrome (version 13.0) on Debian/Linux and Windows 7, Opera (version 11.0) on Debian/Linux and Safari (version 5.0.3) on Mac OS X as clients, the motors can be controlled using a WebSocket-based Web-application. Diffractometer control application use in synchrotron radiation diffraction experiment was also developed.

INTRODUCTION

Web-based applications have many advantages over custom-built applications developed using Graphical User Interface (GUI) building toolkits: 1) platform independence; 2) easy to develop; 3) and easy to distribute the latest version. Based on these advantages, there have been many attempts to introduce Web-based applications for accelerator and experimental control.

Platform independence comes from the fact that most major Web browsers can run not only on traditional personal computer or workstation OSs such as Microsoft Windows, Mac OS X, most Linux distributions, Solaris, and FreeBSD, but can also run on mobile device OSs, such as Android OS and iOS. This gives us many candidates for an operator console and different styles of operation of accelerators and physical experiments. Using a mobile device, operators can access equipment besides it, and can easily troubleshoot the devices, if required.

There are many tools available for developing Web-based applications, including independent design and development tools such as Microsoft Visual Studio and Eclipse, and built-in browser tools. It is easier to develop applications using these tools than to use other language and tool kits.

A Web-based application is loaded from a Web server before every execution. It is easy to maintain the latest versions of all the running applications because you need to replace the Web application on the web server; you need not distribute and ask your application users to install a new version.

Despite these advantages, Web-based applications have a big disadvantage, i.e. a Web-application (HTML-based application) cannot make a full-duplex communications with a server application because the HTTP communication starts only from a client. This means that a server cannot send any information to a client without access from the client.

To solve this problem, a “long polling” mechanism was introduced in “Comet” [3]. A Comet server does not respond immediately to the client request (usually XHR is used). It waits for a server event, and the server then responds to the client with information on the event. Actually, the Comet server has to respond with in a definite time (usually 30 s) because Web browsers detect HTTP session time out.

This mechanism works well. However, it requires that an HTTP communication session be started for every long polling request. This results in a heavy; load on the Comet server and it is impossible to send information on client events to the Comet server during long polling periods. This means that it is difficult to realize a “true” control application using Comet.

The WebSocket [1] protocol was proposed to give true full-duplxy communication ability to Web-based applications in the context of HTML5. WebSocket is currently independent of HTML5 standardization, but is one of the important components of HTML5.

WebSocket is very easy to handle in a Web-based application, i.e., it can be easily handled in a program written using JavaScript. WebSocket can handle text and binary data. However, it is easy to handle text data in a JavaScript application. This means that it is easy to develop a WebSocket-based application for a text-message-based control system like MADOCA[6] or STARS[7]. Once you make a gateway from the WebSocket to your control system, you can make JavaScript-based applications. Therefore, a gateway (WebSocket server) to the MADOCA was built and several Web-based applications were tested, as described in the following sections.

SERVER APPLICATION

A schematic of a control system with WebSocket and MADOCA is shown in Fig.1. A WebSocket server has been newly built as a gateway to the MADOCA control system. The server has to treat the negotiation phase of the WebSocket protocol. This protocol has been revised and the latest version is hybi-10[4] but most Web browsers are implemented on hybi-00[5]; therefore, our WebSocket server is implemented on the basis of hybi-00. The WebSocket server can co-exist with an HTTP server using the same port, i.e., 80 (or 443). In this case, the Web server has to treat the negotiation phase of the WebSocket, while our WebSocket server is running on port 10101 because of continuity of service. To control optical components in the SPring-8 X-ray beamlines, a request server is introduced for the beamline control system on the port 10101.

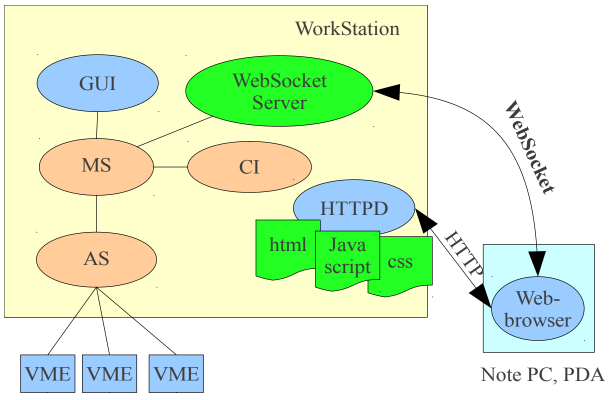


Figure 1: A schematic of WebSocket and MADOCA control system. Orange faced component, MS (Message Server), CI (Command Interpreter) and AS (Access Server) are standard MADOCA components [6]. Green faced components are newly developed for WebSocket Application.

This server is a simple socket server and with a small modification, we can adapt the server for both simple socket service and WebSocket service. The server decides which service is required from the first received string. For the simple socket service, the MADOCA format message will be received and for the WebSocket service, the format defined in hybi-00 or hybi-09 will be received, which is the same as an HTTP request, i.e., " GET /demo HTTP/1.1" or "GET /ws HTTP/1.1". After establishing a connection with a client, the WebSocket server receives MADOCA format from the client and sends it to a MADOCA-based control system. The WebSocket server then receives results from the MADOCA-based control system and sends them back to the client.

CLIENT APPLICATION

Client applications are written in JavaScript. In JavaScirpt, you can open the WebSocket using the "ws" object, as shown below

```
        ws = WebSocket("ws://<hostname>[:port]");,
```

send a message with

```
        ws.send(<message>);,
```

and receive a message with

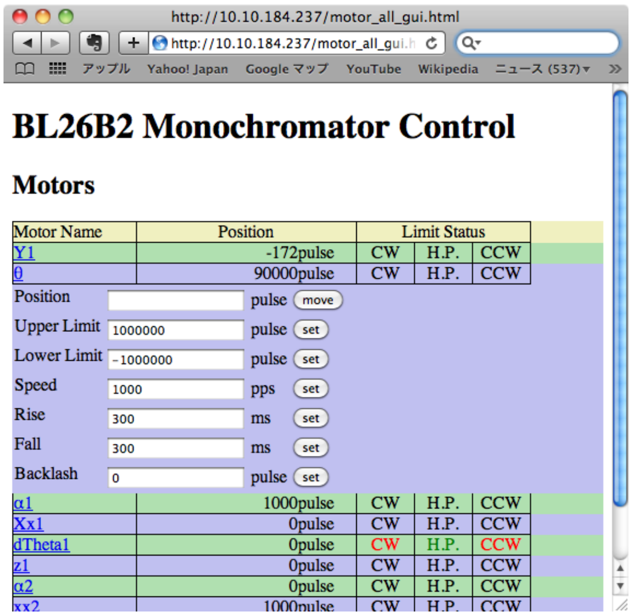
```
        ws.onmessage() = function(event) {
                var message = event.data;
                …
        }.
```

Note that the receive routine is called asynchronously. So you need to handle it with an event-driven handler.

As shown above, the WebSocket interface can be handled very easily.

EXAMPLES

A stepper motor control GUI was developed for the first case of the WebSocket application as shown in Fig.2. The user can watch the motor status, can control motor rotation and can modify motor parameters using this Web page.


http://10.10.184.237/motor_all_gui.html
http://10.10.184.237/motor_all_gui.h
アップル
Yahoo! Japan
Google マップ
YouTube
Wikipedia
ニュース (537)

BL26B2 Monochromator Control

Motors

Motor Name	Position	Limit Status		
Y1	-172pulse	CW	H.P.	CCW
θ	90000pulse	CW	H.P.	CCW

Position		pulse	move
Upper Limit	1000000	pulse	set
Lower Limit	-1000000	pulse	set
Speed	1000	pps	set
Rise	300	ms	set
Fall	300	ms	set
Backlash	0	pulse	set

α1	1000pulse	CW	H.P.	CCW
Xx1	0pulse	CW	H.P.	CCW
dTheta1	0pulse	CW	H.P.	CCW
z1	0pulse	CW	H.P.	CCW
α2	0pulse	CW	H.P.	CCW
xx2	1000pulse	CW	H.P.	CCW

Figure 2: Web-application for stepper motor control.

A more complicated example shown in Fig. 3. is a user interface for a diffractometer control system at the SPring-8 diffuse scattering beamline. This Web page was designed using BlueGriffon^{TM}[8] which is a useful tool for building Web pages. Using this Web page, a diffractometer user can tune the slit and detector conditions and make a series of X-ray diffraction measurements. The scan results are shown in real time in the graph area which is realized by a "canvas" element defined in the HTML5 specification. In this case, the "RGraph" library [9] is used to draw the graph in the canvas element.

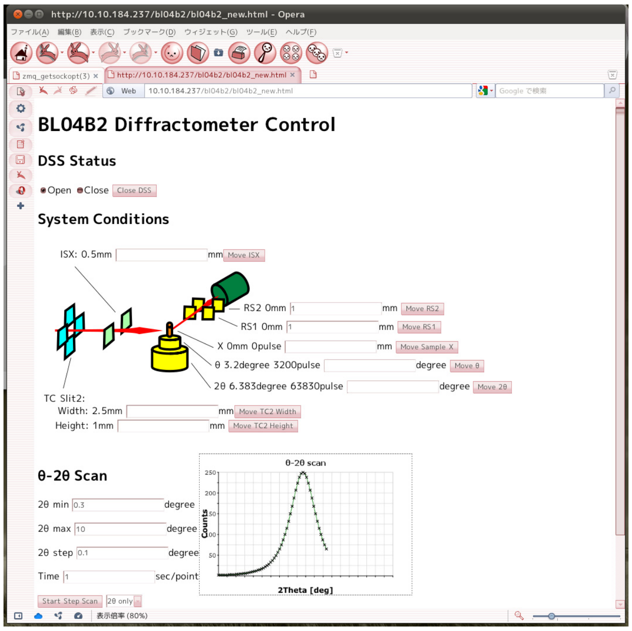

http://10.10.184.237/bl04b2/bl04b2_new.html - Opera
BL04B2 Diffractometer Control
DSS Status
Open Close Close DSS
System Conditions
ISX: 0.5mm mm Move ISX
RS2 0mm mm Move RS2
RS1 0mm mm Move RS1
X 0mm 0pulse mm Move Sample X
θ 3.2degree 3200pulse degree Move θ
2θ 6.383degree 63830pulse degree Move 2θ
TC Slit2:
Width: 2.5mm mm Move TC2 Width
Height: 1mm mm Move TC2 Height
θ-2θ Scan
2θ min 0.3 degree
2θ max 10 degree
2θ step 0.1 degree
Time 1 sec/point
Start Step Scan 2θ only
θ-2θ scan
Counts
2Theta [deg]

Figure 3: A diffractometer control Web-application.

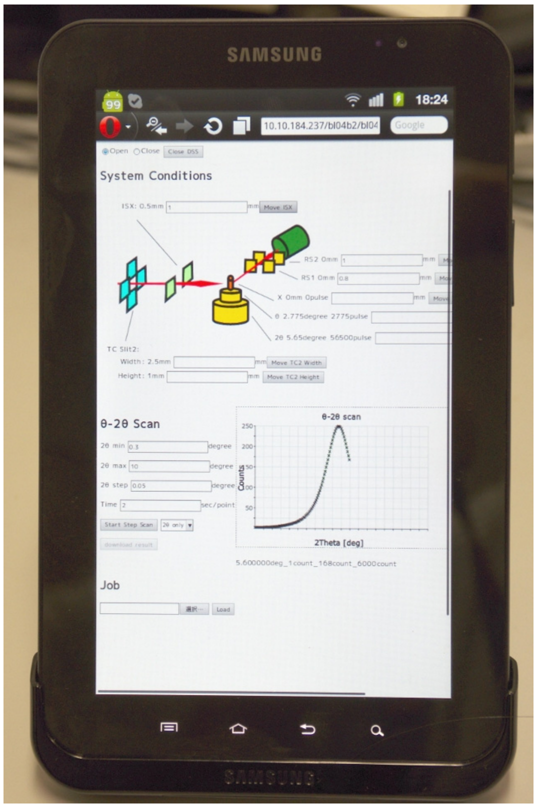


Figure 4: The diffractometer control Web-application running on a mobile device (SAMSUNG Galaxy Tab).

AVAILABLE WEB BROWSERS

The Following Web browsers were tested using the diffractometer-control GUI on a PC, and it was found to work well: Google-Chrome 13.0 on Windows, Debian/GNU Linux and Ubuntu 11.04; Safari on Mac OS X; Firefox 4 on Ubuntu 11.04; and Opera 11.50 on Debian/GNU Linux and Ubuntu 11.04. For Opera and Firefox, WebSocket is disabled as default, so you need to turn on it using "about:config."

Web browsers on mobile devices, Opera Mobile 11.10 on Android OS 2.3.3 (you need to enable WebSocket) and Safari on iPad (1st generation) were available for the WebSocket client. Fig. 4 shows a diffractometer control running on "SAMSUNG Galaxy Tab" which is an Android based tablet with a 7 in LCD screen.

CONCLUSION

As shown in this paper, Web-application can be built using the WebSocket protocol. I have to emphasize that no add-ons are required for executing WebSocket-based applications, just an HTML5 based Web browser. HTML5-based Web-applications provide freedom from the platform, so you can use Web-based applications not only in the central control room but also with various type of equipments. This means the Web-application can be used for local tuning or trouble shooting of the equipment. The web-based application is also useful for remote experiments. You can easily maintain the newest application running on the remote user's PC because the application is loaded from the Web server every time it has to be executed.

REFERENCES

[1] http://websocket.org/
[2] L. Zambon, M. Lonza, "WEB GUIS FOR THE TANGO CONTROL SYSTEM", Proc PCaPAC 2006 P.75 (JLab. USA).
[3] http://infrequently.org/2006/03/comet-low-latency-data-for-the-browser/
[4] http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-10
[5] http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-00
[6] 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", Proc. of ICALEPCS'97, Beijing, China, (1997) p.1
[7] T.Kosuge, et., al., "Recent Progress of STARS", Proc PCaPAC 2005 (2005, Hayama, Japan)
[8] http://bluegriffon.org/
[9] http://www.rgraph.net/