A HTML5 WEB INTERFACE FOR JAVA DOOCS DATA DISPLAY

E. Sombrowski, R. Kammering, K. Rehlich, DESY Hamburg, Germany

Abstract

JAVA DOOCS Data Display (JDDD) [1] is the standard tool for developing control system panels for the FLASH facility and European XFEL. The panels are mainly started on DESY campus. For remote monitoring and expert assistance a secure, fast and light-weight access method is required. One possible solution is using HTML5 as transport protocol, because it is available on many common platforms including mobile ones.

For this reason an HTML5 version of JDDD, running in a Tomcat application server, was developed. WebSocket technology is used to transfer the panel image to the browser. In the other direction, mouse events are sent back from the browser to the Tomcat server. Now thousands of existing JDDD panels can be accessed from remote using standard web technology. No special browser plugins are required.

This article discusses the general issues of the web-based interaction with the control system such as security, usability, network traffic and scalability, and presents the WebSocket approach.

INTRODUCTION

JDDD is a common tool for designing and running control system panels at DESY [2,3,4,5,6]. Around 3000 control panels are started each day from DESY offices and in the control room.

Experience shows that many of these panels are needed by experts for remote assistance. The most comfortable access way is using a web interface, which is available on any PC and mobile device all over the world without the necessity of installing any additional software. The requirement for this interface is that no special web version of the panels should be needed to avoid double work. The communication should be fast and light-weight and has to work also in limited bandwidth environments.

A common technology for modern responsive web communication is AJAX (Asynchronous JavaScript and XML). Using AJAX, the web browser polls the server for data on each update. For the transfer of big data packages at high frequency – as it is needed for the JDDD web interface - the creation of a new HTTP connection every time would be a bottleneck. A better solution is a persistent connection like WebSockets.

WEBSOCKET TECHNOLOGY

WebSockets provide a protocol between client and server, which runs over a persistent TCP connection. A visual representation of such a communication is displayed in Fig. 1:

Handshake

Any network communication that uses the WebSocket protocol starts with an opening handshake. The client web browser sends a request to the server to upgrade the HTTP to WebSocket protocol and if the server supports WebSockets, it sends back a response saying: Ok I am upgrading your HTTP to a single TCP socket connection.

Bidirectional Messages

Through this open connection, bi-directional, full-duplex messages can be exchanged at a high frequency (simultaneously or back and forth). In the case of a JDDD WebSocket connection, panel images, which are created on the server side, are sent to the client and mouse events, which are produced in the browser window, are sent back to the server.

Closing the Channel

If the client or server closes the connection, the panel image creation is stopped on the server and all monitors to control systems values are removed.

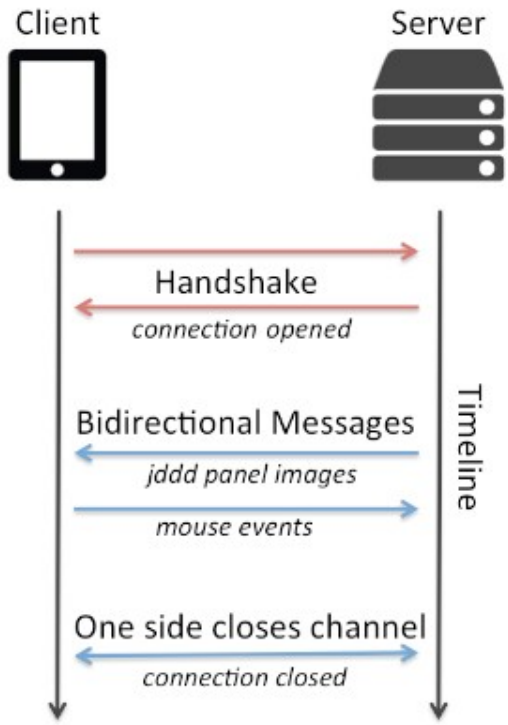

Figure 1: Using WebSockets for client / server communication.

JDDD WEB INTERFACE ARCHITECTURE

Server supporting WebSocket protocol

The WebSocket protocol has been supported since Tomcat 7 and Glassfish 4. Both possibilities were tested and finally the choice was made for Tomcat, because it is more light-weight and the administration is easier.

On the server side one JDDD application is running in the Tomcat web server (see Fig. 2). All panels are started in headless mode in this single JDDD instance. A buffered image is created for each panel with an update rate of currently 0.5 Hz.

Server to Client Communication

To run the JDDD web interface on systems with low bandwidth, the network traffic has to be reduced to a minimum. On the server side this is done by cutting the panel image in 10 times 10 sub-images. Each sub-image is checked for modification and only the parts, which have changed are sent to the client.

Client to Server Communication

In the web browser (client side) all mouse move, click and drag actions are collected using JavaScript. Since the positions differ from browser to browser, they are corrected on the client side before sending them to the server. Now the JDDD web interface emulates the corresponding Java mouse events, so that the panel reacts like on “normal” mouse clicks.

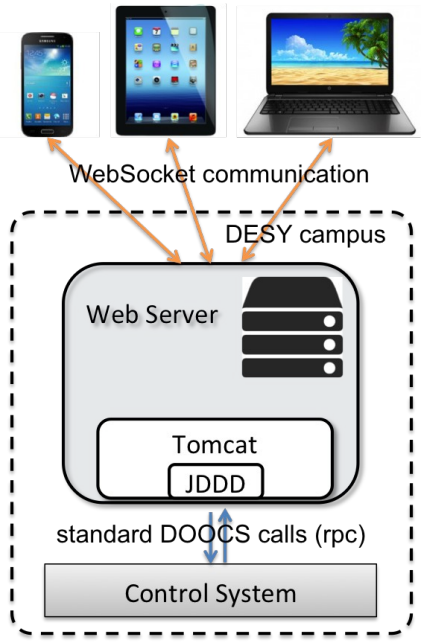

Figure 2: The JDDD web interface architecture using WebSockets for client / server communication.

SECURITY

Before starting the web service a JavaScript login dialog is displayed in the web browser. The user name and password is sent to the server and a DESY Kerberos authentication is executed. The panel is already started in the background and shows up on valid authentication.

The JDDD WebSocket project contains a session manager storing the username, session id and a time stamp for each session. When a new panel is opened on a button click, the session id is inherited from the parent panel and no further authentication is required. The session times out a certain time (currently 1 hour) after the last mouse click. Then all panels of the session are stopped and the session id is removed from the session manager.

During the first test phase, the JDDD web interface has been deployed in a “read only” mode. Changing control system values is not possible.

SCALABILITY

At the moment one Tomcat instance is running on the web server. In this Tomcat server JDDD is started in a single Java Virtual Machine (JVM). The number of panels, which can be operated in one JDDD application, depends on the CPU power, the maximum heap space and on panel complexity. With the existing hardware 15 to 20 standard panels can be started with a proper speed.

Assuming that each user starts approximately 5 control system panels at the same time in his web browser, 3 to 4 people can use the JDDD web interface simultaneously.

To provide the service for a larger number of users, the installation of a Tomcat Cluster is required. A load balancer distributes the incoming requests among the Tomcat servers in the cluster.

CONCLUSION

A proof of concept that the presented technology fulfils our requirements has been done. The web interface has a similar look and feel and functionality as native JDDD (see Fig. 3) and the panel update rate of 0.5 Hz is fast enough for most operational tasks (some restrictions exist: e.g. right mouse clicks cannot be handled, since these are reserved for browser context menus).

The fundamental advantages of the used architecture are:

- Panels have to be designed only once and can be used in standard JDDD as well as in the web interface.
- Changes and improvements in the JDDD source code are instantly available in the web interface.

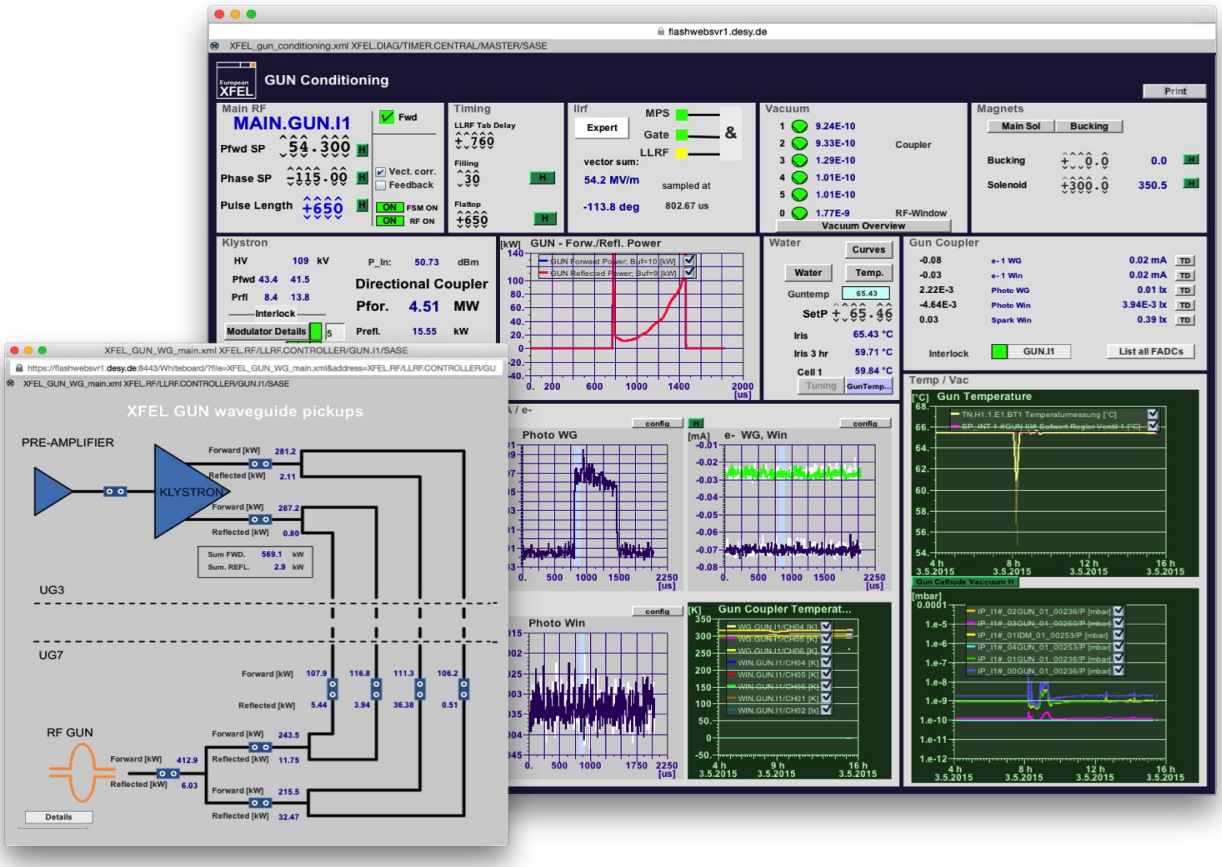

Figure 3: Screenshots of the JDDD web interface running in a Safari web browser (bigger panel) and in a Firefox (smaller panel). All useless browser toolbars are removed and each panel is opened in a separated window. Thus the look and feel and the operation of the panels is equivalent to native JDDD.

OUTLOOK

Some accelerator experts are already working with the current JDDD web interface. To provide the web interface to a larger group of users an improved server setup is required. To support the real operation the “read only” mode has to be changed to full read/write mode, which is planned to be done in the near future.

REFERENCES

[1] jddd website: http://jddd.desy.de

[2] E. Sombrowski. A. Petrosyan, K. Rehlich, W. Schütte, “jddd: a tool for operators and experts to design control system panels”, ICALEPCS’13, San Francisco, USA, October 2013.

[3] E. Sombrowski. A. Petrosyan, K. Rehlich, W. Schütte, “jddd, a state-of-the-art solution for control panel development”, ICALEPCS’11, Grenoble, France, October 2011.

[4] E. Sombrowski, P. Gessler, J. Meyer, A. Petrosyan, K. Rehlich, “jddd in action”, ICALEPCS'09, Kobe, Japan, October 2009.

[5] E. Sombrowski, K. Rehlich, “First Experiences with jddd for Petra Vacuum Controls”, PCAPAC’08, Ljubljana, Slovenia, October 2008.

[6] E. Sombrowski, A. Petrosyan, K. Rehlich, P.Tege, "jddd: A Java Doocs Data Display for the XFEL", ICALEPCS'07, Knoxville, Tennessee, October