A Survey of WebSocket Development Techniques and Technologies

T. Wirasingha#1, N.R. Dissanayake1

1*Informatics Institute of Technology, No.57, Ramakrishna road, Colombo 06, Sri Lanka*
#torinindika@gmail.com

Abstract*—WebSocket protocol has become a sound technology in software industry for real-time web application development, since it provides bidirectional, full-duplex communication between client and server over a single connection. There are several concepts, techniques, and technologies such as frameworks and libraries for WebSocket based development. However, the knowledge existing regarding these concepts, techniques, and technologies is scattered, thus it is not easy to obtain the information required for a specific development environment. In this paper we analyse the literature, then discuss about the existing concepts, techniques, and technologies, available for WebSocket based development. The facts delivered in this paper can be utilized to reduce knowledge search time for engineering of WebSocket based applications, and it can help developers to easily find the needful for the implementation of WebSocket. The knowledge of this paper will be utilized in our ongoing research, towards design and development of a WebSocket server-tool, which will help in rapid development.*

Keywords*—* **Real time web, Survey, WebSocket**

I. INTRODUCTION

The web applications were built on the client-server architecture, in which the client requests for services from the server, and then the server responds with the data or information. Client-server architecture is based on the request/response paradigm of Hyper Text Transfer Protocol (HTTP). HTTP is the fundamental protocol for the service of the web; and, one of the main limitations of HTTP is that it is half-duplex(Lubbers, Albers, & Salim, 2010)In the late 1990s, developers started to develop web pages with dynamic content, with client-side and server-side application development languages.

Whenever a web application runs on a web browser it will be communicating with a server. But if that communication takes a long time, the web page may become an empty page that has no texts and no images. If this occurs frequently, when a user requests "Reload" under heavy loaded servers, it could lead to users' confusion because users lose sight of the operation on a web page. To resolve this issue, AJAX (Asynchronous JavaScript and XML) technology is adopted to web applications (Garrett, 2005). If a web application is constructed based on Ajax technology, the empty web pages will be avoided because client programs communicate asynchronously with web servers. Client programs do not need to wait for the termination of communication with the server.

In 2011, WebSocket was introduced and it provides a full-duplex communication channel over a single WebSocket (WS) connection. WebSocket protocol was standardized by the Internet Engineering Task Force (IETF) as RFC 6455, and WebSocket Application Program Interface (API) is standardized by World Web Consortium (W3C). WebSocket provides a way of creating a persistent, low latency connection, which is capable of handling transactions initiated by either the client or the server, thus full-duplex (Skvorc, Horvat, & Srbljic, 2014) and support both data-push and data-pull.

A. Problem and motivation

There are different concepts, techniques, and technologies regarding WebSocket, based on different programming languages and platforms. In order to develop a WebSocket web application, the developers need to write code for both client and server components using these available concepts, techniques, and technologies. When selecting suitable concepts, techniques, and technologies for the application, they need to spend a considerable amount of time on studying and comparing existing concepts, techniques, and technologies to understand their pros and cons.

Furthermore, we identified that there is a lack of proper details as in surveys related to the WebSocket development which compare real-time web concepts, techniques, and technologies, and their implementations. This setting leads to instil erroneous impressions on software engineers and developers.

To address these issues and fill the gaps, this paper provides an analysis of a survey of the concepts, techniques, and technologies related to WebSocket. In the Background section we discuss and present the pros and cons of the asynchronous communication implementation concepts, techniques, and technologies available for the Rich Internet Application (RIA) development. Then in the next section we present the analysis of the findings of the survey.

B. Methodology

During our literature survey we identified that there are WebSocket server implementations based on different programming languages such as Java, PHP, Scala, JavaScript, C++ (Thorson), ANSI C(libwebsockets.org),etc. We narrowed down the scope of the survey onto JAVA, PHP and python, focusing on the major platforms, emphasising on PHP. For this survey we used three main search engines: Google Scholar, IEEE Xplore, and ACM Digital Library. Through these search engines, we searched for the content using the same set of search strings, as below.

- Rich Internet Applications
- Real time web
- WebSocket
- WebSocket server
- WebSocket server php

We analysed our findings into four main categories: A) Algorithms, Design patterns, and Architectural styles, B) APIs, 3) Frameworks, and 4) WebSocket server-tools, which are presented in the section III of the paper. Under frameworks we are discussing our findings about existing server implementations for WebSocket. We found fifteen existing solutions, and put them into a list and sorted them in descending order based on the popularity in Google search engine and Alexa rankings (http://www.alexa.com/, 1996), we then selected the top most seven to present in this paper.

II. BACKGROUND OF RICH INTERNET APPLICATION DEVELOPMENT TECHNIQUES AND TECHNOLOGIES

In this section we discuss the available asynchronous communication techniques and technologies, and their pros and cons, in order to get a good understanding of the need for WebSocket. Then at the end of the section, we discuss the background of the WebSocket, indicating the importance of the WebSocket towards real-time web application development.

A.AJAX

AJAX (Garrett, 2005) is a group of existing technologies (i.e. HTML, CSS, JavaScript, XML, etc.), which can be seen as the simplest form of the asynchronous communication. AJAX can be defined as a technique rather than a technology, and can also be seen as an architectural style. In AJAX, a client requests from a web server for data/information/resources; then once the response is received, the client processes it and makes changes to the page accordingly, without fully reloading the page but just by updating the necessary sections of the page, which is referred to as partial page rendering.

Since the communication is asynchronous, it does not block the user, and the communication is done faster since less amount of data are communicated, therefore the user experience is enhanced. Client requests are sent as XMLHttpRequest (XHR)(W3C, XMLHttpRequest Level 1, 2014)and the request can point to a static file, which is stored on the server or it's possible to execute a script dynamically. Scripts can communicate with a database server and perform CRUD (Create, Read, Update, and Delete) operations. Data can be communicated in different formats such as XML, JSON, HTML, etc.

The main problem with AJAX is that the communication of it is half duplex in the data-pull mode, thus not suitable for real-time applications (Liu & Sun, 2012). Additionally, in the context of this paper, AJAX exhibits the C10K problem (Kegel, 2014), which states how to provide reasonable service to 10,000 or more concurrent clients using a standard server (Liu & Deters). Technical experiments (Bozdag, Mesbah, & Deursen, 2007) show that if we need high data coherence and high network performance, we should select data-push approach for AJAX.

B. COMET

Comet is an umbrella term, which covers the AJAX based techniques like streaming and long-polling (Gravelle).It tries to reduce and overcome the limitations due to its incapability to start the communication on server side by sending a request to the client. Comet tries to overcome these limitations of the data-pull technique in AJAX, by simulating data-push over data-pull. For example, consider an application developed using long-polling technique; it consists of two steps: In the first step, the client sends a request to the server and the server holds this request open until there is data available for the client; and in the second step, data is sent when available from the server to the client and a new request is started by the client.

With this approach the server can send data to the client immediately. The client polls data from the server by sending a request, which remains open for a long time until data is available to be sent to the client at once.

This technique comparatively good but it has its own drawbacks. For example there can be timeouts if the client-request is open too long. Then in the next step the client has to re-initiate the polling. In addition the behaviour of time-outs can vary for different environments. Also an open request on server side which is not closed requires much of the server resources. Furthermore, Comet techniques also face the 10k issue.

C. Server Sent Events (SSE)

SSE (W3C, Server-Sent Events, 2015)is intended for streaming text-based event data from the server directly to the client, based on data-push mode. The Server-Sent Events EventSource API is standardized as part of HTML5 by the W3C.SSE is used to push notifications into client's components of the web application, which means that it is more suited for real-time communication of server to client. SSE is capable of providing a real-time web application experience to the users by pushing data to the client. However, it is not bi-directional, in that it only supports server push notifications.

D. WebSocket

WebSocket (W3C, The WebSocket API, 2012) provides a way of creating a persistent, low latency connection, which is capable of handling communication initiated by either the client or the server. The client establishes a WebSocket connection through a process known as the WebSocket handshake. This process starts with the client sending a regular HTTP request to the server, in which an upgrade header is included informing the server that the client wishes to establish a WebSocket connection. If the server supports the WebSocket protocol, it agrees to upgrade and communicates this back to the client using an upgrade header in the response (WebSockets - Quick Guide).WebSocket protocol was published as IETF standard and there are many technologies and libraries supporting it. WebSocket protocol is supported by HTML5, Java EE7, and also by a wide range of browsers. Also in Android, IOS applications developers can utilize WebSocket by using different libraries such as Autobahn-library.

For HTTP, the connection is usually closed after every request-response phase. In contrast, in the WebSocket connection started by the client, once the connection is established it is never closed until it is closed explicitly by either the client or the server. While the connection is kept open the client and the server are able to communicate bi-directionally in both data-push and pull modes, since WebSocket supports full-duplex bi-directional communication, allowing the transmission of data in both directions simultaneously. Furthermore, WebSocket addresses the 10k issue, with its small headers and avoiding the overhead of frequent request headers (Pimentel, Bolívar, & Nickerson, 2012).

III. AVAILABLE CONCEPTS, TECHNIQUES, AND TECHNOLOGIES FOR WEBSOCKET BASED DEVELOPMENT

In this section we present the Architectural styles, Application Program Interfaces (APIs), and the frameworks for WebSocket based RIA development, we identified in the survey.

A. Architectural styles

WebSocket supports architectural styles and patterns such as server-push, broadcast, and publish-subscribe (pub/sub) patterns.

1) Push style: Push (also known as server-push)(Zhang & Shen, 2013)describes a way of internet-based communication where the communication is initiated by the server. It is contrasted with pull/get, where the communication is initiated by the client.

2) Broadcast style (Saygin & Ulusoy, 2012) *:* Compared with point-to-point access, broadcast is a more attractive method because a single broadcast of a data item can satisfy all the outstanding requests for that item simultaneously. As such, broadcast can scale up to an arbitrary number of users. There are three kinds of broadcast models as;

- Push-based broadcast(Acharya, Alonso, & Franklin, Broadcast Disks: Data Management for Asymmetric Communication Environments, 1995)(Hameed & Vaidya, 1999) - The server disseminates information using a periodic/aperiodic broadcast program
- On-demand (or pull-based) broadcast(Acharya, Franklin, & Zdonik, Balancing Push and Pull for Data Broadcast, 1997) - The server disseminates information based on the outstanding requests submitted by clients
- Hybrid broadcast(Acharya, Franklin, & Zdonik, Balancing Push and Pull for Data Broadcast, 1997)(Lee, Hu, & Lee, 1999) - Push-based broadcast and on-demand data deliveries are combined to complement each other

2) Pub/sub style (Carzaniga, Papalini, & Wolf, 2011) *:* The publisher-subscriber pattern is a way of passing messages to an arbitrary number of senders and this communication is bidirectional. Pub/sub systems contain information providers, who publish events to the system, and information consumers, who subscribe to particular categories of events within the system. The system ensures the timely delivery of published events to all interested subscribers. A pub/sub system also typically contains message brokers that are responsible for routing messages between publishers and subscribers.

B.APIs

API contains predefine set of protocols, functions/features, and rules, which can be used by developers in software development (Christensson). They come as a package, thus developers can reuse them where they are needed instead of developing the features provided by the APIs from scratch. A powerful API will reduce development time, complexity of the system and it will increase the performance of the application (Jendrock, Cervera-Navarro, & Evans, 2014)

1) JSR 356(Vos, 2013) *:* JSR 356 is an API for java web developers, which can be used to integrate WebSocket in software applications in both server-side and client-side (Vos, 2013). In 2013, Oracle released Java EE 7, which is a rich enterprise software platform, and JSR 356 is a part of that package (Java EE 7 container has the implementation for the API). In Java EE 7, WebSocket client and server components are well balanced, and because of that differences between these components are minimal. Generally, this API follows techniques, which are common to Java EE and it supports two different programming models: Annotation-driven, and Interface-driven.

C. Frameworks

This section presents the top most frameworks (as specified in the methodology) we found for developing WebSocket applications.

1) jWebSocket(jwebsocket) *:* jWebSocket is an open source project, which is released under the Apache License 2.0 on 2010 as a beta version, and it contains support for both client and server component development (jwebsocket). jWebSocket is cross browser compatible and also supports cross platform. Their downloadable package is bundled with WebSocket server (Developed by native java), WebSocket client (Developed by JavaScript) and Clients for Android, Symbian, and IOS. Additionally they have developed a module called "Flash Bridge" for older browsers. According to the inventors, developers can use this framework to build standalone applications and also this can be easily integrated with existing complex web applications.

2) Spring (Spring) *:* Spring is another open source project released under the Apache License 2.0 on 2003. This framework can be used to develop any type of java application. And Spring specifically contains extensions to develop java web applications on top of Java EE platform (Spring). Web framework of Spring is well designed and follows Model-View-Controller (MVC) pattern. One of the key advantages of using this framework is that it allows programmers to develop enterprise-class applications using Plain Old Java Object (POJO). Spring is currently on top of all other java frameworks (Java, 2016) because of its testability and light weight.

Spring Framework 4 includes a module (spring-WebSocket) to support WebSocket and it is compatible with JSR 356. Additionally it provides some other features such as transparent fall-back options to overcome lack of support for WebSocket in some browsers and these are based on SockJS protocol (WebSocket Support). Spring's WebSocket API id is designed in a way that it supports various application servers such as Tomcat (7.0.47+), Jetty (9.1+) and GlassFish (4.1+).

3) Play (Play) *:* Play framework was released on 2009 to the open source community under the Apache License 2.0. It had been developed using both Java and Scala, and this project is still under a process of active development (Play). This framework also supports the MVC pattern.

Play doesn't follow the exact Java EE standards and it is designed with the concept of container-less deployment (Run natively on platforms). Their latest release is Play 2 and they have transferred the entire core and the default template language to Scala. But it still provides server side cooperation with Java libraries, and it could handle single page applications with WebSocket implementation. Another key fact is that Play has more targets in RESTful architecture. Comparing to other Java web frameworks Play is easy to manage. This is because the code need not be compiled, or deployed, nor is it necessary to restart the server to check bug fixes; the page need only be reloaded. LinkedIn (Brikman, 2013) and Coursera(Saeta, 2014) are some examples of popular web applications developed using Play framework.

4) Wrench (Scheirlinck, 2012) *:* Wrench is a WebSocket server and a client package for PHP (5.3/5.4). It is specially targeting the latest stable versions of Chrome and Firefox as the browsers. Wrench is the new version of what was previously known as PHP-WebSocket (PHP-Websockets, 2013). Wrench provides a base class for the WebSocket sever, which implements the recent version of the

WebSocket protocol and which does the socket management and the WebSocket handshake. One of the drawbacks of Wrench is that it uses a single-process server (No thread implementation) (Contributors, 2015). Therefore using this as a standalone package will not be suitable for enterprise level applications, but only for small scale projects. In order to use Wrench for enterprise level applications, developers can use middleware, such as RabbitMQ's STOMP WebSocket plug-in (Pivotal), between the PHP application and the WebSocket clients.

5) Ratchet (WebSockets for PHP) *:* Ratchet is a loosely coupled PHP library, which can be used to develop PHP based real-time web applications over WebSocket. It requires PHP 5.4+ for best performances. It contains nine components, which are developed for different purposes. For example IoServer (base of the application), WsServer component (allows developer's server to communicate with browsers that use W3C WebSocket API), FlashPolicy (allows browsers, which do not natively support WebSocket to connect applications with Flash sockets), etc. By using these components programmers can add functionalities such as HTTP protocol handler, WebSocket protocol handler, etc.

6) Tornado (Tornado) *:* Tornado is a scalable, non-blocking web server and web application framework, released in 2009 for Python. It was developed to be used by FriendFeed (Tornado) and later it was released under the Apache License 2.0 as an open-source project. In their web framework there is a specific module, which supports WebSocket, known as tornado.websocket.

7) Socket.IO (Rauch) *:* Socket.IO is a JavaScript library which is used for real-time web applications in combination with Node.js (Foundation). Node.js is a platform built on Chrome's JavaScript runtime to ease the building of applications in JavaScript that run on the server. Node.js uses an event-driven, non-blocking I/O model, which makes it a popular solution for building real time applications. Socket.IO is one of the third party libraries which can be used with Node.js. It was released in 2010 and was developed by Guillermo Rauch. Latest version is 1.4.5 and it contains separate APIs for the server and the client. Socket.IO uses a separate module called Engine.IO which is the implementation of transport-based cross-browser/cross-device bi-directional communication layer. It made the new version more lightweight.

D. WebSocket Server Tools

During the literature survey we were searching for server tools for WebSocket – such as XAMPP or WAMP for PHP, however we did not come across any. Still any server tool, which can support rapid development of WebSocket applications, is not out there. We assume that the concept and the architectural formalism of the WebSocket applications is too complex to design and develop a server tool to support development of WebSocket applications, however it is worth to be looked into.

In our ongoing research we expect to design, develop and test a WebSocket server tool towards rapid development of WebSocket based applications.

IV. CONCLUSION

The paper presents and discusses the background of the available techniques and technologies for implementing the asynchronous communication of RIAs; also indicating their pros and cons, towards maximising the advantages of WebSocket.

Then the paper presents the analysis of the survey towards the consolidation of the available knowledge of WebSocket development techniques and technologies into a single publication. In the analysis we have briefly discussed the features of the identified concepts, techniques, and technologies. The knowledge delivered in this paper can assist developers of WebSocket based applications, by saving the time that they must otherwise spend on searching and learning the techniques and technologies for WebSocket.

From the survey, we have noted that there are no rapid application development server tools for WebSocket based application development, and we suggest that this be looked into. Furthermore, in the future we hope to introduce a server tool for WebSocket application development.

REFERENCES

(n.d.). Retrieved 03 22, 2016, from Play: https://www.playframework.com/

(n.d.). Retrieved 03 13, 2016, from Spring: https://spring.io/

(n.d.). Retrieved March 28, 2016, from libwebsockets.org: https://libwebsockets.org/

(1996, April 01). Retrieved April 28, 2016, from http://www.alexa.com/: http://www.alexa.com/

Acharya, S., Alonso, R., & Franklin, M. (1995). Broadcast Disks: Data Management for Asymmetric Communication Environments. *ACM SIGMOD Conference.* San Jose: ACM.

Acharya, S., Franklin, M., & Zdonik, S. (1997). Balancing Push and Pull for Data Broadcast. *ACM SIGMOD Conference* (pp. 1-2). Tuscon: ACM.

Bozdag, E., Mesbah, A., & Deursen, A. v. (2007). A Comparison of Push and Pull. *2007 9th IEEE International Workshop on Web Site Evolution* (pp. 5-7). Paris: IEEE.

Brikman, Y. (2013, 02 20). *The Play Framework at LinkedIn*. Retrieved 01 12, 2016, from LinkedIn: https://engineering.linkedin.com/play/play-framework-linkedin

Carzaniga, A., Papalini, M., & Wolf, A. L. (2011). Content-Based Publish/Subscribe Networking and Information-Centric Networking. (pp. 1-2). Toronto: ICN.

Christensson, P. (n.d.). *API*. Retrieved 03 21, 2016, from TechTerms: http://techterms.com/definition/api
Contributors, D. S. (2015, November 06). Wrench Documentation.

Foundation, N. (n.d.). Retrieved April 26, 2016, from Node Js: https://nodejs.org/en/

Garrett, J. J. (2005, February 18). Ajax: A New Approach to Web Applications. Retrieved from https://courses.cs.washington.edu/courses/cse490h/07sp/readings/ajax_adaptive_path.pdf

Gravelle, R. (n.d.). *Comet Programming: Using Ajax to Simulate Server Push*. Retrieved April 26, 2016, from webreference: http://www.webreference.com/programming/javascript/rg28/index.html

Hameed, S., & Vaidya, N. H. (1999). Efficient algorithms for scheduling data broadcast. *Wireless Networks, Volume 5 Issue 3* .

Java. (2016). Retrieved 04 03, 2016, from HotFrameworks : http://hotframeworks.com/languages/java

Jendrock, E., Cervera-Navarro, R., & Evans, I. (2014). *Java Platform, Enterprise Edition The Java EE Tutorial Release 7* . Oracle.

jwebsocket. (n.d.). Retrieved 02 16, 2016, from jwebsocket: http://jwebsocket.org/

Kegel, D. (2014, February 05). *The C10K problem*. Retrieved March 22, 2016, from Dan Kegel's Web Hostel: http://www.kegel.com/c10k.html

Lee, W.-C., Hu, Q., & Lee, D. L. (1999). A study on channel allocation for data dissemination in mobile. *ACM/Baltzer Journal of Mobile Networks and Applications, Volume 4 issue 2* , 2.

Liu, D., & Deters, R. (n.d.). The Reverse C10K Problem for Server-side.

Liu, Q., & Sun, X. (2012). Research of Web Real-Time Communication. *International Journal of Communications, Network and System Sciences* , 2-3.

Lubbers, P., Albers, B., & Salim, F. (2010). Using the HTML5 WebSocket. In P. Lubbers, B. Albers, & F. Salim, *Pro HTML5 Programming* (pp. 138-139). United States of America: Apress.

PHP-Websockets. (2013). Retrieved 01 19, 2016, from GitHub: https://github.com/ghedipunk/PHP-WebSockets

Pimentel, V., Bolívar, U. S., & Nickerson, B. G. (2012). Communicating and Displaying Real-Time Data with WebSocket. *IEEE Internet Computing (Volume:16 , Issue: 4)* , 47.

Pivotal. (n.d.). *Introducing RabbitMQ-Web-Stomp*. Retrieved March 22, 2016, from RabbitMq: http://www.rabbitmq.com/blog/2012/05/14/introducing-rabbitmq-web-stomp/

Rauch, G. (n.d.). *SOCKET.IO 1.0 IS HERE*. Retrieved March 04, 2016, from Socket.io: http://socket.io/

Saeta, B. (2014, 02 18). *Why we love Scala at Coursera*. Retrieved 02 02, 2016, from Coursera: https://building.coursera.org/blog/2014/02/18/why-we-love-scala-at-coursera/

Saygin, Y., & Ulusoy, O. (2012). Exploiting data mining techniques for broadcasting data in mobile computing environments. *IEEE Transactions on Knowledge and Data Engineering (Volume:14 , Issue: 6)* , 1-2.

Scheirlinck, D. (2012). *Wrench*. Retrieved January 25, 2016, from Wrench: http://wrench.readthedocs.io/en/latest/index.html

Skvorc, D., Horvat, M., & Srbljic, S. (2014). Performance evaluation of Websocket protocol for implementation of full-duplex web streams. *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on* (pp. 1-2). Opatija: IEEE.

Thorson, P. (n.d.). *zaphoyd/websocketpp*. Retrieved March 26, 2016, from github.com: https://github.com/zaphoyd/websocketpp
Tornado. (n.d.). Retrieved February 17, 2016, from Tornado: http://www.tornadoweb.org/en/stable/

Vos, J. (2013). *JSR 356, Java API for WebSocket*. Retrieved 03 11, 2016, from Oracle: http://www.oracle.com/technetwork/articles/java/jsr356-1937161.html

W3C. (2015, February 03). Server-Sent Events.

W3C. (2012, September 20). The WebSocket API.

W3C. (2014, January 30). XMLHttpRequest Level 1.

WebSocket Support. (n.d.). Retrieved 01 16, 2016, from Spring: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/websocket.html#websocket-server

WebSockets - Quick Guide. (n.d.). Retrieved March 06, 2016, from Tutorials point: http://www.tutorialspoint.com/websockets/websockets_quick_guide.htm

WebSockets for PHP. (n.d.). Retrieved February 22, 2016, from Ratchet: http://socketo.me/

Zhang, L., & Shen, X. (2013). Research and development of real-time monitoring system based on WebSocket technology. *Mechatronic Sciences, Electric Engineering and Computer (MEC), Proceedings 2013 International Conference on* (p. 2). Shengyang: IEEE.