IOSR Journal of Computer Science (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727
PP 15-19
www.iosrjournals.org

Real Time Data Communication over Full Duplex Network Using Websocket

Shruti M. Rakhunde[1]
[1](Dept. of Computer Application, Shri Ramdeobaba College of Engg. & Mgmt., Nagpur, India)

ABSTRACT : *Internet offers the convenient way of transmitting the information between peers. There is an increased demand of real time data feeds, group communication and teleconferencing which requires a full duplex connection between client and server. But when the real time data transmission is concerned the latency becomes a major issue. The Web Socket offers the better solution as compared to the conventional methods that are considered to be the good solution or delivering real time information. Different methods that have been used over years are HTTP Polling, HTTP Long Polling, Comet etc., but the Web Socket protocol reduces Internet communication overhead and provides efficient, stateful communication between Web Servers and Clients. This paper gives a detailed discussion of various problems of the conventional methods used in real time data transmission and benefits offered by WebSocket with respect to real time data transmission.*

Keywords *- Http Polling, Http Long Polling, WebSocket, HTML5*

I. INTRODUCTION

No one will deny the fact that the web has become the most acceptable way for accessing information in the internet in both business and personal life. WWW has succeeded in distributed information publication system which is unidirectional but the real problem is with the full duplex communication. In today's world with the increasing demand of network services like real time data feeds, group communication and teleconferencing requires the full duplex connection with the server. The HTML5 WebSockets specification defines an API that enables web pages to use the WebSockets protocol for two-way communication with a remote host. It introduces the WebSocket interface and defines a full-duplex communication channel that operates through a single socket over the Web. When real time data communication is concerned the unnecessary network traffic and the latency has become the major issue of concern. Earlier research posits that HTTP wasn't designed for real time, full-duplex communication due to the complexity of real-time HTTP Web applications [1]. Thus the HTTP solution can simulate the real time communication with increased price, increased network traffic and increased latency. The solutions to simulate the full duplex connection were unscalable polling and long polling methods. HTML WebSocket provides the tremendous decrease in the amount of latency and network traffic in communication system. The HTML Websocket is faster than these traditional solutions.

Now from here this paper is organized in various sections. Section 2 gives then brief about the WebSocket protocol, Section 3 gives the literature survey, Section 4 describes various methods for real-time data communication, Section 5 gives a comparative analysis of all these methods, Section 6 is conclusion followed by references.

II. A WEBSOCKET PROTOCOL

The HTML5 WebSockets specification defines an API that enables web pages to use the WebSockets protocol for two-way communication with a remote server. WebSocket is the technology providing the bi-directional, full-duplex communication channels over single transmission control protocol socket. W3C and IETF have standardized the WebSocket API and protocol respectively. Mainstream browser such as Firefox, Google Chrome, Opera and Safari supports Websocket [2].

The websocket protocol has two parts. The handshake consists of a message from the client and the handshake response from the server. The second part is data transfer. HTML5 Web Sockets provides a true standard that you can use to build scalable, real-time web applications. In addition, since it provides a socket that is native to the browser, it eliminates many of the problems. To establish a WebSocket connection, the client and server upgrade from the HTTP protocol to the WebSocket protocol during their initial handshake. Since established, WebSocket data frames can be sent back and forth between the client and the server in full-duplex mode. Both text and binary frames can be sent full-duplex, in either direction at the same time. Simulating bi-directional browser communication over HTTP is error-prone and complex and all that complexity does not scale. Even though end users might be enjoying something that looks like a real-time web application, this "real-time" experience has an outrageously high price tag. It's a price that you will pay in additional latency, unnecessary network traffic and a drag on CPU performance. One of the more unique features WebSockets provide is its ability to traverse firewalls and proxies, a problem area for many applications. A WebSocket detects the presence of a proxy server and automatically sets up a tunnel to pass through the proxy. The following figure shows a basic WebSocket-based architecture in which browsers use a WebSocket connection for full-duplex, direct communication with remote hosts.

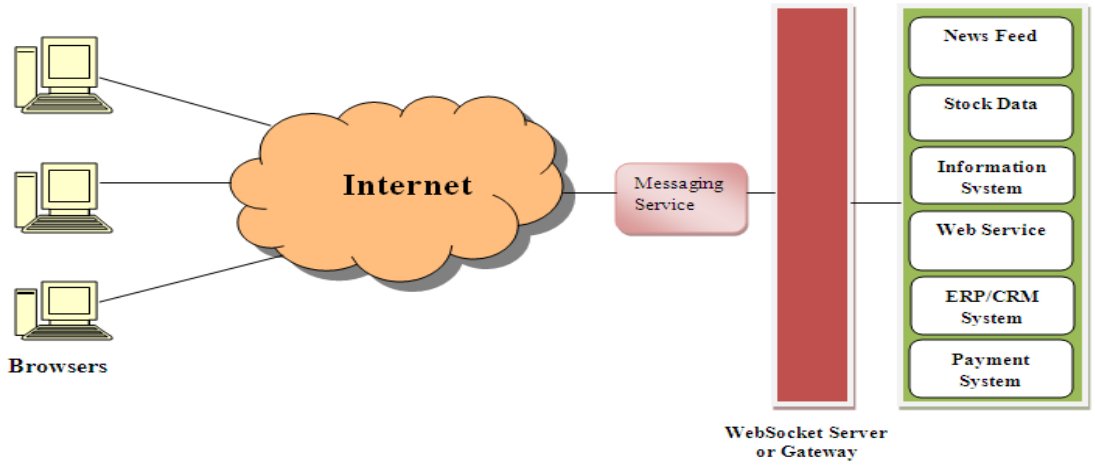


Fig. 1: General WebSocket Based Architecture

The WebSocket standard simplifies much of the complexity around bi directional web communication and connection management. This technology allows developing real time synchronization between multiple users over internet via a web browser.

III. LITERATURE SURVEY

In literature the various researchers have proposed studies related to various aspects of this protocol. In literature there are various work related to real time data communication using websocket, remote data visualization using websocket, etc.

In [1] Victoria Pimentel, Bradford G. Nickerson have compared the one way latency between a client and server using various methods like polling, long polling with WebSocket using WindComm application. The application has components like the wind sensor, the base station computer and client.

In [2] Bo Mao, Zhiang Wu, Jie Cao has proposed a framework for Online Spatio Temporal data visualization based on HTML5 . The data is stored on then sever and it was transmitted to user using WebSocket protocol.

In [3] Andrews Wessels, Mike Purvis, Jahrain Jackson, Syed Rahman have proposed a remote data visualization system architecture using WebSocket that utilizes a few new technologies available in modern web browsers to make remote visualization accessible to anyone without the need to neither download any of the data nor need specialized hardware to visualize it.

In [4] Yan Zhangling and Dai Mao have proposed the real time group communication software architecture called Komm based on WebSocket, which represents the next evolutionary step in web communication compared to comet and Ajax. Comparing with Comet and Ajax implementation, Komm shows better usability, higher performance and lower resource consumption.

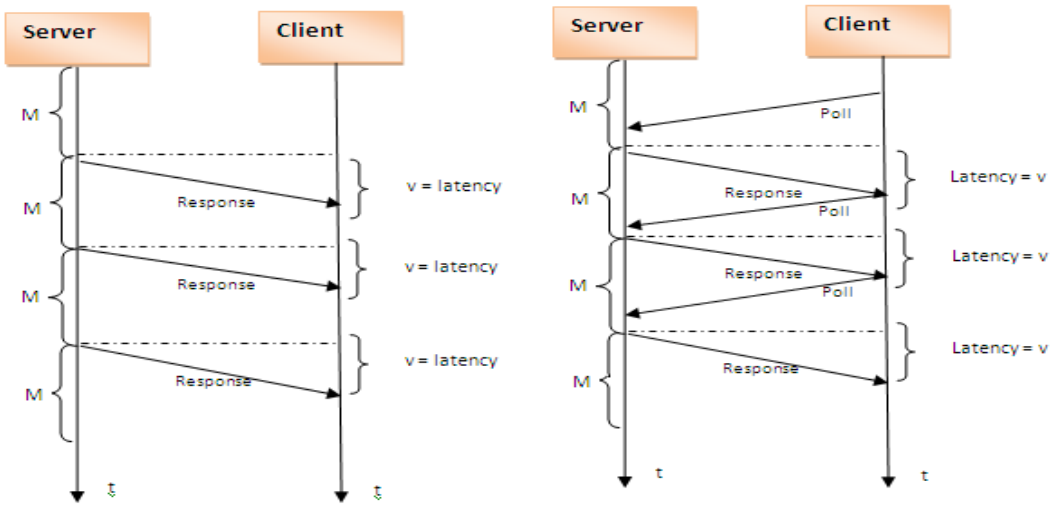
In scientific visualization data are becoming more and more important and usually implies a cooperative effort and experts are usually geographically distributed. To solve the problem the perspective action has been recently initiated with the development of application for collaborative interaction based on two innovative technologies like WebGL and WebSocket. In [5] Charles Marion and Julien Jomier have demonstrated this approach and presented the architecture of the proposed system.

IV. DIFFERENT METHODS FOR REAL TIME DATA COMMUNICATION

HTTP can simulate real time communication environment. The HTTP polling and HTTP long polling can be considered as the good solution for delivering real time information.

4.1. HTTP Polling

HTTP polling consists of a sequence of request response messages. The client sends request-response messages. The client sends a request to a server. Upon receiving this request, the server responds with a new message, if there is any message to send or returns the empty response if no new message is available for client. After a small interval say τ, called as polling interval the server is polled again by client to see if any new messages are available. Various applications like chat rooms, games use HTTP polling.



(a) (b)

Fig.2: General behavior of WebSocket and Long Polling Application. (a) Shows WebSocket behavior and (b) shows long polling behavior, where M is the constant time period in ms at which the actual response become available at the server and v is the latency required in both the application.

4.2. HTTP Long Polling

One weakness associated with polling is the number of unnecessary requests made to the server when it has no new messages for a client. Long polling emerged as a variation on the polling technique that efficiently handles the information push from servers to clients. With the long polling, the server doesn't send an empty response immediately after realizing that no new messages are available for a client. Instead, the server holds the request until a new message is available or a timeout expires. This reduces the number of client requests when no new messages are available.

4.3. WebSocket

With continuous polling, an application must repeat HTTP headers in each request from the client and each response from the server. The WebSocket protocol provides full-duplex, bidirectional communication channel that operates through a single socket over the web and can help build scalable real time Web application. Once the connection is upgraded to WebSocket, messages can flow from the server to the browser the moment they arrive.

Above figure 2 shows the general behavior of the long polling and the Websocket application where the response is sent from the server to the client.

V. COMPARATIVE ANALYSIS OF VARIOUS METHODS

Comparison of the WebSocket protocol with the traditional method like Polling and HTTP long Polling shows the benefits it offers in terms of latency and network utilization. With *polling*, the browser sends HTTP requests at regular intervals and immediately receives a response. This technique was the first attempt for the browser to deliver real-time information. Obviously, it looks like a good solution if the exact interval of message delivery is known, because you can synchronize the client request to occur only when information is available on the server. However, real-time data is often not that predictable, making unnecessary requests inevitable and as a result, many connections are opened and closed needlessly in low-message-rate situations.

With *long-polling*, the browser sends a request to the server and the server keeps the request open for a set period. If a notification is received within that period, a response containing the message is sent to the client. If a notification is not received within the set time period, the server sends a response to terminate the open request. It is important to understand, however, that when you have a high message volume, long-polling does not provide any substantial performance improvements over traditional polling. Here with long polling server doesn't send empty response it holds the request until the message is available or timeout expires.

Now if overhead of network is considered the total HTTP request and response header information calls for overhead. The size of header may vary with respect to application here for example let us consider that header contains 871 bytes and that does not even include any data. The size of header may increase up to 2000bytes in some cases. For analysis let us consider a polling application is deployed for large number of users. Then network throughput for just HTTP request and response header data for two different set of users becomes as follows:

- Use Case A: 1000 Clients polling every second, then network throughput becomes
  871 x 1000= 871000 bytes = 6,968,000 bits per second (6.6 Mbps)
- Use Case B: 10000 clients polling every second, then network throughput becomes
  871 x 10000= 8710000 bytes= 69,680,000 bits per second (665 Mbps)
- **Use Case C**: 100000 clients polling every second: then Network throughput becomes
  871 x 100,000 = 87100000 bytes = 696,800,000 bits per second (665 Mbps)

Above example shows the enormous amount of unnecessary network throughput. Now if this application is rebuild with HTML5 WebSocket there is tremendous amount of reduction in the network throughput due to unnecessary data. Each of these messages is a WebSocket frame that has just *two* bytes of overhead (instead of

871). Now let us consider above example with the WebSocket based application in this case the size of frame header is only 2 bytes. Let us analyze the effect:

- Use Case A: 1000 clients receive 1 message per second, then the Network throughput becomes
  2 x 1000= 2000 bytes= 16000 bits per second (0.015 Mbps)
- Use Case B: 10000 clients receive 1 message per second, then the Network throughput becomes
  2 x 10000 = 20000 bytes = 160000 bits per second (0.153 Mbps)
- Use Case C: 100000 clients receive 1 message per second, then the Network throughput becomes
  2 x 100000 = 200000 bytes = 1600000 bits per second (1.526 Mbps)

As we can see in the following figure, HTML5 Web Sockets provide a dramatic reduction of unnecessary network traffic compared to the polling solution.

Now about the latency issue, the latency of the polling solution, if we assume, that it takes 50 milliseconds for a message to travel from the server to the browser, then the polling application introduces a lot of extra latency, because a new request has to be sent to the server when the response is complete. This new request takes another

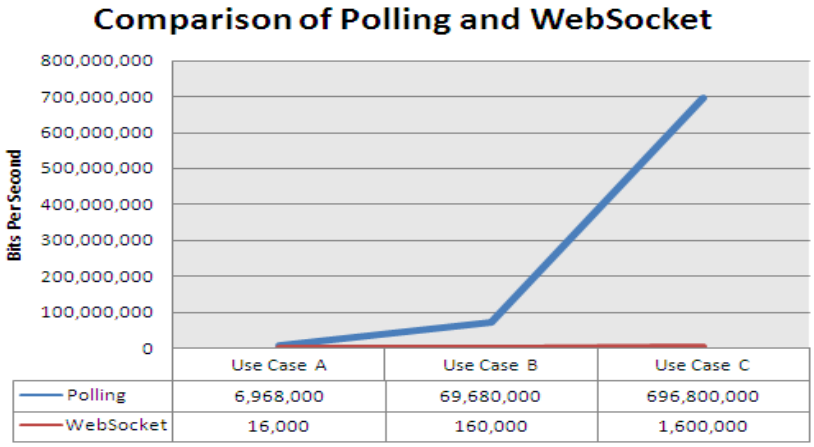


Fig. 3: Comparison of Polling with WebSocket with respect to above example

50ms and during this time the server cannot send any messages to the browser, resulting in additional server memory consumption. The reduction in latency is provided by the WebSocket solution. Once the connection is upgraded to WebSocket, messages can flow from the server to the browser the moment they arrive. It still takes 50 ms for messages to travel from the server to the browser, but the WebSocket connection remains open so there is no need to send another request to the server.

VI. CONCLUSION

Thus this paper provides a detail description of various methods that are used for real time data communication. Here we have discussed various issues involved in designing the application using traditional methods like polling and long polling. Also this paper gives the comparative analysis of methods like polling, long polling with the WebSocket while considering various performance measurement parameters like network overhead and latency. The comparative analysis shows that the websocket offers the better solution. WebSocket protocol offers the tremendous reduction in network overhead and latency when real time data communication is concerned.

REFERENCES

[1]Victoria Pimentel, Bradford G. Nickerson, Communicating and displaying real-time data with WebSocket, *Internet Computing, IEEE, Vol. 16, Issue. 4, Jul-Aug 2012*
[2]Bo Mao, Zhiang Wu, Jie Cao, A framework for online spatio-temporal data visualization based on HTML5, *International Archieves of the Photography, remote Sensing and spatial information Science, Volume XXXIX-B2, 2012,123-127*
[3]Andrew Wessels, Mike Purvis, Jahrian Jacksopn, Syed Rahman, Remote data visualization through WebSockets, *Information Technology, New Generations (ITNG),8th International Conference, 2011,1050-1051*
[4]Yan Zhangling,. Dai Mao, A real-time group communication architecture based on Websocket,, *International Journal of Computer Engineering Vol. 1.No. 4. Nov 2012*
[5] Charles Marion, Julien Jomier, Real-time collaborative scientific webgl visualization with websocket, Web3D'12 Proc. Of 17th International Conference on 3D Web Technology, 2012, 47-50
[6]Vinaitheerthan Sundaram,Lan Zhao, Carol X. Song, Bedrich Benes, Rakesh Veeramacheneni, peter Kristof, Real-time data delivery and remote visualization through multi-layer interfaces, *proc. of Grid Computing Environments Workshop, 2008. GCE '08, Nov 2008, 1-10.*