Nigerian Journal of Technology (NIJOTECH)
Vol. 38, No. 4, October 2019, pp. **1010 – 1020**
Copyright© Faculty of Engineering, University of Nigeria, Nsukka,
Print ISSN: 0331-8443, Electronic ISSN: 2467-8821
www.nijotech.com
http://dx.doi.org/10.4314/njt.v38i4.26

WEBSOCKET IN REAL TIME APPLICATION

K. E. Ogundeyi[1] and C. Yinka-Banjo[2,*]
[1, 2,] DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF LAGOS, AKOKA, LAGOS STATE, NIGERIA.
E-mail addresses: [1] *kehindeogundeyi@gmail.com*, [2] *cyinkabanjo@unilag.edu.ng*

ABSTRACT

There has been an increase in request for real time data feeds, teleconferencing and group communication which entails full duplex connection amongst client and server. Real time web application involves clients which is also known as a browser getting updates from the server as they happen. with regards to the limit observed in old-style real-time communication which include long polling, polling server-sent events and comets, this paper recommends using upcoming Web Socket technology when dealing with real time. The Web Socket offers improved result as compared to the conventional approaches that are considered to be the good solution of providing real time information and lessens overhead acquired while communicating over the internet and offers stateful, efficient communication among Web Servers and Clients.

Keywords: *WebSocket, Http Polling, Http Streaming, Server-Sent Events, Comet*

1. INTRODUCTION

Real time web application involves clients which is also known as a browser getting updates from the server immediately as they occur [1]. An example of real time web application is google doc in which documents can be edited by a user and can also see what is been done by others simultaneously.

A request is initiated by an HTTP client and in which a Transmission Control Protocol (TCP) connection is established [2]. After the client's request has been received, a response is sent back by the server and thereafter the connection is terminated. Using this model, real time data cannot be sent by servers to clients. Consequently, Technologies like Ajax long polling and comet have been used to accomplish real time communication between server and client. Still, real time communication cannot be accomplished in these technologies because some of them needs plug-in to be installed on browser which put heaven burden on the server. Real time data communication was achieved with the advent of Web socket protocol and HTML 5.

WebSocket description depicts an Application Programming Interface which allow the use of web socket protocol by a web page for two-way communication alongside a remote host [3]. Thus, the HTTP solution can simulate the real time communication with high tag in price, increased network traffic and increased latency. The results to act out the full duplex connection were un scalable polling and long polling methods. WebSocket offers the incredible reduction in the quantity of latency and network traffic in communication system. The WebSocket is quicker than these old-fashioned solutions.

2. LITERATURE REVIEW

2.1 Internet Protocol

Internet protocol suite encompass completely different communication protocols that work over the web and different remote communication networks, and it move (from one place to another) most of the extremely important services which run over the network. It offers end-to-end connectivity by establishing, maintaining, and releasing connections among the sender and receiver. Routing of network, IP addressing, error management and flow management is also delivered by the internet protocol [4].

2.2 1real Time Web Definition

Synchronous communication system also known as real-time communication system always includes a large number of passive recipients [5].

* Corresponding author, tel: + 1 – 617 – 852 – 4399

Web clients is able to poll server-side events because of the Light comfort provided by AJAX to the HTTP communication model. Comet presented a better approach by deviating from HTTP model of Communication and permitting communication over HTTP with push-style.
With the advent of HTML 5, web socket brings about a higher development than Ajax and Comet. The condition of HTML 5 WebSocket defines a single socket full duplex (bi directional) connection which is used to pull and push data between a client (browser) and a server, thereby delivering a better result than comet and Ajax polling [6].

2.2.2 Prior Methods for Real Time Data Communication

HTTP can simulate real time communication environment. The HTTP long polling and HTTP polling can be considered as the good solution for delivering real time information [7]:

2.2.2.1 Http Polling

A common feature of HTTP Polling is its sequence in request response messages. A request is sent by the client to the server, when the server receive the message, it returns a new message if there is a message to send otherwise an empty response is returned if there is no message available for them. After a small intermission say t, called as polling interval the server is polled again by client to realize if there is a vacant new message. Examples of application which use HTTP Polling are chat rooms and games.
For HTTP Polling to be adequate in providing real time details, the interval of delivering the message must be known which implies that the rate of transmitting the data must be stable. Thus, the client will only be synchronized by the application developer to request for data when it is certain to be available. However, latency is accumulated when the rate of data request grows which is caused due to the constant repetition of important header details by the overhead intrinsic to HTTP polling. Due to the complexity of real-time HTTPWeb applications, it has been proposed in early research that HTTP was not conceived for real time full duplex communication. Thus, the HTTP solution can simulate the real time communication with high tag in price, increased network traffic and increased latency.

2.2.2.2 Http Long Polling

A flaw in polling is the large amount of futile request it sent to the server when it does not have new message for a client. Therefore, the details which are pushed from server to client are well handled by HTTP Long polling which is a variation of polling [8]. With HTTP long polling when there is no new message to be sent to the client, the server does reply with an empty message but instead keep the request until a fresh message is available to be sent to the client or timeout occurs thereby decreasing the number of pointless request to the server but still HTTP Longpolling did not solve the problem in traditional polling as there is still increased latency and overhead.

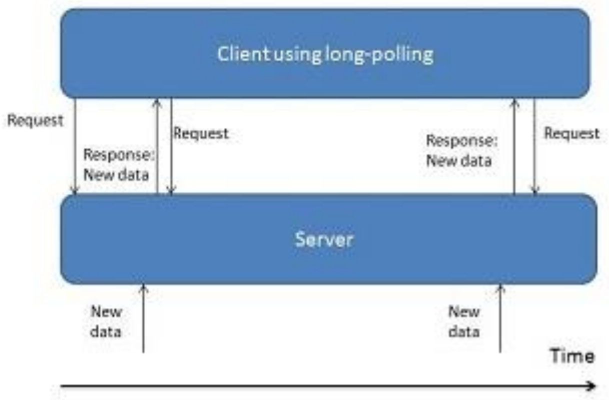


Figure 2: Client Using Long Polling

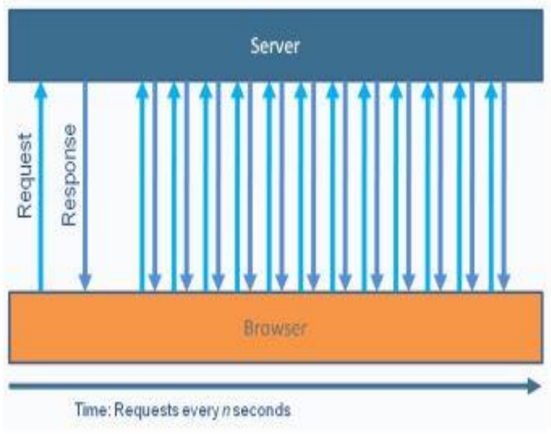


Figure 1: Client Using HTTP Polling

2.2.2.3 Http Streaming

HTTP Streaming which was first made known to the web by Netscape in 1992 originates in two kinds namely page streaming and service streaming. The content of the server streaming in a long-lived HTTP Connection is contained in page streaming as well as service streaming [9]. With HTTP Streaming, the server does not close the connection but keeps it open throughout a client session by running a long loop. A change in state is detected by the server script using some methods like event registration. Once a change is detected, the new data is streamed and flushed while the connection remains open. For the meantime, the new data must be displayed in the user interface while awaiting the server's response [9]. A difference between the forms of streaming is that service streaming is initialized by client's request which is known as XHR-streaming while page streaming uses the request of the initial page to stream data which provide flexibility in the duration of connection [10]. Example of HTTP Streaming implementation is Forever frame (an iframe which obtains script tags from a server in an everlasting response) Codes are implemented in script tags when it is been read by a browser, therefore the data received by a client from the server are enclosed as JavaScript functions [11]. Table 1 shows the comparison between Http Long Polling and Http streaming protocols

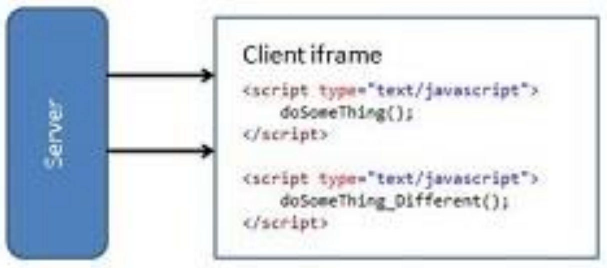
2.2.2.4 Comet

Comet scheme which exist due to the persistence attribute of HTTP/1.1 uses the methods practiced by long polling to accomplish real time behavior [10]. Before the advent of Persistence connection, different URLS are been fetched by different TCP connection which cause congestion on the internet because of the weight being put on the server but with persistence connection the connection between the client and the browser are always open till it is closed by either one of the party sending a precise close connection message or a time/network error take place. As soon as the connection is closed, the server cannot start it back to gain a connection to the browser [11].

With persistence connection, CPU time is saved in both hosts and routers, fewer TCP connections are opened and closed and the memory use in the control blocks of the TCP Protocol can be saved in hosts.

2.2.2.5 Server-Sent Events

Server sent Event uses the text/event stream feature of HTTP/1.1 content type to send message to the browser (client). The channel in which server-sent event use to communicate to the client from the server is a one-way communication channel. However, the client has to subscribe to the channel by first connecting, then when there is a fresh information available the server post events. The connection is always open until it is closed by a proxy or the client itself. The connection can also be configured to close after a duration of time, same as the client reconnection time [12]. As shown in Figure4, Server sent Events behaves quite like long-polling. Server-sent events can be used by developers to access APIs. Event Source interface receive access from the API thereby providing a direct JavaScript code, allowing events to be triggered by the server to the browser and updating the content on the client [6]. When an ID is set on messages sent, the client reconnects and continue where it stops after a look up is been done by the server on its ID. This feature makes server sent event robust.



```
<script type="text/javascript">
    doSomeThing();
</script>

<script type="text/javascript">
    doSomeThing_Different();
</script>
```

Figure 3: HTTP streaming

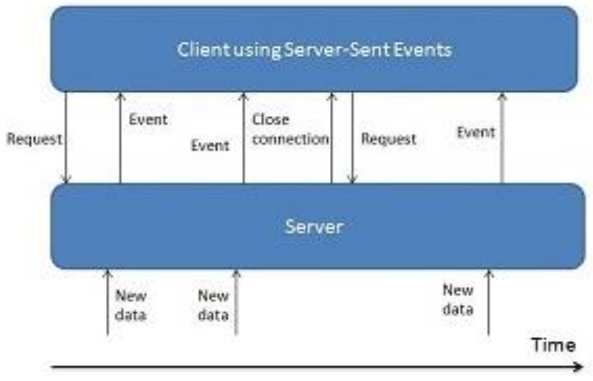


Figure 4: Client Using Server Sent Events

Table 1: Comparison between Http Long Polling and Http streaming protocols

HTTP LONG POLLING	HTTP STREAMING
With HTTP long polling when there is no new message to be sent to the client.	With HTTP Streaming, the server does not close the connection but keeps it open throughout a client session by running a long loop.
The server does reply with an empty message but instead keep the request until a fresh message is available to be sent to the client or timeout occurs thereby decreasing the number of pointless request to the server	A change in state is detected by the server script using some methods like event registration. Once a change is detected, the new data is streamed and flushed while the connection remains open

The model in which client-side events are transmitted to the server (i.e. a user clicks on a link to request a new page from the server) can be referred to as "client-sent events". Server-sent events permits servers to pass information to the client as it becomes accessible, without client polling. To use Server Sent Events, an application implements an Event Source API in the browser and pushes data from the server with Server Sent Event's event stream data format.

2.2.3. Drawbacks of Real Time Web with Http

2.2.3.1 Overhead

Achieving real time with HTTP is a naïve approach because HTTP headers consume excess data which is in most case not useful for real time application while WebSocket protocol is intended for it because it makes available a full-duplex, bidirectional communication channel for use which is carry out with a single socket on the web and can develop scalable, real time Web applications.

2.2.3.2. Unidirectional

HTTP is unidirectional so when using real time application with HTTP, several TCP connections is used to achieve a replicated bi-directional communication [8], even with server sent event, a connection will still be need to push events from the server to the client and also to send message to the server but WebSocket protocol which bring improvement to real time applications is purposely for the aim of bi-directional communication which real time application is all about.

2.3 Web Socket

Communication overhead can be intensified by the use of continuous polling in an application. The WebSocket protocol allows for a robust real time web application because it provide a full-duplex bi-directional communication channel which is done with a single socket [9].

2.4 Web Socket Architecture

WebSocket protocol consist of two categories; the first part is the Handshake which contains the handshake response from the server and the message from the client, the second of the category is data transfer. The Web socket also provide a socket that is innate to the browser which eliminate most of the problems with using HTTP thereby building a scalable real time application.

For a WebSocket connection to be established, both the server and client has to be upgraded from HTTP protocol to web socket protocol at the early stage of handshake. When the connection has been established, WebSocket data, text and binary frame can be sent at the same time in any direction in full duplex mode. Replicating the WebSocket bi-directional communication with HTTP comes with high price to pay in slow CPU performance, increased latency and network traffic. This can be very difficult and prone to error due to the complexity associated with real time application [12]. Another benefit of WebSocket is that it has the ability to transverse proxy and firewall which is an area difficult for many applications. WebSocket pass by proxy, by spontaneously setting up a tunnel when it detects the presence of a proxy server.

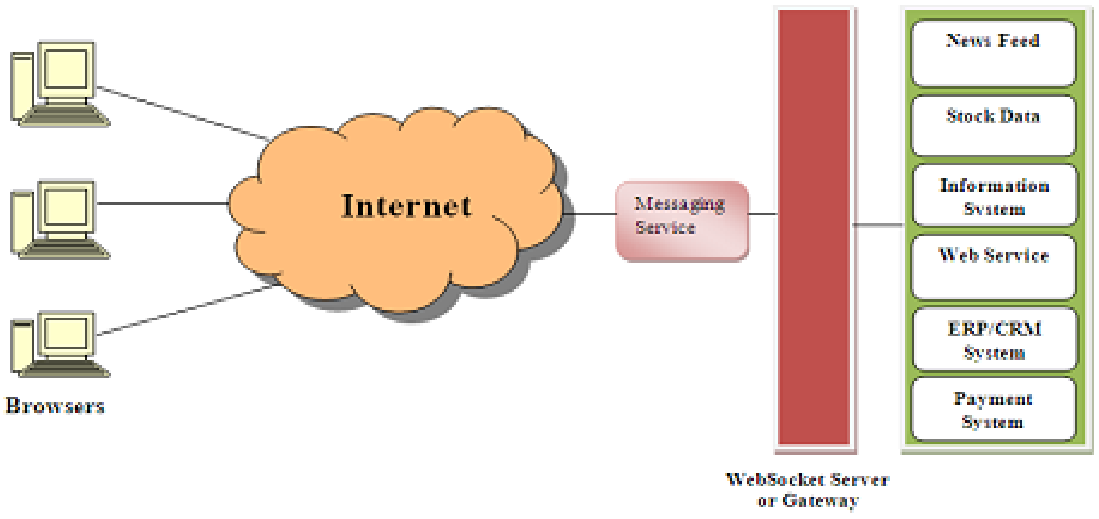


Figure 5: Web socket Architecture

3. THE COMMUNICATION BASED ON WEBSOCKET PROTOCOL

HTTP has been used for real time web to update data before the advent of WebSocket protocol, but it obviously has flaws. To begin with, the packet's head of HTTP Protocol is significantly huge, which increases network overhead. Secondly with HTTP protocol, TCP connection must be established by the server for each user which waste resources and put much pressure on the server. WebSocket Protocol, similar to Socket whose communication is based on TCP/IP is a full duplex communication protocol [3]. The major advantage of WebSocket is that information can be sent between client and server at any time. Ajax communication does not restrain WebSocket because in Ajax technology, it is required that the client initiate the request. Contemporary browser fully supports WebSocket protocol so WebSocket client can push data to the server [2].

3.1 Principle of the Websocket Protocol

The establishment of WebSocket connection is in this manner, the browser first sends a WebSocket connection request, in which the browser responds to the request sent by the client, the process described above is known as 'handshake'. The data transmission at this phase is based on text and use ASCII encoding and is compatible with existing HTTP/1.1 protocols.
In WebSocket, the browser and server only need to establish a connection by handshaking as illustrated in Figure 7. Because the port number is already known by each process through the WebSocket Protocol, information can be sent and received between the browser and the server before the connection is released [13]. To prevent malicious script attack and make the communication more secure, the message needs to be encrypted when the header is being sent by the browser to the server. This is done by the server constructing a SHA-1 message digest with the browser request header "sec WebSocket-Key" and combining it with a SHA-1 encrypted magic string which is encoded in BASE-64. This result is then returned as the value of the Sec-WebSocket Accepted header to the browser.

3.3 State of the Art Applied Areas of Websocket in Realtime Application

The following describes the examples in which WebSocket have been applied to real time applications:

3.3.1 Social Feeds

WebSocket has been applied to social app so feeds can be updated in real time which lets you know what people are doing and when they do them.

3.3.2. Multiplayer Games

Effective interaction between players is very important as the web is at the verge of being game platform independent. WebSocket is been used in games to produce high gaming performance by developers.

3.3.3. Collaborative Editing/Coding

Version control (example is Git) has always been used to merge edited documents but there is still need to monitor users when there is a conflict Git cannot handle. But with a collaborative solution like WebSocket, it is easy knowing people working on the same documents and those editing it.

3.3.4. Clickstream Data

Analyzing how users interacts with an application will help to improve it. The cost of overhead brought by HTTP has necessitate making important and collecting only the most useful data.
Then, realizing a different metric which seemed insignificant but would have been useful in making important decision should have been collected. With WebSocket, the movement of mouse can be tracked and data can also be sent in the back end.

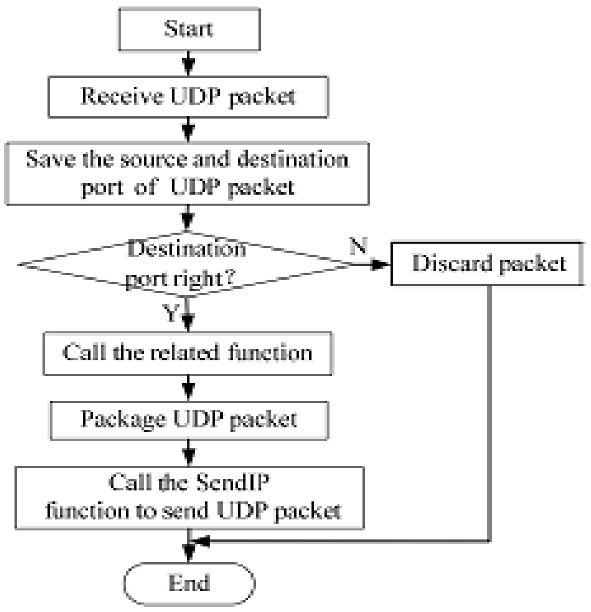

Start
Receive UDP packet
Save the source and destination port of UDP packet
Destination port right?
N
Discard packet
Y
Call the related function
Package UDP packet
Call the SendIP function to send UDP packet
End

Figure 6: Web socket communication principle diagram

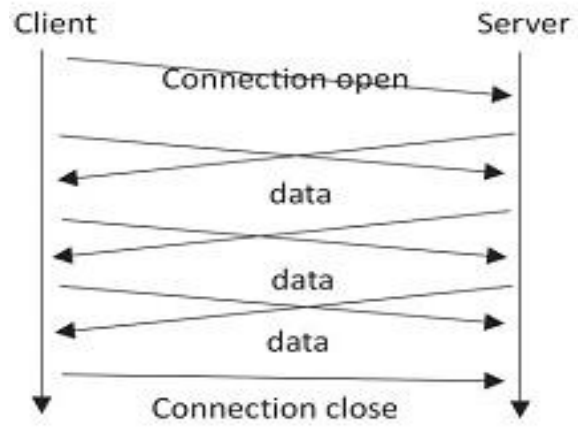


Figure 7: Web socket communication Protocol

3.3.5. Financial Tickers

The finance world is moving fast and human brains cannot sustain processing data at that speed, so algorithms are used. When there is need to track companies worth using a dashboard now and not some seconds ago, WebSocket can be used to stream data without delays.

3.3.6. Sports Updates

Another benefit of WebSocket is its application to sport updates, Users can be kept up to speed if sports information is to be included in Web app.

3.3.7. Multimedia Chat

Person to person meeting cannot be replaced but video conferencing is used when everyone cannot be present. The plug in used in videoconferencing is quite heavy so WebSocket is been used with HTML5, videos elements and get User Media API's as a better option.

3.3.8. Location-Based Apps

Applications are now using GPS capability to get user locations such as tracking user's progress along a route, this will enable useful data to be gathered. WebSocket is been used to do this as HTTP is unnecessarily bulky.

3.3.9. Online Education

Online Education is an excellent medium of learning especially when the cost of education is getting expensive and the internet continue getting cheaper and faster. WebSocket provides a great online education platform for interaction between students and teachers.

3.4 The Implementation of Web Socket Technology

3.4.1. Web Socket's Implementation on the Client Side

Implementing WebSocket on the browser is quite simple. The definition of the interface of WebSocket according to W3c team is explained thus [10]:

Algorithm 1: Implementation of WebSocket on the client side

```
[Constructor(in DOMStringurl, in optional DOMString protocol)]
interface WebSocket
{read only attribute DOMString URL;
// ready state
const unsigned short CONNECTING = 0;
const unsigned short OPEN = 1;
const unsigned short CLOSED = 2;
readonly attribute unsigned short ready State;
readonly attribute unsigned long buffered Amount;
// networking
attribute Function onopen;
attribute Function onmessage;
attribute Function onclose;
oolean send(in DOMString data);
void close();};
```

From Algorithm 1, A valid network address and a protocol type are the parameters which can be initialized by a new WebSocket instance following the definition of construct function and interface. In Algorithm 2,JavaScript handle a WebSocket object and an instance of WebSocket is created with the following code:

Algorithm 2: JavaScript handle a WebSocket object

```
varmyWebSocket = new WebSocket
It is important to check if WebSocket is supported in the browser of the client before initialization is made.
if ("WebSocket" in window)
{varwbs = new WebSocket
("wbs://exampletest.com/service");}
else
{alert ("Does not support WebSocket");}
```

Functions that will be used for handling events like receiving of messages, closing of the message and successful connection establishment must be registered before the messages are sent as demonstrated in Algorithm 3.

Algorithm 3: Establishment of a successful connection

```
myWebSocket.onopen = function(evnt)
{alert("opening connection ..."); };
myWebSocket.onmessage = function(evnt) {alert( "Received Message: "+ evnt.data);};
myWebSocket.onclose = function(evnt)
{alert("Connection closed.");};
```

For a message to be sent, the post message method along with the content of the message which will be used as the default parameter should be called. To terminate the connection after sending the message, the disconnect method should be called as shown in algorithm 4.

Algorithm 4: Terminating a connection

```
myWebSocket.postMessage("HeyWebSocket ");
myWebSocket. disconnect(); stockTickerWebSocket.
disconnect();
```

3.4.2. Websocket Implementation on the Server Side

Implementing WebSocket on the server side is more complex than implementing on the client side because many of the operations on the client side are all done automatically by the browser such as beneficial data extraction, headers generation and header analysis but these are not executed by the servers but done manually by the developer. The server-side implementation of WebSocket depends mainly on Socket programing which is popular for languages like C++, C# and Java. Implementing WebSocket Server side function with C#, to begin with the new request should be monitored in the network by creating a new listener.

Private Socket server Listener = new Socket (Address Family. InterNetwork, SocketType. Stream, protocol lType. IP);

Since the accept function is in charge of listening to new request coming in, it should be in a loop which runs at all time for receiving client request at all time.

Algorithm 5: Looping a new request

```
While(true)
{
Socket sck= serverListener.Accept();
//new connection recieved
If (sck!= null){….} //request been processed
}
```

Quite a lot of function need to be registered when a new connection is received for events handling such as sending of messages, receiving of messages and closing the connection which occur during the communication.

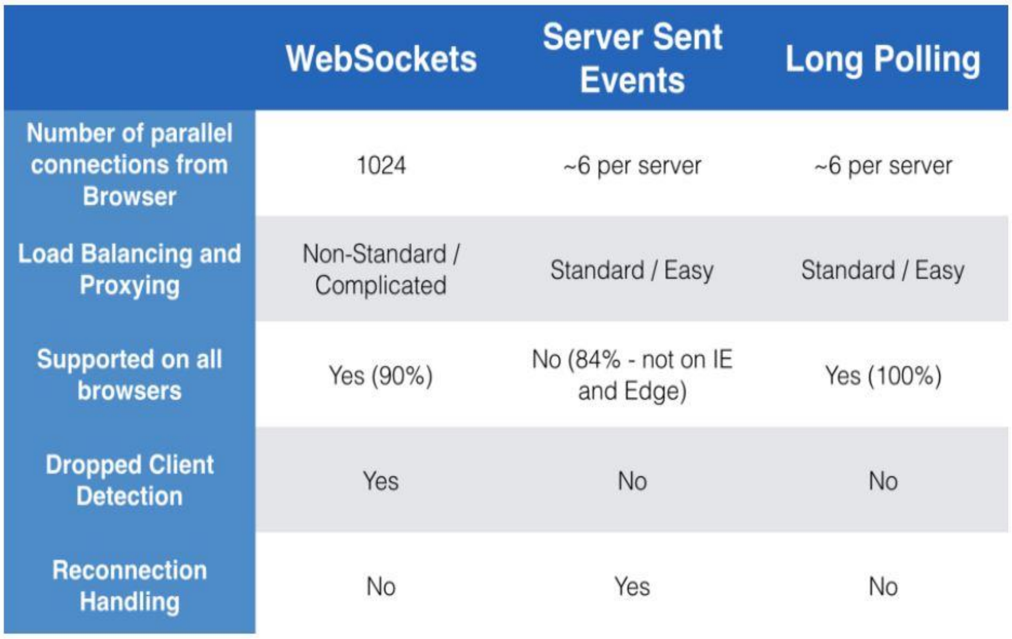
Algorithm 6: Receiving of messages and closing the connection

```
ci. ReceiveData += new ClientSocketEvent (Ci_ReceiveData);
ci. BroadcastMessage += new BroadcastEvent (ci.SendMessage);
ci. DisConnection += new ClientSocketEvent (Ci_DisConnection)
ci. ClientSocket.BeginReceive(ci.receivedDataBuffer,0, ci.receivedDataBuffer.Length, 0, new
AsyncCallback(ci.StartHandshake), ci.ClientSocket.Available);
```

Begin Receive method is call to receive messages from the client request and then handshake with the client browser is made. If the handshake with the client browser is successful then full duplex communication can begin The Start Hand Shake method is responsible for generating handshake data which is grounded on the clients' request. Using the method above, the results gotten from key "Sec-WebSocket-Key1" and key "Sec-WebSocket-Key2" are gotten from the request headers, the computation of MD5 are usually carried out with the two values generated from the two keys and returned at the end of the result. The result gotten from MD5 serves as a mean of keeping data safe during the handshake process [10]. If the handshake process is successful, new connection will be put into the connection poll for reuse next time.
list Connection. Add(ci);
What should be paid attention here is putting the character "\x00" at the beginning of messages and "\xFF" at the end of the message when sending messages and removing these two characters when reading messages. Additionally, message should be encoded or decode byUTF-8 before using.

Algorithm 7: Implementation of WebSocket on the server side

```
public void SendMessage(MessageEntity me)
{
ClientSocket.Send(new byte[] {0x00});
ClientSocket. Send(Encoding. UTF8. GetBytes (Json-
Convert.SerializeObject(me)));
{ ClientSocket.Send(new byte[] { 0xff });
}
```



	WebSockets	Server Sent Events	Long Polling
Number of parallel connections from Browser	1024	~6 per server	~6 per server
Load Balancing and Proxying	Non-Standard / Complicated	Standard / Easy	Standard / Easy
Supported on all browsers	Yes (90%)	No (84% - not on IE and Edge)	Yes (100%)
Dropped Client Detection	Yes	No	No
Reconnection Handling	No	Yes	No

Figure 8: Comparison of WebSocket, Server-Sent Events and Long polling

Finally, call DisConnection method to close the connection when communication is over.

3.5 Comparative Analysis of Websocket With Other Real Time Methods

Comparing other methods such as polling and HTTP Polling with WebSocket, the demonstrations shows that WebSocket has more advantages in term of network utilization and latency.

With polling, Http requests are consistently sent to the server, Polling was the first method used in real time application and would have been a good method if the interval to deliver messages were known but because of the un predictable nature of real time data request, using Polling causes opening of several connections for a low-message situation which are caused by needless requests been sent to the server.

With longpolling, requests are sent by the client to the server, when the server receives the message, it returns a new message if there is a message to send otherwise an empty response is returned if there is no message available for them. After a small intermission say t, called as polling interval the server is polled again by client to realize if there is a vacant new message. With long polling, empty request is not sent back by the server, but it holds the request until there are available messages, or a timeout occurs.

If HTTP request and the information of the response header is referred to as the overhead of the network, considering that the header may be different for different application,

For example, a header (which has the tendency of increasing to almost 2000 bytes) contains 923 bytes excluding data.

Analyzing this data through polling method gives the below network throughput result for response header data and HTTP request for different users:

Case A: if 1000 Clients polls every second, then the network throughput is923 x 1000= 923000 bytes = 7384000 bits per second (6.6 Mbps)

Case B: if 10000 Clients polls every second, then the network throughput is923 x 10000= 9230000 bytes= 73840000 bits per second (665 Mbps)

Case C: if 100000 Clients polls every second, then the network throughput is 923 x 100,000 = 92300000 bytes = 92300000 bits per second (665 Mbps)

The example above shows the massive throughput, which was used during polling method, the below shows the result when the application is rebuilt with WebSocket. Analyzing with just 2 bytes as the header instead of 923, we have:

Case A: if 1000 clients receive 1 message per second, then the Network throughput becomes 2 x 1000= 2000 bytes= 16000 bits per second (0.015 Mbps)

Case B: if 10000 clients receive 1 message per second, then the Network throughput becomes 2 x 10000 = 20000 bytes = 160000 bits per second (0.153 Mbps)

Case C: if 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 shown in figure 9, with WebSocket, there is great reduction in network traffic unlike polling methods. WebSocket also reduces the latency period that could be incurred using polling method, with polling method, if it takes 65milliseconds for a message to be sent from the server to a client, for every poll done to the server, it takes another 65milliseconds which increases latency and continuous server memory utilization but with WebSocket, it takes only 65milliseconds only to send message from the client to the server.

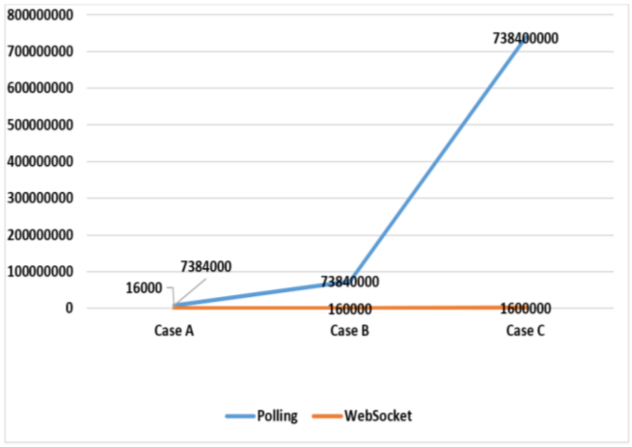


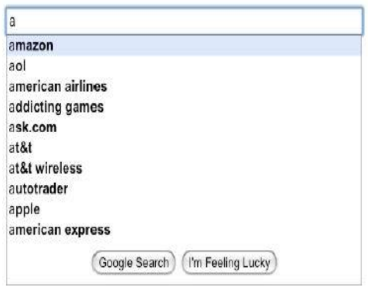
Figure 9: Comparison of Polling with WebSocket with respect to above example

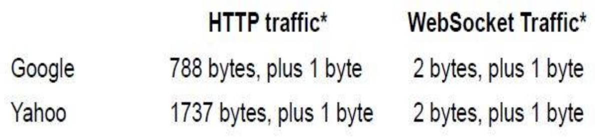
3.6 Advantages of Websocket Over the Previous Methods Used in Real Time Application

1. WebSocket naturally provides a full-duplex bi-directional communication channel which is done with a single socket, which means that an HTTP request uses the same connection from the client to the server and vice versa thereby reducing overhead. Figure 10 shows that in the search bandwidth overhead was reduce up to 1000 times as against the HTTP methods.
2. With WebSocket, latency is reduced. For instance, unlike polling which awaits request from the server and sends messages to the server constantly

without regarding if there are new messages available to be sent. But with WebSocket, Messages are also sent and received by the server and client at any time thereby reducing latency

3. Real time communication with WebSocket becomes proficient as resources such as the CPU and bandwidth are well utilized which in return improves performance unlike server-sent event which consumes more CPU Power and Bandwidth.
4. With WebSocket protocol, other standard protocol can be built.
5. Both Polling and Comet can provide robust end user experience and low perceived latency for a desktop application but WebSocket provides efficient web live streaming applications with insignificant latency.
6. Polling makes needless requests to the server thereby opening several connections for a low-message situation.
7. WebSocket lessens complexity when developing a real time application by eliminating overhead unlike polling and HTTP long polling which HTTP headers consume excess data which is in most case not useful for real time application.





	HTTP traffic*	WebSocket Traffic*
Google	788 bytes, plus 1 byte	2 bytes, plus 1 byte
Yahoo	1737 bytes, plus 1 byte	2 bytes, plus 1 byte

Figure 10: Header information for each character entered into search bar

4. FUTURE TREND

4.1. Embedded Chat

Chat applications in the future are expected to be embedded in other software, products and services which will make real time experience more alive to use.

4.2. 'Omnichannel' And In-Store

Real time Updates can be used in future by in-store to provide preview and recommendations of available products by using real time features like real time communication, updates, notifications combined with scan technology on the customer preferences.

4.3. Crowdsourcing

Data can be crowd sourced and passed to applications which publish data in real time to give valuable services to users and to the community. This can help in avoiding unforeseen circumstances like avoiding traffic congestion

4.4. Smart Automation & Internet of Things (IOT)

With the growth in internet of things, Web socket can be applied in the future to IOT and smart automation

4.5. Out of Home Media

Outdoor media is becoming more digitalized. When connected to the internet, it can be applied to technology like facial recognition and take in live data fields to alter what is been displayed in real time. An example is watching a video which show creative campaign by agency for charity Women's Aid

4.6. Device-to-Device / Machine-to-Machine

The usual chat human is human interacting in real time and other example involves human interaction with machine. In future time machine to machine interaction and device to device interaction will be realized with the web socket.

4.7. Marketing Automation

Marketing automation is a reigning trend in digital marketing and increasingly this involves mobile experiences that are prompt based on customer location. These need to happen in Realtime to be effective.

4.8. Personal Assistants

In future trend, WebSocket can be applied to personal assistance which will takes these Realtime interactions to a whole new level which will engage personal assistance to be more efficient.

4.9. Software

Progressively software runs in the cloud and is always automatically updated example is Google Chrome the browser. But the way software works is also changing to real time.

5. CONCLUSION

Even though HTTP has a higher percentage in being used for real time applications, it still cannot achieve full duplex communication which WebSocket offers because it causes heavy load on the server and increased network overhead. Thus, this article provided a detail description of various methods that are used for real time data communication, 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. WebSocket protocol offers the tremendous reduction in network overhead, Efficient low-latency and high-throughput transport and latency when real time data communication is concerned. If you want low-latency, high-throughput messaging back to the server, WebSocket can do it, Super easy API and Now supported in all modern browsers.

6. REFERENCES

[1] Kristian, J. (2014). "Real Time Web Applications, Comparing frameworks and transport mechanisms". *UIO Department of Informatics,* vol. 1, Number 1 pp. 15.

[2] Xue, L. and Liu, Z. (2014). "Network Real-time Communication Based on WebSocket". *Computer & Digital Engineering*, Volume 1 Number 1. pp 5.

[3] Pimentel, V. and Nickerson, G. (2012). "Communicating and displaying real-time data with WebSocket", *IEEE Internet Computing*, Vol. 16 Number 4, pp. 6.

[4] Anton, D., Yang, Y. A., Bajcsy, R. And Kurillo, G (2017). Platform for Augmented Telemedicine for Real Time Remote Medical Consultation. In *Proceedings of the International Conference on Multimedia Modeling,* Reykjavik, Iceland. pp. 77–89.

[5] Fu-Hau, H., Chia-Hao, L, Cheng-Yu, T., Wei-Tai, C., Yan-Ling, H. and KaiWei, C. (2016). TRAP: A Three-Way Handshake Server for TCP Connection Establishment. *Applied Science Journal*. pp. 3-4.

[6] Lerner, R. (2011), "At the forge: communication in HTML5", *Linux Journal*, Vol.11, Number 7, pp. 2.

[7] Pereira, R. and Pereira, E. G. (2014). Dynamic adaptive streaming over http and progressive download: Comparative considerations. In *Proceedings of the IEEE International Conference on Advanced Information Networking and Applications Workshops,* Victoria, BC, Canada, pp. 95–99.

[8] https://www.javaworld.com/article/2071232/java-app-dev/9-killer-uses-for-websockets.html

[9] I. Fette and A. Melnikov*. The WebSocket Protocol.* RFC 6455.RFC Editor, Dec. 2011. url:http://www.rfc-editor.org/rfc/rfc6455.txt.

[10] Puranik, G., Feiock, C and H. Hill. (2013). "Real-time Monitoring using AJAX and WebSocket". *20th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS).* Scottsdale, AZ, USA: IEEE, pp. 7.

[11] Bozdag, E. (2011). "Push solutions for AJAX Software". *Engineering Research Group Department of Software Technology Faculty EEMCS*, Delft University of Technology, pp. 22.

[12] Rakhunde, S. (2014). "Using WebSocket for Real Time Data Communication in Full Duplex Network". *IOSR Journal of Computer Science Conference (IOSR-JCE) e-ISSN: 2278-0661*, pp. 17-18.

[13] Pan, R., Zhao, H., Wang, J., Liu, D. and P. Cai. (2010). "The Design and Implement of TCP/IP Protocol Cluster on AVR Single chip". School of Information Science and Engineering, Northeastern University, Shenyang, China, pp. 763-764.