Adaptive Polling for Responsive Web Applications

Hatem Aziz, Mick Ridley

Computer Science Department, University of Bradford
Bradford, BD7 1DP, UK

H.M.Aziz@student.bradford.ac.uk,
M.J.Ridley@Bradford.ac.uk

Abstract. The web environment has been developing remarkably, and much work has been done towards improving web based notification systems, where servers act smartly by notifying and feeding clients with subscribed data. In this paper we have reviewed some of the problems with current solutions to real-time updates of multi user web applications; we introduce a new concept “adaptive polling” based on one AJAX technique “Polling” to reduce the high volume of redundant server connections with reasonable latency, we demonstrated a prototype implementation of the new concept which is then evaluated against the existing one; the positive results clearly indicated more efficiency in terms of client-server bandwidth.

Keywords: Real-time updates, AJAX, Bandwidth, Web applications.

1 Introduction

The internet has become the nerve of modern life in many different aspects, especially after the high spread of using small computers, tablets and smart phones. This was the result of the continuing development in Web environment to produce Web technologies that aim to enhance the interactivity and efficiency of Web applications, thus popular web applications and social networks such as Facebook and Twitter have become part of hundreds of millions of users’ life; these applications are based on real time web techniques that can fulfil the need for fast response to specific updates with wise data loading and server contacts, so users can communicate interactively, and be notified with new data matching their interests. AJAX, which can be easily implemented using a variety of tools and languages [3] is one of these techniques that is widely used nowadays benefiting from its ability to work as a real time web technology along with its variant Reverse AJAX [4] it can provide the functionality for what is called RIA: Rich Internet Applications; web applications that have the appearance and functionality of desktop ones.[5]

AJAX in terms of providing real time communications between the client and server has several techniques designed to fulfil that concept such as Polling, Piggyback and Comet; each of them has some advantages and drawbacks. The focus here will be on improving the Polling technique with more consideration to ‘less data transfers’ from the perspective of efficiency. We are motivated by finding a good alternative solution to suit different situations where there are multiuser updates and especially when updates occur at no regular intervals of time as exemplified by e.g. auctions, hotel booking web sites, etc. Although another technology nowadays has gained a considerable attention in terms of real time web communication, which is known as Web Sockets; a bidirectional communication technology introduced in HTML5, where both the client and the server can initiate a data transfer at any time[6], but one of its drawbacks that it can be blocked in some browsers for security issues[7].
In this paper we introduce the main AJAX techniques in Section 2. Outline the principles and prototype implementation of our new approach in Section 3 and report comparative results in Section 4 and Conclusions and future directions in Section 5.

2 AJAX Framework

2.1 AJAX

The AJAX approach is about transferring data between the server and client in the background. That means; the browser does not need to wait until a complete new page is received. The page can be refreshed on the basic of a small amount of data that can be shown within a DOM element. This is the main concept of AJAX, and the web applications that employ this approach are described as 'Asynchronous'[3]. Fig.1 shows how AJAX works.

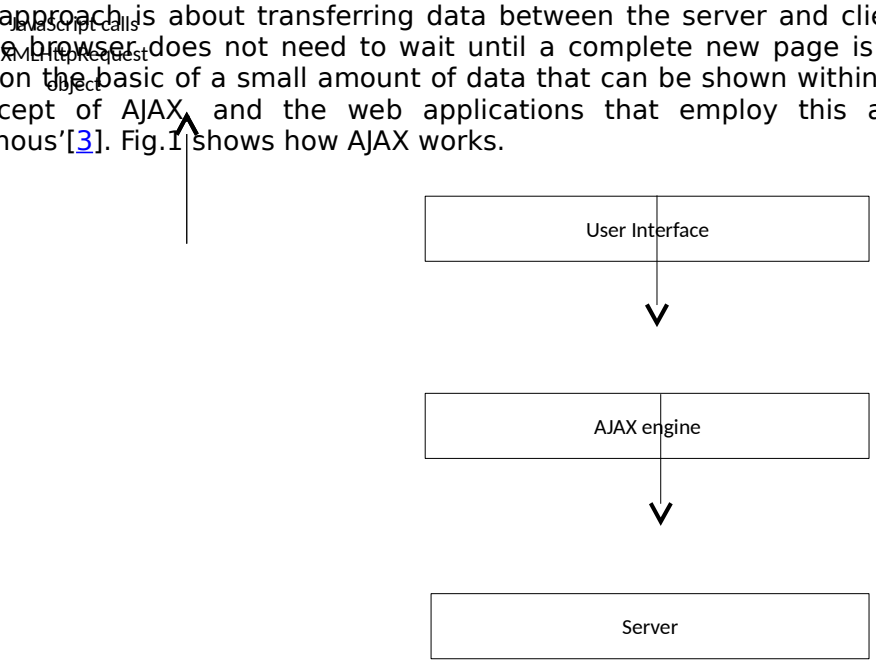

Fig.1. AJAX operation [2]

2.2 AJAX Tools

AJAX combines a number of technologies that work together in an integrated manner to produce asynchronous and interactive web applications. The main primary technologies are listed below [8]

- Standards-based presentation using XHTML and CSS.
- Data interchange and manipulation using JSON or XML and XSLT.
- Dynamic display and interaction using the Document Object Model; the DOM is a platform and language independent interface designed to provide high interactivity with an HTML document, by allowing access and update to the content of that document dynamically. The DOM works effectively with JavaScript to produce interactive web applications [9]. In addition, DOM can be subject to differences between browsers, therefore effective AJAX needs to deal with these differences when using a DOM model in order to obtain the same results on different browsers.
- Asynchronous server communication using XMLHttpRequest (XHR); the XHR object [10] is considered as the core of AJAX. It sends data to a web server in HTTP/HTTPS format and receives the server response as XML, JSON, HTML or plain text. The main feature of XHR is invoking the server with a request, and receiving the response in the background of a web page without the need to reload the page; this process is achieved by initializing a HTTP request to the server via send() method , and obtaining the returned information (if exists) via a response method such as responsXML() [10].
- JavaScript to bind everything together.

2.3 Reverse AJAX

The main target of Reverse AJAX is how to make the server act with more responsiveness, by pushing information to the client once it is available, without the client actually making a request to the server[4]. By default, AJAX requests can only be initiated

from the client to the server[1]. This limitation or problem can be resolved by using reverse AJAX techniques to provide responsive communication between the server and client. These techniques mainly known as Polling, Piggyback and Comet, with their different implementation, raise the issue of simplicity and efficiency; Comet needs specific features in the server, and is complicated in terms of implementation, but features having very low latency: there is no need to wait for the next time the browser connects, Polling in contrast is simple to implement, and should ensure low latency, however it can easily overload a server[11], whereas Piggyback can have very high latency but low overhead connections to the server[1]. Therefore we focused on the possibility of improving one of these techniques to produce a good alternative that can suit some cases with fair latency and reasonable resource consumption in terms of server connections, and we found out that Polling has the potential to be enhanced by considering the frequency of updates on the server.

How Polling works?

In this technique the browser can be set to contact the server at regular and frequent intervals (for example: every second) to check whether an update is available to be sent back to the browser[1]. In this case the client's web page is designed to either show updates about a particular subject e.g. Stock Market of some country, or based on the user's interest of what information should be shown; for instance following someone on Twitter. The later one can be achieved by implementing the Publisher-Subscriber design pattern [12]; where users determine which updates to be notified about at a central server or those made by other users.
Polling can be implemented using the JavaScript function setInterval() [13] to set a fixed time as frequency when contacting the server checking for updates. The advantage of this technique is that it is easy to implement, and does not require any special features on the server side; it also works in all browsers. Its disadvantage however, is the overhead of bandwidth, and redundant connections, therefore it's rarely employed. Fig.2 shows the client-server Polling communication.

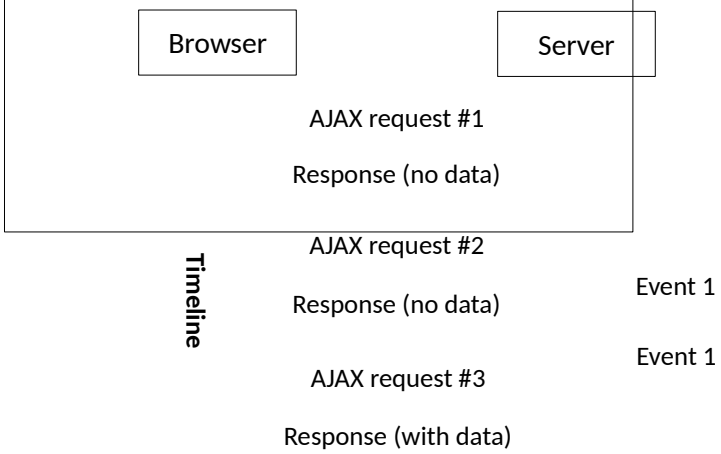

Fig.2. Polling communication [1]

3 Adaptive Polling

The principle of adaptive Polling is controlling the interval time of contacting the server to be flexible and changeable on demand; depending on how frequent updates there are, which means: the time can be increased or decreased due to server response. As mentioned before; Polling has a significant drawback which is the high number of requests (at fixed frequent times) to the server with the potential of returning no data at most periods of connection time. On the other hand, increasing the Polling interval will cause a delay in retrieving updates from the server. We therefore worked on producing a good alternative to Polling by enhancing its performance, and making it more efficient and

intelligent in terms of reducing the number of redundant requests to the server. This improvement is achieved by controlling the interval time of client-server contacts; depending on the frequency of real update at the server.

3.1 Adaptive Polling Prototype

The mechanism of Polling technique of AJAX is based on fixed interval time to send requests to the server and receive replies, either with update or no data returned. Whereas, in Adaptive Polling, the interval time will not be fixed; but adaptive to the server response, this may include the consideration of time to a deadline as for an auction or plane ticket pricing; that means the interval time will be reduced when approaching the time deadline. In this prototype, when the request returns no data from the server, the interval time will be increased until it reaches a maximum, which can be user application set, and if the server responds with an update, the interval time will be decreased until it reaches a minimum which again can be set. In this case, the efficiency of communicating with the server can be improved. Fig.3 illustrates the mechanism of Adaptive Polling (the new technique).

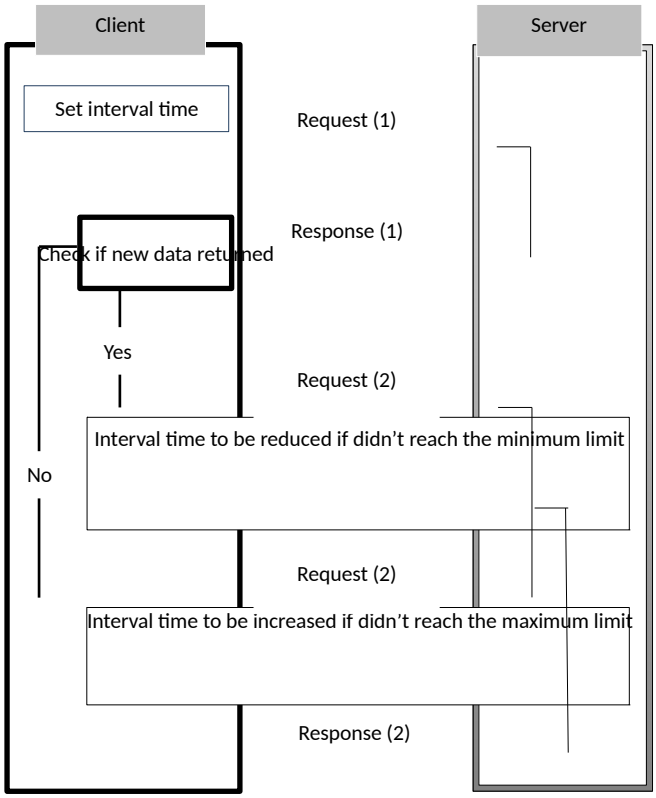

Fig.3. Adaptive Polling prototype

The Listing.1 shows a simple algorithm to clarify the main processes of Adaptive Polling, which then implemented by JavaScript and PHP.

```
Set the maximum interval  Max_int
Set the minimum interval  Min_int
Set the interval time  Interval
Repeat
{
     Delay(Interval)

     Results = Call the function Check_for_update( )
     If Results = true then
```

```
            Call the function ChangeInterval(true)
        Else
            Call the function ChangeInterval(false)
    } while(true)
```

Listing.1. Adaptive Polling Algorithm

3.2 Implementing Adaptive Polling

This implementation is based on the prototype demonstrated in Fig.3; it aims to build a simple multi user web interface representing the work mechanism of the improved Polling. Fig.4 displays a simple chat webpage, where any member can add comments to the chat room; also it shows the current interval time in milliseconds, and the number of times contacting the server for potential updates. These requests are made in different frequencies; depending on the user action, that can be made by adding comments to the chat room. Therefore, when any participant makes changes to the database, the Polling interval time will be decreased, but each time the server responds with no new data, the interval time will be increased until it reaches the maximum limit of interval time.

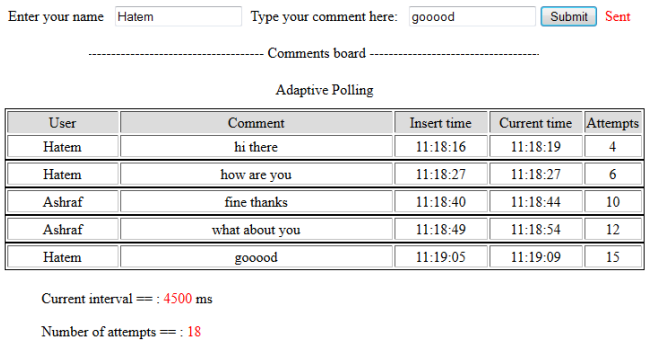

Fig.4. Adaptive Polling implementation

We used JavaScript and XML for best support, libraries and examples, but solutions could be implemented other ways. We are looking to identify generic architectural models for web applications that feature multiuser change/update. Listing.2 shows the main JavaScript function responsible for calling the PHP script at the server to check for new updates, and Listing.3 shows the interval change Function, whose job is to increase or decrease the interval time in response to the server update.

```
function loadXMLDoc()
{
    var xmlhttp;
    if (window.XMLHttpRequest)
      xmlhttp=new XMLHttpRequest();
    else
      xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

    xmlhttp.onreadystatechange=function()
    {
     if (xmlhttp.readyState==4 && xmlhttp.status==200)
     {
       counter1 += 1;
       document.getElementById("myDiv2").innerHTML= counter1;
        if(xmlhttp.responseText != "")
          if(counter1 < 2)
            resultText = xmlhttp.responseText;
          else
            {
             resultText += xmlhttp.responseText;
```

```
            document.getElementById("myDiv").innerHTML=
            resultText;
           changeInterval(false);
          }
       else
           changeInterval(true);
     }
    }
    document.getElementById("myDiv3").innerHTML=intervalTime;
    xmlhttp.open("GET","myscript.php?ct="+counter1 ,true);
    xmlhttp.send();
}
```

Listing.2. Check-for-updates function

```
function changeInterval(flag)
{
 if (!flag)
 {
 if (intervalTime > minInterval)
  {
   intervalTime -= intervalValue;
   clearInterval(si);
   si = setInterval(function(){ loadXMLDoc();},intervalTime);
  }
 }
 else
 {
  if (intervalTime < maxInterval)
  {
   intervalTime += intervalValue;
   clearInterval(si);
   si = setInterval(function(){ loadXMLDoc();},intervalTime);
  }
 }
}
```

Listing.3. Change-interval function

4 Adaptive Polling Vs Polling

We implemented a simple simulation webpage to compare the performance of the two techniques in order to illustrate the improvements in Adaptive Polling over Polling. This simulation allows multiple users to participate in a chat room and see their comments in both forms; Adaptive Polling and normal Polling, where they can notice the difference between them regarding the frequent server connection and the update delay. A simple test was carried out by a group of research students, and the results are shown in Fig.5 and Fig.6.

Adaptive Polling

User	Comment	Insert time	Current time	Attempts
Hatem	Hi there	17:08:36	17:08:40	3
Hatem	Hellooo	17:08:48	17:08:50	6
Abdo	Hi	17:09:02	17:09:02	9
Abdo	how are you	17:09:08	17:09:11	11
Hatem	goood	17:09:24	17:09:25	14
Hatem	where is the others	17:09:33	17:09:34	16
Ashraf	here man	17:09:47	17:09:50	19
Ashraf	how is it going	17:10:00	17:10:06	22
Hatem	good	17:10:16	17:10:18	24
Abdo	gooood	17:10:28	17:10:30	26
Hatem	where is Ibrahim and Muktar	17:10:52	17:10:55	30
Ibrahim	Yes I m here man	17:11:16	17:11:20	34
Ibrahim	how are you all	17:11:27	17:11:33	36
Hatem	oh nice to see u here Ibrahim	17:11:47	17:11:48	38
Ashraf	How r u Ibrahim where are you	17:12:00	17:12:01	40
Ibrahim	buzy a little bit	17:12:15	17:12:15	42
Ibrahim	what about you Ashraf	17:12:29	17:12:35	45
Ashraf	I m fine coping	17:13:06	17:13:12	51
Muktar	sorry guys was stuck	17:13:43	17:13:44	56
Muktar	Hatem is that ok	17:14:01	17:14:04	59
Hatem	yes that s ok Muk. Don t worry	17:14:17	17:14:17	61
Hatem	Thank you all guys	17:14:33	17:14:37	64
Hatem	for responding	17:14:44	17:14:50	66
Hatem	and see you later	17:14:57	17:15:03	68
Muktar	No prob man	17:15:16	17:15:18	70
Muktar	that s fine	17:15:30	17:15:31	72
Abdo	it s alright man	17:15:56	17:15:57	76
Ashraf	see ya	17:16:21	17:16:23	80
Ibrahim	see you later	17:16:34	17:16:36	82

Current interval: == **2000** ms
Number of attempts: == **82**

Fig.5. Adaptive Polling simulation

Polling

User	Comment	Insert time	Current time	Attempts
Hatem	Hi there	17:08:36	17:08:38	5
Hatem	Hellooo	17:08:48	17:08:48	15
Abdo	Hi	17:09:02	17:09:02	29
Abdo	how are you	17:09:08	17:09:09	36
Hatem	goood	17:09:24	17:09:25	52
Hatem	where is the others	17:09:33	17:09:33	60
Ashraf	here man	17:09:47	17:09:48	75
Ashraf	how is it going	17:10:00	17:10:01	88
Hatem	good	17:10:16	17:10:16	103
Abdo	gooood	17:10:28	17:10:29	116
Hatem	where is Ibrahim and Muktar	17:10:52	17:10:53	140
Ibrahim	Yes I m here man	17:11:16	17:11:16	163
Ibrahim	how are you all	17:11:27	17:11:28	175
Hatem	oh nice to see u here Ibrahim	17:11:47	17:11:47	194
Ashraf	How r u Ibrahim where are you	17:12:00	17:12:01	208
Ibrahim	buzy a little bit	17:12:15	17:12:15	222
Ibrahim	what about you Ashraf	17:12:29	17:12:29	236
Ashraf	I m fine coping	17:13:06	17:13:07	274
Muktar	sorry guys was stuck	17:13:43	17:13:43	310
Muktar	Hatem is that ok	17:14:01	17:14:01	328
Hatem	yes that s ok Muk. Don t worry	17:14:17	17:14:18	345
Hatem	Thank you all guys	17:14:33	17:14:33	360
Hatem	for responding	17:14:44	17:14:44	371
Hatem	and see you later	17:14:57	17:14:58	385
Muktar	No prob man	17:15:16	17:15:17	404
Muktar	that s fine	17:15:30	17:15:30	417
Abdo	it s alright man	17:15:56	17:15:57	444
Ashraf	see ya	17:16:21	17:16:21	468
Ibrahim	see you later	17:16:34	17:16:34	481

Current interval: == **1000** ms
Number of attempts: == **481**

Fig.6. Polling simulation

According to the results of the test; we found out that Adaptive Polling only needed 1/6 of the contacts compared to Polling in contacting the server, where Adaptive Polling sent about 78 requests searching for updates, Polling needed about 475 contacts to the server. Therefore there was a significant amount of redundant connections made by Polling comparing to Adaptive Polling which is more efficient from this point of view.

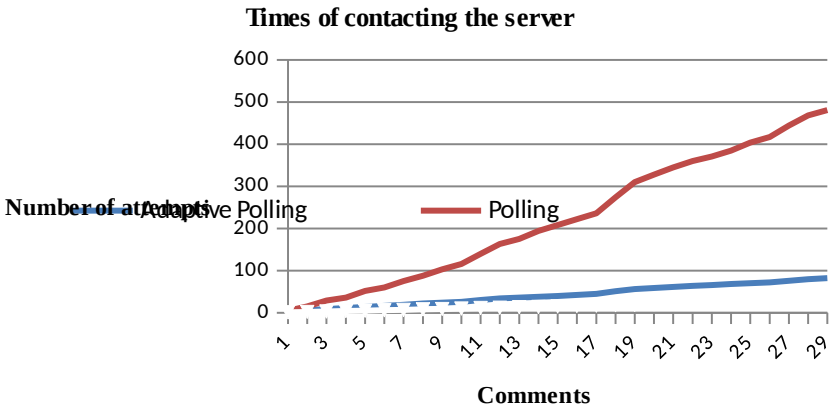

Fig.7. Server communication frequency

Fig.7 illustrates the frequency of communication between both techniques. In terms of latency; Adaptive Polling has more delay than normal Polling as shown in Fig.8. However, in many times they share the same response time for update, which is a positive point when considering the low number of times connecting the server.

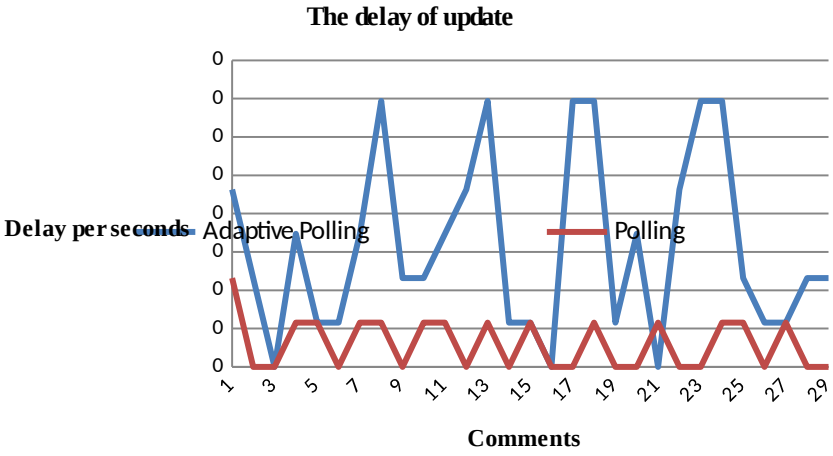

Fig.8. Time delay of retrieving updates

5 Conclusion

The most popular web applications in the last decade are real time based, that have the ability to notify users with any updates (often from database) in their areas of interest, therefore a variety of tools and technologies have been introduced to serve such features; AJAX is one of the widely used technologies in many popular websites, and Facebook is one example. This loose name 'AJAX' combines different languages and tools that can work together according to the concept of AJAX, but in the same time, not all AJAX techniques are suitable for all cases; it varies depending on the ease of implementation, efficiency and accuracy. So that has encouraged us to combine the simplicity and efficiency in one technique as an alternative solution to the others.

Therefore, our focus is on manipulating and extending one AJAX implemented technique, Polling, to eliminate the high bandwidth and loss of resources, and the idea was to deal with dynamic interval time based on server response, producing a modified technique that we call 'Adaptive Polling' and the results were positive after comparing the two techniques, as the

redundant connections remarkably reduced with reasonable latency; which suggests it is a good alternative that can be implanted in some cases especially when updates occur in no regular interval manner.

In addition to the positive results of our new concept; further improvements can be achieved by considering different cases when the time interval can be changed in response to a deadline event such as the closure of an auction. Also the real time updates may be led by the user activity at the client side of a website e.g. how fast the user enters data through to no activity or the browser being out of focus.

References

[1] M. Carbou. Reverse Ajax. IBM. Available: http://www.ibm.com/developerworks/library/wa-reverseajax1/

[2] X. Wang, "AJAX technology applications in the network test system," in *Electrical and Control Engineering (ICECE), 2011 International Conference on*, 2011, pp. 1954-1956.

[3] S. Holzner, *Ajax bible*. Indianapolis, Ind.: Wiley Pub., 2007.

[4] D. Crane, P. McCarthy, and S. Tiwari, *Comet and Reverse Ajax the next-generation Ajax 2.0.* Berkeley, CA. New York, N.Y.: Apress . 2008.

[5] R. Kay. Rich Internet Applications. Computerworld. Available: http://www.computerworld.com/article/2551058/networking/rich-internet-applications.html

[6] Z. Kessin, *Programming HTML5 applications.* Beijing ; Farnham: O'Reilly, 2012.

[7] Websockets. F-List Wiki. Available: https://wiki.f-list.net/Websockets

[8] R. Larsen. An introduction to Ajax. IBM. Available: http://www.ibm.com/developerworks/library/wa-aj-ajaxhistory/

[9] P. L. Hégaret. Document Object Model. W3C. Available: http://www.w3.org/DOM/

[10] XMLHttpRequest. WHATWQ. Available: https://xhr.spec.whatwg.org/

[11] K. Zyp. Easing into Comet. cometdaily.Available: http://cometdaily.com/2007/12/05/easing-into-comet/

[12] E. Gamma, *Design patterns : elements of reusable object-oriented software.* Reading, Mass. ; Wokingham: Addison-Wesley, 1995.

[13] J. Pollock, *JavaScript : a beginner's guide*, 3rd ed. New York: McGraw Hill, 2010.