HTML5 Features Integration in Green Ajax to Implement More Efficient Push Technology

Ridwan Sanjaya
Department of Information Systems
Faculty of Computer Science
Soegijapranata Catholic University
Semarang, Indonesia
ridwan@unika.ac.id

Abstract - **Data communication in the web was changed since Ajax was used to send and request the data without bothering the web interface. However, the most common implementation in using Ajax is to request data from the server only. Data delivery initiatives from the server to the client can not be done in the HTTP architecture. Several approaches were invented to overcome the limitations in allowing the web servers to send data to the browser without any client initiative. Green Ajax is one of the approaches that has dependency with additional non-standard software on the web browser to receive the notification from the server.**

The new HTML standard called HTML5 has Server-Sent Events feature that is offering the technology to push a message from server to the client and natively supported by most of the web browsers. However, it should not work solely to get the objectives in obtaining the complete data from the server using minimum bandwidth. The integration HTML5 features in Green Ajax could be the best implementation of Push Technology. It will be able to solve the Green Ajax limitation in web browser's plug-in dependency but reduce the large bandwidth consumption caused by Server-Sent Events feature.

This paper will describe the technique of using the Server-Sent Events feature in Green Ajax to find better approach on two-way data communication in HTTP architecture using minimum bandwidth. The result could be a model for web programmers to display the real time data from the server. The Flight Information Display Screen (FIDS) will be used as the case in the implementation.

Keywords - **Ajax, HTML5, Push, Technology, Server-Sent Events, Two Way**

I. INTRODUCTION

The concept of Green Ajax to minimize the bandwidth in providing the real time data which is implemented by sending a notification from the server to the browser which is already equipped with a server plug-in [1]. The notification from server is used to trigger the web browser in requesting a new data to the server. This concept is able to overcome the conventional Ajax which is repeatedly request to the server to get the new data [2]. The limitations of conventional Ajax are the large bandwidth consumption in cumulative because of repeated requests; the incomplete data collection due the differences between request time in the client and updating time in the server; and the redundant data collection when the server do not update in long period [3].

However, there are several requirements in the Green Ajax implementation. First, the server is able to identify the clients who will receive the notification. Second, the web browser should be able to receive a notification. Third, the web browser could do

a request to the server after the notification was received. Those requirements that make this approach will get high Successful Receptive Percentage (SRP), and low Data Loss Percentage (DLP) [4].

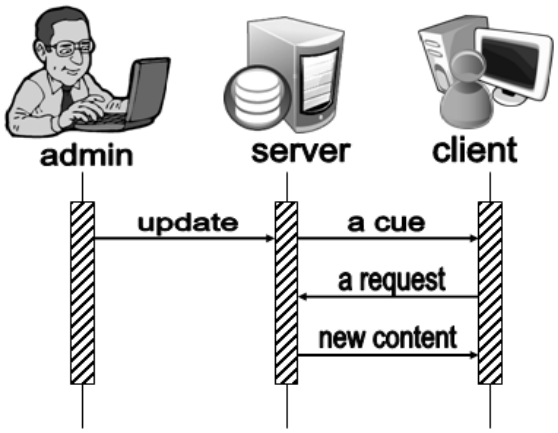

Figure1. A notification was sent when the administrator update the data.

In the preliminary experiments, the web browser should be equipped with a server plug-in which is non-standard software on the common browser. The plug-in will make the browser able in receiving the notification from the server. Then, the server-side code should identify the computers connected to the server. It should be done to direct the server notifies the client when there is a new update. However, modifications on the proxy or router settings must be done to eliminate the problem in the communication between the clients behind a proxy or router.

The use of non-standard software on the common browsers should be avoided to make the Green Ajax approach become widely used by programmers. The new HTML standard called HTML 5 brings the new technology that might be needed by Green Ajax. The new technology in a feature called Server-Sent Events (SSE) makes the clients be able to receive the data from the server without the client's initiatives [5]. The programmers should not create any server-side code to identify the clients who need the notification because the mechanism will be done by the protocol of new HTML standard. Currently, HTML5 is widespreadly adopted in the most web browsers in the desktops, laptops, tablets, smartphones, and also televisions [6, 7].

The limitation of this feature is repeatedly sending data to the clients based on value of retry parameter. Its value can be defined in the server-side code to control the reconnection time from server to the client. It will have the same effect with the classical Ajax which will send the redundant data and/or the incomplete data collection to the client. The new technology should not be used alone to achieve the same goal of the Green Ajax approach. This paper will describe the application of Server-Sent Events in the Green Ajax to solve the Green Ajax's limitation on dependency of the server plug-in and inability to communicate with the computers behind proxy or router, but to reduce the large bandwidth consumption caused by Server-Sent Events feature.

II. HTML5 INTEGRATION IN GREEN AJAX

Pushing the data from server to the client in a given interval time is the behavior of Server-Sent Events in HTML5. This new feature in the new HTML standard brings benefit to the data communication in the web. In the previous HTML standard, the data will be sent to the client only if there is a request from the client to the server. On the other side, the server should record the client ID if the server has iniative to send the data to the client without any client's request. This communication will be possible when the client has a server plug-in inside the web browser. However, it would create a complexity when the number of clients is increasing and the clients are behind a proxy or router. This complexity can be solved by Server-Sent Events integration in Green Ajax.

However, the behavior of Server-Sent Events also creates a disadvantage in the bandwidth overhead. If the data size from the server is large and there is no new data in a certain time, the same data will be kept and

sent to the client. This disadvantage is also found in the previous research when classical Ajax is used in data communication [8]. To avoid the effect in bandwidth overhead, the same concept in Green Ajax could be used. The data pushed by the server should be in small size as a notification to trigger the client in making a request to server. The data could be a-single digit number from zero to nine or any unique number combination of hour, minute, and second of the updating time.

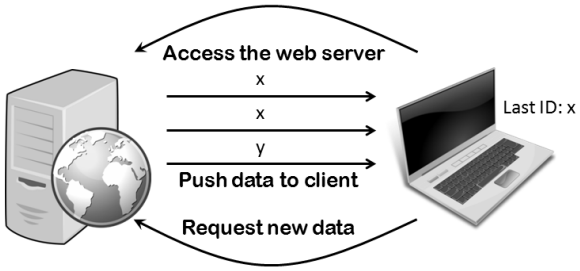

Figure2. Each ID of notifications were compared with the last ID in client.

The number is pushed by the server will be compared with the number saved in the web browser. If both numbers are different, the web browser will replace the number in the local storage with a new number from the server. It also triggers the client to request new data to the server by using Ajax. When the client receives the data from the server, it will be displayed in a web browser. The local storage feature could be found as a new feature in the HTML5 [8]. The local storage is used to substitute a cookie and able to be stored in a web browser without expiration date until the programmer removes it. However, this feature is only supported in Internet Explorer 8+, Firefox 3.5+, Opera 10.5+, Chrome 4+, and Safari 3.1+ [9].

The flowchart of the above concept that explains the flow of communication and works of the server-side and client side could be seen in the figure below.

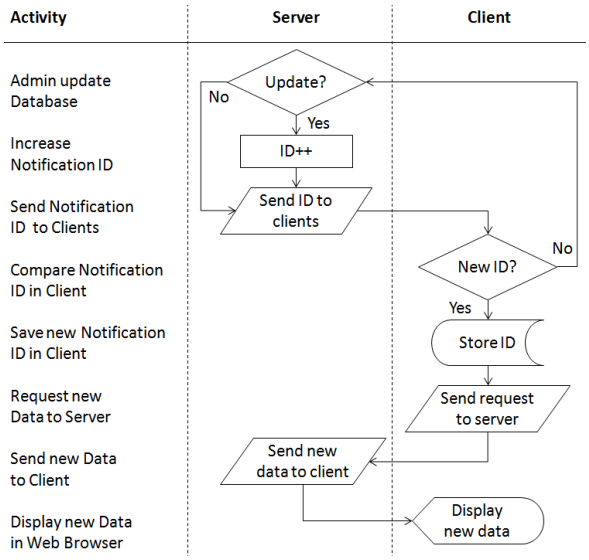

Figure3. Flowchart of communication using Green Ajax and Server-Sent Events.

III. IMPLEMENTATION

In the implementation, the existing technology could be used such as a trigger in the database. When the administrator changes the database, it will also change the notification ID in another table. The notification ID will be used in the server-side application connected to the client using the Server-Sent Events feature. If the administrator did not update the database, the notification ID will not changed.

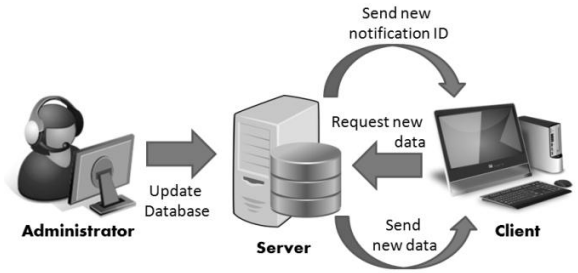

Figure4. The model of communication in the implementation.

The example code of trigger in the database in order to increase the notification ID when the database was updated could be seen below:

```
CREATE TRIGGER 'updateID' AFTER UPDATE
ON 'arrival'
 FOR EACH ROW BEGIN
DECLARE x INTEGER;
SET x=(SELECT id from id);
IF x=9 THEN
 update id set id=0;
ELSE
 update id set id=id+1;
END IF;
END
```

As a behavior of Server-Sent Events, the notification ID will be sent to the client in a given time from fids.php. The client is triggered to request to the server when the notification ID is different with the notification ID inside the local storage.

```
var source=new EventSource("fids.php");
source.onmessage=function(event) {
  if(EventID!=localID ||
document.getElementById("ID").innerHTML=="") {
    localStorage.fids = EventID;

document.getElementById("ID").innerHTML="Last
ID: "+EventID;
    fids(); //request using Ajax
  }
};
```

By using Ajax, the request will be sent in the process behind to fidsboard.php without bothering the user interface. After the client receives the data from the server, the web browser will show the data.

```
function fids() {

document.getElementById("result").innerHT
ML="";

  xmlHttp=GetXmlHttpObject()
  if (xmlHttp==null) {
    alert ("Your browser does not support AJAX!");
    return;
  }
  var url="fidsboard.php?sid="+Math.random();
  xmlHttp.onreadystatechange=stateChanged;
  xmlHttp.open("GET",url,true);
  xmlHttp.send(null);
}

function stateChanged() {
  if (xmlHttp.readyState==4) {
    var data = xmlHttp.responseText;
    var lines = data.split("<br/>"), output = [], i;
    for (i = 0; i < lines.length; i++)
    output.push("<tr><td>"
      + lines[i].split("|").join("</td><td>")
      + "</td></tr>");
    output =
"<table><tr><td>Flight</td><td>Origin</td><td>Ti
me</td><td>Remark</td></tr>"
      + output.join("")
      + "</table>";

document.getElementById("result").innerHTML=out
put;
  }
}
```

IV. RESULT

The figure below is the network activity diagram of the web applications accessed by the web browser. This diagram is currently provided by most of web browsers in their Inspect Element feature.

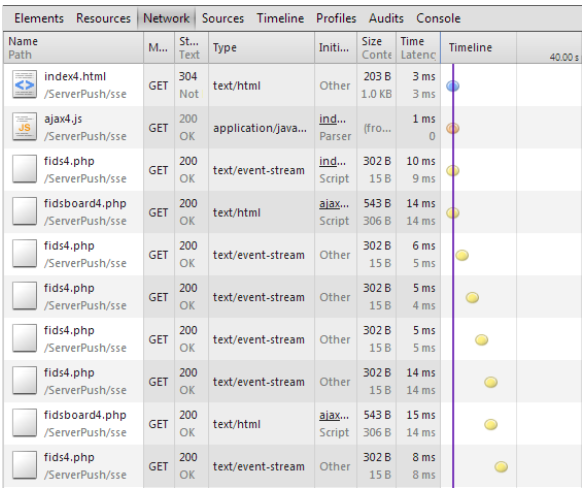

Figure5. The network activity diagram of the web application

Server-side Program fidsboard.php will be accessed when the client receives a new ID of notification. From the figure above, the size of information is 543 Bytes. Without new information, server is only transferring the same notification ID from fids.php with size 302 Bytes in the network. Green Ajax using existing Server-Sent Events will keep transferring 543 Bytes in every interval time. On the other hand, when Server-Sent Events is only sending the small number as

notification, the bandwidth consumption is decreasing. The table below is the calculation of the bandwidth saving using Green Ajax with modified Server-Sent Events when there are three updates within 1 minute.

Table1. The bandwidth usage by existing and modified SSE.

Green Ajax	Existing SSE	Modified SSE
3 secs	543 Bytes	543 Bytes
6 secs	543 Bytes	302 Bytes
9 secs	543 Bytes	302 Bytes
12 secs	543 Bytes	302 Bytes
15 secs	543 Bytes	302 Bytes
18 secs	543 Bytes	302 Bytes
21 secs	543 Bytes	302 Bytes
24 secs	543 Bytes	302 Bytes
27 secs	543 Bytes	302 Bytes
30 secs	543 Bytes	543 Bytes
33 secs	543 Bytes	302 Bytes
36 secs	543 Bytes	302 Bytes
39 secs	543 Bytes	302 Bytes
42 secs	543 Bytes	302 Bytes
45 secs	543 Bytes	302 Bytes
48 secs	543 Bytes	302 Bytes
51 secs	543 Bytes	302 Bytes
54 secs	543 Bytes	302 Bytes
57 secs	543 Bytes	302 Bytes
60 secs	543 Bytes	543 Bytes

The results show that Green Ajax with existing Server-Sent Events will consume 10,860 Bytes within 1 minute. In the other side, Green Ajax with modified Server-Sent Events will only consume 6,763 Bytes within 1 minute. The larger file size of new information transferred by the server will show the significant bandwidth saving in the network.

CONCLUSION

Server-Sent Events feature helps Green Ajax in sending a notification to the clients without having problem on the computers behind a router or proxy. The reason of using Server-Sent Events only to notify the clients is because the behavior of Server-Sent Events to reconnect the client using an interval time, not by triggering from any activity on the server. To accomplish the minimum bandwidth usage as the Green Ajax requirement, the server should only send a small data as a notification. In these experiments, a single digit number is used to be a notification. Even though the small number is used to be sent as a notification, it will keep increasing the bandwidth usage compared to the Green Ajax without Server-Sent Events.

When the client receives the new number from the server, the client will request data to the server. But, if the client receives the same number from the server, the client will not do any request to the server. In this case, the clients should have a mechanism to store the number which is sent by the server. HTML5 provides local storage feature to be used in storing the number. The number stored in the web browser will be compared with the notification sent by the server. If both of numbers are equal, the client still receives the same number from the server. When the clients receive the new number in the notification, the common technique in Ajax is used in the clients to send a request to the server. The reply will be received and equipped with the appropriate format in the client-side for a better view.

Even though the bandwidth efficiency is decreasing, HTML5 adoption in Green Ajax is increasing the possibility of Green Ajax approach to be widely implemented in all computers. The Server-Sent Events and local storage features in the HTML5 have the big role for reducing the limitation of Green Ajax in the implementation. This approach could be used by any web browser who supports HTML5 natively.

ACKNOWLEDGMENT

This study is supported by Beasiswa Unggulan scholarship from Bureau for Planning and International Cooperation, Ministry of Education and Culture of Indonesia under the Grant No. 96733/A2.4/LL/2012.

REFERENCES
(Arranged in the order of citation in the same fashion as the case of Footnotes.)

[1] R. Sanjaya. "Green Ajax Implementation on the Wireless Local Area Network using Randomized Cue Applications". International Journal of Information and Education Technology. Vol.1(1), pp. 63-67.

[2] R. Sanjaya. "Trade-off analysis for web application using Green Ajax". Proceeding of IEEE International Conference on Management of Innovation and Technology (ICMIT), Singapore, pp. 1050-1054.

[3] R. Sanjaya. "Mobile Traffic Evaluation for Fuzzy-Based Application Using Green Ajax". Proceeding of the 2011 IEEE International Conference on Information and Education Technology (ICIET), Guiyang, pp. 409-416.

[4] R. Sanjaya. "Web traffic reduction for infrequent update application using Green Ajax". Proceeding of the 2nd IEEE International Conference on Information Management and Engineering (ICIME), Chengdu, pp. 170-176.

[5] M. MacDonald. "HTML5: The Missing Manual". O'Reilly Media, Inc.

[6] K. Kapetanakis, S. Panagiotakis, AG. Malamos. "HTML5 and WebSockets; Challenges in Network 3D Collaboration". Proceedings of the 17th Panhellenic Conference on Informatics (PCI '13), New York, pp. 33-38.

[7] M. Olan. "HTML5 Jumpstart". Journal of Computing Sciences in Colleges, Vol. 28(6), pp. 65-66.

[8] W. West, SM. Pulimood. "Analysis of Privacy and Security in HTML5 Web Storage". Journal of Computing Sciences in Colleges, Vol. 27(3), pp. 80-87.

[9] NC. Zakas. "Professional JavaScript for Web Developers". John Wiley & Sons.