Traffic Analysis for Real-Time Web Applications using Green Ajax

Ridwan Sanjaya

A Doctoral Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Computer Information Systems
Graduate School of Information Technology
Assumption University
Academic Year 2011

Traffic Analysis for Real-Time Web Applications using Green Ajax

Ridwan Sanjaya

A Doctoral Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Computer Information Systems
Graduate School of Information Technology
Assumption University
Academic Year 2011

ABSTRACT

The classic web applications usually need a lot of bandwidth to provide the rich user interfaces. Since Ajax was introduced, it has reduced the web server load and the data transfer to/from users' computers (Sanjaya, 2007). By using Ajax, only a specific part in the web page can be requested to the web server (Lin, 2008). However, it still cannot provide the real time data updating. The common approach to provide the real time data updating uses a timer to request a new data from the web server periodically (Kletsch, 2008; Chen, 2008). But, the requests sometimes do not get any new data. If the interval time to renew the data is too long, the data updating will come to the client late and some data received by the client may be lost.

The proposed approach to solve the problem is creating an Ajax application which can receive a signal from the web server. The received signal will trigger the web application to renew the data from the web server. By limiting of requests to the web server only if a new data in the web server arises, as a consequence the traffic between the web server and the client can be reducible. The efficiency of web traffic will be measured by two matrices, the successful receptive percentage of the web application request to the web server and the bandwidth consumption of web application.

In this research, an innovation of Green Ajax will be proposed. The idea of low bandwidth and low resource consumption are introduced. Mozilla Firefox and Firebug will be employed as the tools to measure the experimental results. The experiments test not only infrequent update applications, but also frequent update application and fuzzy-based application. From the experiments, Green Ajax is expected to be a suitable approach for the interactive web based applications.

Keywords: ajax, bandwidth saving, frequent update, fuzzy-based, green ajax, infrequent update, local area network, real-time, data update, web application, web traffic

ACKNOWLEDGEMENTS

Praise to the Lord for His wisdom, grace, and blessings so that this dissertation can be completed properly. This dissertation can be completed properly because of encouragement and support of various parties. Therefore, I give appreciation and gratitude to:

1. My excellent advisor, Dr. Chanintorn Jittawiriyanukoon, for his time, patience, ideas, supports, guidance, great comments, and correction during the process of publications and my dissertation.
2. Lucia Sianny Octavia, my beloved wife who always gives her prayer, love, patience, and support to me.
3. Kezia Patricia Sanjaya, my lovely daughter who always smiles and keeps being cheerful in every moment. She makes me proud as her father and makes me keep my struggle on doing researches.
4. Ms. Henny Putri Saking Wijaya, my Proofreader for her time and help in grammar checking of the dissertation.
5. Mr. Sanga Rujiphongpai, for his time and cooperation in the process of my study in Graduate School of Information Technology.
6. Saarce Elsye Hatane, Jerry Rumkeny, and all members of Ramkhamhaeng-Bangna who always give supports and pray during the process of my dissertation.
7. Other parties who have assisted in the completion of this dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

1.1. Importance of the Study

Even though internet speed is increasing significantly in the present, the web traffic is busier. The classic web applications usually need a lot of bandwidth to provide the rich user interfaces. Reloading the big size of web page every time will cause the big consumption of bandwidth.

The new technique, named Ajax, can reduce the web server load and the data transfer between server and users' computer. Ajax is not new but it is a set of web technologies which can provide the interaction between the web server and client. It consists of HTML, JavaScript technology, DHTML, and DOM (Mesbah, 2007; Lin, 2008).

If the web application uses Ajax, only a specific part in the web page can be requested to the web server (Sanjaya, 2007). The new data will appear on the specific part in the web page. The users will not see the blank page when the new data is requested to the server.

However, it still cannot provide the real time data updating. The common approach to provide the real time data updating uses a regular user-definable intervals known as Time to Refresh (TTR) to request a new data from the web server periodically (Kletsch, 2008; Chen, 2008). But this approach does meaningless requests because the requests sometimes do not get any data updating. In contrast, changing the longer interval time to update the data is not efficient because the client will not receive the data on time.

In this research, an alternative way of classical Ajax, called Green Ajax is proposed. The proposed Green Ajax will be able to receive a signal from the web server. This signal will command the client to request a new data from the web server. Requesting a new data of the web application will depend on the received signal. It will reduce the traffic between the

web server and the client. The efficiency of web traffic will be measured from the successful receptive percentage of the web application requests to the web server and the bandwidth consumption of web application.

A research by Merrill found the total bandwidth savings of 61% happened when using the Ajax approach. Another research by White showed 71% of performance improvement and 32% time savings occurred when using Ajax (Dahlan, 2008). From the experiment, by combining with Ajax, the expected results of this research compared to the previous researches are increasing 80% of the successful receptive percentage of the web application requests, reducing 100% of data loss, and reducing 80% of bandwidth consumption.

Mozilla Firefox and Firebug will be used as the tools to measure the results. The importance of study not only get a faster real time data updating approach, but also help to solve the problem of the limited bandwidth in the developing countries around the world. The experiments test not only infrequent update applications, but also frequent update application and fuzzy-based application. From the experiments, Green Ajax is expected to be a suitable approach for the interactive web based applications.

In this research, a Green Ajax will be used as a proposed term of the approach because it will bring the idea of low bandwidth and low resource consumption. The differences between the classic Ajax and the Green Ajax can be shown on table 1.1.

Table 1.1. Green Ajax Comparisons

	Ajax	**Green Ajax**
Update in the a specific part of page	Yes	Yes
Update by client request	Yes	Yes
Update by server initiative	No	Yes
Approach for infrequent update data	Interval timer	Server's signal
Real time data updating	Sometimes	Yes

1.2. General Background

Several decades ago, desktop-based programming was the most favorite programming-style in the world. The main characteristic of desktop-based application is installed in each user's computer. If there are 100 users, the application must be installed on 100 computers. When the internet was gaining popularity, the programming paradigm was shifted to web-based application (Gal, 2001; Redouane, 2002; Zapeda, 2007).

Some enterprises prefer to use web-based programming because the application and data are centralized. The users do not need to install the application on their computers. Once they have a web browser and able to connect to the server by using intranet or internet, they can use the application to work.

A centralized application on the server is considered as an advantage of web-based programming because the users can access the application from any computer, without depending on the location or the workplace. For the security, the application can be equipped with authentication. The users must enter a username and password to use the application.

Not surprisingly, the term of Cloud Computing is becoming popular after the web-based development is gaining the maturity. This term is used to describe the absence of dependence on the device or specific locations in carrying out the work using a computer. Applications and databases are pooled, stored, and executed on the server computer (Zhang, 2001).

An online retail sale is one example of a web-based program. Because the application and data are managed centrally, the business owners can find out real-time stock at each store. It will help the manager to decide the stock moving from one store to other stores.

However, at that moment, the smooth interaction between users and computers only can be easily translated by using desktop-based programming languages. The old-style internet programming can not display the smooth interaction between users and computers better than desktop-based programming. The users will see the blank page every time the application is

doing a process. It causes the users feel uncomfortable every time they see the blank page.

Another disadvantage, the bandwidth consumption will increase because all of the web page should be loaded. Even though the application does not need to reload the whole page, the application must do it anyway. It will make the traffic on the network busier if the application has a lot of process in a page.

Since Ajax was introduced by Jesse James Garret (Zapeda, 2007), the web-based programming has never been the same again. The users will not see the blank page or page transitions even though the application does a process. The XMLHttpRequest in Ajax is the main technology that makes Ajax engine able to receive any data from the web server without altering the web page (Sanjaya, 2007). The process is done on the background without interrupting the user's interfaces.

However, based on the web architecture, the client has to request to the web server to get a response. The web server only can give a response but it cannot send the reply without the client's request. The common approach to get the latest data on the server is using an interval to run a request. Even though the server does not have any new data, the client still keeps on requesting based on the timer. It is wasting not only the time and bandwidth but also the resource on the computer to send the requests and receive the replies. However, if the interval is increased, some data received by the client will be lost. Even though the approach is not wasting the bandwidth, but the web application will not show the real time data.

Real time data updating is becoming a crucial topic in the web application development. It needs an alternative method to provide a capability of the client to receive a signal from the web server without the client's request. The approaches should combine the existing or new technologies in the web application development. The application should have a new approach to show a real time data which is not wasting the bandwidth and the resource.

This research will explore and look for the appropriate approaches or techniques to get

the real-time data on the server. In the reality, the data from the server can be produced in the schedule (fixed time) and also in an unpredicted updated time (random). The proposed approach should be able to work in those conditions. Green Ajax should work effectively on triggering the client to request to the latest data on the web server.

1.3. Research Objectives

The main objective of this research is to discover the solution for the web application to display interactive contents using small bandwidth.

1. To create a concept to reduce the web traffic of the web application.

   This research will form the concept to minimize the bandwidth and resources for the web application which depends on the web server's data updating. This concept has to replace the use of interval time to update the data from web server.

2. To design an approach to provide a real time data updating on the web application.

   Even though the time for updating data at the server side is random and unpredictable, the web application can detect the updating activity. By detecting the updating activity, the web application can show the latest data at the right time.

3. To develop a model to provide a capability of client on receiving a signal from the web server without the client's request.

   In this research, a model for the web server signal to the client without client's request will be created to be an alternative of the current web architecture.

4. To design the model of Green Ajax implementation for the web based application.

   All of the concept which is found in the research will be a model to implement Green Ajax in the web application. It models not only the architecture but also the form of supporting technologies in the Green Ajax.

5. To implement Green Ajax in the web programming.

   After the concept is modeled, there is a need to implement Green Ajax in the web applications. In this research, PHP will be used as a language scripting to make the web applications. The new programming approach in PHP that is object oriented programming will be used on the code.

1.4. Scope of Research

All of the existing technologies will be included to create the model of web application. Ajax will be used as the main technique in this research because it has been proven by some researches as a good approach to decrease the bandwidth. However, Ajax is not efficient when the web page needs the data which is updated in the random time.

The experiments focus not only on the reducing the web traffic of infrequent update web application, but also on web traffic of frequent update application and fuzzy-based application. The proposed approach will be compared to Ajax applications which have different interval time to update. The bandwidth consumption, accuracy of requests, and data loss of those approaches will be summarized as the efficiency of proposed approach.

To implement the approach in the current web architecture which does not support the signaling from the web server yet, the design of the application and the network implementation will be developed in the end of the research. Local area network will be used for the scope of research.

II. LITERATURE REVIEW

2.1. Internet

The Internet is a system of billions computers connected to the global computer networks. Its history began since 1969 when the U.S. Defense Advanced Research Projects Agency (DARPA) decided to find how to connect several computers. This research project was named as ARPANET. In 1970, more than 10 computers were connected to each other. They could communicate to each other and form as a network (Bhasin, 2002).

The network of the research project was grown to be a network of networks and served as a global data communication system that linked millions of private, public, academic and business networks. Because it was decentralized by design, nobody owned the Internet and no one led its authority. The decentralization had an intention to make the internet less vulnerable to wartime or terrorist attacks.

The World Wide Web (WWW) and Electronic Mail (E-mail) is the popular applications today. Other inventions, e.g. newsgroup, IRC, and FTP, were completing the facility inside the ARPANET. In 1982, the term "Internet" was used for the first time and TCP/IP was used to be its universal protocol.

2.2. World Wide Web (WWW)

In 1992, Tim Berners-Lee invented the World Wide Web (WWW). By that time, the audio and video were able to be presented over the Internet. The concept of World Wide Web was to use hypertext to organize a distributed document system. The hypertext refers to each document by using cross-references (also known as links). It makes one web page contain text, images, videos, multimedia, and hyperlink. The hyperlink will be used to navigate to other web pages (Kressin, 1997).

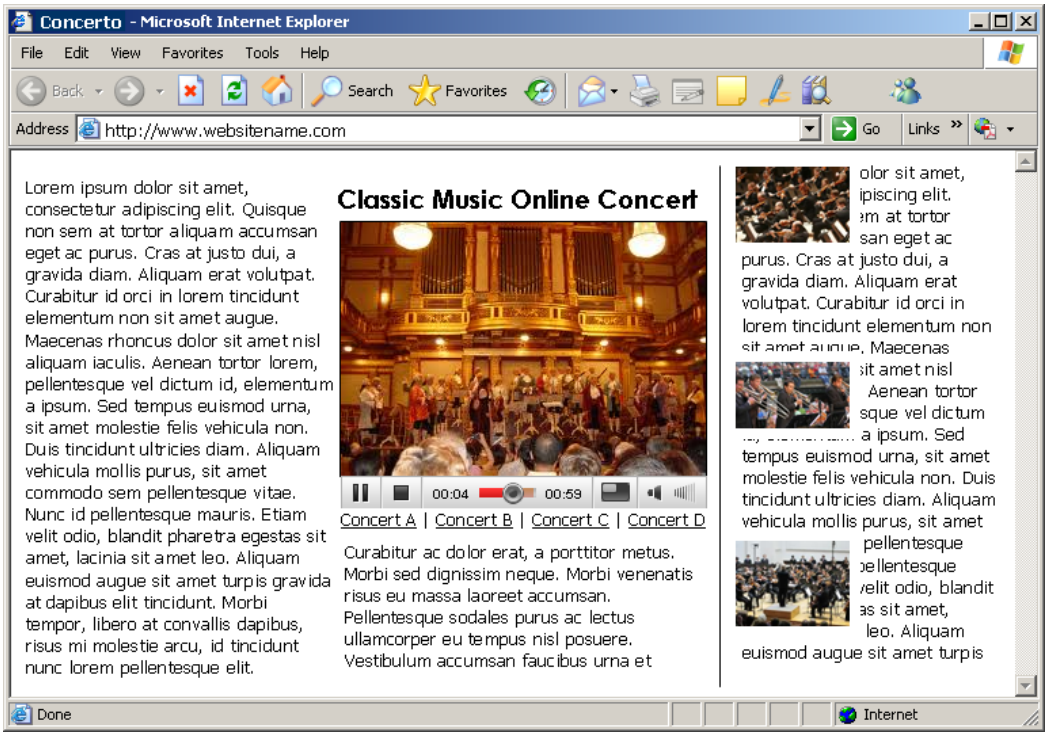


Figure 2.1. World Wide Web contains text, images, videos, multimedia, and hyperlink

Many people consider the Internet and Web the same thing, but in fact they are different. The Internet provides the basic structure, but the web uses the structure to provide content, documents, multimedia, and other documents. In short, the Internet is the road where the web is one of the vehicles that runs on it.

2.3. TCP/IP

Protocols are needed to rule the network communication because there are several different types of communication inside the network. Each protocol will do a part of job in the network by ruling them based on the types of communication. The Internet and other similar networks use TCP/IP as standard protocol to rule its communication. TCP/IP consists of two protocols (Chappell et al, 2004) which are the Transmission Control Protocol (TCP) and the Internet Protocol (IP).

TCP/IP provides end-to-end connectivity which allows the direct connection from hosts

to other hosts. The protocol also rules the format of data, addressing, transmitting, routing, and receiving at the destination. It has four abstraction layers which are the Link Layer, the Internet Layer, the Transport Layer, and the Application Layer. Each layer has its operational scope of the protocols and function to solve problems in its scope.

The Link Layer, the lowest layer on TCP/IP, manages the connection from host to host, without intermediate. It rules the connection between the hardware of computer and interface-to-interface messaging.

The Internet Layer rules the communication from links to other links of a computer and helps the networks interconnection. This layer is the basic of the Internet and Internet Protocol (IP) is its primary protocol. Internet Protocol has job to deliver datagram (packet) from the source to the destination host only based on their addresses. For this purpose, IP defines the basic addressing namespaces to identify and locate hosts on the network and structures for datagram encapsulation.

The Transport Layer manages the direct host-to-host communication and data transmission framework between hosts. The Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) are used in this layer.

The Application Layer, the highest layer on TCP/IP, handles interaction of the application on a process-to-process level between Internet hosts. This layer contains protocols to manage data communications services function.

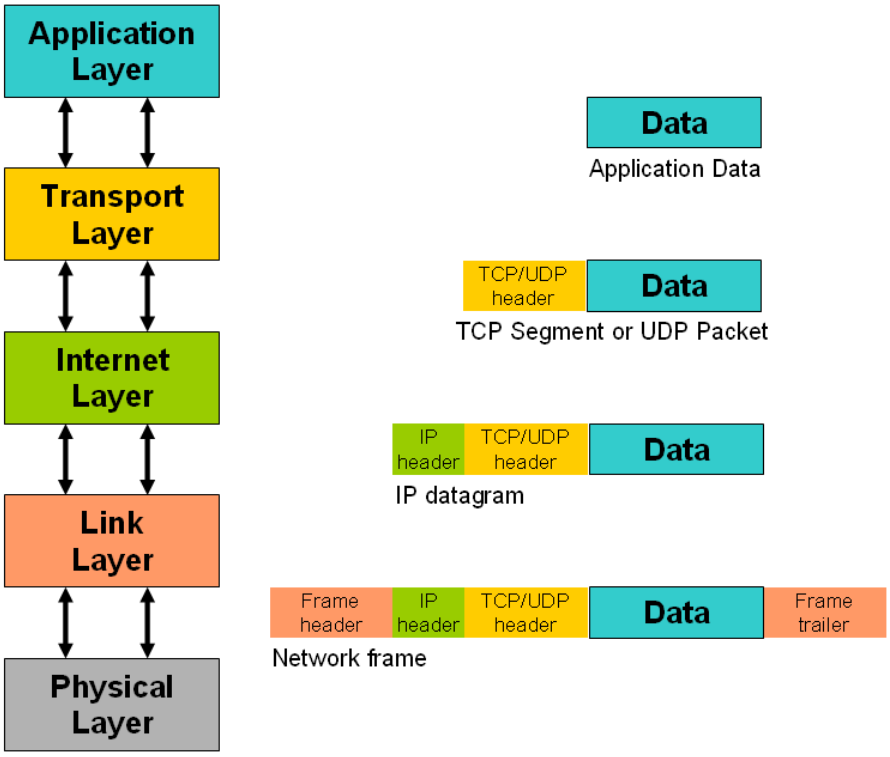


Figure 2.2. TCP/IP header encapsulation

(source: http://uw713doc.sco.com)

The four abstraction layers of TCP/IP are often compared with the seven-layer OSI Reference Model, which is the product of the Open Systems Interconnection effort at the International Organization for Standardization. The seven-layer division is intended to divide the communication system into smaller parts (Forouzan, 2002).

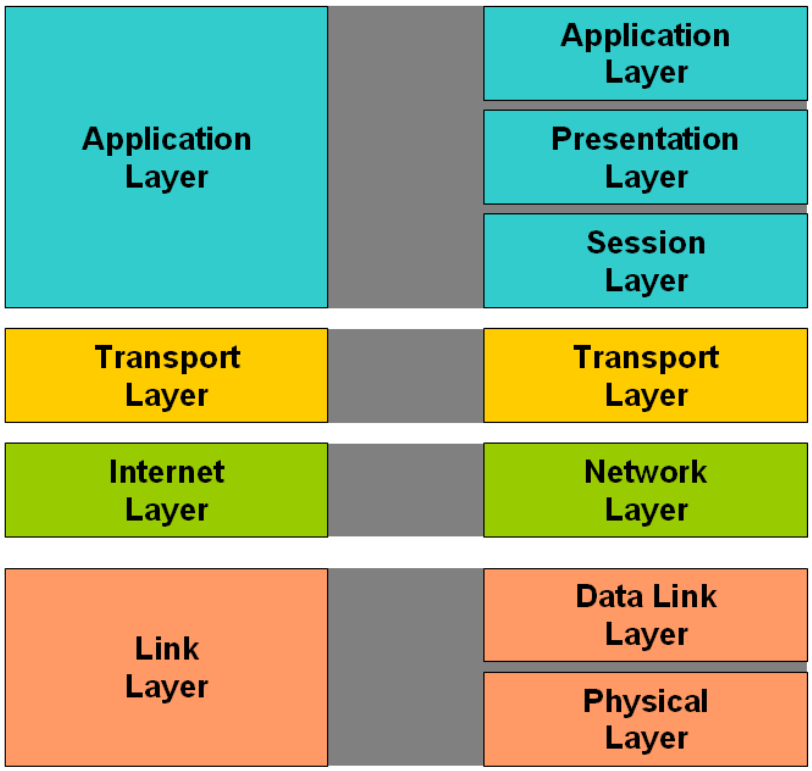


Figure 2.3. TCP/IP Layer and OSI Layer

2.4. Domain Name Service (DNS)

Domain Name Service (DNS) was developed to allow the users to connect to the host easily. Previously, the users had to remember the IP address of computer which consisted of some numbers. Commonly, the collection of numbers is not easily remembered by human. By using DNS, the users do not have to know the IP address of the computer. They also do not have to know the route to the computer. The computer which is connected to the network can be visited by typing its name address (Panwar et al, 2004).

The Domain Name System (DNS) manages the names by using a hierarchical system on distributed databases which is connected to the Internet or a private network. It will translate any requests of domain names to the numerical identifiers related with networking equipment. Humans will be easier to find any devices around the world by using their meaningful names.

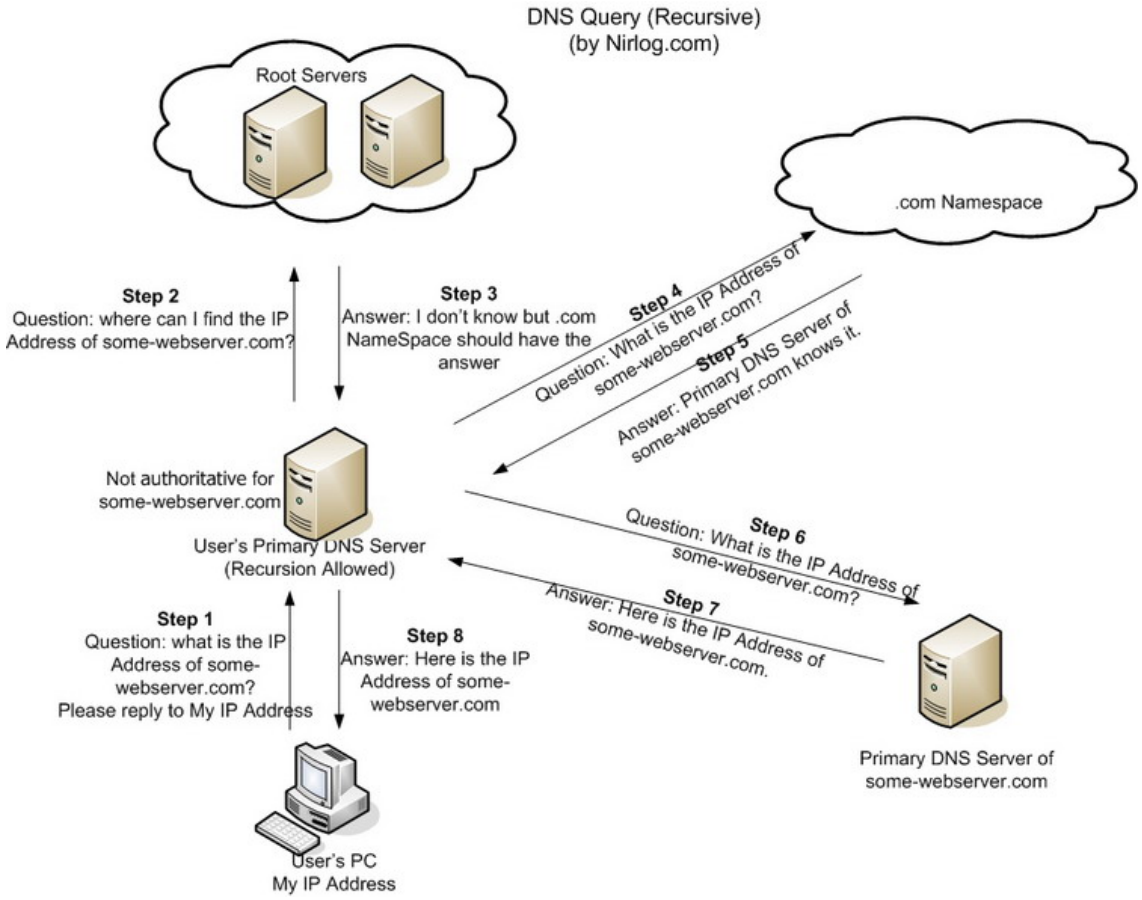


Figure 2.4. Distributed databases for Domain Name System

(source: www.nirlog.com)

2.5. Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) is the foundation on the World Wide Web who has one of functions to define the rule for any web browsers to put the documents on the server (Forouzan, 2002). A web browser is needed to display the HTML document on the client's screen. Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, and Opera are the examples of the web browser. HTTP is located on the Application Layer of TCP/IP but not limited on TCP/IP only. HTTP is possible to be implemented on the upper of any protocols on the Internet, or on other networks.

Uniform Resource Identifiers (URIs) or Uniform Resource Locators (URLs) is used to

identify any the resources connected to HTTP by using http or secure http (https) schemes. HTTP is a protocol to request or answer between client and server. A client of HTTP or user agent (such as web browser, robot, etc) initiates to request by creating a connection to a port (commonly 80 or 8080) in the web server. Server or origin server will give a status as a response, following by HTML documents or other contents. In the connection between user agent and origin server, there may be a third party connector, such as a proxy, gateway, and tunnel.

2.6. Hypertext Markup Language (HTML)

Hypertext Markup Language (HTML) is the major markup language to build web pages on the Internet. The HTML consists of a set markup tags to describe the format of the content on the web pages (Kressin, 1997). Each tag is enclosed in angle brackets, such as <h1>, and normally comes in pairs like <h1> and </h1>. The first tag is called an opening tag and the second tag is called a closing tag. Some tags are single tags which usually have slash character before the end of angle bracket, such as
.

Even though the contents have HTML tags, the web browser does not display the tags. The web browser uses the tags to format the content of the page. It will display the texts, paragraphs, lists, links, quotes, images, and other embedded objects. It also can be used to create interactive forms.

```
<html>

<head>
<title>Example of a Web Page</title>
</head>

<body>
<p><b>This is example of a Web Page</b><br/>
Thank you</p>
</body>

</html>
```

Figure 2.5. Example of HTML code on a web page

Additionally, JavaScript and Cascading Style Sheets (CSS) can be embedded on the web pages to enrich the interactivity. JavaScript, a client side script, will bridge the interaction between users and the web page by giving the instructions to the web browser. Cascading Style Sheets (CSS) will manage the appearance and layout of text and other material on the web pages.

2.7. Web Based Application

A development on the World Wide Web is continuing. Since a static web application represented by HTML was not enough to catch speed of data updating, a dynamic web application was introduced. JavaScript and VBScript are used to provide a dynamic web application in the client side. CGI/Perl, PHP, ASP, JSP, and others are used to provide a dynamic web application in the server side (Jablonski, 2004; McFarland, 2005). Combining with any database, the web page does not show the static information, but it changes dynamically anytime based on the updating data. (Coverse, 2004)

2.7.1. Client-side Scripting

The term of client-side scripting is referred to the script embedded into HTML that is executed on the client side. The script will give instructions to the web browser in the client side to do some actions. It will enable the web pages to give different behaviors and change the content, based on the user's input or other conditions. Moreover, the users can see the script in the web browsers by viewing the source code of web page.

JavaScript and VBScript are well-known client-side scripting language used nowadays. However, JavaScript is widely accepted by most of web browsers, compared to the VBScript. VBScript developed by Microsoft is only able to run in Internet Explorer web browser. On the contrary, all web browsers are able to execute JavaScript (Bhasin, 2002). Both of scripts can be embedded within an HTML code as an embedded script and referenced by the documents as an external script.

In the beginning, the client-side scripting is used by the majority to enrich the appearance of web page, such as display the time, give effects of the cursor, page, or any objects of web page, and display some warnings. Now, major implementation of client-side scripting is used for input form. Client-side scripts give instructions for the browser to response certain user actions, (e.g., clicking a button). It will help the web page to reduce the need to contact the server, before every input is completed and validated.

Figure 2.6. JavaScript is used for timer countdown

2.7.2. Server-side Scripting

Based on its term, server-side scripting is referred to the script which is executed on the server (Jablonski, 2004; McFarland, 2005). The result of the execution is a normal HTML page which has been processed previously by server. It is usually combined with the databases or other data stores to provide interactive web sites.

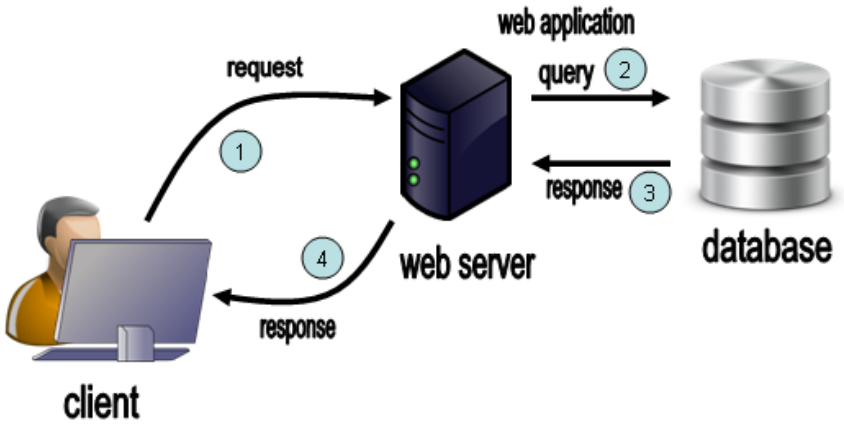


Figure 2.7. Web application gets information from database

Compared to client-side scripting, the code of server-side scripting cannot be seen from web browser. The page is customized for each user or different clients may get different response of page because the page is produced from the response based on the user's requirements, access rights, or queries into data stores.

PHP (PHP: Hypertext Preprocessor), ASP (Active Server Pages), ASP.NET, and JSP (Java Server Pages) are well-known languages of server-side scripting used in the world wide. The web developer who likes open-source usually prefers to use PHP because it is free and it can be used in Linux, Windows, Mac OS, and other operations systems. The commercial web developer may prefer to use ASP or ASP.NET using its development tools. However, the development tends to be more expensive than PHP because it uses license, not only for development tools but also for the web server and additional plug-ins. The Java Programmers prefer to develop the web application by using JSP because it uses the same environment on the desktop.

2.7.3. Dynamic HTML (DHTML)

Dynamic HTML (DHTML) is a big concept which combines some technologies to create interactive web pages (Powell et al, 2007). The main elements of DHTML are a web page written by a static markup language (such as HTML), an embedded program in the client-side (such as JavaScript), a language to manage the presentation definition of each element on a web page (such as CSS), and a structure of the objects on a web page (DOM). The term has intention to emphasize on the way to display the interactive page. After the page is fully loaded, it will give effects for each action of the users on the web page.

Figure 2.8. Dynamic HTML provides richer user interface

Even though the characteristic of DHTML does not emphasize on the unique page on each loaded, the page can be produced by server-side scripting (such as PHP, ASP.NET, or JSP) where the web server generates contents before sending them to the client. Those elements of DHTML must be included on the page generated by the server.

2.7.4. Document Object Model (DOM)

The Document Object Model (DOM) provides the structure of the web page to identify and modify the property of each element on a page. By learning DOM, the web developers are able to manipulate the contents of a web page (or document) because DOM provides methods to retrieve the property of objects, set the property of objects, add and remove the objects. All of the methods will enable the web developer to create the dynamic web (Marcheto et al, 2008).

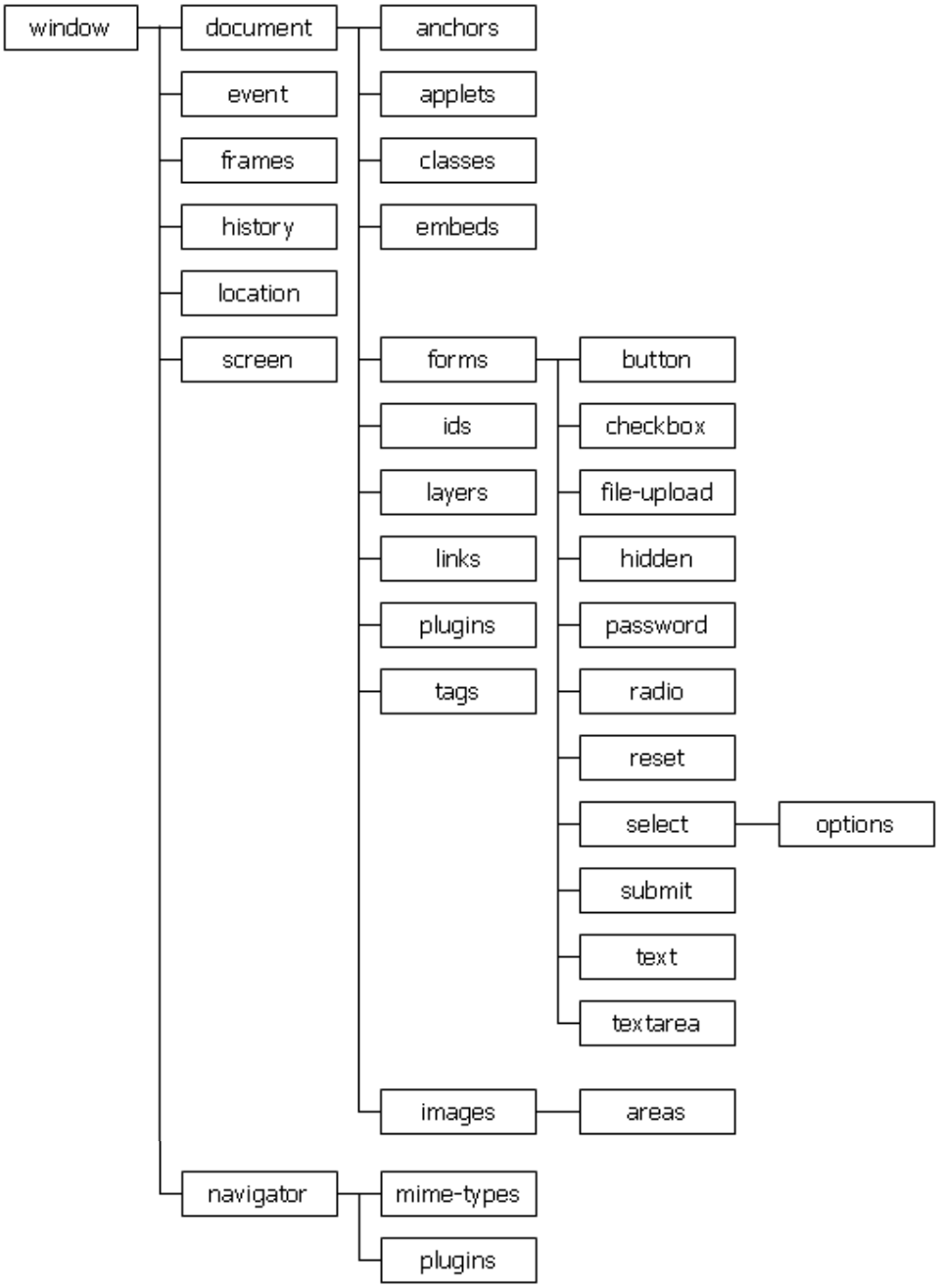


Figure 2.9. The structure of Document Object Model

The Document Object Model (DOM) is able to be used in many platforms and any languages to represent and interact with any objects in a web document. By using client-side scripting (such as JavaScript), the web developers are able to modify the content of each element via DOM.

2.7.5. Cascading Style Sheet (CSS)

Cascading Style Sheets (CSS) is a style sheet language used to format the layout, colors, and fonts of the web page (Thomas, 1999). It can be embedded on the web page as inline style sheet or embedded style sheet, or referenced by the web page as linked style sheet. By using CSS, it is possible to display the page in several different formats, such as on-screen and in print.

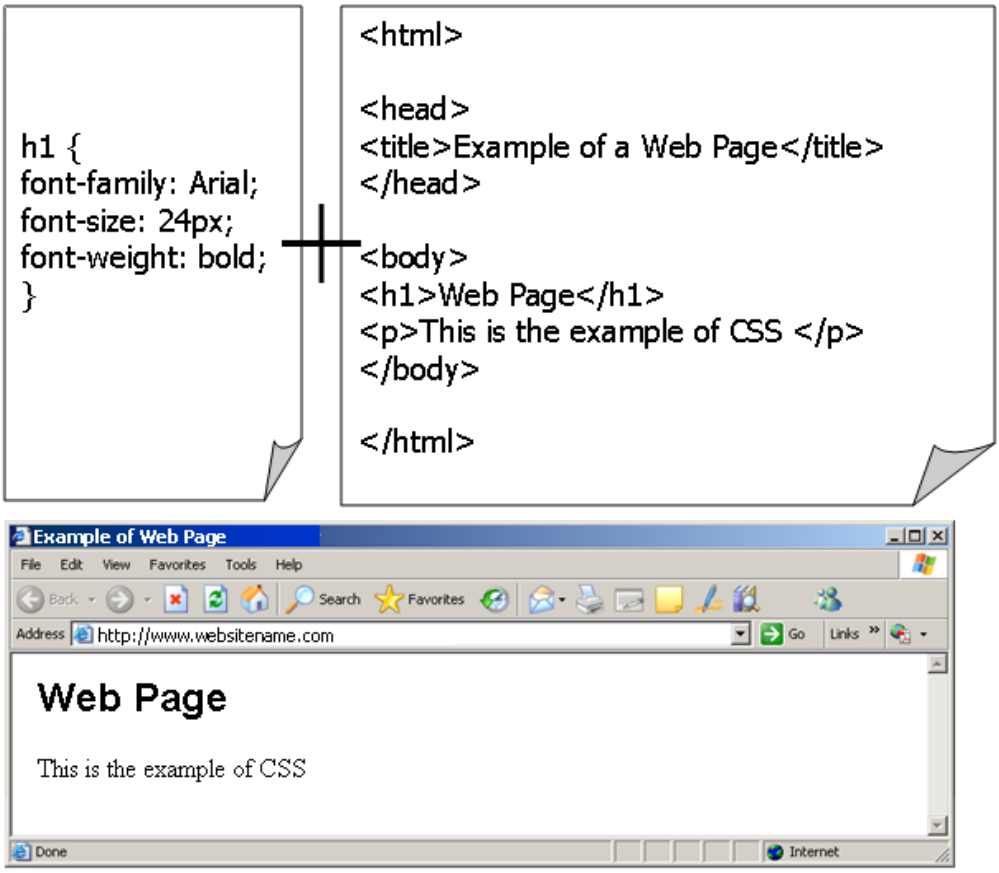


Figure 2.10. CSS format the layout of a web page

The term of Cascading Style Sheets is derived from the fact that each different style declarations can be placed in sequence, then form a parent-child relationship in each style. CSS also uses a priority scheme to decide which style rules apply if more than one rule

matches against a particular element.

2.8. Ajax

Ajax (Asynchronous JavaScript and XML) is not a new technology, but a set of technologies used to provide richer user interfaces for dynamic web application. It consists of HTML, CSS, Document Object Model (DOM), XMLHttpRequest, and JavaScript. Integration of those technologies will provide an interactive user interface and ability to show the dynamic contents asynchronously (Marchetto, 2009).

Ajax creates a new way to communicate between server and client by using an asynchronous interactive technology. The result will give more efficient response and better user experience (Jiaqi, 2009). The web application will not have to load the entire HTML. It will not leave the users waiting for responses from web server. The data can be requested and be transmitted when the users browse the page (Lin, 2008). The other benefits, several pages can be migrated to be a single page interface with the independent control on each component (Mesbah, 2007).

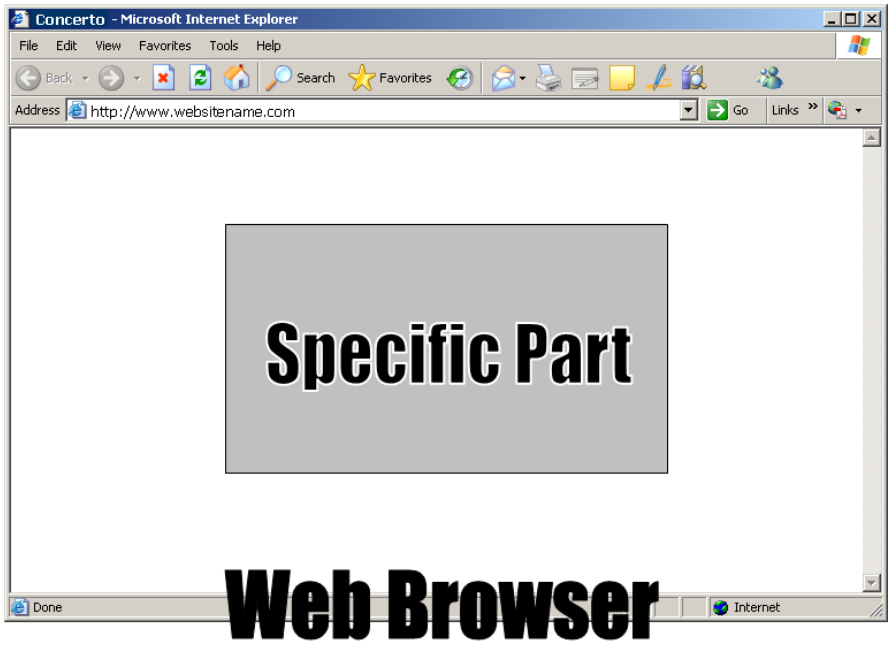


Figure 2.11. Ajax reloads only specific part on a web page

Ajax has been used in some applications created by several famous web providers, such as Google Maps, Google Docs, Google Suggest, Google Chat, Flickr, Yahoo! Mail, and Outlook Web Application (Thiesen, 2007). Those increase the popularity of Ajax.

Ajax is using Time to Refresh (TTR) to renew the contents without user's intervention. The duration of TTR can be based on the web developer or the analyzing of user's internet speed. However, it has limitation if the users want to get the real-time data from the server. The user-definable intervals cannot predict when the server issues the new data.

2.9. Green Ajax

Green Ajax irons out the problem by providing the new approach based on initiating the signal from web server to the clients whenever an update or a change of data occurs. As the client receives the server's signal, the client will then send out a request (Sanjaya, 2010a). This scheme cuts required bandwidth down to necessary due to the update. From the experiment, it is proved that the bandwidth saving is based on some forms of the update including infrequent and frequent application.

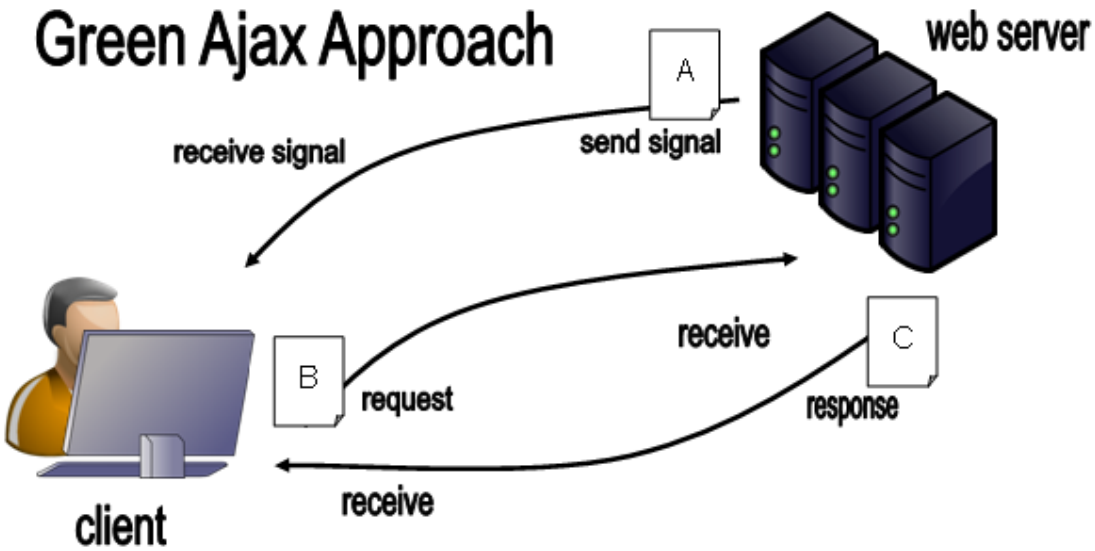


Figure 2.12. Green Ajax approach

Green Ajax is an enhancement of Classic Ajax which has been recognized in the web application. Inheriting from the origin, Green Ajax also has the ability to decrease the bandwidth consumption through delivering data only in the specific part of web page (Sanjaya, 2010a). The user interfaces will display any information without any interruption on screen because the communication between web application and server is done in the background.

The improvement is located in the ability of web application to receive notification from the server whenever an update or a change of data occurs. When the application receives the server's signal (Sanjaya, 2010a), the application will then send out a request. The applications will display the latest information without any involvement from user. This method reduces required bandwidth below than necessary for data updating.

2.10. Fuzzy-based Application

Fuzzy logic provides an alternative way on using non-crisp value. Some values are uncertain and cannot be figured out by a Boolean function or other logics which use crisp value sets. On the other hand, Fuzzy will result any doubtful figures lying between 0 and 1 based upon the degree of membership. The degree of membership is decided by the function (Saritas et al, 2007).

In the Fuzzy logic, the fuzzifier will map the crisp inputs into fuzzy sets on the data entry side. On the other hand, defuzzifier will map the fuzzy sets output into a single crisp point on the result side. The fuzzifier is needed in fuzzy logic because the system has to convert crisp data to fuzzy data in the beginning. Commonly, if the system receives crisp inputs, the outputs are also data in crisp format. Furthermore, the defuzzier will be used to convert fuzzy data into crisp data as the output (Mouzouris, 1997).

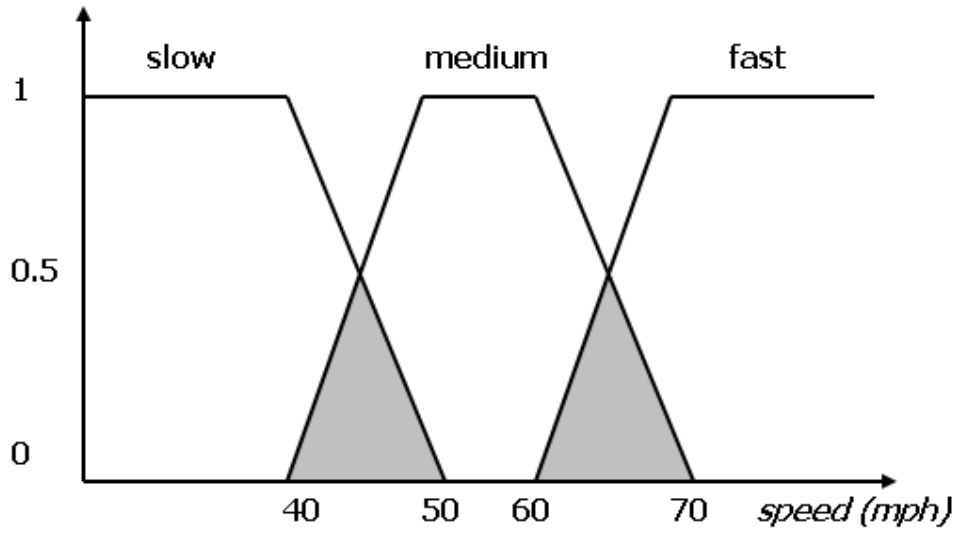


Figure 2.13. Unclear definition of speed in the crisp area

Some ranges of time in the experiment can be realistic and over crossed situations. From these conditions, results will look more practical than the ones demonstrated (Sanjaya, 2011a). By applying Matlab (Mathworks, 2002), the Fuzzy Inference System introduced by Sugeno can be utilized to solve the uncertain ranges of update time. Results due to these fuzzy based applications can be employed to solve the uncertainty in each input region. After the clarification of each fuzzy-based input is accomplished then the bandwidth saving will be further evaluated based on these Matlab results.

III. RESEARCH METHODOLOGY

This research attempts to develop a concept to reduce the web traffic of any update web-based application and design an approach to provide a real time data updating on the web-based application.

3.1. Theoretical Frameworks

Based on the web architecture, the client has to request to the web server to get a response. To get a response from the web server frequently, the client has to do the request periodically. In the web application, a timer is usually used to request to the web server periodically. In the smaller interval, web application will request to the web server with the high frequency. In reverse, web application will request to the web server with the low frequency.

Ajax can reduce the web server load and the data transfer to/from users' computers because only a specific part in the web page is requested to the web server. However, the data updating from Ajax depends on the web application request to the web server. If the request is done when the data is still not updated, it will waste the bandwidth and the resource.

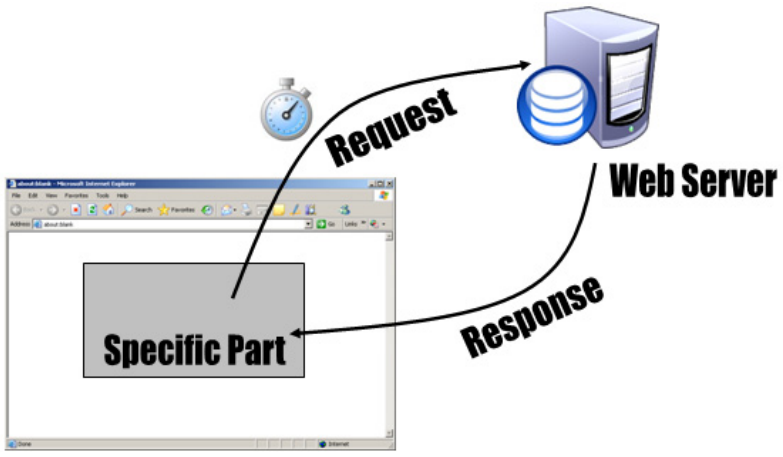


Figure 3.1. Timer to request new data to the web server periodically

3.2. Conceptual Frameworks

Based on the above statement, the client's request should be done at the appropriate time. The time to request should depend on the data updating activity on the web server. The approach should design the way to give a notification from the web server to the client if there is a new update. A web server in the client side or embedded on the web browser might be needed. It should give the capability to the client to receive a signal from the web server.

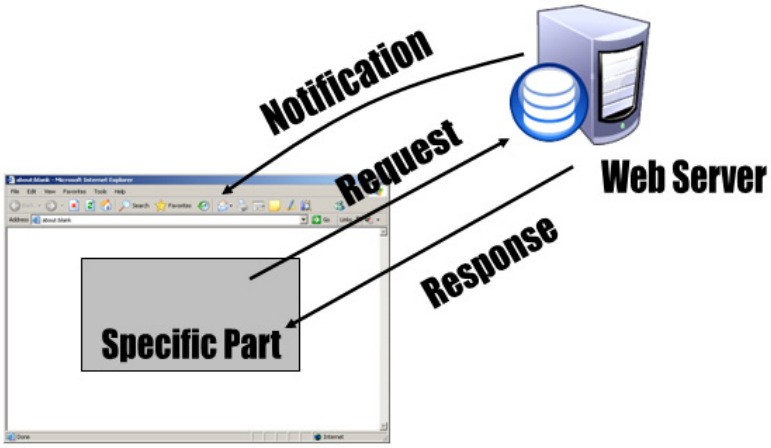


Figure 3.2. A web server send a signal to the client if there is a new update

To reduce the web traffic, the approach can be combined with Ajax to limit the size of data updating. The combination of these will bring the idea of low bandwidth and the low resource together to provide a real time data updating which will be named as Green Ajax.

Ajax will be used as a main technique in the Green Ajax because Ajax has been proven by some researches as a good approach to decrease the bandwidth. However, Ajax only does some meaningless requests in the condition of infrequent data updating. Green Ajax will cover the limitation of Ajax in this case. Furthermore, Ajax will be named Classic Ajax to clarify the difference with Green Ajax as a new approach.

3.3. A Conceptual Model

To provide the Green Ajax approach on the applications, the web server must have the

ability to signal the client. If the client gets a signal from the server, the client will request the new data to the server to provide a real time data updating on the web application. The conceptual model to provide the Green Ajax approaches can be shown below.

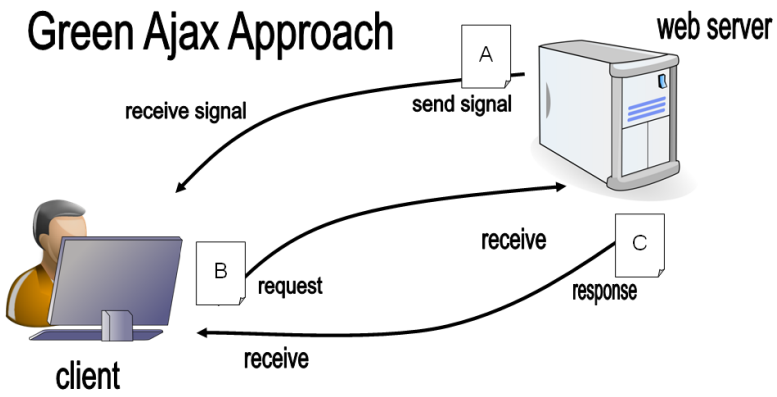


Figure 3.3. The model of Green Ajax Approach

The client will receive a small size of signal from the server for each updating activity to trigger a request to the web server. The web server will send all of the responses based on the client's request. The proposed approach to receive the signal from the web server without clients' request is providing a virtual server on the clients.

The way to calculate the bandwidth consumption of Classic Ajax and Green Ajax can be shown on the formula below. In case there are 15 updates from the server, each type of Ajax will consume different bandwidth using the following calculation.

- Classic Ajax

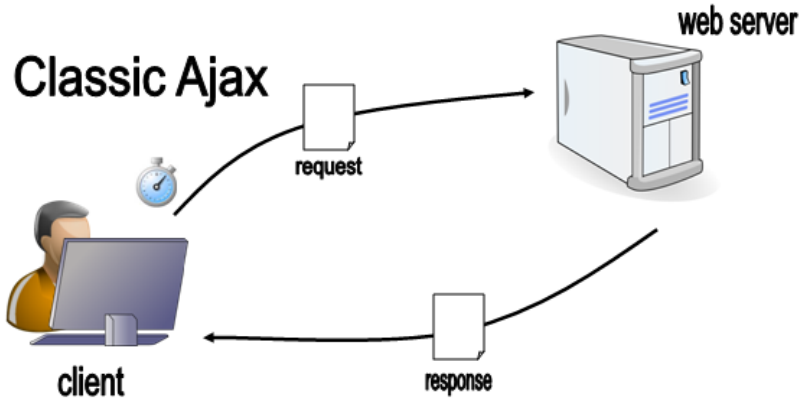


Figure 3.4. Data transfer on Classic Ajax

Formula:

Bandwidth consumptions = x (Req + Rcv) bytes per hour, in which x = 1 hour / TTR

Time to Request (TTR) = 2 minutes or 30 times in one hour

Request size = 10 bytes

Response size = 100 bytes

Bandwidth consumptions = 30 x (10 + 100) bytes per hour

= 30 x 110 bytes per hour

= 3,300 bytes per hour

- Green Ajax

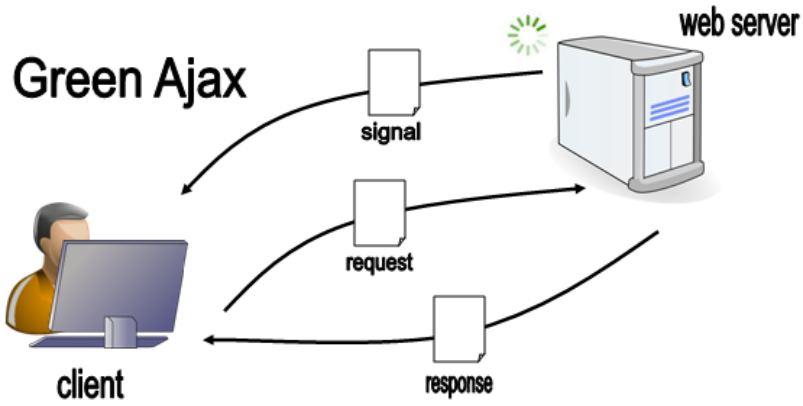


Figure 3.5. Data transfer on Green Ajax

Formula:

Bandwidth consumptions = y (Sig + Req + Rcv) per hour, in which y = numbers of update.

There are 15 times of update in one hour

Signal size = 1 byte

Request size = 10 bytes

Response size = 100 bytes

Bandwidth consumptions = 15 x (1 + 10 + 100) bytes per hour

= 15 x 111 bytes per hour

= 1,665 bytes per hour

The size of the signal, the request of client, and the received response can be counted by using Firebug on Mozilla Firefox.

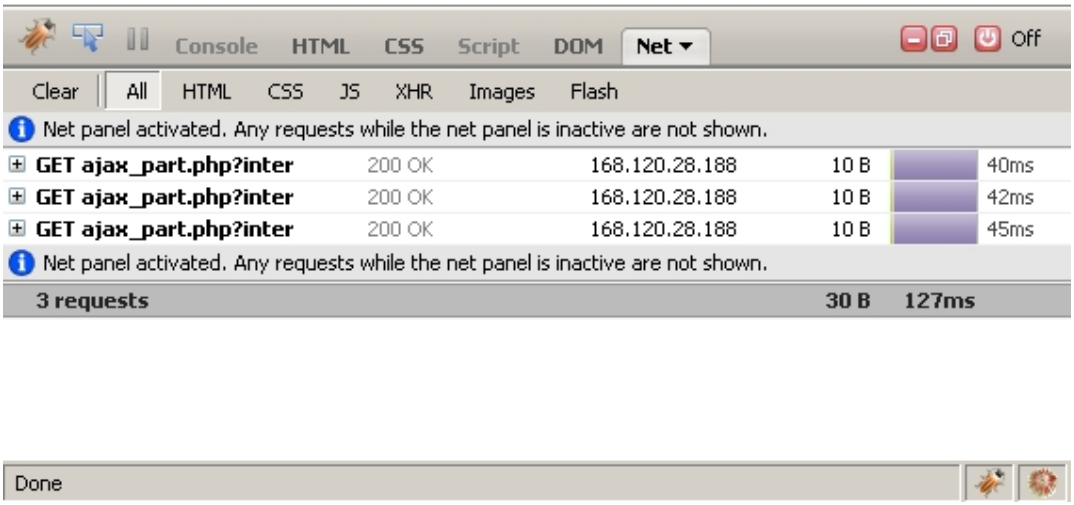


Figure 3.6. Firebug on the Mozilla Firefox

3.4. The Environment of Experiments

To implement the Green Ajax in the local area network, the experiment will be done on the computer laboratory which is connected on the network. Those computers are Acer Veriton 3700GX which have specification Intel Pentium 4 3.2 GHz Processor, 1 GB RAM, Hard Drive 80 GB 7200 RPM S-ATA, LAN Card Broadcom 4401/5751 10/100/1000 Controller, and Windows XP Operating System. However, the experiment can be done by using other computers that have different specification. Even though the Networks Interface Card (NIC) or other peripherals are changed, the trend of results will not have significant differences.

Figure 3.7. Computers on the Experiment

The topology of the network on the computer laboratory is using star topology. The client is connected to the web server or another client via the switch/hub. Each client will be equipped with a virtual server to provide the capability on the signal receiving from the web server.

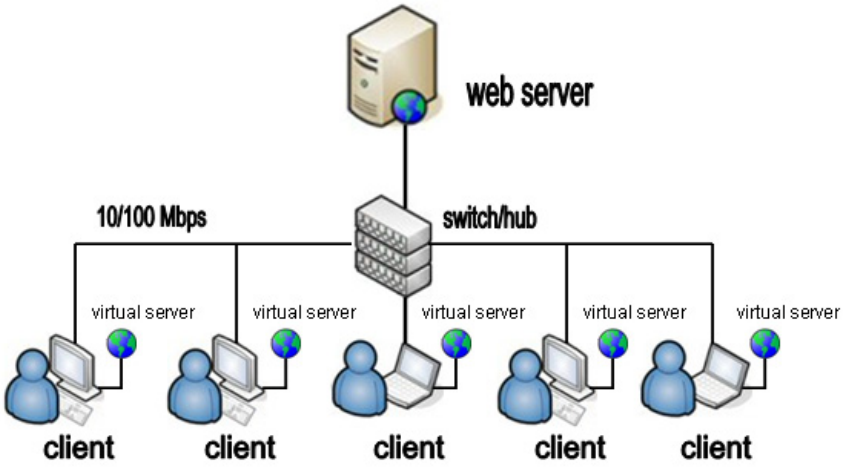


Figure 3.8. The Example of Topology

Each client will have a Mozilla Firefox 3.5.14 as the web browser to test the Classic Ajax and Green Ajax. Firebug 1.4.5 will be used in the Mozilla Firefox to count the size of the signal, the request of client, and the received responses in the experiment. The POW 0.1.9 add-on will be added on each client to provide the virtual server.

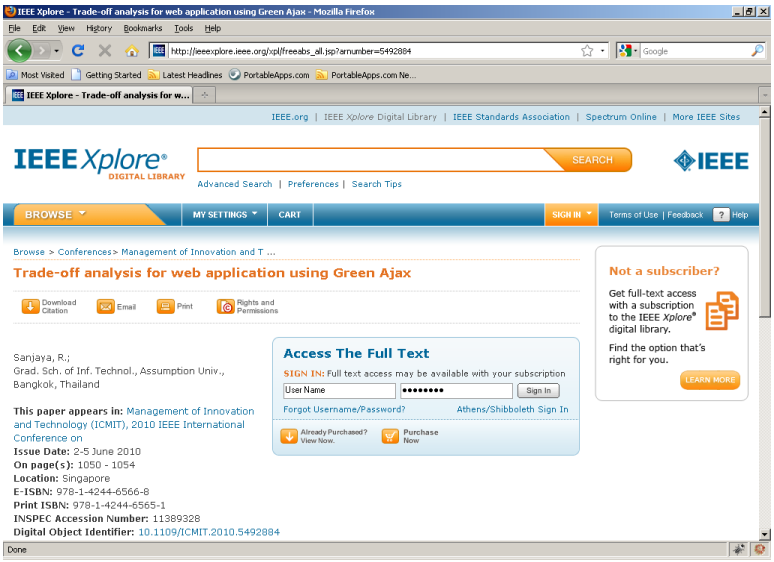


Figure 3.9. Mozilla Firefox as the web browser in the research

On the other side, the server will have XAMPP 1.7.4, a package of Apache web server which contains MySQL, PHP, and Perl. The web-based applications will be placed in the server to communicate with clients. In this research, the web-based applications will adapt the approach to compare the Classic Ajax and the Green Ajax. Matlab 6.5 is used also to cross-check the Fuzzy-based application which is tested on this research.

In the preliminary experiment, the bandwidth consumption of Green Ajax will be compared to the Ajax applications which have different interval times. The interval time for Ajax applications can be selected randomly, such as 30 seconds, 60 seconds, 5 minutes, 10 minutes, and 15 minutes. All of data will be collected in the experiment, such as bandwidth consumption, number of updates activity, number of client's request, number of received data in the client, number of wasted request of the client, and the non-received data.

Table 3.1. Experiment Data Form

Data	**Green Ajax**	**Classic Ajax**				
		30"	***60"***	***5'***	***10'***	***15'***
Requests (Req)						
Received (Rcv)						
Wasting (Wst)						
Updates (Upd)						

Data	Green Ajax	Classic Ajax				
		30”	*60”*	*5’*	*10’*	*15’*
Un-received (Los)						
Server (Srv)						

These data can be used to summarize the Data Loss Percentage (DLP) of each experiment. The value of DLP will identify the ineffective requests done by the client. The formula of DLP can be shown below.

$$\text{Data Loss Percentage (DLP)} = \frac{\text{number of times data was not received timely (Los)}}{\text{number of times update data was activated (Upd)}} \times 100$$

On the other side, these data can also be utilized to calculate the Successful Receptive Percentage (SRP) of each experiment. The amount of effective request done by the client can be seen from the value of SRP. The formula of SRP can be shown below.

$$\text{Successful Receptive Percentage (SRP)} = \frac{\text{number of times data has been received (Rcv)}}{\text{number of times requests had been issued (Req)}} \times 100$$

Other experiments will be continued using other random interval time, fixed intervals time of update on the web server, and fuzzy-based application. Those experiments will be used to make conclusions of the benefit between Green Ajax and Classic Ajax.

3.5. The Experiments

There are four experiments that will be conducted for this research to compare and implement the Green Ajax. First, Green Ajax will be compared to Classic Ajax by using random update activities issued by the server. Second, Green Ajax will be compared to Classic Ajax when the server does an update based on the fixed interval time. Third, Fuzzy-

based application will be used on both Green Ajax and Classic Ajax. Fourth, an application on the server will send some random signals to the Green Ajax.

3.5.1. Web Traffic Reduction for Infrequent Update Application Using Green Ajax

In this experiment, some random update activities (Upd) are sent by the web server. In the client side, Classic Ajax sends some requests based on the timer. Those numbers of request will be reckoned as Request (Req). If the client gets a new data from the reply sent by the server, it will be counted as received (Rcv). However, if the client does not get a new data as the reply, it will be counted as wasting request (Wst).

When the application did not get the complete data from the web server, it will be counted as Data Loss (Los). In the Green Ajax concept, the signal will be sent by the web server to give a notification to the client when the web server has a new data. The activity will be identified as Server (Srv). The data from one hour experiment will be recorded on the table 3.2.

Table 3.2. Summary of Experiment Using Green Ajax

Data	**Green Ajax**	**Classic Ajax**				
		30"	***60"***	***5'***	***10'***	***15'***
Requests (Req)						
Received (Rcv)						
Wasting (Wst)						
Updates (Upd)						
Un-received (Los)						
Server (Srv)						

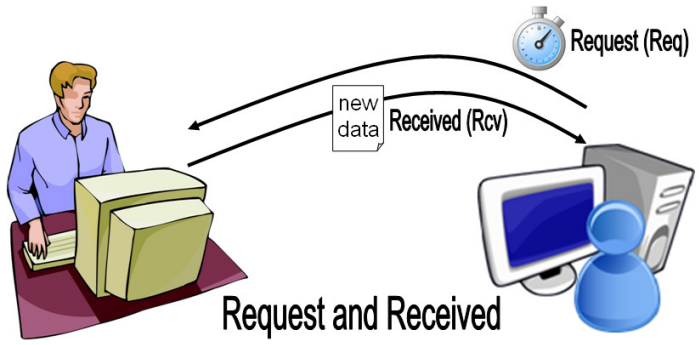


Figure 3.10. Request (Req) and Received (Rcv)

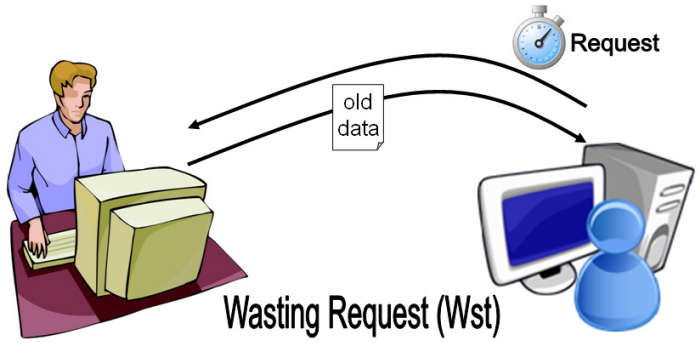


Figure 3.11. Wasting Request (Wst) when the old data replied to the client

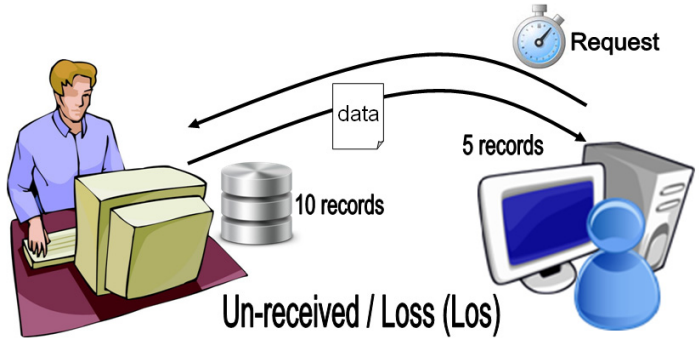


Figure 3.12. Data loss (Los) when the client gets partial data released by the server

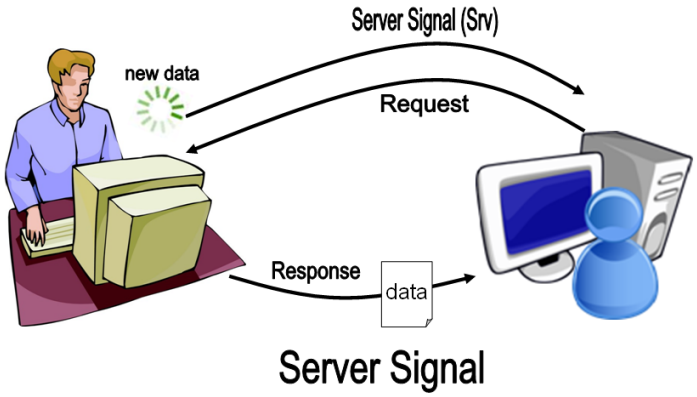


Figure 3.13. Server will signal (Srv) to the client when the server has a new data

The chart will be created based on the table 3.2 to give the clear illustration of requests (Req), received (Rcv), and un-received (Los) of Green Ajax and each Classic Ajax application.

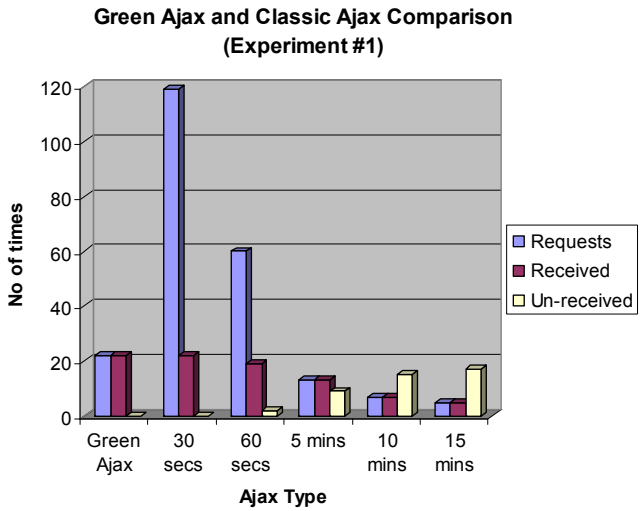


Figure 3.14. Example of the Experiment Chart

By using the formula, SRP and DLP of Green Ajax and some Classic Ajax applications can be summarized on the table 3.3.

Table 3.3. SRP and DLP on the Experiment

Results	**Green Ajax**	**Classic Ajax**				
		30"	***60"***	***5'***	***10'***	***15'***
SRP (%)						
DLP (%)						

Based on the table above, the chart below will be created to show the SRP and DLP of Green Ajax and some Classic Ajax applications.

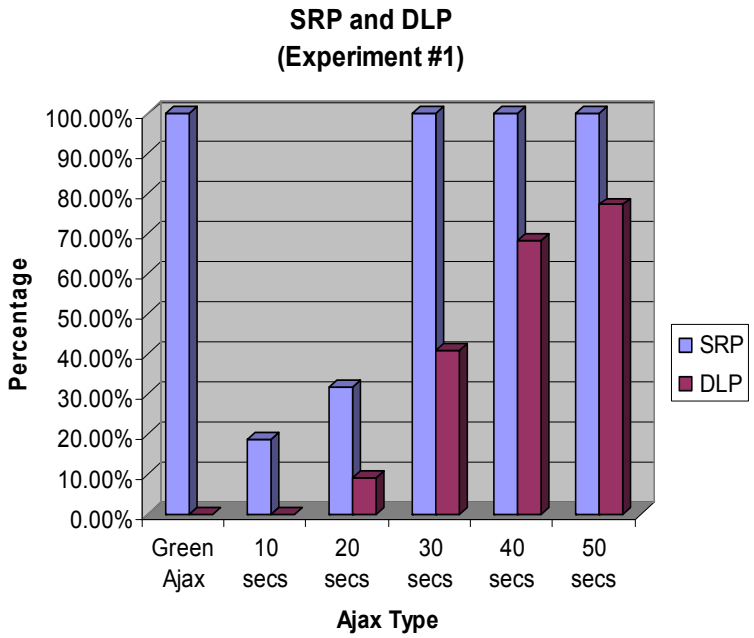


Figure 3.15. Example of SRP and DLP chart

The bandwidth consumption and data loss for each type of Ajax in the experiment will be recorded on the table below.

Table 3.4. Bandwidth Consumption on the Experiment

Results	**Green Ajax**	**Classic Ajax**				
		30"	***60"***	***5'***	***10'***	***15'***
Bandwidth						
Data Loss						

From the data on table 3.4, the chart will be created to give the clear view of bandwidth and data loss of each Ajax applications.

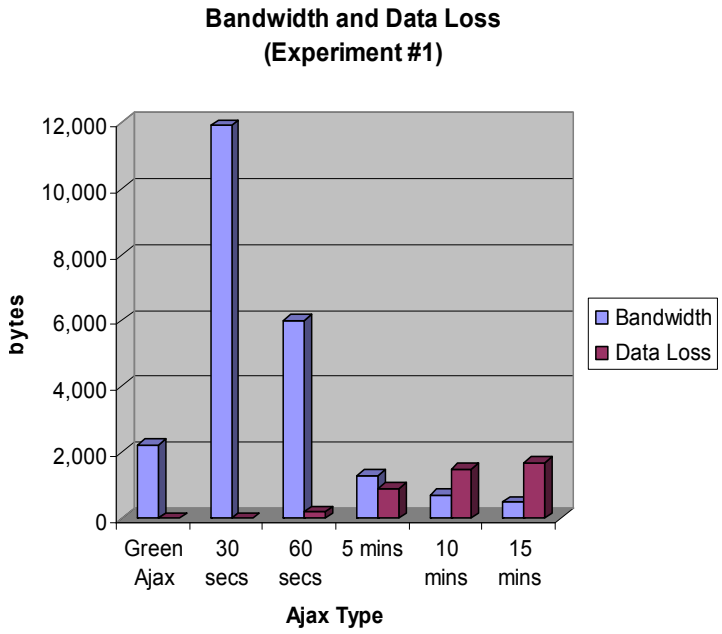


Figure 3.16. Example of Bandwidth and Data Loss Chart

The experiment will be repeated three times using different interval of time for Classic Ajax. The new interval time will be bigger than the interval time of previous experiment. The results from three experiments will give the comparison between Green Ajax and Classic Ajax. The interval time of Classic Ajax in all of experiments can be seen on the table below.

Table 3.5. Interval Time of Classic Ajax in All of Experiments

Data	**Green Ajax**	**Classic Ajax**				
		#1	***#2***	***#3***	***#4***	***#5***
1st Experiment	auto	30 secs	60 secs	5 mins	10 mins	15 mins
2nd Experiment	auto	5 secs	10 secs	15 secs	20 secs	25 secs
3rd Experiment	auto	10 mins	20 mins	30 mins	40 mins	50 mins

3.5.2. Trade-off Analysis for Web Application Using Green Ajax

In this experiment, the server will update the data based on the fixed interval time to compare Green Ajax and Classic Ajax. The interval time to update will be classified as the high frequency, medium frequency, and low frequency. The range of each classification can be seen on the table 3.6. Both the web server and Classic Ajax use the same interval time to do their tasks. As the concept, Green Ajax will request only if there is a signal from the web

server.

Table 3.6. Scenario of Experiments

Group #	Scenario	
	Situations	***Interval Time***
1	High Frequency Update	5 seconds – 25 seconds
2	Medium Frequency Update	30 seconds – 15 minutes
3	Low Frequency Update	30 minutes – 5 hours

The first experiment on high frequency will use interval time of 5 seconds, 10 seconds, 15 seconds, 20 seconds, and 25 seconds. In the second experiment on medium frequency examines 30 seconds, 1 minute, 5 minutes, 10 minutes, and 15 minutes of interval time of data updating. The last experiment using low frequency will examine every 30 minutes, 1 hours, 2 hours, 3 hours, and 5 hours.

Illustration of the experiments can be seen on the figures below. All of experiments will update using scheduled time: Classic Ajax will request based on the interval of time, and Green Ajax will request based on the signal released from the server, as seen on figure below.

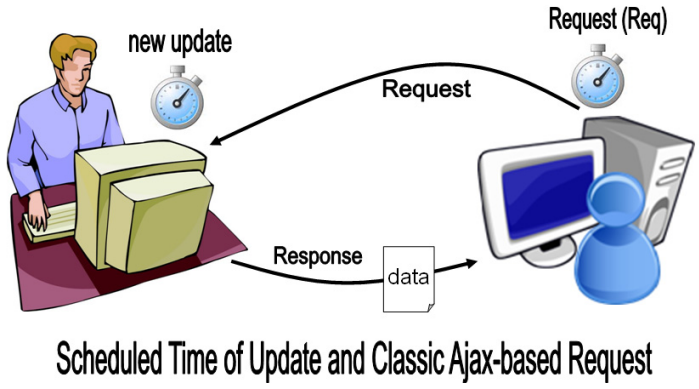


Figure 3.17. Web server and Classic Ajax use the interval time to do their tasks

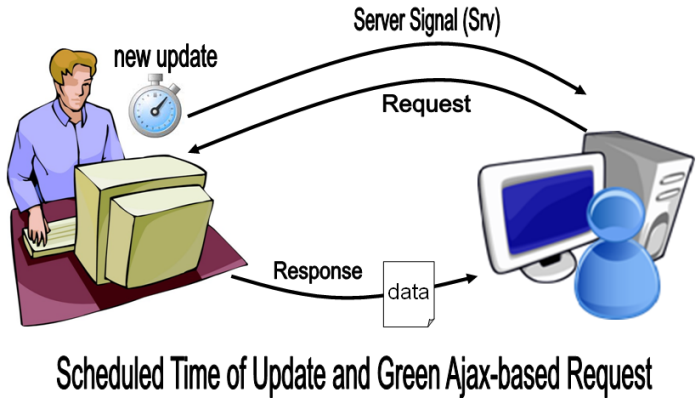


Figure 3.18. Green Ajax will request based on the signal released from the server

The results of bandwidth consumption in the experiments will be recorded on the table below. The table below will be adjusted for the experiment results of Medium Frequency Update and Low Frequency Update.

Table 3.7. Bandwidth Consumption (in bytes)

Scenario[a]	**Interval Time**				
	5 secs	***10 secs***	***15 secs***	***20 secs***	***25 secs***
Classic Ajax					
Green Ajax					
Margin (%)					

a. Experiment time: x hour

From the table above, the chart will be created to see the comparison of bandwidth consumption between Green Ajax and Classic Ajax using each interval time.

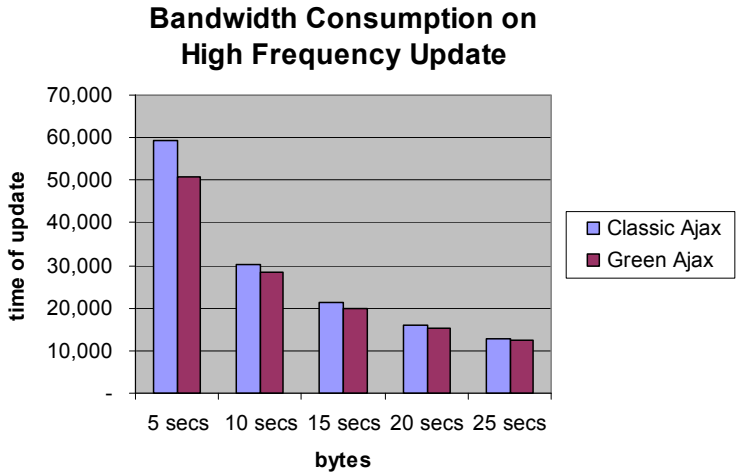


Figure 3.19. Example of Bandwidth Consumption Chart

Using the formula of Successful Receptive Percentage (SRP), the result of Green Ajax and Classic Ajax application will be summarized on the table below.

Table 3.8. SRP Data Collections

Scenario[a]	**Interval Time**				
	5 secs	***10 secs***	***15 secs***	***20 secs***	***25 secs***
Classic Ajax					
Green Ajax					

a. Experiment time: x hour

The Data Loss Percentage (DLP) of both Green Ajax and Classic Ajax application will be summarized on the table below.

Table 3.9. DLP on High Frequency Update

Scenario[a]	**Interval Time**				
	5 secs	***10 secs***	***15 secs***	***20 secs***	***25 secs***
Classic Ajax					
Green Ajax					

a. Experiment time: 1 hour

3.5.3. Mobile Traffic Evaluation for Fuzzy-Based Application Using Green Ajax

Previously, experiments are tested using crisp three range of time. However, they might be different with the actual conditions. Each range of time in the reality can be overlapping. In this research, the ranges of time of the previous experiments will be modified to examine the Green Ajax by using Fuzzy-based application. The example range of time can be seen on the table 3.10.

Table 3.10. Example Range of Time

Group #	**Scenario**	
	Situations	***Interval Time***
1	High Frequency Update	30 seconds – 10 minutes
2	Medium Frequency Update	5 minutes – 60 minutes
3	Low Frequency Update	30 minutes – 5 hours

Based on table above, the range of interval time to update on high frequency update

application is from 30 seconds to 600 seconds (10 minutes). The range of interval time to update on medium frequency update application is from 300 seconds (5 minutes) seconds to 3,600 seconds (60 minutes). The range of interval time to update on medium frequency update application is since 1,800 seconds (30 minutes) to 18,000 seconds (5 hours).

The overlapping area between High and Medium Frequency Update and also between Medium and Low Frequency Update can be seen in Figure 3.20. These overlapping conditions on each range will be faced in the real situations.

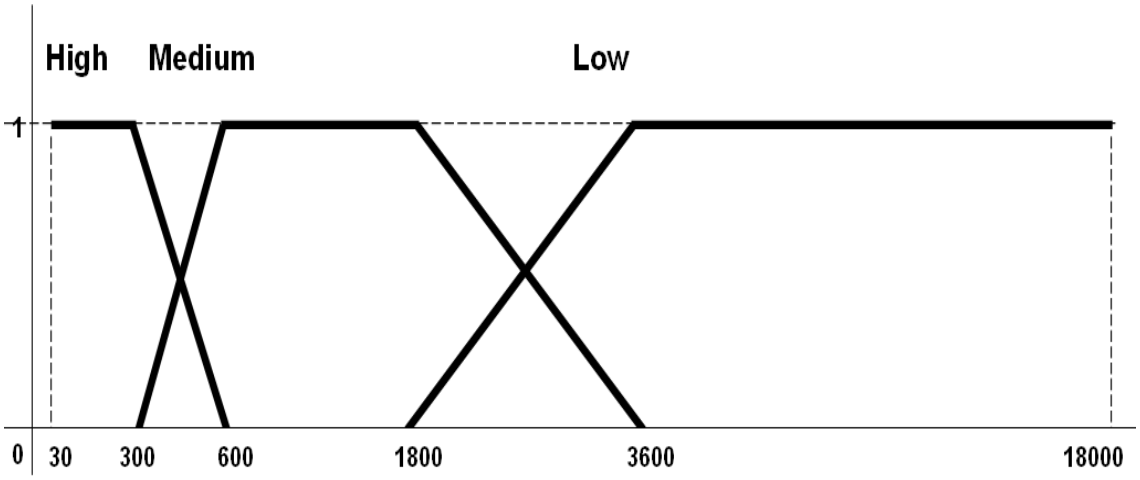


Figure 3.20. Range of interval time to update based on Table 3.10

The membership of each situation on the figure 3.20 is listed in the equation below. The range of time is assumed in three basic types (high, medium and low) as shown on the equation below.

$$High = \begin{cases} 1, & 30 \leq x \leq 300 \\ \dfrac{x - 300}{300}, & 300 < x \leq 600 \end{cases}$$

$$Medium = \begin{cases} \dfrac{x + 300}{300}, & 300 < x < 600 \\ 1, & 600 \leq x \leq 1800 \\ \dfrac{x - 1800}{1800}, & 1800 < x < 3600 \end{cases}$$

$$Low = \begin{cases} \dfrac{x + 1800}{1800}, & 1800 < x < 3600 \\ 1, & 3600 \leq x \leq 18000 \end{cases}$$

The result of the experiments based on the above ranges of time will be recorded in table 3.11.

Table 3.11. Bandwidth Consumption (in bytes) in Fuzzy-Based Application

Interval Time	**Bandwidth Consumption**		
	Classic Ajax	***Green Ajax***	***Margin***
x1 secs			
x2 secs			
x3 secs			
x4 secs			
.			
.			
.			
xn secs			

The table above will be used to record the result of all experiments. Those experiments will use three ranges of time namely high frequency, medium frequency, and low frequency. Based on those data, some charts will be created to compare the bandwidth consumption between Green Ajax and Classic Ajax.

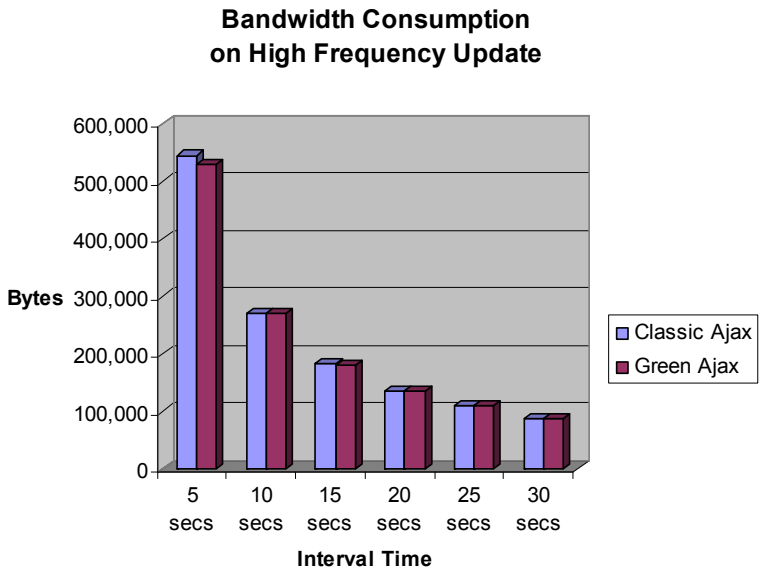


Figure 3.21. Example of Bandwidth Consumption Chart

The complete result in these experiments will be summarized in the table 3.12 to conclude the better approach on each situation.

Table 3.12. Final Results of Fuzzy-based Application Experiment

Group #	**Scenario**	
	Situations	***Better Approach***
1	High Frequency Update	
2	Medium Frequency Update	
3	Low Frequency Update	

3.5.4. Implementation on the Local Area Network using Randomized Cue Applications

In this experiment, the Randomized Cue Application will be created to work as a simulator to send a cue to the clients in the random time (Sanjaya, 2011b). The application will decide the time of sending the next cue based on the random value. All of data from this experiment will be recorded on the Experiment Data Form (Table 3.1).

From those data, the bandwidth consumption of Green Ajax will be compared to the bandwidth consumption of Classic Ajax application. The Classic Ajax application uses 2 minutes TTR to request the data to the web server. The bandwidth consumption and data loss between Green Ajax and Classic Ajax will be listed on table 3.13.

Table 3.13. Bandwidth Consumption and Data Loss (in bytes)

Results[a]	Green Ajax	Classic Ajax
Bandwidth		
Data Loss		

a. Experiment time: x hours

Using the formula of SRP and DLP, the result of calculation on this experiment will be recorded on the table 3.14 to see the performance of both Green Ajax and Classic Ajax.

Table 3.14. SRP and DLP

Results[a]	Green Ajax	Classic Ajax
SRP (%)		
DLP (%)		

a. Experiment time: x hours

The above results will show the performance of Green Ajax when it is implemented on local area network by using Randomized Cue Applications.

3.6. Source Code

In the Green Ajax implementation, a routine program is required in the server side to trigger a cue transmission to the client whenever an administrator does an updating. The routine program can be integrated to the update program, or can be separated modules which can be used together by other programs. A one byte character will be used as a cue that is sent to the clients. The reason of using one byte character is to limit the bandwidth usage, keep the client secure, and avoid harmful applications.

Each client on the local area network has their own identity in the form of an IP address. When the client connects to the server, the IP address of the client will be recorded by the server. That identity will be used as a target to receive the cue from the server when an administrator does an update.

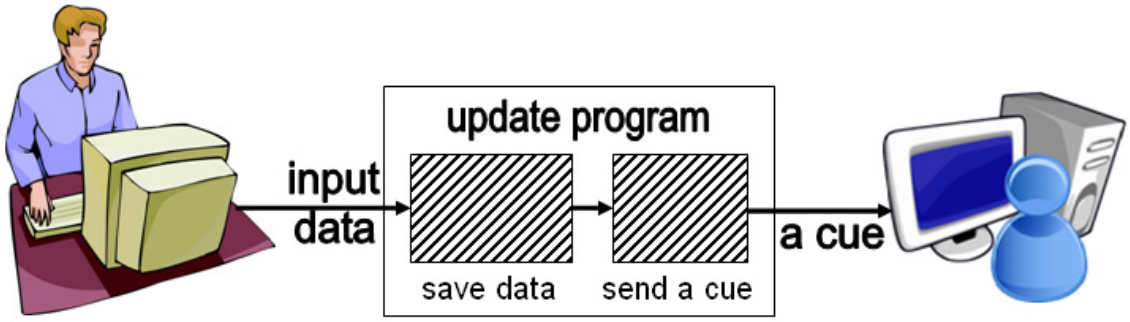


Figure 3.23. A routine program to trigger a cue transmission

The following PHP codes describe how the update program works. The RandomRefresh() function will set the next update time, CreateData() function will create random data for the experiments, and SendCue() function will trigger a cue transmission to the clients.

```
<?php
include “GreenAjax.php”;
$myRandomApp = new GreenAjax();
$myRandomApp->RandomRefresh();
$myRandomApp->CreateData();
$myRandomApp->SendCue();
?>
```

The small program on the client has the duty to receive the cue. It can be either a plug-in or extension that is installed in the web browser. POW (Plain Old Webserver) is an example of plug-in used in this research. A small program on the client will interpret the character received as a command to make a request to the server. This method is more effective than using a repetition of the request based on TTR. Furthermore, new data will be sent to the client after the server receives such request.

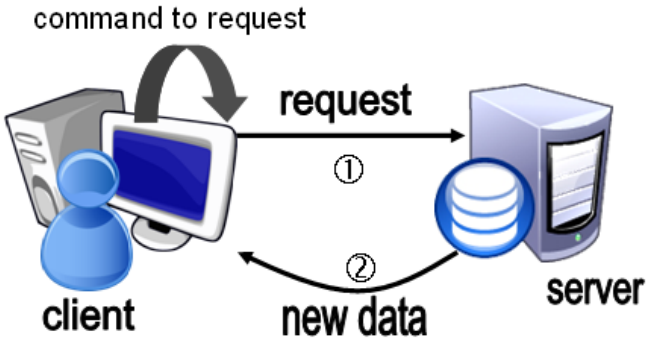


Figure 3.24. A small program embedded on the web browser

To make the plug-in works, the application has to be connected with the plug-in. The following PHP codes describe how the main program works. The initAjax() function will enable Ajax on the application. Then, initPlugin() function will create a connection to the plug-in. The Listen() function will receive a cue to trigger the client in making request to the server.

```
<?php
include “GreenAjax.php”;
$myClientApp = new GreenAjax();
$myClientApp->initAjax();
$myClientApp->initPlugin();
$myClientApp->Listen();
?>
```

The initPlugin() function above will listen the cue sent by the server. Whenever a cue received, the client will request using the following code to update the data. The readData() function will read the new data on the server and the Output() function will print the result. The request will be done in the background by using XMLHttpRequest, DOM, and JavaScript.

```
<?php
include “GreenAjax.php”;
$myDataApp = new GreenAjax();
$myDataApp->readData();
$myDataApp->Output();
?>
```

The GreenAjax.php that is included on every script above has several methods as seen below.

```
<?php
class GreenAjax{
  function RandomRefresh() { … }
  function CreateData() { … }
  function SendCue() { … }
  function initAjax() { … }
  function initPlugin() { … }
  function Listen() { … }
  function readData() { … }
  function Output() { … }
}
?>
```

After sending a cue, the clients who are still requesting to the server will be recorded by the server using their IP address. This mechanism makes the server work effectively in sending the cue to the clients on the next update. A Session Time can also be used to monitor the clients who are accessing the server.

IV. RESULTS AND ANALYSIS

Overall, this research examines the Green Ajax in four conditions, namely (1) Green Ajax compared to Classic Ajax using a variety of intervals time to request while the server does an update in unpredictable time, (2) Green Ajax compared to Classic Ajax using different intervals time to request that equals with updates time on the server, (3) Green Ajax compared to Classic Ajax while the server does an update using Fuzzy scheme, and (4) Green Ajax tested by web-based applications that command the server to send a cue to the client in the random time.

Some data has been collected in the experiments to see the benefit of Green Ajax compared to Classic Ajax. The number of update done by the web server will be recorded as update activities (Upd). The classic Ajax issues some requests based on the interval time in the client side. Those numbers of request will be counted as Request (Req). However, some of those requests that do not get any new data will be counted as wasting request (Wst). If the application gets a new reply, the reply will be counted as received (Rcv). However, sometimes the application does not get the complete data. Those data loss will be counted as Loss (Los). In Green Ajax approach, the server sends a signal to the client to give the information if there is a new data released. The activity will be recorded as Server (Srv). The result of experiment will be recorded on the table 4.1.

Table 4.1. Experiment Data Form

Data	**Green Ajax**	**Classic Ajax**				
		30"	***60"***	***5'***	***10'***	***15'***
Requests (Req)						
Received (Rcv)						
Wasting (Wst)						
Updates (Upd)						
Un-received (Los)						
Server (Srv)						

The table above shows a form to record the data of experiments using 30 seconds, 60 seconds, 5 minutes, 10 minutes, and 15 minutes as interval time for Classic Ajax. Other forms will be used for other interval time and other experiments in the research. From those data collection, the size of bandwidth consumption, size of data loss, Data Loss Percentage (DLP), and Successful Receptive Percentage (SRP) can be calculated. The formula of Data Loss Percentage (DLP) and Successful Receptive Percentage (SRP) can be seen in the figure below.

$$\text{Data Loss Percentage (DLP)} = \frac{\text{number of times data was not received timely (Los)}}{\text{number of times update data was activated (Upd)}} \times 100$$

$$\text{Successful Receptive Percentage (SRP)} = \frac{\text{number of times data has been received (Rcv)}}{\text{number of times requests had been issued (Req)}} \times 100$$

Figure 4.1. Formula of DLP and SRP

From the data of experiments and the calculations based on those formulas, the result of the research will be reviewed in the following discussion.

4.1. Web Traffic Reduction for Infrequent Update Application Using Green Ajax

In first experiment of this research, there are 22 random updates activity (Upd) in one hour issued by the web server. On the client side, the classic Ajax issues some requests based on the interval timer. The numbers of request will be counted as Request (Req). Some requests that do not get a new data will be counted as wasting request (Wst). If the application gets a new reply, the reply will be counted as received (Rcv). However, sometimes the application does not get the complete data. Some data loss will be counted as

Loss (Los). In Green Ajax approach, the server should signal the client to give the information if there is a new data. The activity will be recorded as Server (Srv). The data of the experiment within one hour can be seen on the table 4.2.

Table 4.2. Summary of Experiment #1 Using Green Ajax

Data	Green Ajax	Classic Ajax				
		30"	*60"*	*5'*	*10'*	*15'*
Requests (Req)	22	119	60	13	7	5
Received (Rcv)	22	22	19	13	7	5
Wasting (Wst)	0	97	41	0	0	0
Updates (Upd)	22	22	22	22	22	22
Un-received (Los)	0	0	2	9	15	17
Server (Srv)	22	0	0	0	0	0

From table above, the SRP of Green Ajax is 100%. There are 22 update activities on the server. Green Ajax received 22 updates and issued 22 times requests. It shows no wasting requests having been issued by Green Ajax. The Green Ajax shows 0% of DLP. There is no un-received data for 22 updates activity on the server. The chart of the comparison can be seen in Figure 4.2.

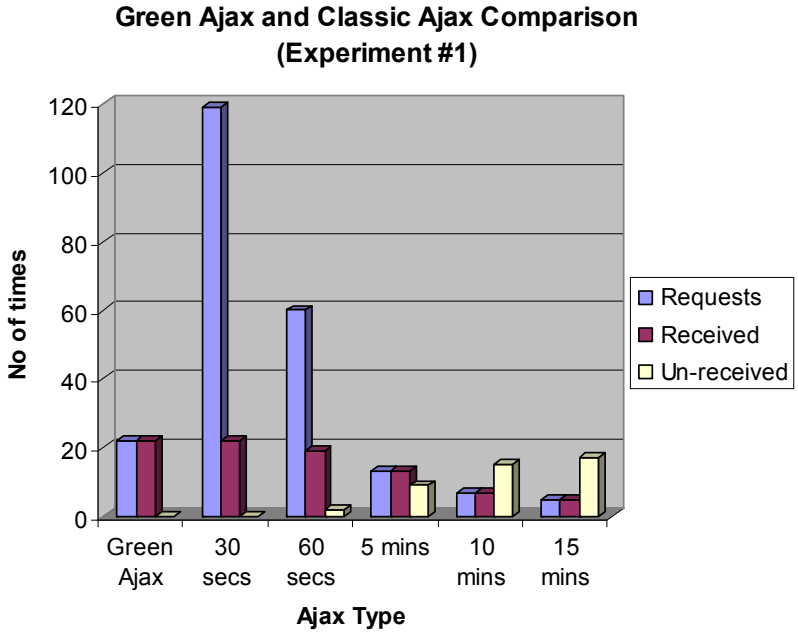


Figure 4.2. Comparison of Ajax Experiment's Result

The calculation of SRP and DLP of each Ajax experiment can be summarized on Table 4.3. Even though three classic Ajax applications (5 minutes, 10 minutes, and 15 minutes) show 100% of DLP, there are some data losses. Although the classic Ajax using interval timer of 30 seconds has 0% of DLP, it has 18.49% of SRP only. Green Ajax has the best result on both of results. It has 100% SRP and 0% DLP.

Table 4.3. SRP and DLP on Experiment #1

Results	**Green Ajax**	**Classic Ajax**				
		30"	***60"***	***5'***	***10'***	***15'***
SRP (%)	100.00	18.49	31.67	100.00	100.00	100.00
DLP (%)	0.00	0.00	9.09	40.91	72.73	77.27

Based on Table 4.3, chart of SRP and DLP can be shown in Figure 4.3.

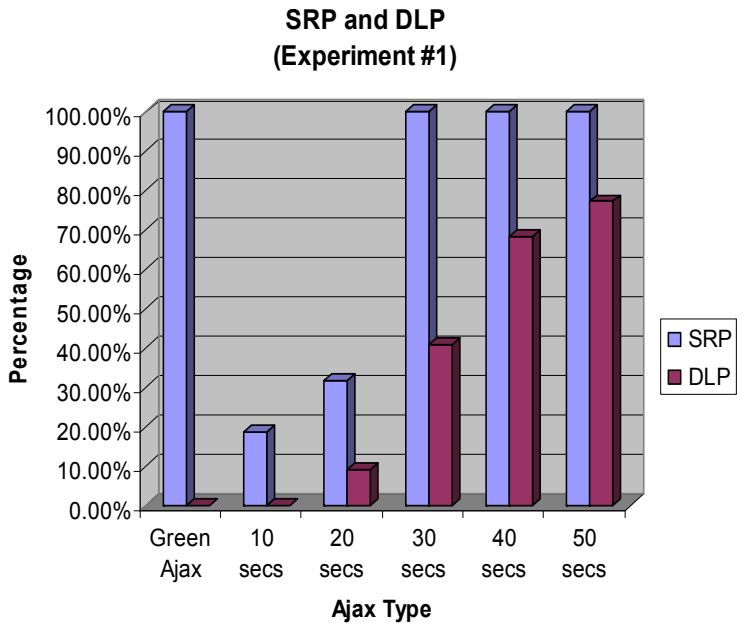


Figure 4.3. SRP and DLP of Experiment #1

The bandwidth consumption for each type of Ajax is shown on the following table. The bandwidth consumption of Green Ajax is 2,222 bytes without data loss. Even though three Ajax applications (5 minutes, 10 minutes, and 15 minutes) show less consumption than Green Ajax, they do not have the complete data. The higher interval of application has bigger data loss.

Table 4.4. Bandwidth Consumption on Experiment #1

Results	Green Ajax	Classic Ajax				
		30"	*60"*	*5'*	*10'*	*15'*
Bandwidth	2,020	64,020	32,020	21,320	16,020	12,820
Data Loss	0	0	0	0	0	0

Based on Table 4.4, the chart of bandwidth consumption and data loss can be shown below in Figure 4.4. The bars of bandwidth consumption on the classic Ajax applications look significantly different compared to the Green Ajax. However, the longer interval of classic Ajax application has smaller bar of bandwidth consumption but it has taller bar of data loss. In this experiment, the graphic representation of Green Ajax looks more ideal then others.

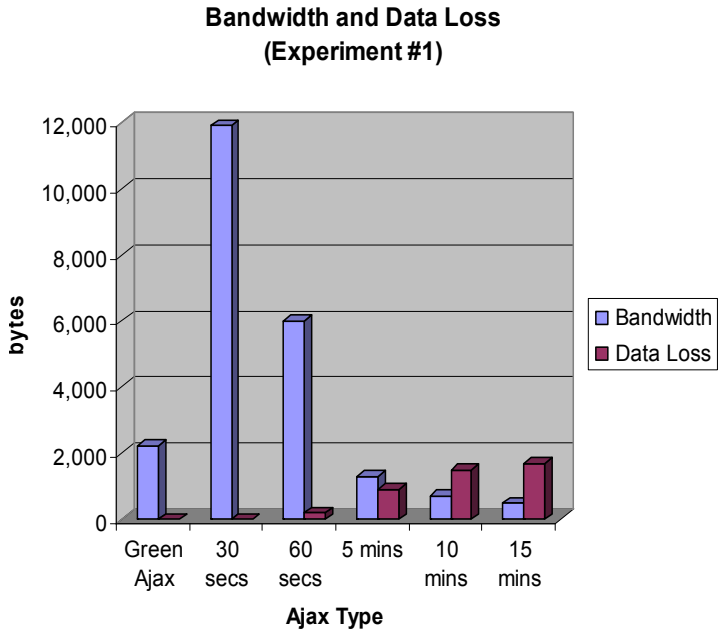


Figure 4.4. Bandwidth and Data Loss

To get the valid results, another experiment using different scenario is created. From the experiment #2 within one hour using different interval time scenario, there are 27 updates activity (Upd) on the web server which is in random time. The scenario of this experiment uses the shorter interval time than the previous experiment to update the data on the classic Ajax. Those intervals are 10 seconds, 20 seconds, 30 seconds, 40 seconds, and 50 seconds.

The result from this experiment can be shown below.

Table 4.5. Summary of Experiment #2 Using Green AJAX

Data	Green Ajax	Classic Ajax				
		10"	*20"*	*30"*	*30"*	*50"*
Requests (Req)	27	365	181	122	92	73
Received (Rcv)	27	26	25	25	25	24
Wasting (Wst)	0	339	156	97	67	49
Updates (Upd)	27	27	27	27	27	27
Un-received (Los)	0	1	2	2	2	3
Server (Srv)	22	0	0	0	0	0

From Table 4.5, Green Ajax still shows 100% SRP because there are 27 updates activity on the server. Green Ajax received 27 updates without any wasting requests. Green Ajax shows 0% DLP because there is no un-received data for 27 updates activity on the server. Compared to the other Ajax applications, the Green Ajax has the ideal results.

Based on data of the above table, the chart of the comparison can be shown in Figure 4.5. All kinds of Ajax application receive almost all of the data but they have different significant number of requests. The bars of request on all of classic Ajax applications are much higher than the bar of request on the Green Ajax.

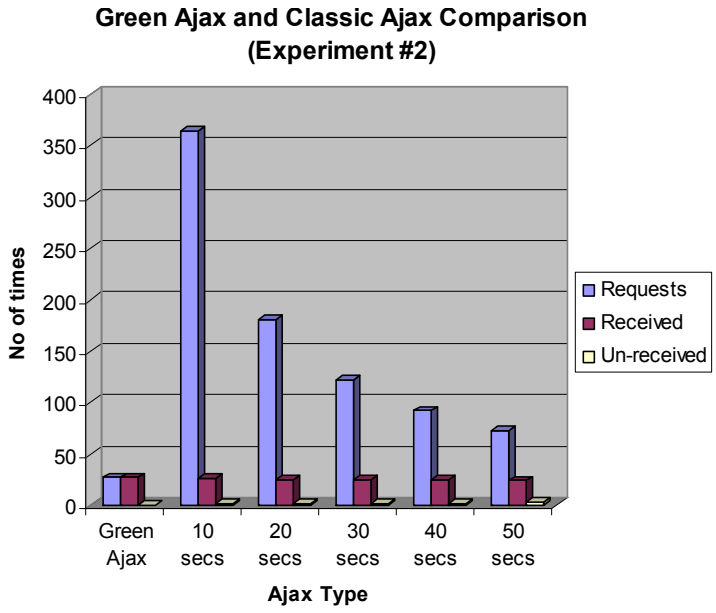


Figure 4.5. Comparison of Ajax Experiment's Result

The calculation of SRP and DLP of each Ajax experiment can be summarized on the following table. Green Ajax has the best result on both of results. It has 100% SRP and 0% DLP. The other classic Ajax applications show low SRP and get some loss data.

Table 4.6. SRP and DLP on Experiment #2

Results	Green Ajax	Classic Ajax				
		10"	*20"*	*30"*	*30"*	*50"*
SRP (%)	100.00	7.12	13.81	20.49	27.17	32.88
DLP (%)	0.00	3.70	7.41	7.41	7.41	11.11

Based on Table 4.6, chart of SRP and DLP can be shown in Figure 4.6.

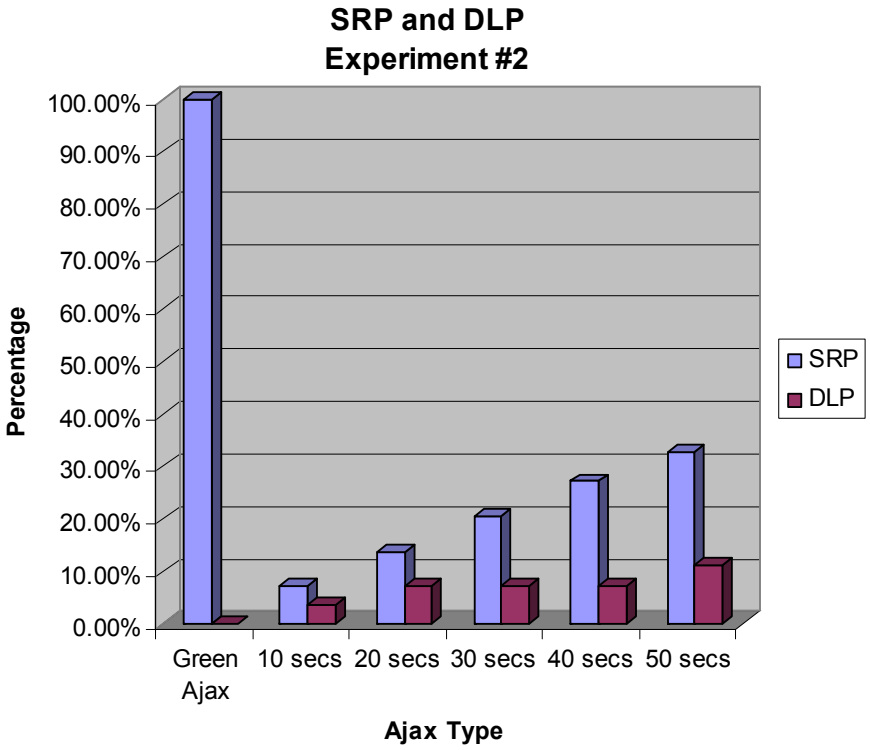


Figure 4.6. SRP and DLP of Experiment #2

The bandwidth consumption for each type of Ajax is shown on the following table. The bandwidth consumption of Green Ajax is 2,727 bytes without data loss. The other Ajax applications consume more bandwidth than the Green Ajax. Those Classic Ajax applications get some data lost 100 bytes, 200 bytes, and 300 bytes.

Table 4.7. Bandwidth Consumption on Experiment #2

Results	Green Ajax	Classic Ajax				
		10"	*20"*	*30"*	*30"*	*50"*
Bandwidth	2,727	36,500	18,100	12,200	9,200	7,300
Data Loss	0.00	100	200	200	200	300

Based on Table 4.7, chart of bandwidth consumption and data loss can be shown in Figure 4.7.

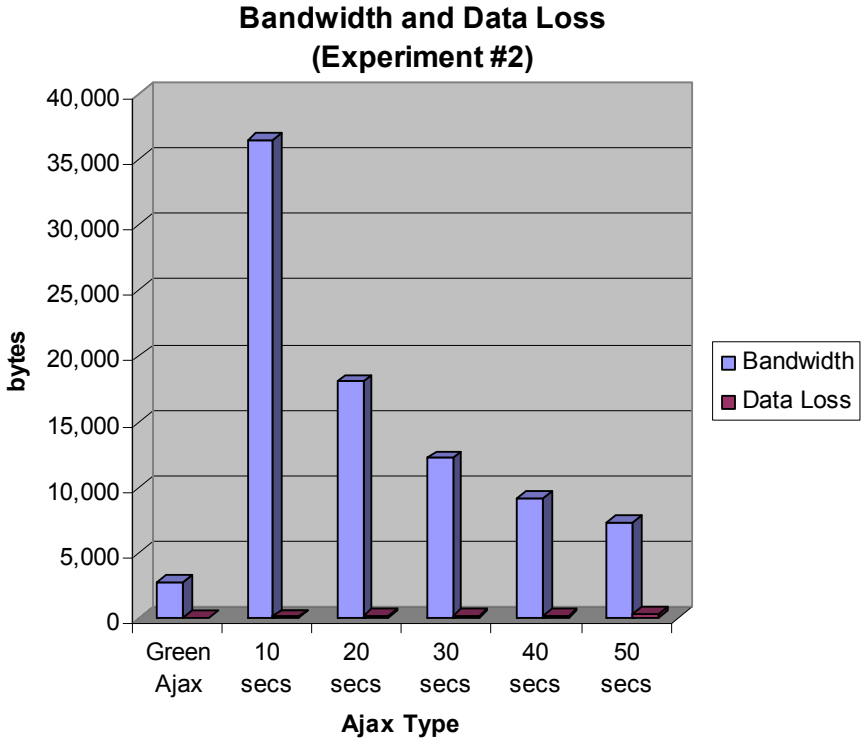


Figure 4.7. Bandwidth and Data Loss

The scenario of the experiment #3 is changing the interval time for data updating. The classic Ajax of the experiment #3 uses longer interval than the two above experiments. The interval times of the experiment are 10 minutes, 20 minutes, 30 minutes, 40 minutes, and 50 minutes.

Table 4.8. Summary of Experiment #3 Using Green AJAX

Data	Green Ajax	Classic Ajax				
		10'	*20'*	*30'*	*40'*	*50'*
Requests (Req)	24	7	4	3	2	2
Received (Rcv)	24	7	4	3	2	2
Wasting (Wst)	0	0	0	0	0	0
Updates (Upd)	24	24	24	24	24	24
Un-received (Los)	0	17	20	21	22	22
Server (Srv)	24	0	0	0	0	0

From Table 4.8, Green Ajax shows 100% SRP. There are 24 updates activity on the server. Green Ajax received 24 updates without wasting requests. Green Ajax shows 0% DLP. There is no un-received data for 24 updates activity on the server. The chart of the comparison can be shown in Figure 4.8.

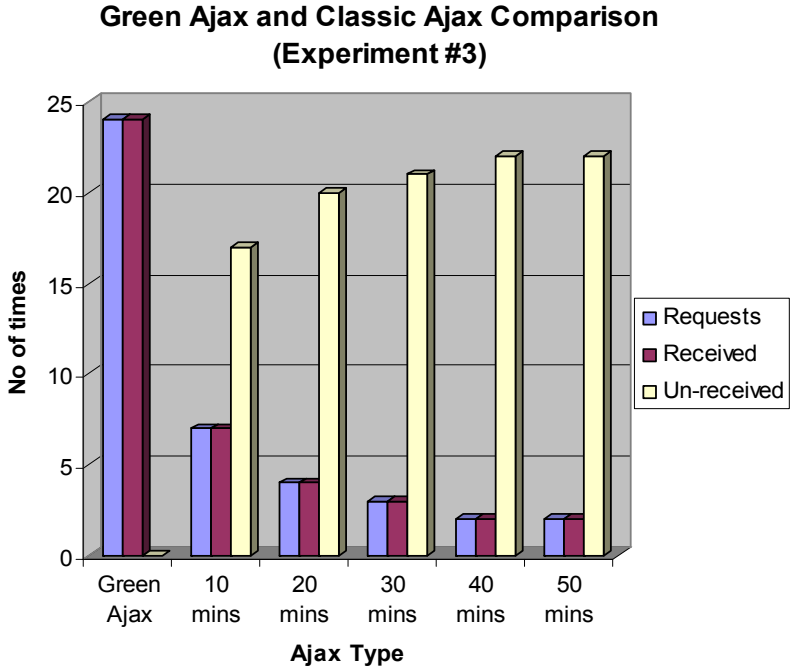


Figure 4.8. Comparison of Ajax Experiment's Result

The summary of SRP and DLP of each Ajax experiment can be shown on below. Even though all of the classic Ajax applications show 100% SRP, there are some data losses. The greater interval creates bigger loss. Green Ajax still has the best result on both of results. It shows 100% SRP and 0% DLP.

Table 4.9. SRP and DLP on Experiment #3

Results	Green Ajax	Classic Ajax				
		10'	*20'*	*30'*	*40'*	*50'*
SRP (%)	100.00	100.00	100.00	100.00	100.00	100.00
DLP (%)	0.00	70.83	83.33	87.50	91.67	91.67

Based on Table 4.9, chart of SRP and DLP can be shown in Figure 4.9.

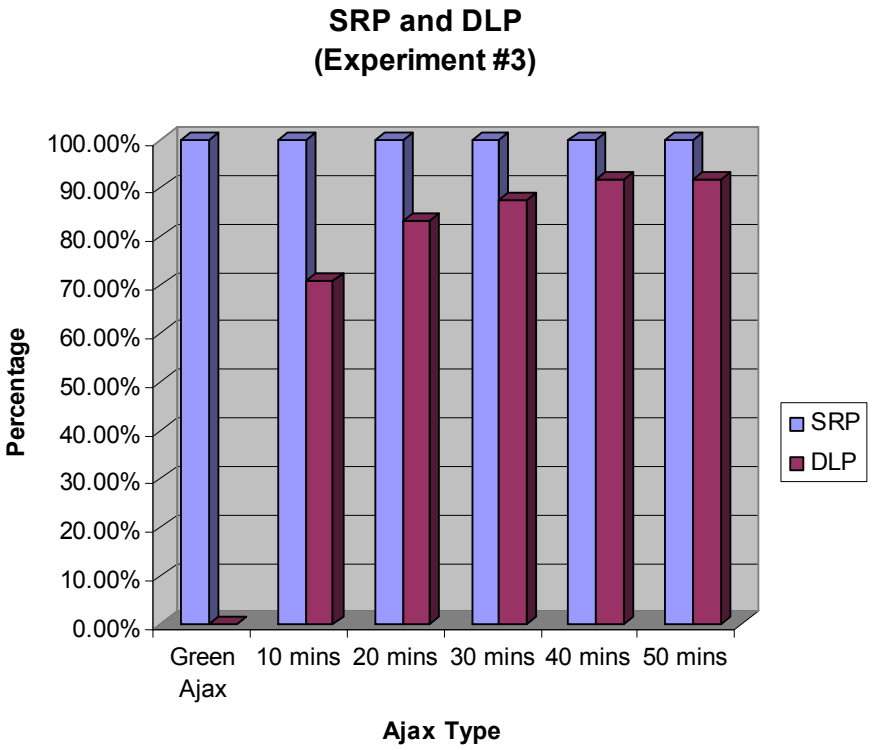


Figure 4.9. SRP and DLP of Experiment #3

The bandwidth consumption for each type of Ajax is shown on the following table. The bandwidth consumption of Green Ajax is 2,424 bytes without data loss. Even though all of the classic Ajax applications consume less bandwidth than Green Ajax, they get some data loss and the data loss becomes greater for the longer interval.

Table 4.10. Bandwidth Consumption on Experiment #3

Results	Green Ajax	Classic Ajax				
		10'	*20'*	*30'*	*40'*	*50'*
Bandwidth	2,424	700	400	300	200	200
Data Loss	0	1,700	2,000	2,100	2,200	2,200

Based on Table 4.10, chart of bandwidth consumption and data loss can be shown in Figure 4.10.

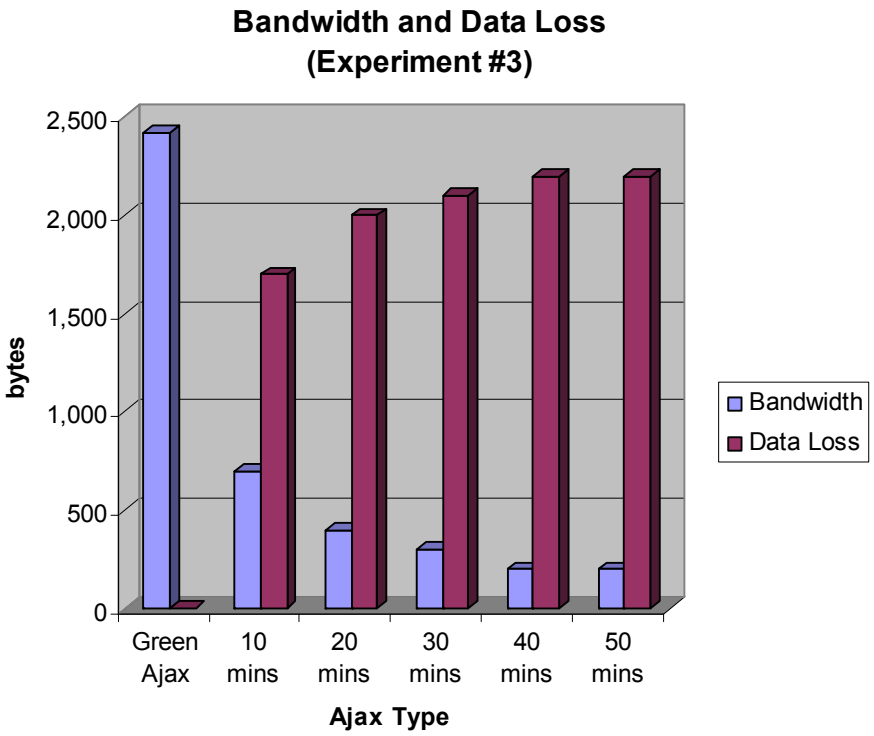


Figure 4.10. Bandwidth and Data Loss

4.2. Trade-off Analysis for Web Application Using Green Ajax

In the first experiments using the high frequency update scenario, the web server will update every 5 seconds, the classic Ajax will keep request every 5 seconds, and the Green Ajax will request on condition that the application receives a signal from the web server. Other experiments of high frequency update will have interval time of 10 seconds, 15 seconds, 20 seconds, and 25 seconds.

Table 4.11. Scenario of Experiments

Group #	Scenario	
	Situations	***Interval Time***
1	High Frequency Update	5 seconds – 25 seconds
2	Medium Frequency Update	30 seconds – 15 minutes
3	Low Frequency Update	30 minutes – 5 hours

On the medium frequency update this research will examine 30 seconds, 1 minute, 5 minutes, 10 minutes, and 15 minutes of interval time of data updating. For the low frequency update, it will test the updating data on the server every 30 minutes, 1 hour, 2 hours, 3 hours, and 5 hours.

From the experiment of the first part, the results can be seen on Table 4.12. Even though the classic Ajax uses the same interval time as the server' interval time to update, its bandwidth consumption is still higher than the Green Ajax. Some requests of the classic Ajax do not get any new data. The Green Ajax is still the best but the longer interval creates shorter margin bandwidth consumption between The Green Ajax and the classic Ajax.

Table 4.12. Bandwidth Consumption (in bytes) on High Frequency Update

Scenario[a]	**Interval Time**				
	5 secs	***10 secs***	***15 secs***	***20 secs***	***25 secs***
Classic Ajax	59,500	30,200	21,200	16,100	12,900
Green Ajax	50,803	28,583	19,897	15,150	12,322
Margin (%)	17.12%	5.66%	6.55%	6.27%	4.69%

a. Experiment time: 1 hour

The decreasing margin of bandwidth consumption between both Ajax can be seen in Figure 4.11.

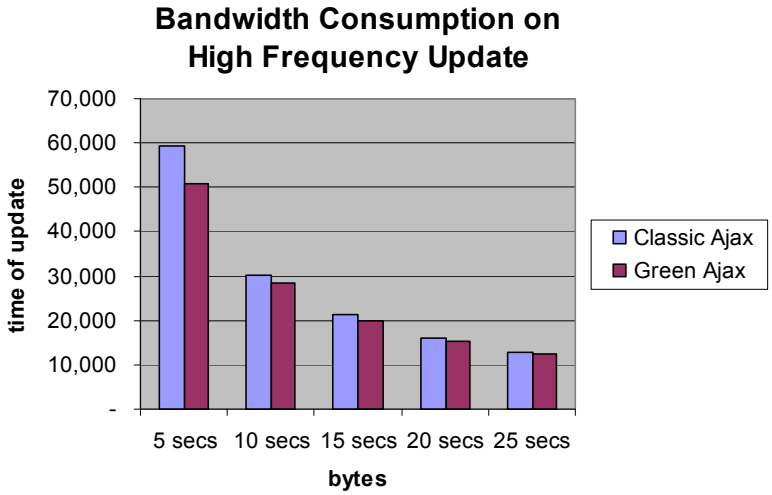


Figure 4.11. Bandwidth Consumption on High Frequency Update

From the previous research (Sanjaya, 2010b), the Successful Receptive Percentage (SRP) of each application can be seen below. The formula in Figure 4.1 will show the percentage of the accurate requests to the web server.

On Table 4.13, SRP of the classic Ajax is not fully 100% because some requests on the first, second, and third interval time do not get any new data. It means there are some wasting requests.

Table 4.13. SRP on High Frequency Update

Scenario[a]	**Interval Time**				
	5 secs	***10 secs***	***15 secs***	***20 secs***	***25 secs***
Classic Ajax	99.16%	99.00%	98.36%	100.00%	100.00%
Green Ajax	100.00%	100.00%	100.00%	100.00%	100.00%

a. Experiment time: 1 hour

The SRP of the Green Ajax applications are 100%. All of the requests from the Green Ajax applications to the web server get all the new data. The Green Ajax applications do not waste their requests.

The Data Loss Percentage (DLP) of each application can be seen below. On Table 4.14, the DLP of both applications are 0%. There are no data loss on the classic Ajax and the Green Ajax applications. All applications get all the new data from the server.

Table 4.14. DLP on High Frequency Update

Scenario[a]	**Interval Time**				
	5 secs	***10 secs***	***15 secs***	***20 secs***	***25 secs***
Classic Ajax	0%	0%	0%	0%	0%
Green Ajax	0%	0%	0%	0%	0%

a. Experiment time: 1 hour

Table 4.15 shows the results from the second part of experiments. Some classic Ajax applications show smaller bandwidth consumption than the Green Ajax. However, there are no significant margin of bandwidth consumption between the classic Ajax and the Green Ajax. The margins of the bandwidth consumption are below than one percent.

Table 4.15. Bandwidth Consumption (in bytes) on Medium Frequency Update

Scenario[a]	Interval Time				
	30 secs	***1 mins***	***5 mins***	***10 mins***	***15 mins***
Classic Ajax	59,700	30,000	6,100	3,000	2,100
Green Ajax	59,792	29,997	6,060	3,030	2,121
Margin (%)	(0.15%)	0.01%	0.66%	(0.99%)	(0.99%)

a. Experiment time: 5 hour

The bandwidth consumption of both applications can be seen in Figure 4.12. Because of little margins, the bars inside the chart do not show the significant differences.

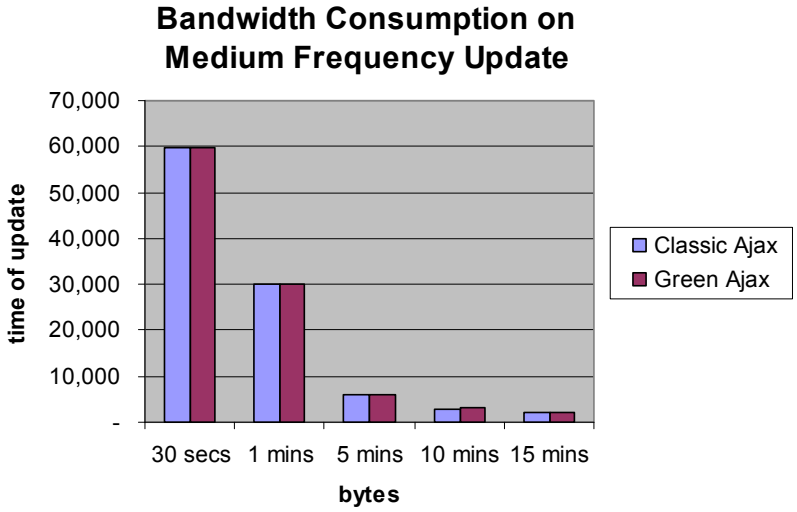


Figure 4.12. Bandwidth Consumption on Medium Frequency Update

Even though the differences of margins between both applications are insignificant, the Green Ajax shows the best performance on the SRP. On Table 4.16, some of classic Ajax applications do not have 100% SRP. Some requests from the classic Ajax do not get the new data from the web server. However, the Green Ajax applications get all the new data from their requests.

Table 4.16. SRP on Medium Frequency Update

Scenario[a]	Interval Time				
	30 secs	***1 mins***	***5 mins***	***10 mins***	***15 mins***
Classic Ajax	99.16%	99.00%	98.36%	100.00%	100.00%
Green Ajax	100.00%	100.00%	100.00%	100.00%	100.00%

a. Experiment time: 5 hour

On the medium frequency, both applications show 0% of DLP. The experimental results of 30 seconds, 1 minute, 5 minutes, 10 minutes, and 15 minutes have the same result as listed on Table 4.14. There are no data lost when the applications work. The classic Ajax and the Green Ajax get all the new data from the web server.

Similar with the second part of experiment, the third part also shows insignificant margins. However, in this case, the classic Ajax applications show the best performance. Their bandwidth consumption is smaller than the Green Ajax. On Table 4.17, the bandwidth consumption on the higher interval time of the classic Ajax is close to the Green Ajax. The average margins on both applications are below than one percent. The differences are found only on the signal size for each data updating on the web server.

Table 4.17. Bandwidth Consumption (in bytes) on Low Frequency Update

Scenario[a]	**Interval Time**				
	30 mins	***1 hr***	***2 hrs***	***3 hrs***	***5 hrs***
Classic Ajax	1,100	600	300	200	200
Green Ajax	1,111	606	303	202	202
Margin (%)	(0.99%)	(0.99%)	(0.99%)	(0.99%)	(0.99%)

a. Experiment time: 5 hour

In Figure 4.13, there are no significant differences on the bars.

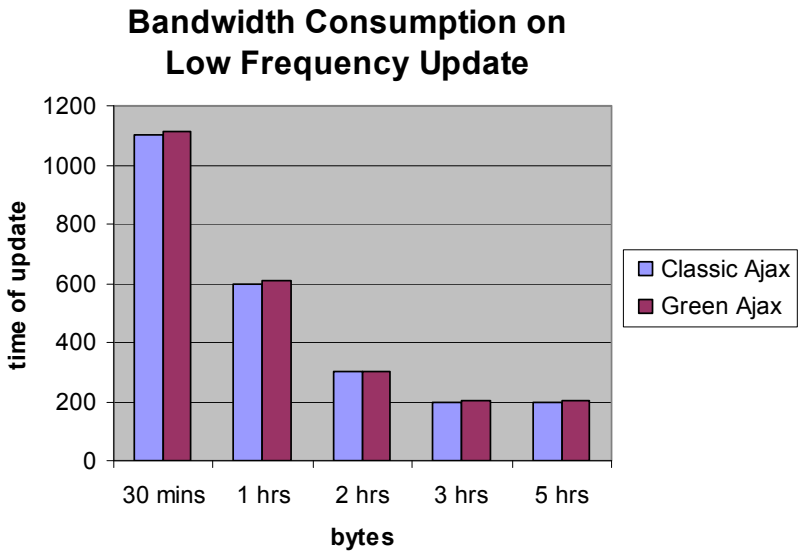


Figure 4.13. Bandwidth Consumption on Low Frequency Update

From Table 4.18, the classic Ajax application and the Green Ajax applications get 100% SRP. There are no wasting requests on both applications. They do effective requests to the server because all requests get the new data from the web server.

Table 4.18. SRP on Low Frequency Update

Scenario[a]	Interval Time				
	30 mins	***1 hr***	***2 hrs***	***3 hrs***	***5 hrs***
Classic Ajax	100.00%	100.00%	100.00%	100.00%	100.00%
Green Ajax	100.00%	100.00%	100.00%	100.00%	100.00%

a. Experiment time: 5 hour

The DLP results of both applications on the Low Frequency Update experiments are also 0%. The result of 30 minutes, 1 hour, 2 hours, 3 hours, and 5 hours are the same as listed on Table 4.14. All applications receive all of data on the server.

4.3. Mobile Traffic Evaluation for Fuzzy-Based Application Using Green Ajax

The ranges of time to update in the previous experiment are usually used in the theory. The real condition of each range of time to update sometimes overlaps. In this research, the previous scenario of experiments will be adjusted to evaluate the Green Ajax by using Fuzzy-based application. The adjusted scenario can be seen on Table 4.19.

Table 4.19. Adjusted Scenario of Experiments

Group #	Scenario	
	Situations	***Interval Time***
1	High Frequency Update	5 seconds – 60 seconds
2	Medium Frequency Update	15 seconds – 30 minutes
3	Low Frequency Update	15 minutes – 5 hours

Compared to the previous experiment on the desktop application, it has more real situation. In addition, the assumptions of environment are zero delay between server and the clients, the speed up to 100 mbps (Lin et al, 2009), close to access point, and the clients have Ajax support on its web browser.

In this experiment, the range of time displayed in Figure 4.14 is used. The high frequency update application has interval time for data updating ranging from 5 seconds to 60 seconds. The medium frequency update has interval time to update from 15 seconds to 1,800 seconds or 30 minutes. The low medium frequency update has interval time since 900 seconds (15 minutes) to 18,000 seconds (5 hours).

In the figure shown below, there are overlapping area between High and Medium Frequency Update and also between Medium and Low Frequency Update. These conditions are common in the real situations.

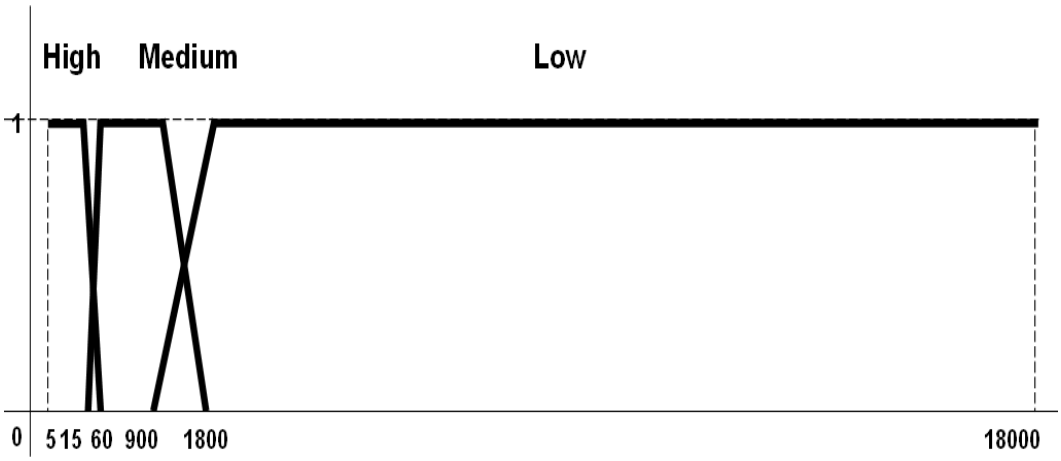


Figure 4.14. Range of time in the real situation

The membership of each situation in Figure 4.14 is listed in equation below. The range of time is also assumed to be three basic types (high, medium and low) as shown below.

$$High = \begin{cases} 1, & 5 \le x \le 15 \\ \dfrac{x - 15}{45}, & 15 < x \le 60 \end{cases}$$

$$Medium = \begin{cases} \frac{x+15}{45}, & 15 < x < 60 \\ 1, & 60 \leq x \leq 900 \\ \frac{x-900}{900}, & 900 < x < 1800 \end{cases}$$

$$Low = \begin{cases} \frac{x+900}{900}, & 900 < x < 1800 \\ 1, & 1800 \leq x \leq 18000 \end{cases}$$

Experiments based on the above ranges of time produce the results as shown on table 4.20. In the results, bandwidth consumption of Green Ajax is the lowest. Moreover, the margin of bandwidth consumption between Classic Ajax and Green Ajax is significant. It strongly confirms Green Ajax is still the most recommended approach for high frequency update in Fuzzy-based Application. In this experiment, overall results show Green Ajax is suitable to apply in Ajax-based applications.

Table 4.20. Bandwidth Consumption (in bytes) on High Frequency Update in Fuzzy-Based Application

Interval Time	**Bandwidth Consumption**		
	Classic Ajax	***Green Ajax***	***Margin***
5 secs	545,243	529,493	15,750
10 secs	271,170	269,670	1,500
15 secs	182,543	181,043	1,500
20 secs	136,343	135,593	750
25 secs	109,830	109,080	750
30 secs	87,509	86,793	716

From the chart shown in Figure 4.15, the bandwidth consumption of Green Ajax is lower significantly than Classic Ajax, especially in the lower interval time. Overall, Classic Ajax has higher bar graphs than Green Ajax. It indicates Classic Ajax consumes higher bandwidth than Green Ajax on the Fuzzy-based application with high frequency updates.

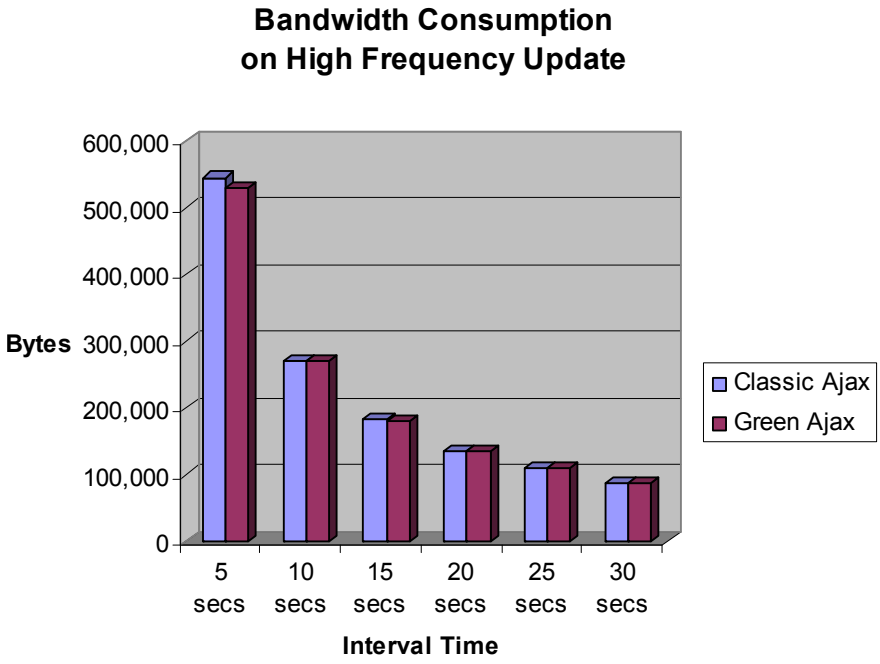


Figure 4.15. Range of time in the real situation

The evaluation is also done on the application with medium frequency update. The results of experiments can be demonstrated on table 4.21. In many experiments, Green Ajax shows that it consumes smaller bandwidth than Classic Ajax. Only in few cases, Classic Ajax is a bit more efficient than Green Ajax. The large proportion of Green Ajax in the experimental results above indicates that the Fuzzy-based applications are more suitable to Green Ajax.

Table 4.21. Bandwidth Consumption (in bytes) on Medium Frequency Update

Interval Time	**Bandwidth Consumption**		
	Classic Ajax	***Green Ajax***	***Margin***
15 secs	182,543	181,043	1,500
20 secs	136,343	135,593	750
25 secs	109,830	109,080	750
30 secs	87,509	86,793	716
3 mins	14,458	13,742	716
5 mins	10,119	9,403	716
10 mins	5,063	5,063	0
15 mins	3,616	3,616	0
30 mins	1,074	1,085	-11

The comparison of bandwidth consumptions between Classic Ajax and Green Ajax are displayed in Figure 4.16. Majority, Green Ajax has lower bar charts than Classic Ajax. This graph shows the Green Ajax is more efficient in bandwidth consumption. Even though in the last experiment whose interval time is 30 minutes Classic Ajax has lower bar graph, but it is not too significant.

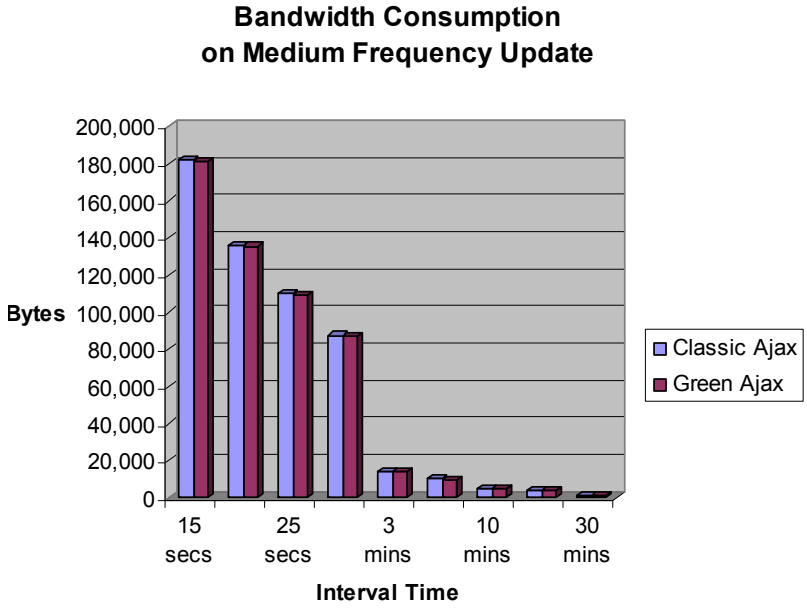


Figure 4.16. Bandwidth Consumption on Medium Frequency Update

Contrary with two previous experiments, evaluations on the applications with low frequency update show the superiority of Classic Ajax compared to Green Ajax. Classic Ajax uses smaller bandwidth than Green Ajax. It shows Classic Ajax is suitable for application with low frequency update even though the margin is not significant.

Table 4.22. Bandwidth Consumption (in bytes) on Low Frequency Update

Interval Time	**Bandwidth Consumption**		
	Classic Ajax	***Green Ajax***	***Margin***
15 mins	3,616	3,616	0
30 mins	1,074	1,085	-11
1 hr	573	579	-6
2 hrs	286	289	-3
3 hrs	191	193	-2
5 hrs	191	193	-2

In Figure 4.17, the bar graphs of Green Ajax are higher than Classic Ajax. The graphs indicate the Classic Ajax consumes lower bandwidth. However, the difference between two bars is very small.

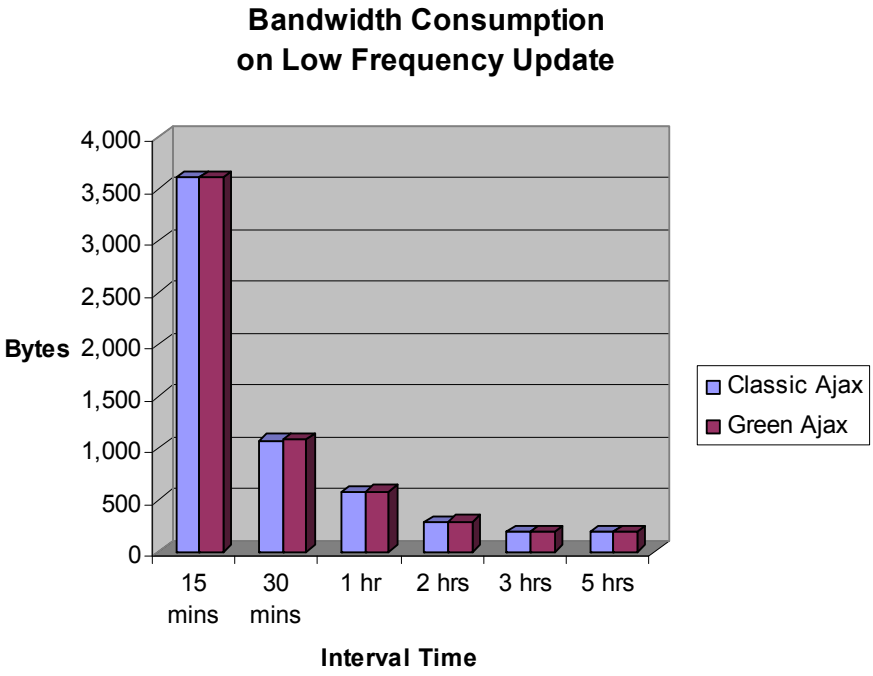


Figure 4.17. Bandwidth Consumption on Low Frequency Update

The complete result in this research can be seen on table 4.23. Green Ajax is recommended to be implemented on the Fuzzy-based application with high frequency update. Moreover, majority of Fuzzy-based application with medium frequency update, Green Ajax is still suitable to be used. However, Classic Ajax is better than Green Ajax on handling data updating with low frequency in the Fuzzy-based application.

Table 4.23. Final Results of Fuzzy-based Application Experiment

Group #	**Scenario**	
	Situations	***Better Approach***
1	High Frequency Update	Green
2	Medium Frequency Update	Majority Green
3	Low Frequency Update	Classic

4.4. Implementation on the Local Area Network using Randomized Cue Applications

In this experiment, Randomized Cue Applications are used as a simulator to produce an update at the random time. Time decision of the next update is based on the random value generated by the application. The results obtained can describe the actual conditions when Green Ajax is used. Then, the bandwidth saving will be calculated using Mozilla Firefox 3.5.14 and Firebug 1.5.4.

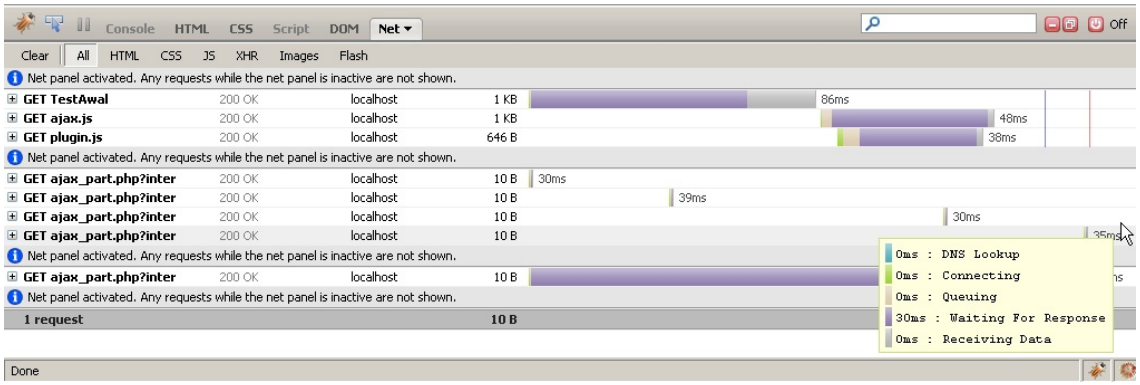


Figure 4.18. Firebug 1.5.4 on Mozilla Firefox 3.5.14

The bandwidth consumption of Randomized Cue Applications will be compared with the classic Ajax Application's bandwidth consumption. The classic Ajax Application uses 2 minutes TTR to request the update data from the server. From the results, the number of times data that had been received by the client (Rcv), and the number of times requests that had been issued by the server (Req) were compared to get the Successful Receptive Percentage (SRP) (Sanjaya, 2010a). Data Loss Percentage (DLP) were calculated by comparing the number of times data was not received timely by the client (Los) and the number of times update data was activated by the server (Upd).

Table 4.24. SRP and DLP

Results[a]	Green Ajax	Classic Ajax
SRP (%)	100.00	69.54
DLP (%)	0.00	18.60

a. Experiment time: 5 hours

Table 4.24 shows the SRP of Green Ajax gets 100%. It shows no wasting requests having been issued by Green Ajax. However, classic Ajax only gets 69.54% data. It shows there are 30.56% requests to the server but do not get new data. Another result, DLP of Green Ajax is 0% or all data has been received by Green Ajax. However, the classic Ajax gets 18.60% DLP. It shows there are 18.60% un-received data from server by classic Ajax.

The result of 5 hours experiment can be seen on Table 4.25. It shows that bandwidth consumption of Green Ajax is smaller than classic Ajax. Moreover, there are 14,400 bytes data transferred by server which are not received by the clients. It is different from Green Ajax that does not lose any data.

Table 4.25. Bandwidth Consumption and Data Loss (in bytes)

Results[a]	Green Ajax	Classic Ajax
Bandwidth	77,529	90,729
Data Loss	0.00	14,400

a. Experiment time: 5 hours

The above results show the benefit of Green Ajax Implementation on the local area network has been successfully proved by using Randomized Cue Applications.

5. CONCLUSION AND RECOMMENDATION

In this chapter, the research summary, conclusions, contribution to knowledge, and recommendations for further research are discussed as follows.

5.1. Research Summary

Basically, this research sees the weakness of the Classic Ajax that is very influential on the bandwidth consumption when it is used to display the real-time data from the server. If the time interval for the request is reduced, the frequency of requests to the server will be more frequent (High Frequency). The request will be wasted if the new data from the server is not obtained. If the time interval for the request is increased, the frequency of requests to the server will be less frequent (Low Frequency). Some data may be lost because there is some data on a server that is not requested by the client.

This research sees the need for sending the data from the server to the client every time there is an update. However, if the content delivery is centralized on the server, it will cause the accumulation of load at one point. In addition, there is the possibility of malicious content that will be received by the client.

Green Ajax irons the problem by sending the signal from the server to the client when data is updated. The client still has to request to get the data from the server, not only depends on the content delivery from the server as discussed above. In addition, the received content is absolutely the part of the web page which is requested by the client. It minimizes the delivery of another content which is not related to the client's request that may be harmful to the client.

Green Ajax tests not only on the server which updates the data in random time, but also on the server that performs scheduled updates. Fuzzy-based application is also used on the

server to test the range of time to update that is often overlapping. Furthermore, a simulation was created to implement and test the Green Ajax via the delivery of signals in random time to the client as a notification when a new data exists on the server.

Based on several experiments, Green Ajax is able to save more bandwidth compared to the Classic Ajax, especially when the server updates periodically using short interval time (High Frequency) and medium interval (Medium Frequency). Green Ajax can be used when the server updates using long interval time even though there is insignificant loss compared to Classic Ajax.

5.2. Conclusions

As listed on the objective of this research in chapter one, Green Ajax could be the solution for the web application to display interactive contents using small bandwidth.

1. The concept is able to reduce the web traffic on the web application by signaling the client when the server updates the data. This concept replaces the use of interval time to request the data from web server.
2. The approach to provide a real time data updating on the web application should be designed as follows. The application of data updating should trigger the signal to the client when the data has been saved. Even though the time for updating data on the server side is random and unpredictable, the signal can be used by the client to detect the updating activity. By detecting the updating activity, the client can request the latest data at the right time.
3. To provide the capability of client on receiving a signal from the web server, the client should have a plug-in embedded on the web browser or small program like an applet embedded on the web application to act as a virtual server. By providing the plug-in or small application, the client has capability to receive the signal from the server without

client's request.

4. The implementation of Green Ajax in the web application discussed in chapter four, which are the signal produced by the web application and a virtual server embedded on the client, could be the model to support Green Ajax.
5. The implementation of Green Ajax in the web programming is successful. The new programming approach in PHP that is object oriented programming can also be used to implement Green Ajax in the web programming. The code in chapter three and appendix could be used to produce a random signal from the web server and receive the signal on the client side.

5.3. Contribution to Knowledge

Optimization of bandwidth in web programming is essential, especially for web-based applications that need the latest data from the server. However, by using existing approaches, namely Classic Ajax using Time to Refresh (TTR) to request data periodically, the bandwidth consumption often becomes greater than the requirement or received incompletely by the client.

Green Approach Ajax is a new alternative in web-based programming using Ajax. Previously, programmers needed Time to Refresh (TTR) to update web content periodically, but now the web content can be updated quickly when there is an update on the server. By using Green Ajax, a signal from the server to the client will make the client request the latest data from the server immediately. It will not only reduce the bandwidth consumption and avoid the data loss, but also display the information on the client faster than previous.

5.4. Recommendation for Further Research

Green Ajax testing is still limited to a Local Area Network using wired or wireless. In

the further research, the scope of computer network will get more attention not only on a wired and wireless Local Area Network but also on a larger network. The existence of proxy server and router also needs to be included in the future research. Making a tunnel between client and web server in the signal transmission is required if the client is behind a proxy server or router.

The identification of clients should not merely depend on the IP address listed on the server that is sometimes not associated with a real IP address on the clients. However, using the current HTTP Protocol should be prioritized because it has been used widely. As a result, the benefits of Green Ajax can be implemented widely on the computer network.

Moreover, adoption of Green Ajax on the web-based mobile application is recommended to reduce the mobile traffic. Adaptation of environment on mobile technologies has to get more attention besides a network bottleneck which probably occurs when numerous mobile gadgets access the network at the same time continuously.

APPENDIX A

PUBLICATIONS

Several publications published during the research are the followings:

Title: **Web Traffic Reduction for Infrequent Update Application Using Green Ajax**

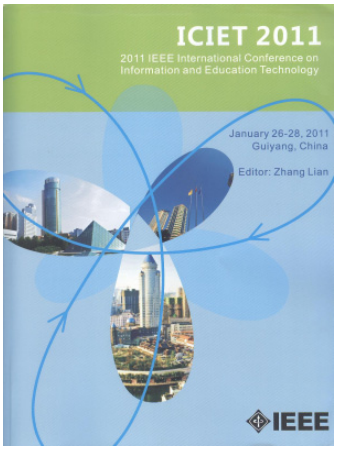
Proceeding of 2010 2nd IEEE International Conference on Information Management and Engineering (ICIME)

Chengdu, China, April 16-18, 2010

Title: **Trade-off Analysis for Web Application Using Green Ajax**

Proceeding of 2010 5th IEEE International Conference on Management of Innovation and Technology (ICMIT)

Singapore, June 2-5, 2010



Title: **Mobile Traffic Evaluation for Fuzzy-Based Application Using Green Ajax**

Proceeding of 2011 IEEE International Conference on Information and Education Technology (ICIET)

Guiyang, China, January 26-28, 2011

Title: **Green Ajax Implementation on the Wireless Local Area Network using Randomized Cue Applications**

International Journal of Information and Education Technology, Vol. 1, No. 1, April 2011

IACSIT, Singapore

Web Traffic Reduction for Infrequent Update Application Using Green Ajax

Ridwan Sanjaya
Graduate School of Information Technology
Assumption University
Bangkok, Thailand
ridwan.sanjaya@gmail.com

Abstract— **The classic web applications usually need a lot of bandwidth to provide the rich user interfaces. Since Ajax [8] was introduced, it has reduced the web server load and the data transfer to/from users' computers. By using Ajax, only a specific part in the web page can be requested to the web server [6]. However, it still cannot provide the real time data updating. The common approach to provide the real time data updating uses a timer to request a new data from the web server periodically [3, 4]. But, the requests sometimes do not get any new data. If the interval time to renew the data is too long, the data updating will come to the client late and some data received by the client may be lost.**

The proposed approach to solve the problem is creating an Ajax application which can receive a signal from the web server. The received signal will trigger the web application to renew the data from the web server. By limiting of requests to the web server only if a new data in the web server arises, as a consequence the traffic between the web server and the client can be reducible. The efficiency of web traffic will be measured by two matrices, the successful receptive percentage of the web application request to the web server and the bandwidth consumption of web application.

In this research, an innovation of Green Ajax will be proposed. The idea of low bandwidth and low resource consumption are introduced. Mozilla Firefox and Firebug will be employed as the tools to measure the experimental results. From the experiments, the expected results of this research achieve 80% of the successful receptive percentage of the web application requests, no data loss percentage, and reduce 80% of bandwidth consumption.

ajax, green ajax, web application, web traffic

I. INTRODUCTION

Even though internet speed is increasing significantly in the present, the web traffic is busier. The classic web applications usually need a lot of bandwidth to provide the rich user interfaces. Reloading the big size of web page every times will cause the big consumption of bandwidth.

The new technique, named Ajax, can reduce the web server load and the data transfer between server and users' computer. Ajax is not new but it is a set of web technologies which can provide the interaction between the web server and client. It consists of HTML, JavaScript technology, DHTML, and DOM [2, 6].

If the web application uses Ajax, only a specific part in the web page can be requested to the web server [8]. The new data will appear on the specific part in the web page. The users will not see the blank page when the new data is requested to the server.

However, it still cannot provide the real time data updating. The common approach to provide the real time data updating uses a timer to request a new data from the web server periodically [3, 4]. But this approach does meaningless requests because the requests sometimes do not get any data updating. In contradiction, to change the longer interval time to update the data is not efficient because the client will not receive the data on time.

In this research an alternative way of classical Ajax, called Green Ajax, is proposed. The proposed Green Ajax will be able to receive a signal from the web server. This signal will command the client to request a new data from the web server. Requesting a new data of the web application will depend on the received signal. It will reduce the traffic between the web server and the client. The efficiency of web traffic will be measured from the successful receptive percentage of the web application requests to the web server and the bandwidth consumption of web application.

A research by Merrill found the total bandwidth savings of 61% happened when using the Ajax approach. Another research by White showed 71% of performance improvement and 32% time savings occurred when using Ajax [1]. From the preliminary experiment, by combining with Ajax, the expected results of this research compared to the previous researches are increasing 80% of the successful receptive percentage of the web application requests, reducing 100% of data loss, and reducing 80% of bandwidth consumption.

Mozilla Firefox and Firebug will be used as the tools to measure the results. The importance of study is not only getting a faster real time data updating approach, but also help solve the problem of the limited bandwidth in the developing countries around the world.

In this research, a Green Ajax will be used as a proposed term of the approach because it will bring the idea of low bandwidth and low resource consumption. The differences

between the classic Ajax and the Green Ajax can be shown on table 1.

TABLE I. GREEN AJAX COMPARISONS

Features	Ajax	
	Classic	*Green*
Update in the a specific part of page	Yes	Yes
Update by client request	Yes	Yes
Update by server initiation	No	Yes
Approach for infrequent update data	Interval timer	Server's signal
Real time data updating	Sometimes	Yes

II. LITERATURE REVIEW

A development on the World Wide Web is continuing. Since a static web application represented by HTML was not enough to catch speed of data updating, a dynamic web application was introduced. JavaScript and VBScript are used to provide a dynamic web application on the client side. CGI/Perl, PHP, ASP, JSP, and others are used to provide a dynamic web application on the server side [5, 10]. Combining with any databases, the web page does not show the static information, but it changes dynamically anytime based on the updating data [11].

However, the dynamic web application is still lack of interfaces. Ajax was introduced to overcome the limitations. Ajax is not a new technology but a set of the existing web technologies. It consists of HTML, JavaScript technology, DHTML, and DOM [2]. By using Ajax, the web application can provide an interactivity response. Commonly, a classic web application will show a blank page when there is an activity to request a new data on the page. On the Ajax application, the users will not see the blank page if there are requests to the web server. Only a specific part in the web page can be requested to the web server [8]. Updating a specific part can reduce the web server load and the data transfer to/from users' computers.

Currently, Ajax is included as a component on the Web 2.0 [7]. The term of Web 2.0 is used to emphasize the second phase of the World Wide Web development [9]. It was introduced by O'Reilly Media in 2004. It refers to the new generation of Web applications which provide online participation, collaboration and interaction. By including Ajax as a component on the Web 2.0, shows the capability of Ajax on the Web development.

III. CONCEPTS

This research attempts to develop a concept to reduce the web traffic of non-frequent update web application and design an approach to provide a real time data updating on the web application.

A. Theoretical Frameworks

Based on the web architecture, the client has to request to the web server to get a response. To get a response from the web server frequently, the client has to do the request periodically. In the web application, a timer is usually used to request to the web server periodically. In the smaller interval, web application will request to the web server with the high frequency. In reverse, web application will request to the web server with the low frequency.

Ajax can reduce the web server load and the data transfer to/from users' computers because only a specific part in the web page is requested to the web server. However, the data updating from Ajax depends on the web application request to the web server. If the request is done when the data is still not updated, it will waste the bandwidth and the resource.

B. Conceptual Frameworks

Based on the above statement, the client's request should be done on the appropriate time. The time to request should depend on the data updating activity on the web server. The approach should design the way for a web server to send a signal to the client if there is a new update. A web server in the client side or embedded on the web browser might be needed. It should give the capability to the client to receive a signal from the web server.

To reduce the web traffic, the approach can be combined with Ajax to limit the size of data updating. The combination of these will bring the idea of low bandwidth and the low resource together to provide a real time data updating which will be named as Green Ajax.

Ajax will be used as a main technique in the Green Ajax because Ajax has been proven by some researches as a good approach to decrease the bandwidth. However, Ajax only does some meaningless requests in the condition of infrequent data updating. Green Ajax will cover the limitation of Ajax in this case.

C. A Conceptual Model

To provide the Green Ajax approach on the applications, the web server must have the ability to signal the client. If the client got a signal from the server, the client will request the new data to the server to provide a real time data updating on the web application. The conceptual model to provide the Green Ajax approaches can be shown below.

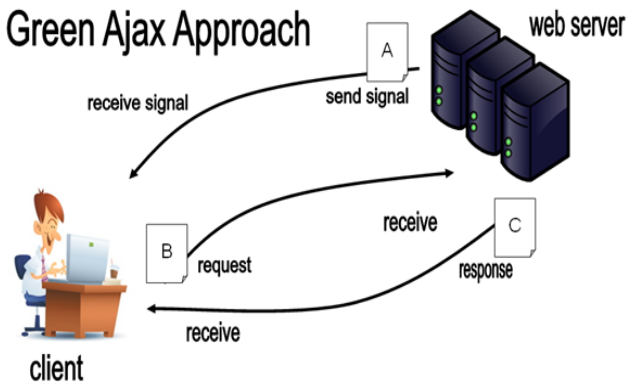


Figure 1. The model of Green Ajax Approach

The client will receive a small size of signal from the server for each updating activity to trigger a request to the web server. The web server will send all of the responses

based on the client request. The proposed approach to receive the signal from the web server without clients' request is providing a virtual server on the clients.

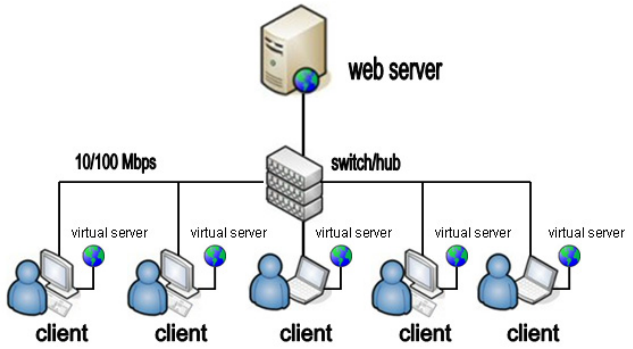


Figure 2. The Example Topology

The size of the signal, the request of client, and the received response will be counted as bandwidth consumption by using Mozilla Firefox and Firebug.

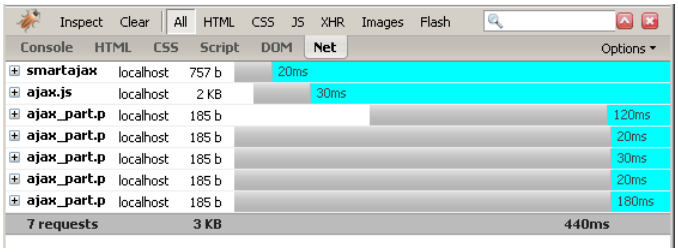


Figure 3. Firebug on the Mozilla Firefox

In the preliminary experiment, the bandwidth consumption of Green Ajax will be compared to the Ajax applications which have different interval times. The interval time for Ajax applications can be selected randomly as 30 seconds, 60 seconds, 5 minutes, 10 minutes, and 15 minutes. Other data will be collected on the experiment, such as bandwidth consumption, number of updates activity, number of client's request, number of received data in the client, number of wasted request of the client, and the non-received data.

These data can summarize the Data Loss Percentage (DLP) between these applications. The formula of DLP can be shown below.

$$\text{Data Loss Percentage (DLP)} = \frac{\text{number of times data was not received timely (Los)}}{\text{number of times update data was activated (Upd)}} \times 100$$

From the Data Loss Percentage (DLP), the Successful Receptive Percentage (SRP) of each application can also be summarized. The formula of SRP can be shown below.

$$\text{Successful Receptive Percentage (SRP)} = \frac{\text{number of times data has been received (Rcv)}}{\text{number of times requests had been issued (Req)}} \times 100$$

IV. RESULTS

From the experiment within one hour, there are 22 updates activity (Upd) on the web server which is unpredictable time. Each type of classic Ajax on the experiment request several times based on the interval timer. The number of request within one hour will be counted as Request (Req) Some request did not get a new data, will be count as wasting request (Wst). If the application gets a new reply, the reply will count as received (Rcv). However, sometimes the application did not get the complete data. Some data loss will be counted as Loss (Los). For Green Ajax approach, the server should signal the client to give the information if there is a new data. The activity will be recorded as Server (Srv).

TABLE II. SUMMARY OF EXPERIMENT #1 USING GREEN AJAX

Data	Green Ajax	Ajax				
		30"	*60"*	*5'*	*10'*	*15'*
Requests (Req)	22	119	60	13	7	5
Received (Rcv)	22	22	19	13	7	5
Wasting (Wst)	0	97	41	0	0	0
Updates (Upd)	22	22	22	22	22	22
Un-received (Los)	0	0	2	9	15	17
Server (Srv)	22	0	0	0	0	0

From the table 2, the SRP of Green Ajax is 100%. There are 22 updates activity on the server. Green Ajax received 22 updates and issued 22 times requests. It shows no wasting requests had been issued by Green Ajax. The Green Ajax shows 0% of DLP. There is no un-received data for 22 updates activity on the server. The chart of the comparison can be shown on figure 4.

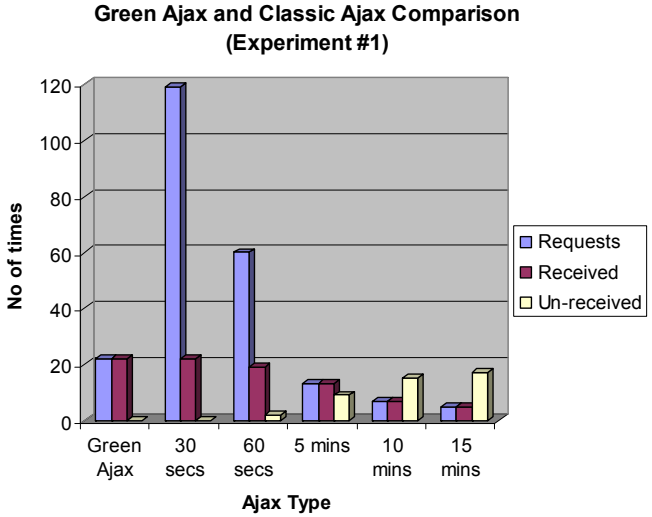


Figure 4. Comparison of Ajax Experiment's Result

The calculation of SRP and DLP of each Ajax experiment can be summarized on the following table. Even

though three Ajax (5 minutes, 10 minutes, and 15 minutes) show 100% of DLP, there are some data losses. Although the Ajax with interval timer of 30 seconds has 0% of DLP, it has 18.49% of SRP only. Green Ajax has best result on both of results. It has 100% SRP and 0% DLP.

TABLE III. SRP AND DLP ON EXPERIMENT #1

Results	Green Ajax	Ajax				
		30"	*60"*	*5'*	*10'*	*15'*
SRP (%)	100.00	18.49	31.67	100.00	100.00	100.00
DLP (%)	0.00	0.00	9.09	40.91	72.73	77.27

Based on the table 3, chart of SRP and DLP can be shown on figure 5.

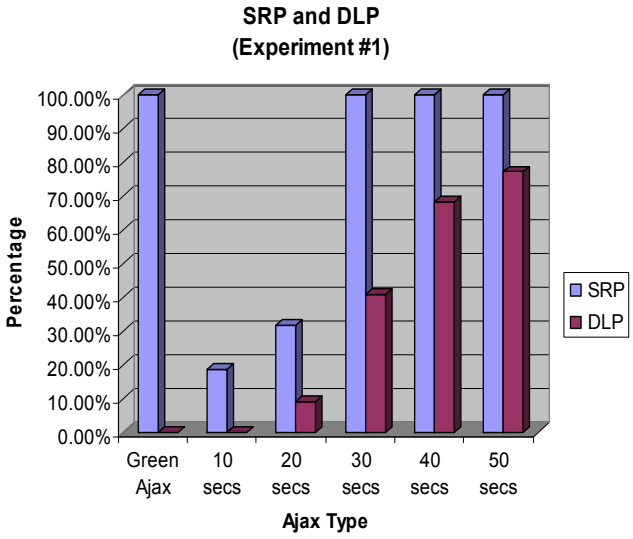


Figure 5. SRP and DLP of Experiment #1

The bandwidth consumption for each type of Ajax is shown on the following table. The bandwidth consumption of Green Ajax is 2,222 bytes without data loss. Even though three Ajax applications (5 minutes, 10 minutes, and 15 minutes) show less consumption than Green Ajax, they do not have the complete data. The higher interval of application has bigger data loss.

TABLE IV. BANDWIDTH CONSUMPTION ON EXPERIMENT #1

Results	Green Ajax	Ajax				
		5 secs	*10 secs*	*15 secs*	*20 secs*	*25 secs*
Bandwidth	2,020	64,020	32,020	21,320	16,020	12,820
Data Loss	0	0	0	0	0	0

Based on the table 4, the chart of bandwidth consumption and data loss can be shown below on figure 6. The bars of bandwidth consumption on the classic Ajax applications look significant different compared to the Green Ajax. However, the longer interval of classic Ajax application has smaller bar of bandwidth consumption but it has taller bar of data loss. In this experiment, the graphic representation of Green Ajax looks more ideal then others.

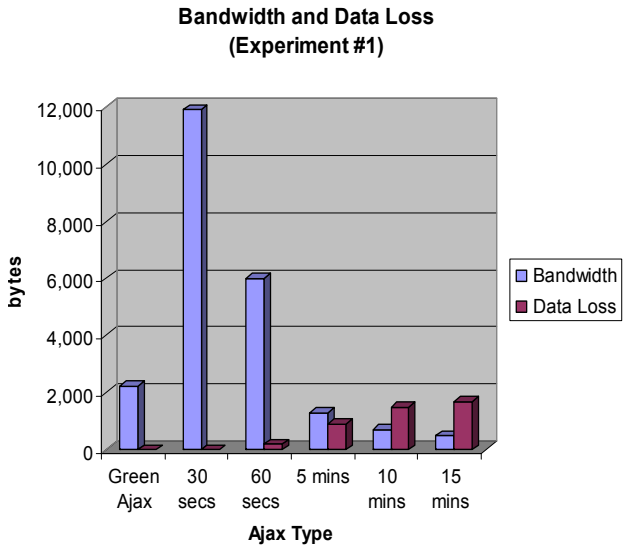


Figure 6. Bandwidth and Data Loss

To get the valid results, another experiment using different scenario is created. From the experiment #2 within one hour using different interval time scenario, there are 27 updates activity (Upd) on the web server which is in random time. The scenario of this experiment uses the shorter interval time then the previous experiment, to update the data on the classic Ajax. Those intervals are 10 seconds, 20 seconds, 30 seconds, 40 seconds, and 50 seconds. The result from this experiment can be shown below.

TABLE V. SUMMARY OF EXPERIMENT #2 USING GREEN AJAX

Data	Green Ajax	Ajax				
		10"	*20"*	*30"*	*40"*	*50"*
Requests (Req)	27	365	181	122	92	73
Received (Rcv)	27	26	25	25	25	24
Wasting (Wst)	0	339	156	97	67	49
Updates (Upd)	27	27	27	27	27	27
Un-received (Los)	0	1	2	2	2	3
Server (Srv)	22	0	0	0	0	0

From the table 5, Green Ajax still shows 100% SRP because there are 27 updates activity on the server. Green Ajax received 27 updates without any wasting requests. Green Ajax shows 0% DLP because there is no un-received data for 27 updates activity on the server. Compared to the other Ajax applications, the Green Ajax has the ideal results.

Based on data of the above table, the chart of the comparison can be shown on figure 7. All kind of Ajax applications receive almost all of the data but they have different significant number of requests. The bars of request on all of classic Ajax applications are much higher than the bar of request on the Green Ajax.

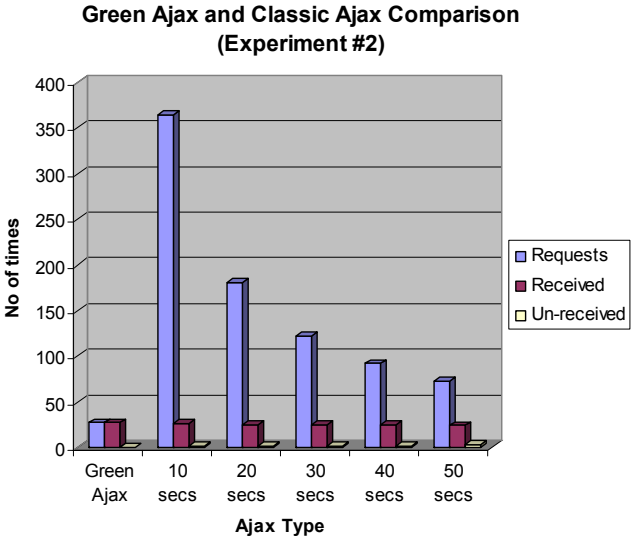


Figure 7. Comparison of Ajax Experiment's Result

The calculation of SRP and DLP of each Ajax experiment can be summarized on the following table. Green Ajax has best result on both of results. It has 100% SRP and 0% DLP. The other classic Ajax applications show low SRP and get some loss data.

TABLE VI. SRP AND DLP ON EXPERIMENT #2

Results	Green Ajax	Ajax				
		10"	*20"*	*30"*	*40"*	*50"*
SRP (%)	100.00	7.12	13.81	20.49	27.17	32.88
DLP (%)	0.00	3.70	7.41	7.41	7.41	11.11

Based on the table 6, chart of SRP and DLP can be shown on figure 8.

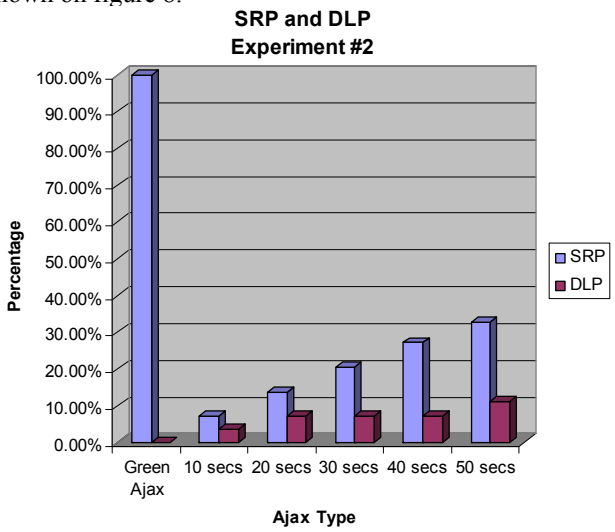


Figure 8. SRP and DLP of Experiment #2

The bandwidth consumption for each type of Ajax is shown on the following table. The bandwidth consumption of Green Ajax is 2,727 bytes without data loss. The other Ajax applications consume more bandwidth than the Green Ajax. Those Classic Ajax applications get some data lost 100 bytes, 200 bytes, and 300 bytes.

TABLE VII. BANDWIDTH CONSUMPTION ON EXPERIMENT #2

Results	Green Ajax	Ajax				
		10"	*20"*	*30"*	*40"*	*50"*
Bandwidth	2,727	36,500	18,100	12,200	9,200	7,300
Data Loss	0.00	100	200	200	200	300

Based on the table 7, chart of bandwidth consumption and data loss can be shown on figure 9.

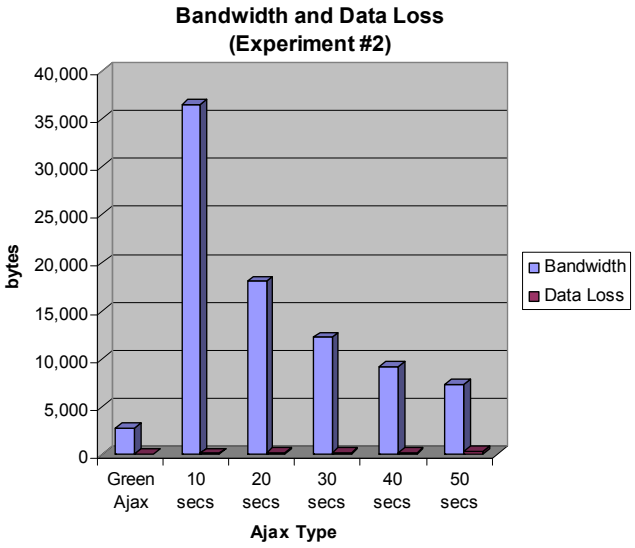


Figure 9. Bandwidth and Data Loss

The scenario of the experiment #3 is changing the interval time for data updating. The classic Ajax of the experiment #3 uses longer interval than the two above experiments. The interval times of the experiment are 10 minutes, 20 minutes, 30 minutes, 40 minutes, and 50 minutes.

TABLE VIII. SUMMARY OF EXPERIMENT #3 USING GREEN AJAX

Data	Green Ajax	Ajax				
		10'	*20'*	*30'*	*40'*	*50'*
Requests (Req)	24	7	4	3	2	2
Received (Rcv)	24	7	4	3	2	2
Wasting (Wst)	0	0	0	0	0	0
Updates (Upd)	24	24	24	24	24	24
Un-received (Los)	0	17	20	21	22	22
Server (Srv)	24	0	0	0	0	0

From the table 8, Green Ajax shows 100% SRP. There are 24 updates activity on the server. Green Ajax received 24 updates without wasting requests. Green Ajax shows 0% DLP. There is no un-received data for 24 updates activity on

the server. The chart of the comparison can be shown on figure 10.

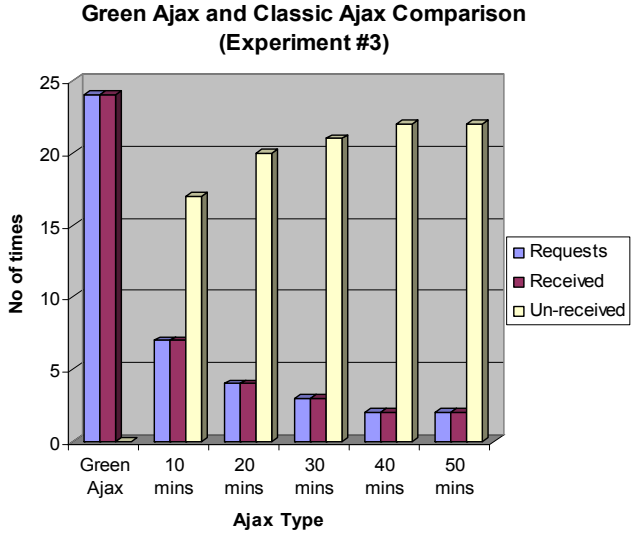


Figure 10. Comparison of Ajax Experiment's Result

The summary of SRP and DLP of each Ajax experiment can be shown on below. Even though all of the classic Ajax applications show 100% SRP, there are some data losses. The greater interval creates bigger loss. Green Ajax still has best result on both of results. It shows 100% SRP and 0% DLP.

TABLE IX. SRP AND DLP ON EXPERIMENT #3

Results	Green Ajax	Ajax				
		10'	*20'*	*30'*	*40'*	*50'*
SRP (%)	100.00	100.00	100.00	100.00	100.00	100.00
DLP (%)	0.00	70.83	83.33	87.50	91.67	91.67

Based on the table 9, chart of SRP and DLP can be shown on figure 11.

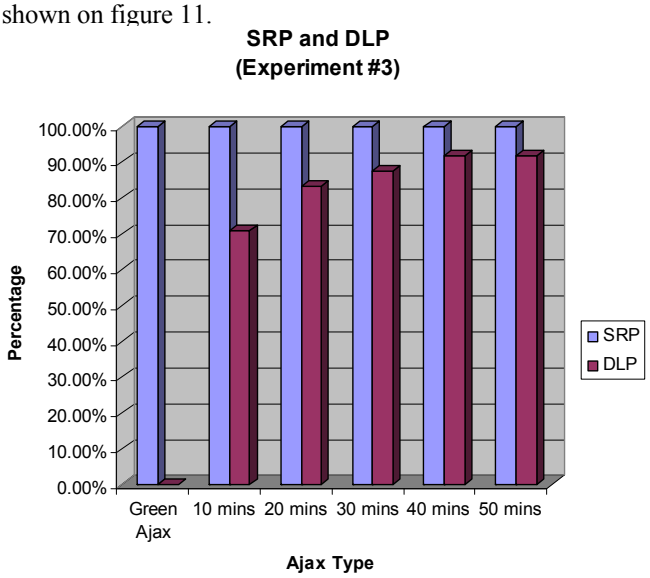


Figure 11. SRP and DLP of Experiment #3

The bandwidth consumption for each type of Ajax is shown on the following table. The bandwidth consumption of Green Ajax is 2,424 bytes without data loss. Even though all of the classic Ajax applications consume less bandwidth than Green Ajax, they get some data loss and the data lost become greater for the longer interval.

TABLE X. BANDWIDTH CONSUMPTION ON EXPERIMENT #3

Results	Green Ajax	Ajax				
		10'	*20'*	*30'*	*40'*	*50'*
Bandwidth	2,424	700	400	300	200	200
Data Loss	0	1,700	2,000	2,100	2,200	2,200

Based on the table 10, chart of bandwidth consumption and data loss can be shown on figure 12.

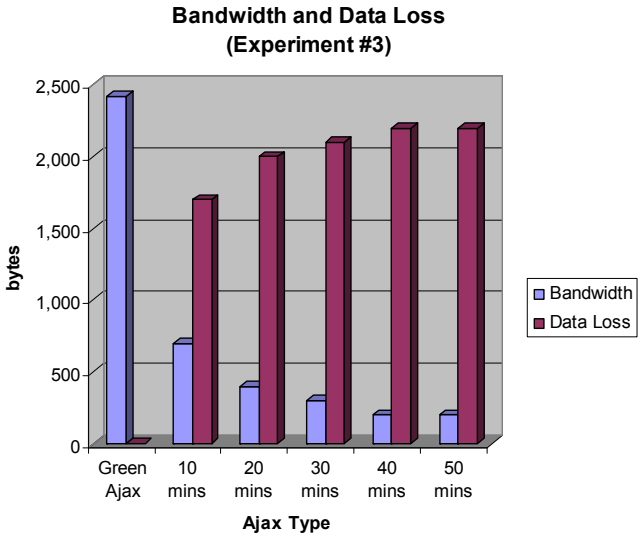


Figure 12. Bandwidth and Data Loss

V. CONCLUSIONS

The classic Ajax has been proven as a good approach to create the richer user interface for the web application. Some researches show the Ajax capability to reduce the bandwidth consumption. However, it is not proven as a good approach if the application is used to show the real-time data updating from the server. The above three experiments show all the interval does not effective. Even though some applications using longer interval show 100% successful receptive, but they do not have the complete data from the server.

If the application uses shorter interval, it has a possibility to get all of the data from the server, but the successful receptive percentage of application become very low. The application request very often, but most of the requests do not get any new data from the server. It will consume more bandwidth than necessary.

In these experiments, Green Ajax is the solution for real-data updating application. Green Ajax has a better approach than the classic Ajax. The successful receptive percentage of Green Ajax is 100% without data loss. It also has the best

result of the combination of total bandwidth consumption and data loss. The approach makes the web application faster and low resources.

ACKNOWLEDGMENT

This manuscript is fully granted from the Ministry of National Education of the Republic of Indonesia since 2008.

REFERENCES

[1] A.A. Dahlan and T. Nishimura, "Implementation of Asynchronous Predictive Fetch to Improve the Performance of Ajax-Enabled Web Applications", Proceeding of the 10th International Conference on Information Integration and Web-based Applications & Services, ACM, New York, USA, November 2008, pp. 345-350. doi: http://doi.acm.org/10.1145/1497308.1497371.

[2] Mesbah and A.v. Deursen, "An Architectural Style for Ajax" Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture, IEEE Computer Society Washington, DC, USA, pp. 9-9. doi: http://dx.doi.org/10.1109/WICSA.2007.7.

[3] Kletsch and D. Volk, "Towards an AJAX-based Game Engine" Proceedings of the 2008 Conference on Future Play: Research, Play, Share, ACM, New York, USA, November 2008, pp. 270-271. doi: http://doi.acm.org/10.1145/1496984.1497050.

[4] C.L. Chen and T.V. Raman, "AxsJAX: A Talking Translation Bot Using Google IM : Bringing Web-2.0 Applications to Life" Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A), ACM International Conference Proceeding Series, vol. 317, ACM New York, USA, April 2008, pp. 54-56. doi: http://doi.acm.org/10.1145/1368044.1368056.

[5] D.S. McFarland, "Dreamweaver 8: The Missing Manual", Sebastopol: CA, O'Reilly Media, Inc., 2004.

[6] L. Zhijie, W. Jiyi, Z. Qifei, and Z. Hong, "Research on Web Applications Using Ajax New Technologies" Proceeding of the 2008 International Conference Multimedia and Information Technology, IEEE Computer Society, Washington, DC, USA, December 2008, pp. 139-142. doi: http://doi.ieeecomputersociety.org/10.1109/MMIT.2008.107

[7] Pilgrim and J. Chris, "Improving the Usability of Web 2.0 Applications" Proceedings of the 19th ACM Conference on Hypertext and Hypermedia, ACM, New York, USA, June 2008, pp. 239-240. doi: http://doi.acm.org/10.1145/1379092.1379144.

[8] R. Sanjaya and C. Brahmawong, "Distance Examination using Ajax to Reduce Web Server Load and Student's Data Transfer" International Journal of the Computer, the Internet and Management, Volume 15 SP3, Assumption University of Thailand, November, 2007, pp. 24.1-6.

[9] R. Sanjaya and P. Sribhadung, "Web 2.0 and Its Implementation to eLearning" International Journal of the Computer, the Internet and Management, Volume 14 SP1, Assumption University of Thailand, August 2006, pp. 47.1-8.

[10] S. Jablonski, I. Petrov, C. Meiler, and U. Mayer, "Guide to Web Application and Platform Architectures", New York: NY, Springer-Verlag Berlin Heidelberg, 2004.

[11] T. Converse, J. Park, and C. Morgan, "PHP5 and MySQL Bible", Indianapolis, Indiana: Wiley Publishing, 2004.

Trade-off Analysis for Web Application Using Green Ajax

Ridwan Sanjaya
Graduate School of Information Technology
Assumption University
Bangkok, Thailand
ridwan.sanjaya@gmail.com

***Abstract*—The Green Ajax outperforms than the classical Ajax [10]. It can decrease the bandwidth consumption to/from the web server. In the previous research, the Green Ajax results the best performance for the case of random and unpredictable update time on the web server.**

However, from practical point of view the web server will not always update at the random pattern of time. In this research, the web server will update the data at fixed interval of time. The classical Ajax applications will be set to request to the web server using the same duration of time to update on the web server. In other hand, the Green Ajax will request only if the web server sends a signal.

However, the web server's signal in the Green Ajax will collectively affect the bandwidth consumption. The investigation of the overhead signal generation by Green Ajax introduces the trade-off of the Green Ajax employment. Crucial results conclude that Green Ajax is appropriate for the applications with high frequency update.

Green Ajax; classical Ajax; traffic analysis; real-time; data update; web application

I. INTRODUCTION

The classical Ajax has been proven as a good approach to decrease the bandwidth consumption on the web application. The main advantages of using Ajax are less waiting time and more control for the user [11], because the web page specifically changes in portion, not a whole page [9]. This powerful mechanism can be viewed on some well known websites such as Google Suggest, Google Map, Google Mail, Flickr, Yahoo! Mail, and Microsoft Outlook Express Web Version [7].

However, it is not a good approach to display data updates in real-time. The common Ajax applications will check the data updating to the server by using regular user-definable intervals [5]. By using the interval time to refresh, the classical Ajax generates many redundant requests. It results higher bandwidth consumption than normal. To avoid this impact, Green Ajax can be a better approach. The approach makes the web application faster and lower resources to support the real-time data updating. [10]

With the Green Ajax, a web server will have to signal the client prior the updates. Even though the signal size is one byte only, it will collectively affect the bandwidth consumption. To investigate the affect overhead signal to the total bandwidth consumption, a scenario has been created. It will be used comparing the bandwidth consumption of the classical Ajax and Green Ajax. The scenario divides into three parts. The first part will evaluate the data requesting in the high interval time. The second part will use the medium interval time for data requesting. And the low interval time to request will be used on the third part.

Each part will assess five different intervals time to request. Therefore, the total experiments of this scenario are fifteen times. All experiments will use both the classical Ajax and Green Ajax. The web server will update the data in a fixed interval time. The classical Ajax application will be set to request to the web server using the same duration time to update to on the web server. In other hand, the Green Ajax will use its own characteristic, which will only request after the web server gives a signal [10]. This signal arises when the administrator do an update on the web server. It can be seen on the figure 1.

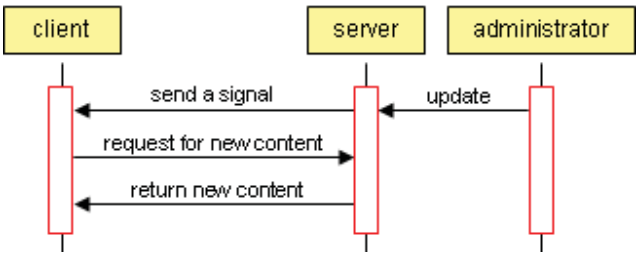


Figure 1. The signal arises when the administrator update the data

The example illustration of each experiment can be seen on the Figure 2. For each experiment, the web server will update the data in the fixed time. Thus, the classical Ajax will request in the same duration time to update on the web server. However, the interval time on the Green Ajax is not set as other applications because it will update anytime when it receives the signal from the web server.

On figure 2, the web server will update the new data every five seconds and the client will request to the server every five seconds also. The first part of experiment uses 5, 10, 15, 20, 25 seconds, and they will be grouped as the high

978-1-4244-6567-5/10/$26.00 ©2010 IEEE

frequency update. The second part uses 30 seconds, 1 minutes, 5 minutes, 10 minutes, 15 minutes, and they will be grouped as the medium frequency update. The third part uses 30 minutes, 1 hour, 2 hours, 3 hours, 5 hours, and they will be grouped as the low frequency update.

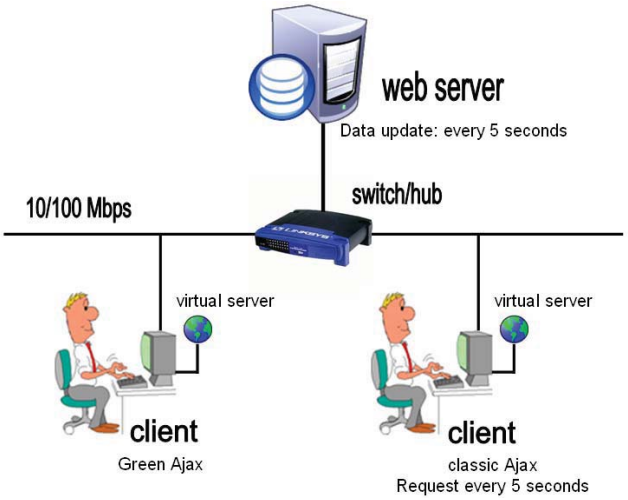


Figure 2. Example of an experiment using both Green Ajax and the classical Ajax

II. LITERATUR REVIEW

Ajax (Asynchronous Javascript And XML) is not a new technology, but a set of technologies used to provide richer user interfaces for dynamic web application. It consists of HTML, CSS, Document Object Model (DOM), XMLHttpRequest, and Javascript. Integration of those technologies will provide an interactive user interface and ability to show the dynamic contents asynchronously [1].

Ajax creates a new way to communicate between server and client by using an asynchronous interactive technology. The result will give more efficient response and better user experience [12]. The web application will not have to load the entire HTML. It will not leave the users waiting for responses from web server. The data can be be requested and be transmited when the users browse the page [13]. The other benefits, several pages can be migrated to be a single page interface with the independent control on each component [2].

Ajax has been used in some applications created by several famous web providers, such as Google Maps, Google Docs, Google Suggest, Google Chat, Flickr, Yahoo! Mail, and Outlook Web Application [8]. Those are increasing the popularity of Ajax.

III. TESTING ENVIRONMENT

The examples of real-time dynamic web data which used in this paper are news headlines, stock exchange display board, the web based flight status on the airport, and the information board of the foreign currency exchange.

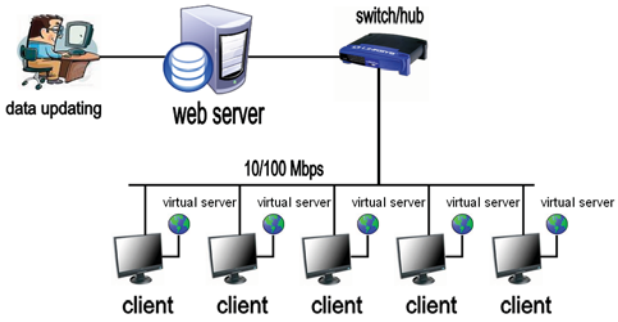


Figure 3. The clients display the updated information from the server

From figure 3, the person in charge will update the new information to the server. In order to display the real-time information, the clients have two alternatives. First, the client requests to the server based on the Time to Refresh (TTR) [4]. Second, the client will requests to the server after the client receive the server's signal. By using the second option, the server will push the signal to the client automatically when the server has a new data.

However, to provide the clients' capability to receive the signal from the server, the clients should have a web browser plug-in whose function as a virtual server. By using the approach, initial communication can be started by the server or the client. Otherwise; the signal can not be received by the clients without their requests. On HTTP, the initial communication is always initiated by the client and never by the server. The protocol was designed for retrieving data from servers [3].

Based on the common situation, the data of stock exchange, flight status, and foreign currency exchange can be changed anytime. However, the management can set the interval time of data updating after the management can predict the average time to update. Grouping the time of data updating can be made to make easier the observations. By time grouping, it can be seen the appropriate situation for the Green Ajax and the classical Ajax if the time for data updating is set already.

The TTR period can differ widely, depending on usage context. For example, news headlines might update every 10 minutes and online statuses of other users on chat application might update every minute [6]. This paper examines three cases used on the web applications based on the time of data updating activity. The cases grouped into three situations, see table I.

TABLE I. SCENARIO OF EXMPERIMENTS

Group #	Scenario	
	Situations	***Interval Time***
1	High Frequency Update	5 seconds – 25 seconds
2	Medium Frequency Update	30 seconds – 15 minutes
3	Low Frequency Update	30 minutes – 5 hours

The time grouping on above is created based on the most commonly used interval time in several applications with the

very high level update to the lowest level update. The stock exchange application is one of the examples which most often in changes, which updates the data within range 5 to 25 seconds. The example applications do not update very often can be seen on the flight status on the airport, which usually update the data between 30 seconds and 15 minutes. The example application of the last group can be seen on the foreign currency display board, which updates in range 30 minutes to 5 hours.

IV. EXPERIMENT METHODOLOGY

In the first experiment using the high frequency update scenario, the web server will update every 5 seconds, the classical Ajax will keep request every 5 seconds, and the Green Ajax will request on condition that the application receive a signal from the web server. The bandwidth consumption of the classical Ajax and the Green Ajax will be summarized to get the percentage of the margin between the classical Ajax and the Green Ajax's bandwidth consumption. The number of times data received on the client (Rcv) will be divided by the number of times requests issued by the client (Req). The result of this calculation is the Successful Receptive Percentage (SRP).

$$\text{Successful Receptive Percentage (SRP)} = \frac{\text{number of times data has been received (Rcv)}}{\text{number of times requests had been issued (Req)}} \times 100$$

Figure 4. Formula of Successful Receptive Percentage (SRP)

If there is data lost, the number of times data was not received (Los) will be divided by the number of times update data activated (Upd) to get the Data Loss Percentage (DLP).

$$\text{Data Loss Percentage (DLP)} = \frac{\text{number of times data was not received timely (Los)}}{\text{number of times update data was activated (Upd)}} \times 100$$

Figure 5. Formula of Data Loss Percentage (DLP)

Other experiments of high frequency update will have interval time of 10 seconds, 15 seconds, 20 seconds, and 25 seconds. On the medium frequency update this paper will examines 30 seconds, 1 minutes, 5 minutes, 10 minutes, and 15 minutes of interval time of data updating. For the low frequency update, it will test the updating data on the server every 30 minutes, 1 hours, 2 hours, 3 hours, and 5 hours.

By using the above formula, this experiment will get the comparison of SRP and DLP between high frequency update, medium frequency update, and low frequency update. The percentage margin will show the significance of using the classical Ajax and the Green Ajax when the updating time on web server was set already.

V. RESULTS

From the experiment of the first part, the results can be seen on the table II. Even though the classical Ajax uses the same interval time as the server' interval time to update, its bandwidth consumption is still higher than the Green Ajax. Some requests of the classical Ajax do not get any new data. The Green Ajax is still the best but the longer interval creates shorter margin bandwidth consumption between The Green Ajax and the classical Ajax.

TABLE II. BANDWIDTH CONSUMPTION (IN BYTES) ON HIGH FREQUENCY UPDATE

Scenario[a]	Interval Time				
	5 secs	*10 secs*	*15 secs*	*20 secs*	*25 secs*
Classical Ajax	59,500	30,200	21,200	16,100	12,900
Green Ajax	50,803	28,583	19,897	15,150	12,322
Margin (%)	17.12%	5.66%	6.55%	6.27%	4.69%

a. Experiment time: 1 hour

The decreasing margin of bandwidth consumption between both Ajax can be seen on the Figure 6.

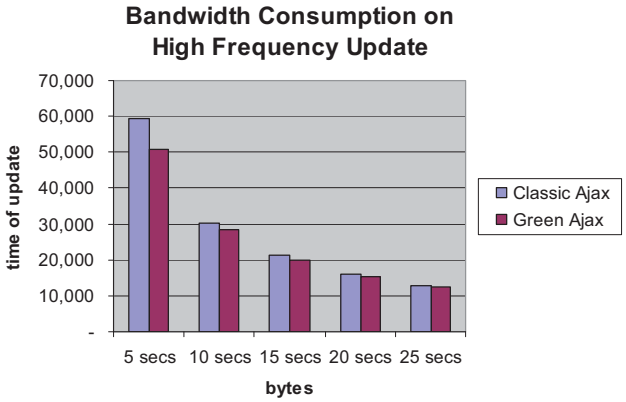


Figure 6. Bandwidth Consumption on High Frequency Update

From the previous research [10], the Successful Receptive Percentage (SRP) of each application can be seen below. The formula on the Figure 4 will show the percentage of the accurate requests to the web server.

On the Table III, SRP of the classical Ajax is not fully 100% because some requests on the first, second, and third interval time, do not get any new data. It means there are some wasting requests.

TABLE III. SRP ON HIGH FREQUENCY UPDATE

Scenario[a]	Interval Time				
	5 secs	*10 secs*	*15 secs*	*20 secs*	*25 secs*
Classical Ajax	99.16%	99.00%	98.36%	100.00%	100.00%
Green Ajax	100.00%	100.00%	100.00%	100.00%	100.00%

a. Experiment time: 1 hour

The SRP of the Green Ajax applications are 100%. All of the requests from the Green Ajax applications to the web server get all the new data. The Green Ajax applications do not waste their requests.

The Data Loss Percentage (DLP) of each application can be seen below. The formula on the Figure 5 will show the percentage of uncompleted data from the server. On the table IV, the DLP of both applications are 0%. There are no data loss on the classical Ajax and the Green Ajax applications. All applications get all the new data from the server.

TABLE IV. DLP ON HIGH FREQUENCY UPDATE

Scenario[a]	Interval Time				
	5 secs	*10 secs*	*15 secs*	*20 secs*	*25 secs*
Classical Ajax	0%	0%	0%	0%	0%
Green Ajax	0%	0%	0%	0%	0%

a. Experiment time: 1 hour

Table V shows the results from the second part of experiments. Some classical Ajax applications show smaller bandwidth consumption than the Green Ajax. However, there are no significant margin of bandwidth consumption between the classical Ajax and the Green Ajax. The margins of the bandwidth consumption are below than one percent.

TABLE V. BANDWIDTH CONSUMPTION (IN BYTES) ON MEDIUM FREQUENCY UPDATE

Scenario[a]	Interval Time				
	30 secs	*1 mins*	*5 mins*	*10 mins*	*15 mins*
Classical Ajax	59,700	30,000	6,100	3,000	2,100
Green Ajax	59,792	29,997	6,060	3,030	2,121
Margin (%)	(0.15%)	0.01%	0.66%	(0.99%)	(0.99%)

a. Experiment time: 5 hour

The bandwidth consumption of both applications can be seen on the Figure 7. Because of little margins, the bars inside the chart do not show the significant differences.

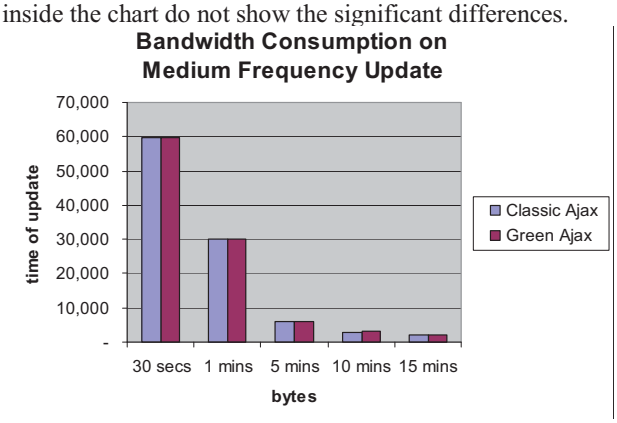


Figure 7. Bandwidth Consumption on Medium Frequency Update

Even though the differences of margins between both applications are insignificant, the Green Ajax shows the best performance on the SRP. On table VI, some of classical Ajax applications do not have 100% SRP. Some requests from the classical Ajax do not get the new data from the web server. However, the Green Ajax applications get all the new data from their requests.

TABLE VI. SRP ON MEDIUM FREQUENCY UPDATE

Scenario[a]	Interval Time				
	30 secs	*1 mins*	*5 mins*	*10 mins*	*15 mins*
Classical Ajax	99.16%	99.00%	98.36%	100.00%	100.00%
Green Ajax	100.00%	100.00%	100.00%	100.00%	100.00%

a. Experiment time: 5 hour

On the medium frequency, both applications show 0% of DLP. The experimental results of 30 seconds, 1 minutes, 5 minutes, 10 minutes, and 15 minutes have the same result as listed at table IV. There are no data lost when the applications work. The classical Ajax and the Green Ajax get all the new data from the web server.

Similar with the second part of experiment, the third part also shows insignificant margins. However in this case, the classical Ajax applications show the best performance. Their bandwidth consumption is smaller than the Green Ajax. On table VII, the bandwidth consumption on the higher interval time of the classical Ajax is close to the Green Ajax. The average margins on both applications are below than one percent. The differences are found only on the signal size for each data updating on the web server.

TABLE VII. BANDWIDTH CONSUMPTION (IN BYTES) ON LOW FREQUENCY UPDATE

Scenario[a]	Interval Time				
	30 mins	*1 hr*	*2 hrs*	*3 hrs*	*5 hrs*
Classical Ajax	1,100	600	300	200	200
Green Ajax	1,111	606	303	202	202
Margin (%)	(0.99%)	(0.99%)	(0.99%)	(0.99%)	(0.99%)

a. Experiment time: 5 hour

On the Figure 8, there are no significant differences on the bars.

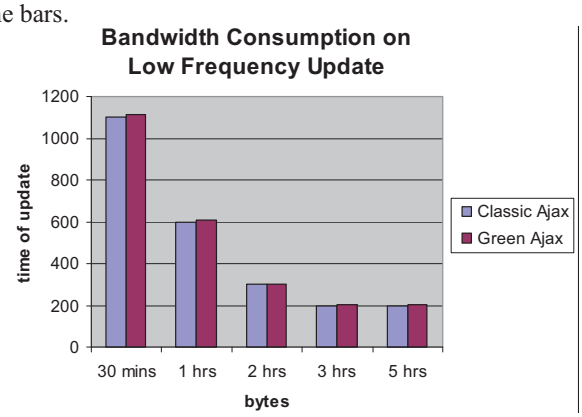


Figure 8. Bandwidth Consumption on Low Frequency Update

From table VIII, the classical Ajax application and the Green Ajax applications get 100% SRP. There are no wasting requests on both applications. They do effective requests to the server because all requests get the new data from the web server.

TABLE VIII. SRP ON LOW FREQUENCY UPDATE

Scenario[a]	Interval Time				
	30 mins	*1 hr*	*2 hrs*	*3 hrs*	*5 hrs*
Classical Ajax	100.00%	100.00%	100.00%	100.00%	100.00%
Green Ajax	100.00%	100.00%	100.00%	100.00%	100.00%

a. Experiment time: 5 hour

The DLP results of both applications on the Low Frequency Update experiments are also 0%. The result of 30 minutes, 1 hour, 2 hours, 3 hours, and 5 hours are the same as listed at table IV. All applications receive all of data on the server.

VI. CONCLUSIONS

Green Ajax shows better performance than the classical Ajax on previous research. Those experiments were tested on the case of random and unpredictable update time on the web server. Green Ajax has been proven able to make the web application give faster responses and consume low resources. However, those experiments were not tested in the fixed update time on the web server.

In the case of fixed update time on the web server, Green Ajax is still able to show the best performance compared to the classical Ajax. Green Ajax succeeds on the high frequency update. In other hand, the classical Ajax defeats the Green Ajax on the medium and low frequency update. However, there are insignificance differences of bandwidth consumption on the classical Ajax and the Green Ajax. The margins of bandwidth consumption between both applications always show below than one percent. The comparison of SRP on both applications on the high and medium frequency update are little differences, even though the Green Ajax shows outperform than the classical Ajax. For DLP, both applications show zero data lost.

Even though the signals to the Green Ajax are consuming bandwidth more than the classical Ajax, it is not too significant. The differences bellow than one percent is not a large number. Based on the above comparison, Green Ajax is still appropriate for the applications with high frequency update while there is no difference between Green Ajax and classical Ajax for the applications with either low or medium frequency update. The Green Ajax can be used not only in the unpredictable time of update but also when time of update on the web server was set already.

ACKNOWLEDGMENT

This manuscript is fully granted from the Ministry of National Education of the Republic of Indonesia since 2008.

REFERENCES

[1] A. Marchetto and P. Tonella, “Search-Based Testing of Ajax Web Applications,” in *Proceedings of the 2009 1st International Symposium on Search Based Software Engineering*, IEEE Computer Society, May 2009, pp. 3-12. [Online]. Available: http://dx.doi.org/10.1109/SSBSE.2009.13

[2] A. Mesbah and A.v. Deursen, “Migrating Multi-page Web Applications to Single-page AJAX Interfaces,” in Proceedings of the 11th European Conference on Software Maintenance and Reengineering, IEEE Computer Society, March 2007, pp. 181-190. [Online]. Available: http://dx.doi.org/10.1109/CSMR.2007.33

[3] A. Serbinski and A. Abhari, “Improving the Delivery of Multimedia Embedded in Web Pages” in *Proceedings of the 15th international conference on Multimedia*, ACM Press, Sept 2007, pp. 779-782.

[4] E. Bozdag, A. Mesbah, and A. van Deursen, "A comparison of push and pull techniques for ajax," in *Web Site Evolution (WSE) 2007. 9th IEEE International Workshop on Oct 2007*, pp. 15-22. [Online]. Available: http://dx.doi.org/10.1109/WSE.2007.4380239

[5] E. Bozdag and A. van Deursen, "An Adaptive Push/Pull Algorithm for AJAX Applications," in *Third International Workshop on Adaptation and Evolution in Web Systems Engineering on 2008*. AEWSE 2008, pp. 95-100.

[6] M. Mahemoff, *Ajax Design Patterns*. Sebastopol, CA: O'Reilly, 2006, pp. 216-217.

[7] N. C. Zakas, J. Mcpeak, and J. Fawcett, *Professional Ajax*. Indianapolis, Indiana: Wiley Publishing, 2006, pp. 13-16.

[8] P. Thiesen and C. Chen, “Ajax Live Regions: Chat as a Case Example,” in *Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, ACM Press, May 2007, pp. 7-14.

[9] R. Sanjaya and C. Brahmawong, “Distance Examination using Ajax to Reduce Web Server Load and Student’s Data Transfer”, in *International Journal of the Computer, the Internet and Management*, vol. 15 SP3, Nov 2007, pp. 24.1-6.

[10] R. Sanjaya, “Web Traffic Reduction for Infrequent Update Application Using Green Ajax,” in *Proceeding of the 2nd IEEE International Conference on Information Management and Engineering 2010*, in press.

[11] Smith, Keith. “*Simplifying Ajax-Style Web Development*”, Computer, May 2006, pp. 98-101.

[12] W. Jiaqi, L. Jie, W. Shujuan, “The Realization of Google Suggest in Mis System by Using Ajax,” in *Proceedings of the 2009 International Forum on Computer Science-Technology and Applications*, Vol. 02, IEEE Computer Society, December 2009, pp 93-96. [Online]. Available: http://dx.doi.org/10.1109/IFCSTA.2009.144

[13] Z. Lin, J. Wu, and Q. Zhang, and H. Zhou, “Research on Web Applications Using Ajax New Technologies,” in *Proceeding of the 2008 International Conference Multimedia and Information Technology*, IEEE Computer Society December 2008, pp. 139-142. [Online]. Available: http://dx.doi.org/10.1109/MMIT.2008.107

Mobile Traffic Evaluation for Fuzzy-Based Application Using Green Ajax

Ridwan Sanjaya
Graduate School of Information Technology
Assumption University
Bangkok, Thailand
ridwan.sanjaya@gmail.com

Abstract— **Green Ajax previously has been proved to be a suitable approach for infrequent updates applications [8]. The low bandwidth requirement, high Successful Receptive Percentage (SRP), and low Data Loss Percentage (DLP) assure that Green Ajax outperforms comparable to Classic Ajax. It is also proved that Green Ajax is a good approach for frequent updates applications [9]. The introduction of the Green Ajax to mobile-based applications will be highly beneficial in terms of cost savings and less resources as lower transmission of traffic.**

The paper will focus the traffic of web-based mobile applications as Green Ajax is employed. Fuzzy-based applications are also considered in order to cope with the practical conditions. It is common in reality that each basic type of three traffics, which are high, medium and low, will be overlapping. Fuzzy Inference System by Sugeno [11] is utilized to solve these ambiguous conditions.

In this paper, Green Ajax will be proposed into the web-based mobile environment. In comparison to classic Ajax, results confirm that the Green Ajax requires lower bandwidth as real-time data update from the server to the handheld units. These experimental results demonstrate that Green Ajax can help save transmission costs and requires a smaller amount resource for the web-based applications over mobile communication.

fuzzy-based, green ajax, mobile, web-based application, traffic

I. INTRODUCTION

The needs of applications on mobile device always increase. New technology on mobile gadget constantly creates needs of users by itself, unless the internet connection is a major problem. Internet connection became a key requirement on mobile devices. Currently, many applications on any mobile devices require internet connection. Some data based on the application residing on mobile devices need to be synchronized with the data on the server.

Compared to stand-alone mobile applications, the development of web-based application for mobile devices is essential. On the stand-alone applications, developers have to provide several solutions in order to support multiple platforms. It is more efficient to develop the web-based application employed by multi-devices independently and support the "write once, run many" paradigm [10].

However, the current interface on the mobile device will not technically support real-time information update. User applications depend on requests for updated data from the server. It is a waste in both time and resources consumption, particularly the request with no update. In the web-based environment using Green Ajax, any updates, whenever arisen, will appear automatically on the mobile device with low bandwidth requirement as introduced in [8].

Classic Ajax has been investigated over the wired network environment. From these experiments, it is proved that a good approach for web-based applications is to reduce the communication bandwidth. It is in fact a band limited over the wireless communication environment. Apparently, the Classic Ajax approach cannot apply to the web-based mobile application due to real-time updates situation [12]. The interaction between server and mobile devices over the wireless environment will be increasing. As a consequence, the whole bandwidth by all means will be occupied and degraded by the interactive overhead. The benefits on bandwidth saving is crucial in the mobile environment.

Green Ajax is an improvement of Classic Ajax which has been well known in the network environment. Inherits from the ancestor, Green Ajax also has capability to reduce the bandwidth consumption. Because not all parts of the web page will be requested from the web server [7]. The utilization of this approach in mobile devices has a good impact in reducing the mobile traffic.

Figure 1. Green Ajax on mobile web-browser

In figure 1, the requested part is the predetermined location on the page to display the new data. By using Green

Ajax, only the data located on the specific parts are requested to the server. Large bandwidth will be saved by updating only on the certain parts of the page.

II. Literature Review

Classic Ajax is a set of the existing web technologies. It consists of HTML, JavaScript technology, DHTML, and DOM [6]. For web-based application over the network environment, Classic Ajax is a common way to provide the dynamic communication between client and server [3]. Based on HTTP architecture, the server will not update the client if no request. The clients have to issue requests then wait for responses from the web server. If they are the same old data in the responses, the client will keep on with the same displayed information. It wastes the request time, response time and resources for these unnecessary requests.

Green Ajax irons out the problem by providing the new approach based on initiating the signal from web server to the clients whenever an update or a change of data occurs. As the client receives the server's signal, the client will then send out a request [8]. This scheme cuts required bandwidth down to necessary due to the update. From the experiment, it is proved that the bandwidth saving based on some forms of the update including infrequent and frequent application. However, experimental update patterns are not coping with realistic conditions, namely, they are not involving with ambiguous types. They are referred as a Fuzzy based application which may indeed reflect the reality.

Fuzzy logic provides an alternative way on using non-crisp value. Some values are uncertain and cannot to be figured out by a Boolean function or other logics which use crisp value sets. On the other hand, Fuzzy will result any doubtful figures lying between 0 and 1 based upon the degree of membership. The degree of membership is decided by the function [4].

In the Fuzzy logic, the fuzzifier will map the crisp inputs into fuzzy sets on the data entry side and the other hand, defuzzifier will map the fuzzy sets output into a single crisp point on the result side. The fuzzifier is needed in fuzzy logic because the system has to convert crisp data to fuzzy data in the beginning. Commonly, if the system receives crisp inputs, the outputs are also data in crisp format. Furthermore, the defuzzier will be used to convert fuzzy data into crisp data as the output [2].

Some ranges of time in the experiment can be realistic and over crossed situations. From these conditions, results will look more practical than the ones demonstrated in [9]. By applying Matlab [11], the Fuzzy Inference System introduced by Sugeno can be utilized to solve the uncertain ranges of update time. Results due to these fuzzy based applications can be employed to solve the uncertain in each input regions. After the clarification of each fuzzy-based input is accomplished then the bandwidth saving will be further evaluated based on these Matlab results.

Web-based mobile application will benefit the bandwidth savings as mentioned earlier. In other words, the bandwidth saving will increase speed in communication. Web-based mobile application experiences in dealing with many limitations, ranging from the handy size of handheld devices to inadequate bandwidth over the internet connection in some regions. In order to reduce space required on mobile devices, most part of application will reside onto server side. Moreover, bandwidth saving will help iron out the problem of low speed internet.

III. Green Ajax upon Network Traffic

In the research, Green Ajax will be evaluated in the network environment, which will focus on three types of the traffic, namely, Infrequent Update, Frequent Update, and the Fuzzy-based Applications. The experimental environment has been implemented in a local area network (LAN) with five clients using star topology as shown in figure 2. The size of the signal, the request of client, and the received response will be counted as bandwidth consumption [8]. The evaluation is to compare Green Ajax versus Classic Ajax.

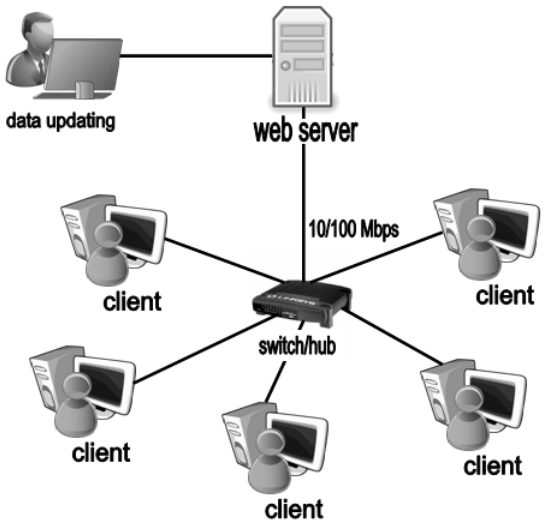


Figure 2. Experimental Star Topology for the research

A. Infrequent Update Application

The assessment in the earlier research [8] focused on the bandwidth consumption between Green Ajax and Classic Ajax. However, in this paper, interval time for updates is different and regrouped into three sets. The first set comprises of five web applications with short Time to Refresh (TTR). Each application refreshes every 5, 10, 15, 20 and 25 seconds. In the second set, five web applications with longer TTR, which each of them refreshes every 0.5, 3, 5, 10 and 15 minutes. The last set of five applications with the longest TTR, which each keeps refreshing every 30, 60, 120, 180 and 300 minutes.

The three sets of five applications, each as listed previously, will be applicable to Classic Ajax alone while Green Ajax application will only refresh whenever it receives an update signal from web server. In this case, the web server randomizes data updating time. The experiment which had been run for one hour, results confirm Green Ajax requires lower bandwidth than Classic Ajax. Table I and figure 3 show Green Ajax's lower bandwidth consumption compared to others.

TABLE I. BANDWIDTH CONSUMPTION AND DATA LOSS

Results	Green Ajax	Ajax				
		30"	*60"*	*5'*	*10'*	*15'*
Bandwidth	2,222	11,900	6,000	1,300	700	500
Data Loss	0	0	200	900	1,600	1,700

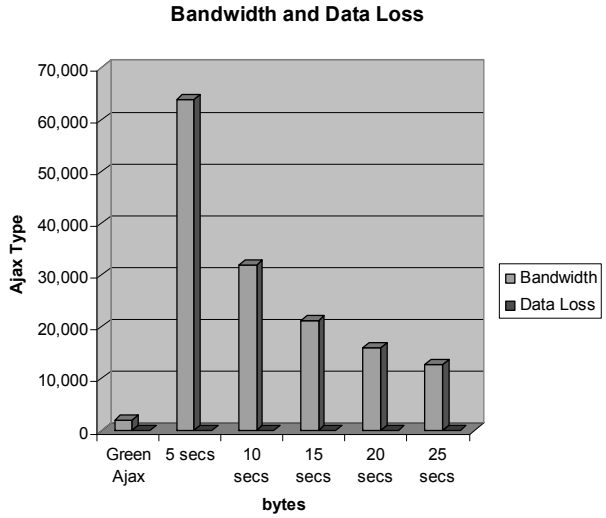


Figure 3. Bandwidth and Data Loss

Moreover, all of Successful Receptive Percentage (SRP) of Green Ajax applications is always 100%. The Data Loss Percentage (DLP) of Green Ajax applications is always 0%. It technically confirms Green Ajax is successfully receiving an update on demand, but all of data from the server was lossless.

TABLE II. SRP AND DLP

Results	Green Ajax	Ajax				
		5 secs	*10 secs*	*15 secs*	*20 secs*	*25 secs*
SRP	100.00%	3.13%	6.25%	9.39%	12.50%	15.63%
DLP	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

The Classic Ajax applications that use shorter interval, have a possibility to get all of the data from the server (DLP equals 0%), but the SRP of application becomes low. The application requests very often, but most of the requests do not get any new data from the server. It will consume more bandwidth than necessary. The figure 4 explains SRP and DLP experiments based on the Infrequent Update Application.

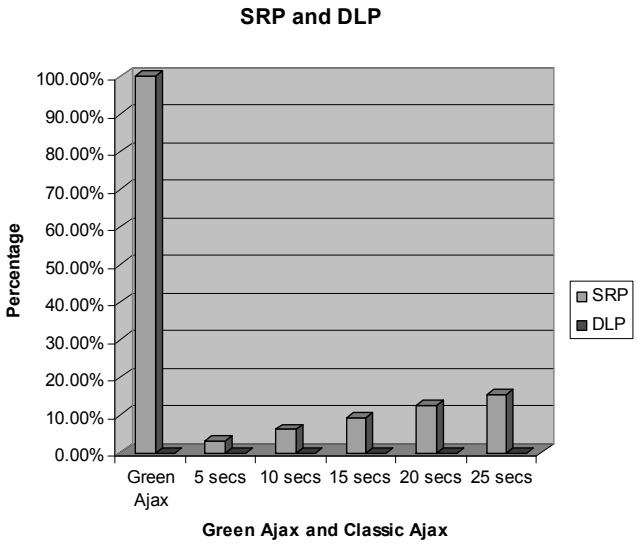


Figure 4. SRP and DLP

B. *Frequent Update Application*

The previous research [9] focused on the overhead signal generation by Green Ajax. The web server's signals in the Green Ajax collectively influence the bandwidth consumption in the network. In this case, the web server has a fixed schedule time for data updating. The evaluation results will be compared with Classic Ajax whose same duration time as the server updates time. As Green Ajax characteristic, the application will only refresh whenever it receives an update signal from web server.

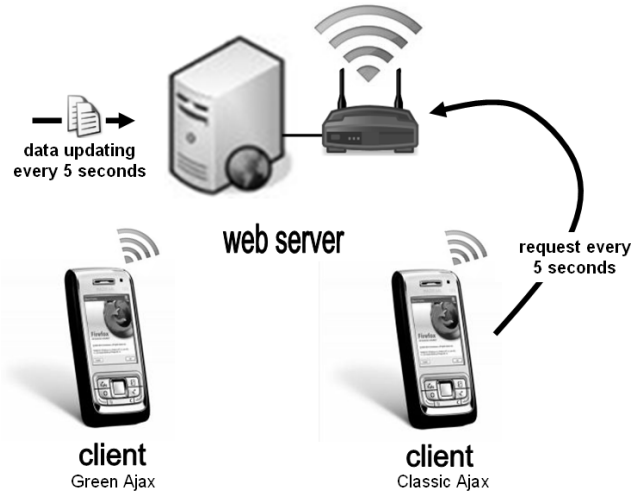


Figure 5. Server and Classic Ajax have the same interval time for update

In this paper, the experiments were regrouped into three sets. Each set will be compared to Green Ajax application. The first set has five web applications with TTR ranging from 5 seconds to 25 seconds. The Classic Ajax application in the first set refreshes every 5, 10, 15, 20, and 25 seconds. The web server also updates in the same interval time. The second set also has five web applications with Time to

Refresh (TTR) ranging from 30 seconds to 15 minutes. Each Classic Ajax application in this set refreshes every 0.5, 3, 5, 10, and 15 minutes. The web server also updates in the same interval time. In the third set, TTR of Classic Ajax and interval time to update on the web server are set to be longer than previous sets. Five Classic Ajax applications and the web server have repetition activities on the same time to refresh or update in every 0.5, 1, 2, 3, and 5 hours. The complete sets of experiment can be summarized in the table III.

TABLE III. SCENARIO OF EXPERIMENTS

Group #	Scenario	
	Situations	*Interval Time*
1	High Frequency Update	5 seconds – 25 seconds
2	Medium Frequency Update	30 seconds – 15 minutes
3	Low Frequency Update	30 minutes – 5 hours

The range of time based on the above scenario can be drawn and shown in the figure 6. There are strict ranges of time in each set which are labeled as high, medium, and low frequency update.

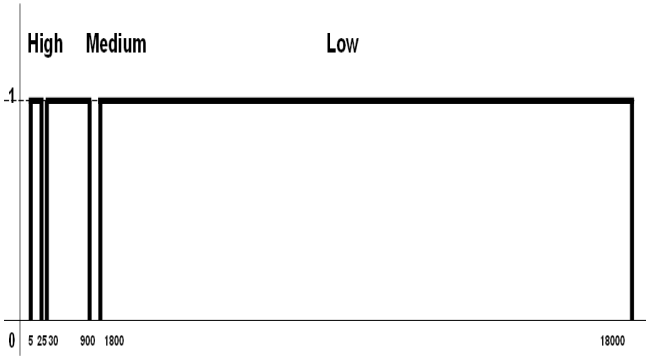


Figure 6. Range of time in the previous experiment

In that experiment, High Frequency Update has the exact range of time between 5 and 25 seconds. The Medium Frequency Update has the crisp area from 25 until 900 seconds. The memberships of Low Frequency Update start from 1800 to 18000 seconds. From one of experiment result on the Frequent Update, table IV and figure 7 show the Green Ajax's efficiency on the bandwidth usages.

TABLE IV. BANDWIDTH CONSUMPTION

Ajax	Bandwidth Consumption				
	5 seconds	*10 seconds*	*15 seconds*	*20 seconds*	*25 seconds*
Classic	72,699	36,156	24,339	18,179	14,644
Green	70,599	35,956	24,139	18,079	14,544

The figure 7 shows the significant margin of bandwidth consumption between Classic and Green Ajax.

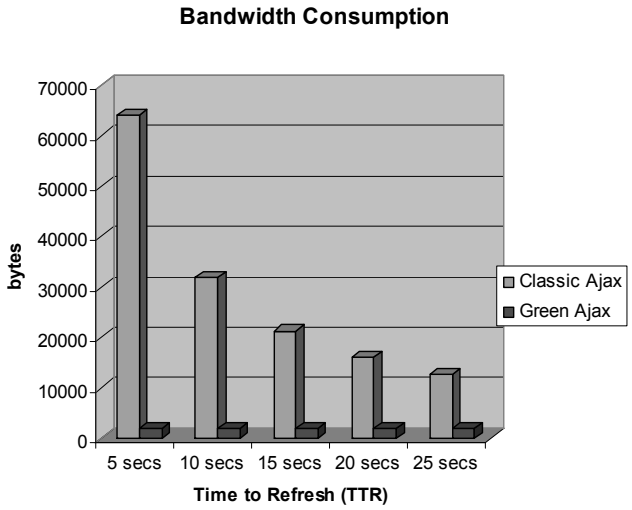


Figure 7. Bandwidth Consumption on High Frequency Update

Successful Receptive Percentage on Green Ajax is 100%. It technically confirms Green Ajax is successfully receiving an update on demand, but all of data from the server was lossless. In other hand, some requests from Classic Ajax applications to the web server do not get any new data, especially in the High Frequency. Moreover, the graph shows no data loss (DLP) in all sets of experiments. It verifies both Classic and Green Ajax succeed in all data obtained from the server.

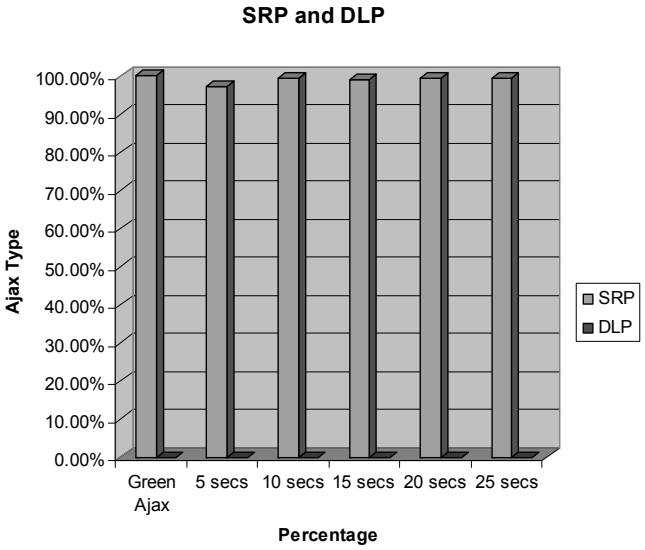


Figure 8. SRP and DLP

However in the third set, the experiment results show Green Ajax application consumes higher bandwidth than Classic Ajax. The complete results of experiments in the frequent update application can be listed on the table V.

TABLE V. RESULTS OF EXPERIMENTS

Group #	Situation	Better Results		
		Bandwidth	*SRP*	*DLP*
1	High Frequency Update	Green	Green	Both
2	Medium Frequency Update	Both	Green	Both
3	Low Frequency Update	Classic	Both	Both

In table V, Green Ajax outperforms on the High Frequency Update, in terms of bandwidth consumption, SRP, and DLP. Green Ajax also shows the better result on the Medium Frequency Update, especially in SRP. However, in the Low Frequency Update, Classic Ajax was proven as better approach but the margin of bandwidth consumption between them is not too significant.

C. *Fuzzy-based Application*

In this research, Fuzzy is utilized to make the experimental conditions close to real condition. Previously, experiments are tested using crisp three range of time. But they might be different with the actual conditions. Each range of time in the reality can be overlapping. As shown in the experiment in section 3.2, the range of time is assumed to be three basic types (high, medium and low) as listed in equation 1, 2 and 3.

$$High = \begin{cases} 1 & , \quad 5 \le x \le 25 \\ 0 & , \quad otherwise \end{cases} \tag{1}$$

$$Medium = \begin{cases} 1 & , \quad 25 \le x \le 900 \\ 0 & , \quad otherwise \end{cases} \tag{2}$$

$$Low = \begin{cases} 1 & , \quad 1800 \le x \le 18000 \\ 0 & , \quad otherwise \end{cases} \tag{3}$$

However, range of time in the real situation is never as precise as the division of time shown in table III. Each range of time can be overlapping as displayed in the table VI.

TABLE VI. ADJUSTED SCENARIO OF EXPERIMENTS

Group #	Scenario	
	Situations	*Interval Time*
1	High Frequency Update	5 seconds – 60 seconds
2	Medium Frequency Update	15 seconds – 30 minutes
3	Low Frequency Update	15 minutes – 5 hours

Evaluation of Green Ajax operation in mobile devices will be done by using the scenario in figure 9.

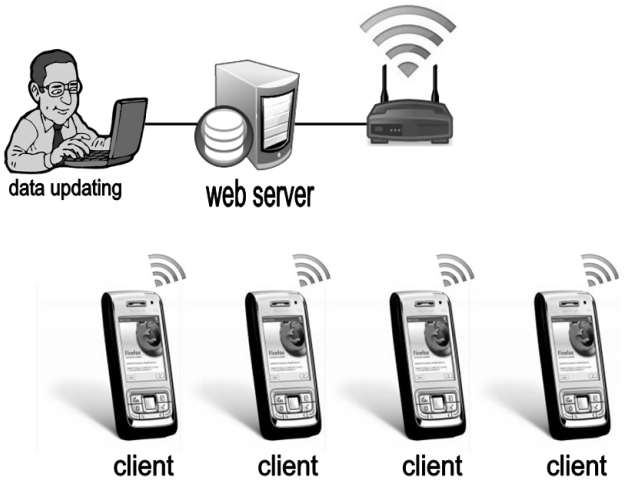


Figure 9. Experiment using Green Ajax on mobile devices

In figure 9, the experiment is performed by using the wireless network as a medium for accessing to the server. The mobile devices will perform the initial request on the first access to the web server. Every time the server performed the update, the server will launch a notification to the clients. Furthermore, the clients will request to the server for updated data and display the data on the specific parts of the web page.

The flow of process on signaling and requesting can be seen in figure 10. The web server will send a small size of signal to the mobile device when a new data arises. After the mobile device got the signal, the request will be sent to the web server. Furthermore, the web server will send all of the responses based on demand. A virtual server on the mobile device is required to receive a signal from the web server.

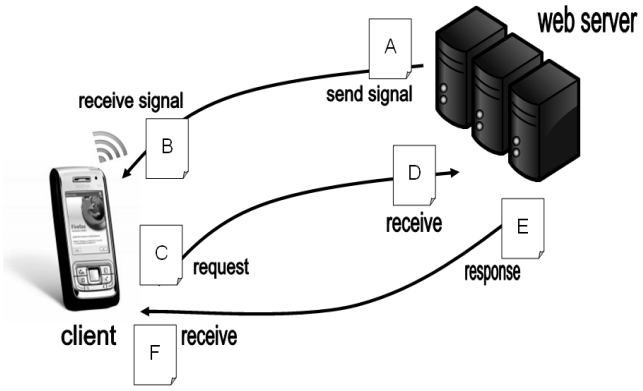


Figure 10. The model of Green Ajax Approach on Mobile Device

IV. GREEN AJAX UPON MOBILE TRAFFIC

The evaluation of Green Ajax upon mobile traffic is using the adjusted scenario shown in the table VI. Compared to the previous experiment on the desktop application, it has

more real situation. In addition, the assumptions of environment on mobile communication are zero delay between server and mobile devices, the speed up to 100 mbps [5], close to access point, and the mobile devices have Ajax support on its web browser.

In this experiment, the range of time displayed in figure 10 is used. The high frequency update application has interval time for data updating ranging from 5 seconds to 60 seconds. The medium frequency update has interval time to update from 15 seconds to 1,800 seconds or 30 minutes. The low medium frequency update has interval time since 900 seconds (15 minutes) to 18,000 seconds (5 hours).

In figure shown below, there are overlapping area between High and Medium Frequency Update and also between Medium and Low Frequency Update. These conditions are common in the real situations.

Figure 11. Range of time in the real situation

The membership of each situation on the figure 11 listed in equation 4, 5, and 6. The range of time is also assumed to be three basic types (high, medium and low) as shown below.

$$High = \begin{cases} 1, & 5 \le x \le 15 \\ \frac{x - 15}{45}, & 15 < x \le 60 \end{cases} \quad (1)$$

$$Medium = \begin{cases} \frac{x + 15}{45}, & 15 < x < 60 \\ 1, & 60 \le x \le 900 \\ \frac{x - 900}{900}, & 900 < x < 1800 \end{cases} \quad (2)$$

$$Low = \begin{cases} \frac{x + 900}{900}, & 900 < x < 1800 \\ 1, & 1800 \le x \le 18000 \end{cases} \quad (3)$$

Experiments based on the above ranges of time produce the results as shown in table VII. In the results, bandwidth consumption of Green Ajax is the lowest. Moreover, the margin of bandwidth consumption between Classic Ajax and Green Ajax is significant. It strongly confirms Green Ajax still being the most recommended approach for high frequency update in Fuzzy-based Application. In this experiment, overall results show Green Ajax is suitable to apply in Ajax-based applications.

TABLE VII. BANDWIDTH CONSUMPTION (IN BYTES) ON HIGH FREQUENCY UPDATE IN FUZZY-BASED APPLICATION

Interval Time	Bandwidth Consumption		
	Classic Ajax	*Green Ajax*	*Margin*
5 secs	545,243	529,493	15,750
10 secs	271,170	269,670	1,500
15 secs	182,543	181,043	1,500
20 secs	136,343	135,593	750
25 secs	109,830	109,080	750
30 secs	87,509	86,793	716

From the chart shown in the figure 12, the bandwidth consumption of Green Ajax is lower significantly than Classic Ajax, especially in the lower interval time. Overall, Classic Ajax has higher bar graphs than Green Ajax. It indicates Classic Ajax consumes higher bandwidth than Green Ajax on the Fuzzy-based application with high frequency updates.

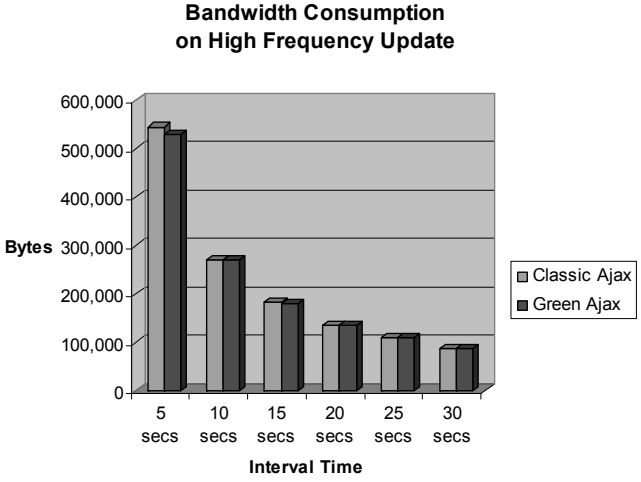


Figure 12. Range of time in the real situation

The evaluation is also done on the application with medium frequency update. The results of experiments can be demonstrated in the table VIII. In many experiments, Green Ajax shows that it consumes smaller bandwidth than Classic Ajax. Only few cases, Classic Ajax is a bit more efficient than Green Ajax. The large proportion Green Ajax in the experimental results above indicates that the Fuzzy-based applications are more suitable to Green Ajax.

TABLE VIII. BANDWIDTH CONSUMPTION (IN BYTES) ON MEDIUM FREQUENCY UPDATE

Interval Time	Bandwidth Consumption		
	Classic Ajax	*Green Ajax*	*Margin*
15 secs	182,543	181,043	1,500
20 secs	136,343	135,593	750
25 secs	109,830	109,080	750
30 secs	87,509	86,793	716
3 mins	14,458	13,742	716
5 mins	10,119	9,403	716
10 mins	5,063	5,063	0
15 mins	3,616	3,616	0
30 mins	1,074	1,085	-11

The comparison of bandwidth consumptions between Classic Ajax and Green Ajax are displayed in figure 13. Majority, Green Ajax has lower bar charts than Classic Ajax. This graph shows the Green Ajax is more efficient in bandwidth consumption. Even though in the last experiment whose interval time is 30 minutes Classic Ajax has lower bar graph, but it is not too significant.

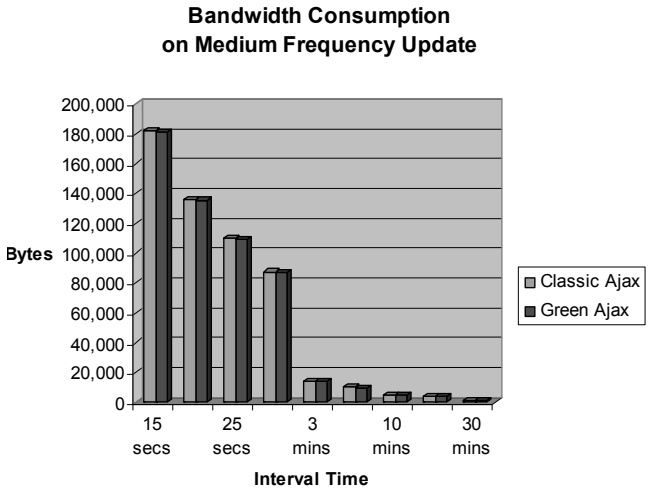


Figure 13. Bandwidth Consumption on Medium Frequency Update

Contrary with two previous experiments, evaluations on the applications with low frequency update shows the superiority of Classic Ajax compared to Green Ajax. Classic Ajax uses smaller bandwidth than Green Ajax. It shows Classic Ajax is suitable for application with low frequency update even though the margin is not significant.

TABLE IX. BANDWIDTH CONSUMPTION (IN BYTES) ON LOW FREQUENCY UPDATE

Interval Time	Bandwidth Consumption		
	Classic Ajax	*Green Ajax*	*Margin*
15 mins	3,616	3,616	0
30 mins	1,074	1,085	-11
1 hr	573	579	-6
2 hrs	286	289	-3
3 hrs	191	193	-2
5 hrs	191	193	-2

On figure 14, the bar graphs of Green Ajax are higher than Classic Ajax. The graphs indicate the Classic Ajax consume lower bandwidth. However, the difference between two bars is very small.

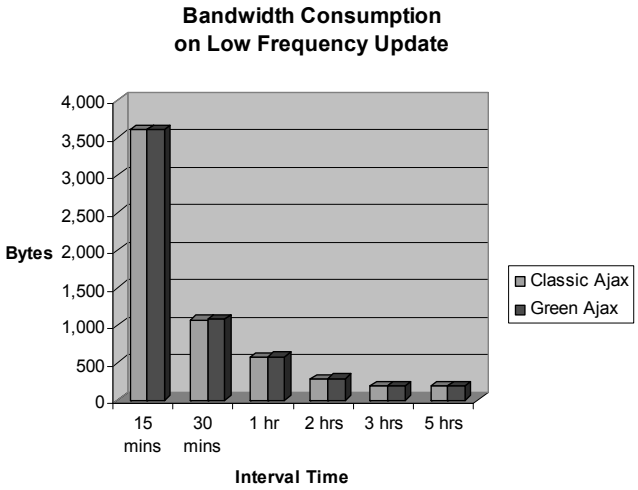


Figure 14. Bandwidth Consumption on Low Frequency Update

The complete result in this research can be seen in the table X. Green Ajax is recommended to be implemented on the Fuzzy-based application with high frequency update. Moreover, majority Fuzzy-based application with medium frequency update, Green Ajax is still suitable to be used. However, Classic Ajax is better than Green Ajax on handling data updating with low frequency in the Fuzzy-based application.

TABLE X. EXPERIMENT RESULTS OF FIXED TIME UPDATE APPLICATION

Group #	Scenario	
	Situations	*Better Approach*
1	High Frequency Update	Green
2	Medium Frequency Update	Majority Green
3	Low Frequency Update	Classic

V. CONCLUSIONS

Green Ajax has been proven as a good approach for web-based application on the desktop computer. The earlier experiment shows the advantages of Green Ajax than Ajax

Classic. Introducing the approach to the mobile devices is very helpful in easing mobile traffic. Low-traffic communication will help increase the speed of displaying information on mobile devices.

In this research, Fuzzy is utilized to make the experimental conditions close to real conditions. Each situation in the reality can be overlapping because no range of time which has really precise in division of time. In the Fuzzy-based application, mobile traffic has been reduced by using Green Ajax.

Overall, the consumption of bandwidth by Green Ajax is lower than classic Ajax. Adoption of Green Ajax on the Web-based mobile application is recommended to reduce the mobile traffic. In further research, adaptation of environment on 3G and 4G technologies will get more attention, besides a network bottleneck which probably occurs when numerous mobile gadgets access the network at the same time continuously.

ACKNOWLEDGMENT

This manuscript is fully granted from the Ministry of National Education of the Republic of Indonesia since 2008.

REFERENCES

[1] E.F. Martinez, "Universal Fuzzy System to Takagi-Sugeno Fuzzy System Compiler", Proceedings of the 2002 IEEE International Conference on Fuzzy Systems, Honolulu, HI, pp. 305-309 May 2002. doi: http://dx.doi.org/10.1109/FUZZ.2002.1005006

[2] G.C. Mouzouris and J.M. Mendel, "Dynamic Non-Singleton Fuzzy Logic Systems for Nonlinear Modeling", IEEE Transactions on Fuzzy Systems, Vol. 5, No. 2, pp. 199-208 May 1997. doi: http://dx.doi.org/10.1109/91.580795

[3] H. Kasai and N. Uchihara, "Mobile video Ajax technology for time-directional quick access", Proceeding of IEEE International Symposium on Wireless Communication Systems 2008, Reykjavik, pp. 144-148, December 2008. doi: http://dx.doi.org/10.1109/ISWCS.2008.4726035

[4] I. Saritas, N. Etik, N. Allahverdi, and I.U. Sert, "Fuzzy Expert System Design For Operating Room Air-Condition Control Systems", Proceedings of the 2007 International Conference on Computer Systems and Technologies, ACM New York, NY, USA, pp. IIIA.1-1-8, June 2007. doi: http://doi.acm.org/10.1145/1330598.1330625

[5] M.F. Lin, J.Y. Tzu, L. Lin, and H.M. Lee, "The IEEE802.11n Capability Analysis Model Based on Mobile Networking Architecture", Proceeding of IEEE International Conference on Systems, Man and Cybernetics 2009, San Antonio, TX, pp. 1857-1860, October 2009. doi: http://dx.doi.org/10.1109/ICSMC.2009.5346215

[6] Mesbah and A.v. Deursen, "An Architectural Style for Ajax", Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture, IEEE Computer Society Washington, DC, USA, pp. 9-9, January 2007. doi: http://dx.doi.org/10.1109/WICSA.2007.7

[7] R. Sanjaya and C. Brahmawong, "Distance Examination using Ajax to Reduce Web Server Load and Student's Data Transfer" International Journal of the Computer, the Internet and Management, Volume 15 SP3, Assumption University of Thailand, pp. 24.1-6, November 2007.

[8] 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, China, pp. 170-176, April 2010. doi: http://dx.doi.org/10.1109/ICIME.2010.5477773

[9] 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, June 2010. doi: http://dx.doi.org/10.1109/ICMIT.2010.5492884

[10] S. Blom, M. Book, V. Gruhn, R. Hrushchak, and A. K¨ohler, "Write Once, Run Anywhere A Survey of Mobile Runtime Environments," Proceeding of the 3rd International Conference on Grid and Pervasive Computing Workshops, Kunming, pp.132-137, May 2008. doi: http://dx.doi.org/10.1109/GPC.WORKSHOPS.2008.19

[11] The Mathworks, "MATLAB Fuzzy Logic Toolbox - User's Guide Version 2", July, 2002.

[12] Y. Xiao, Y. Tao, and W. Li, "A Dynamic Web Page Adaptation for Mobile Device Based on Web2.0", Proceeding of Advanced Software Engineering and Its Applications 2008, Hainan Island, pp. 119-122, December 2008. doi: http://dx.doi.org/10.1109/ASEA.2008.13

Green Ajax Implementation on the Wireless Local Area Network using Randomized Cue Applications

Ridwan Sanjaya, *Member, IACSIT*

***Abstract*—Green Ajax as a new approach in the web development has an aim to create interactive communication between server and client but avoiding the useless redundant requests. Some researches have been conducted to prove the benefit of Green Ajax, compared to Classic Ajax. The results of experiment show that the performance of Green Ajax is better than Classic Ajax in almost of all conditions. These results strengthen the conclusion that Green Ajax is a good approach in the web-based application development.**

However, some further researches have to be conducted in different environments to prove that the conclusion is also valid for a wider variety of conditions. One of those conditions is the wireless Local Area Network which is now very commonly available in schools, hotels, offices and business centers. Thus, the implementation of Green Ajax for the applications that support business, not just an experiment, can be realized immediately by using Randomized Cue Applications.

This paper will focus on the analysis of Green Ajax performance on the wireless Local Area Network. The program discussed will demonstrate the ability of Green Ajax in providing a good interaction between users and the web server. The results of the experiment confirm that the Green Ajax is suitable for the applications on the wireless Local Area Network. Network traffic reduction is obtained as the final result in the use of Green Ajax.

***Index Terms*—Green Ajax, network traffic, web-based application, wireless LAN.**

I. Introduction

Green Ajax is an enhancement of Classic Ajax which has been recognized in the web application. Inheriting from the origin, Green Ajax also has the ability to decrease the bandwidth consumption by delivering data only in the specific part of web page [8]. The user interfaces will display any information without any interruption on screen because the communication between web application and server is done in the background.

The improvement is located in the ability of web application to receive notification from the server whenever an update or a change of data occurs. When the application receives the server's cue, the application will then send out a request [7]. The applications will display the latest information without any involvement from the user. This method reduces required bandwidth below than necessary for data updating.

In the previous experiments done in the wired Local Area Network, Green Ajax has been proven as a suitable approach for web-based application in variety conditions. When the server updates the data in unpredictable time, Green Ajax applications are always able to be a better approach in displaying the latest data. Green Ajax sends fewer requests to the server compared to Classic Ajax. The less bandwidth consumption of Green Ajax reduces the network traffic [7].

Furthermore, Green Ajax is still able to display the latest data using smaller amount of requests than Classic Ajax even though the server updates the data at the scheduled time and Classic Ajax uses the same interval time of data updating in the server for its TTR [8]. In addition, Green Ajax is also able to save the bandwidth when the frequency of updates is very frequent (High Frequency Update). The same benefit can be seen when the frequency of update activity is medium (Medium Frequency Update).

Even though it is insignificant, Green Ajax uses larger bandwidth than Classic Ajax when the frequency of data updating is rare (Low Frequency). But in overall results, Green Ajax is dominating the success in comparison with Classic Ajax.

In the experiments using Fuzzy-based applications, the server extends the range of time for data updating on each classification of frequency [9]. The extension of range is done because there is no exact crisp between each classification in the reality. The Fuzzy exists between the High to Medium Frequency Update and Medium to Low Frequency Update. With the expansion area, this experiment became more real than previous experiments. As the results, the margin of bandwidth consumption between Green Ajax and Classic Ajax is very significant. Green Ajax uses less bandwidth than Classic Ajax when the frequency of updates is often and medium. When the frequency of updates is rare, Green Ajax uses slightly bigger bandwidth but insignificant compared to the Classic Ajax.

In this research, the server will perform the update in unpredictable time but limited 5 minutes as the maximum interval time of update. One client who uses Classic Ajax will perform data requests to the server periodically. The client uses a shorter time than the maximum value of data updating interval time on the server, which is 2 minutes. At the same

Manuscript submitted on March 18, 2011.

Ridwan Sanjaya is with the Graduate School of Information Technology, Assumption University, Bangkok, Thailand as the Ph.D. candidate, on leave from the Computer Science Faculty, Soegijapranata Catholic University, Semarang, Central Java Province, Indonesia (e-mail: ridwan.sanjaya@gmail.com)

time, the four clients who use the Green Ajax depend on the cue sent by the server. If the cue is received by the client, the client will make a request to the server. Even though the experiments use only five clients as seen on Fig. 1, it is possible to add more clients if the organization has more than five clients.

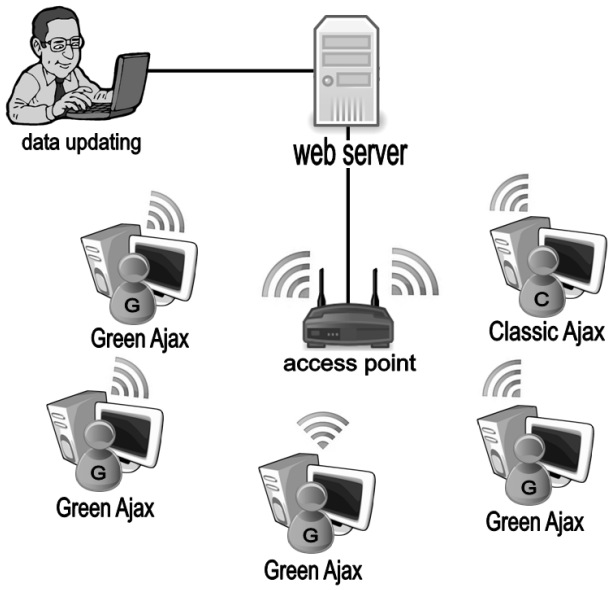


Figure 1. Five clients of Green Ajax and 1 client of Classic Ajax

The experiment was performed on a wireless Local Area Network using the assumption: the clients are located near the access point and there is no network disruption during the experiment.

The structure of the discussion in this paper is divided into five sections. Introduction and Literature Review will be discussed in section 1 and section 2. Then, Green Ajax assessments in wireless LAN environments will be discussed in section 3. The results of experiment on each client will be analyzed in section 4 and concluded in section 5.

II. Literature Review

The discussion of Green Ajax and Classic Ajax is not far from the discussion of client-side scripting. Both of them are built using JavaScript as a popular client-side language of scripting in the web programming. JavaScript is the client side scripting, which can control the interaction between user and the web page by calling the XMLHttpRequest to request data to the server without disturbing the web page [4]. After the data is obtained from XMLHttpRequest, the JavaScript will be used to manipulate the contents of a web page in Document Object Model (DOM) [1].

XMLHttpRequest is the main technology that makes Ajax engine able to request to the server and receive data from the server [10]. DOM is a structure within a web page, in which its content and visibility can be modified by using JavaScript [5]. When the value of the object is changed, the specific parts of the web page are also changed. The benefit, web application does not need to replace the entire page when requesting the latest data from the server. Only the needed data will be delivered to the specific part of the web page as seen on Fig. 2. It will reduce the bandwidth and increase the speed to display the content of web page.

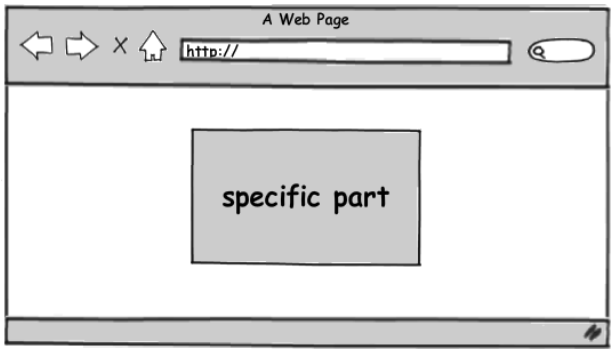


Figure 2. Delivering data only in the specific part of web page

However, the server will not send the data to the client automatically. JavaScript must issue requests from the client to the server by calling the methods in XMLHttpRequest then wait responses from the web server. It is common in practice that web application will repeat the requests at regular user-definable intervals known as Time to Refresh (TTR) [8] as seen on Fig. 3. If the responses are the same as the old data, the same information will be displayed on the client side. These unnecessary requests will waste the request time, response time and resources. In some circumstances, the interactivity of web application has to be paid with wasted bandwidth.

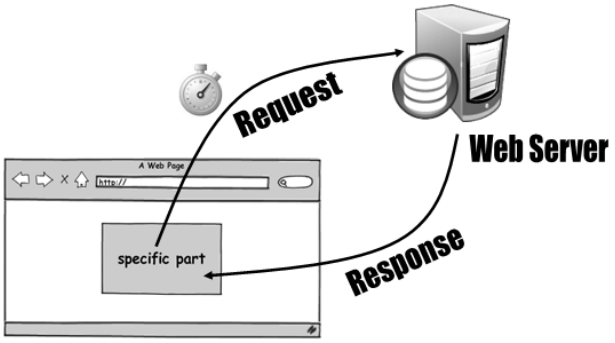


Figure 3. The content is requested from the server periodically

Green Ajax solves the problem by providing a new approach to send a cue from web server to the clients whenever an update arises. After the client receives a cue from web server, the client will issue the request to the web server [7]. Fig. 4 shows how the Green Ajax works that has been explained above.

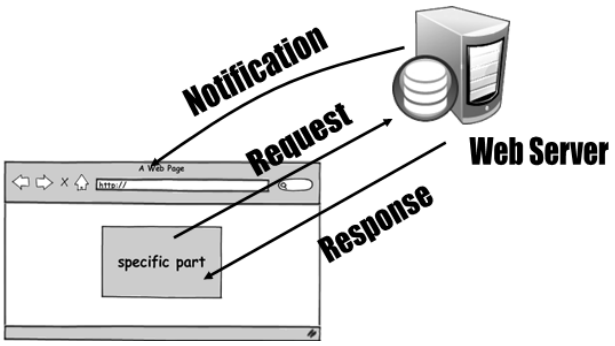


Figure 4. The content is requested from the server based on notification

This method reduces required bandwidth below than necessary for data updating. It is a suitable approach for bandwidth saving in any conditions, including infrequent and frequent update applications. From the experiment, Green Ajax is proved to cut successfully required bandwidth in presenting the latest updates from the server.

Fig. 5 shows when the administrator updates the database on the server, a cue will be released to the client by the server. Then, the client will send a request to the server and a new content will be received by the client from the server. The latest contents will appear inside a web page that is displayed on the screen the client.

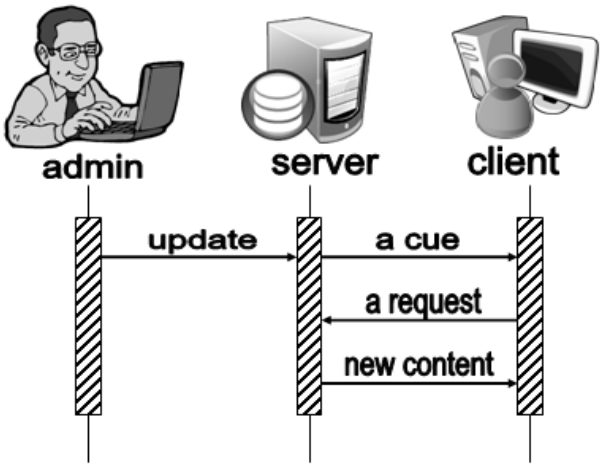


Figure 5. The cue arises when the administrator update the data

From the results, Green Ajax shows the best performance in Infrequent Update Applications. Those experiments were tested on the case of random and unpredictable update time on the web server. Green Ajax also shows its best performance in the case of fixed update time on the web server. Majority, Green Ajax can be used in any situation of update time.

TABLE I. EXPERIMENTS RESULTS OF THE CASE OF FIXED UPDATE TIME ON THE WEB SERVER

#	Situation	Better Results		
		Bandwidth	*SRP*	*DLP*
1	High Frequency Update	Green	Green	Both
2	Medium Frequency Update	Both	Green	Both
3	Low Frequency Update	Classic	Both	Both

In the experiments using Fuzzy-based Applications, Green Ajax is still able to be the most suitable approach in the real situation [9]. Those experiments have five time-zone-update times which consist of three independent zones and two strips.

TABLE II. EXPERIMENT RESULTS OF FUZZY-BASED APPLICATION

#	Scenario	
	Situation	*Better Approach*
1	High Frequency Update	Green
2	Medium Frequency Update	Majority Green
3	Low Frequency Update	Classic

The comparison of performance between Green Ajax and Classic Ajax will be done on the wireless Local Area Network. The client will connect to the server via an access point. Wireless Local Area Network is using high frequency radio waves 2.4 GHz (802.11b, 802.11g) or 5 GHz (802.11a) for its communications [11]. Wireless Local Area Network became popular because every computer is now equipped with Wi-Fi facility. The clients will join easily the Local Area Network without having to plug or unplug the network. Once the client is identified and validated on the network, the client can be connected to other computers in the network.

III. IMPLEMENTATION

In the Green Ajax implementation, a routine program is required in the server side to trigger a cue transmission to the client whenever an administrator does an updating. The routine program can be integrated to the update program, or can be separated modules which can be used together by other programs. A one byte character will be used as a cue that is sent to the clients. The reason of using one byte character is to limit the bandwidth usage, keep the client secure, and avoid harmful applications.

Each client on the wireless Local Area Network has their own identity in the form of an IP address. When the client connects to the server, the IP address of the client will be recorded by the server. That identity will be used as a target to receive the cue from the server when an administrator does an update. The illustration of the implementation can be seen in Fig. 6.

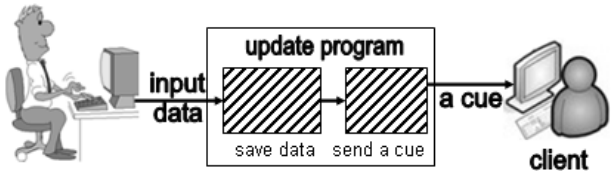


Figure 6. A routine program to trigger a cue transmission

The following PHP codes describe how the update program works. The RandomRefresh() function will set the next update time, CreateData() function will create random data for the experiments, and SendCue() function will trigger a cue transmission to the clients.

```
<?php
include “GreenAjax.php”;
$myRandomApp = new GreenAjax();
$myRandomApp->RandomRefresh();
$myRandomApp->CreateData();
$myRandomApp->SendCue();
?>
```

The small program on the client has the duty to receive the cue. It can be either a plug-in or extension that is installed in the web browser. POW (Plain Old Webserver) is an example of plug-in used in this research. A small program on the client will interpret the character received as a command to make a request to the server as seen in Fig. 7. This method is more

effective than using a repetition of the request based on TTR. Furthermore, new data will be sent to the client after the server receives such request.

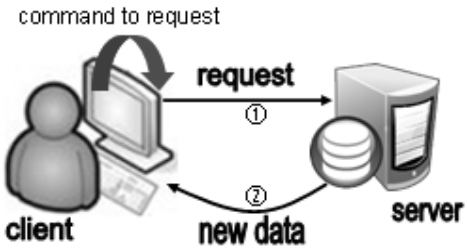


Figure 7. A small program embedded on the web browser

To make the plug-in works, the application has to be connected with the plug-in. The following PHP codes describe how the main program works. The initAjax() function will enable Ajax on the application. Then, initPlugin() function will create a connection to the plug-in. The Listen() function will receive a cue to trigger the client in making request to the server.

```
<?php
include "GreenAjax.php";
$myClientApp = new GreenAjax();
$myClientApp->initAjax();
$myClientApp->initPlugin();
$myClientApp->Listen();
?>
```

The initPlugin() function above will listen the cue sent by the server. Whenever a cue is received, the client will request using the following code to update the data. The readData() function will read the new data on the server and the Output() function will print the result. The request will be done in the background by using XMLHttpRequest, DOM, and JavaScript.

```
<?php
include "GreenAjax.php";
$myDataApp = new GreenAjax();
$myDataApp->readData();
$myDataApp->Output();
?>
```

After sending a cue, the clients who are still requesting to the server will be recorded by the server using their IP address. This mechanism makes the server work effectively in sending the cue to the clients on the next update. A Session Time can also be used to monitor the clients who are accessing the server.

In this research, a Randomized Cue Applications are used as a simulator to produce an update in the random time. Time decision of the next update is based on the random value generated by the application. The results obtained can describe the actual conditions when Green Ajax is used. Then, the bandwidth saving will be calculated using Firebug 1.5.4 as one of additional plug-ins on Mozilla Firefox 3.5.14. Firebug as seen in Fig. 8 has functions to debug, edit, and monitor any object on a web page [6]. In this research, its function to monitor is used to calculate the size of data transfer from/to the server.

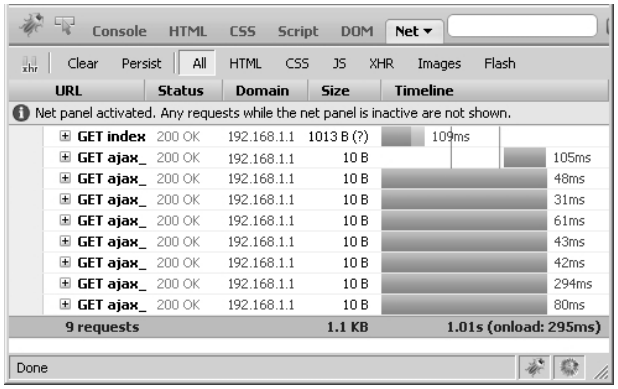


Figure 8. Mozila Firefox 3.5.14 and Firebug 1.5.4

IV. RESULTS AND ANALYSIS

The bandwidth consumption of Randomized Cue Applications will be compared with the Classic Ajax Application's bandwidth consumption. The Classic Ajax Application uses 2 minutes TTR to request the update data from the server.

From the results, the number of times data that had been received by the client (Rcv), and the number of times requests that had been issued by the server (Req) were compared to get the Successful Receptive Percentage (SRP) [7]. Data Loss Percentage (DLP) were calculated by comparing the number of times data was not received timely by the client (Los) and the number of times update data was activated by the server (Upd).

TABLE III. SRP AND DLP

Results[a]	Green Ajax				Classic Ajax
	1	2	3	4	
SRP (%)	100.00				69.54
DLP (%)	0.00				18.60

a. Experiment time: 5 hours

Table III shows the SRP of Green Ajax applications are 100%. It shows no wasting requests had been issued by Green Ajax. However, Classic Ajax only gets 69.54% data. It shows there are 30.56% requests to the server but do not get new data (wasting requests). In another result, DLP of Green Ajax applications are 0%. It means all data has been received by each Green Ajax application. However, the Classic Ajax gets 18.60% DLP. It shows there are 18.60% data updating from server that un-received by Classic Ajax.

The visualization results of experiments to compare Green Ajax and Classic Ajax can be seen on the Fig. 9. The SRP bars show that all SRP of the Green Ajax applications are very high compared to SRP of the Classic Ajax application which is only 69.54%. The DLP bars indicate that the DLP from all Green Ajax is very low compared to the Classic Ajax which can reach 18.60%.

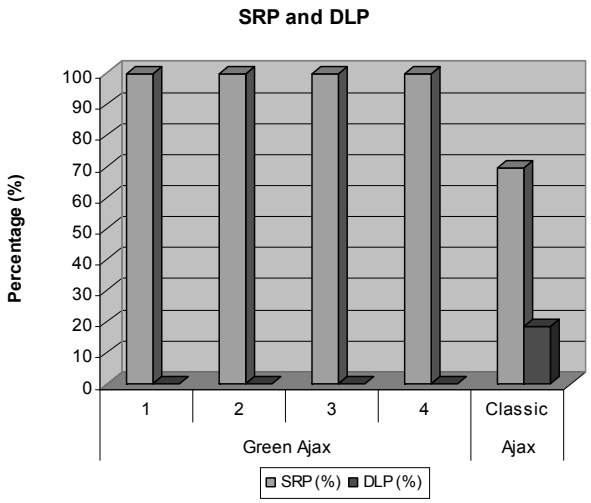


Figure 9. The result of SRP and DLP

The bandwidth consumption of five hours experiment can be seen on Table IV. It shows that bandwidth consumption of Green Ajax applications is smaller than Classic Ajax. In the Data Loss, there are 14,400 bytes data transferred by server which are not received by the clients. It is different from Green Ajax that does not lose any data.

TABLE IV. BANDWIDTH CONSUMPTION AND DATA LOSS (IN BYTES)

Results[a]	Green Ajax				Classic Ajax
	1	2	3	4	
Bandwidth	77,529				90,729
Data Loss	0.00				14,400

a. Experiment time: 5 hours

Based on table IV, the visualization of bandwidth consumption and data loss of each application can be seen on Fig. 10. The bars of bandwidth consumption of Green Ajax are shorter than the bar of bandwidth consumption of Classic Ajax. Other bars show that data loss of Green Ajax applications is very low compared to Classic Ajax which never receives 14,400 bytes data on the server.

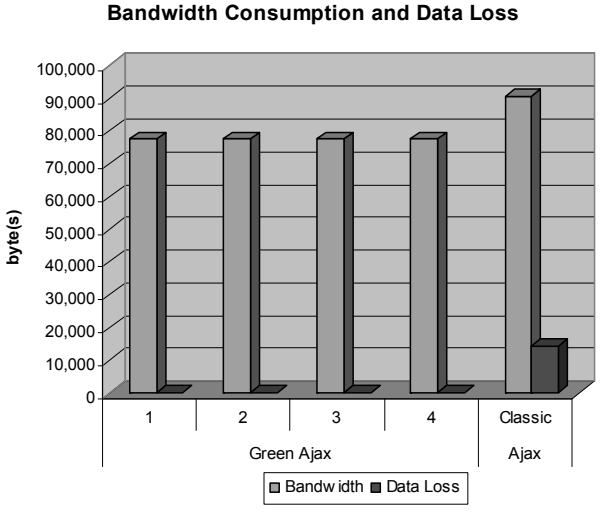


Figure 10. The Bandwidth Consumption and Data Loss

The above results show the benefit of Green Ajax Implementation on the wireless Local Area Network has been successfully proved by using Randomized Cue Applications. Even though the cue has a size that can lead to ineffectiveness, the results indicate that the cue helps the application to request more effectively.

V. CONCLUSIONS

Green Ajax performs excellently on the wireless Local Area Network. The implementation of Green Ajax is not limited to the media used for data transmission. Green Ajax performance in wireless Local Area Network is as good as tested on the wired LAN. By using Randomized Cue Applications, the real-time data will be showed on screen using less bandwidth. The bandwidth reduction is caused by two main reasons. First, not all portion of page should be updated. The technique to update on specific part of page can be done by collaboration between the XMLHttpRequest, DOM, and JavaScript [2]. Second, the number of requests to the server has been reduced by the clients. The cue from a server arises to tell the client to make requests. The technique will consume bandwidth only if there are any updates.

A small application on the client will receive a maximum of one byte character to trigger the client to make a request to the server. The reason to limit only one byte character is to limit the bandwidth usage, keep the client secure, and avoid harmful applications. The small application can be either a plug-in or extension that is installed in the web browser. POW is an example of plug-in used in this research. However, this application has common functions that are not used fully by Green Ajax. It is necessary to make a small application specifically required for the purposes of Green Ajax.

In further research, the scope of computer network will get more attention not only on a wired and wireless Local Area Network but also on a larger network. The identification of clients should not only depend on the IP address listed on the server that is sometimes not associated with a real IP address on the clients. However, using the current HTTP Protocol should be prioritized because it has been used widely. As a result, the benefits of Green Ajax can be implemented widely on the computer network.

ACKNOWLEDGMENT

This manuscript is fully granted from the Ministry of National Education of the Republic of Indonesia since 2008.

REFERENCES

[1] A. Marchetto, P. Tonella and F. Ricca, "State-Based Testing of Ajax Web Applications", *Proceeding of the 2008 International Conference on Software Testing, Verification, and Validation (ICST)*, Lillehammer, pp. 121-130, April 2008. doi: http://dx.doi.org/10.1109/ICST.2008.22.

[2] C.W. Smullen and S.A. Smullen, "AJAX Application Server Performance", *Proceeding of the IEEE SoutheastCon 2007*, Richmond, VA, March 2007, pp. 154-158. doi: http://dx.doi.org/10.1109/SECON.2007.342873.

[3] L. Zhijie, W. Jiyi, Z. Qifei, and Z. Hong, "Research on Web Applications Using Ajax New Technologies", *Proceeding of the 2008 International Conference Multimedia and Information Technology (ICIMT)*, Washington, DC, USA, December 2008, pp. 139-142. doi: http://doi.ieeecomputersociety.org/10.1109/MMIT.2008.107

[4] N. Hanakawa and N. Ikemiya, “A Web Browser for Ajax Approach with Asynchronous Communication Model”, *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI-06)*, Hong Kong, China, pp. 808-814. doi: http://dx.doi.org/10.1109/WI.2006.30.

[5] N. Matthijssen, A. Zaidman, M.A. Storeyy, I. Bully and A.v. Deursen, “Connecting Traces: Understanding Client-Server Interactions in Ajax Applications”, *Proceeding of the 2010 IEEE 18th International Conference on Program Comprehension (ICPC)*, Braga, Minho, pp. 216-225, July 2010. doi: http://dx.doi.org/10.1109/ICPC.2010.14.

[6] P. Hope and B. Walther, *Web Security Testing Cookbook*, Sebastopo;, CA: O'Reilly Media, 2009.

[7] 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, China, pp. 170-176, April 2010. doi: http://dx.doi.org/10.1109/ICIME.2010.5477773

[8] 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, June 2010. doi: http://dx.doi.org/10.1109/ICMIT.2010.5492884

[9] 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)*, to be published.

[10] R. Sanjaya and C. Brahmawong, “Distance Examination using Ajax to Reduce Web Server Load and Student’s Data Transfer”, *International Journal of the Computer, the Internet and Management (IJCIM)*, Volume 15 SP3, Assumption University of Thailand, pp. 24.1-6, November 2007.

[11] T. Soungalo, L. Renfa, and Z. Fanzi, "Evaluating and Improving Wireless Local Area Networks performance," Proceeding of the 2010 3rd International Conference on Information Sciences and Interaction Sciences (ICIS), pp.367-372, June 2010. doi: http://dx.doi.org/10.1109/ICICIS.2010.5534802

Ridwan Sanjaya has been working as a lecturer in Soegijapranata Catholic University, Semarang, Indonesia since 2002. This author became a member of IEEE, IACSIT, and IEICE. He received the Master of Science in Internet and E-Commerce Technology (MS.IEC) degree from Assumption University, Bangkok, Thailand in 2007. Currently he is pursuing his Doctoral degree in Computer Information Systems (CIS) at Graduate School of Information Technology, Assumption University, Bangkok, Thailand. He have been publishing more than 60 books related to computer area such as Web Development with JSP, Graphic Engineering using PHP, PDF Report Development with PHP 5.0, Cross-Platform Computer Network Administration, Creative Digital Marketing, Business-Driven Information System, etc.

APPENDIX B

EXPERIMENT DATA PROCESSING

Each experiment automatically records the update time on server and the refresh time on clients. Those data will be summarized and stored on the Experiment Data Form. For the example is the experiment of Low Frequency Update on Trade-off Analysis for Web Application Using Green Ajax. The server updates the new data every 2 hours and there are 1 client using Classic Ajax and 1 client using Green Ajax. The Classic Ajax client uses 2 hours as the interval time to request (TTR) or equal with the time to update on the server.

Table B.1. Recorded Data using Interval Time 2 hours

No	Server Update Time	Refresh Time	
		Green Ajax	Classic Ajax (TTR 2 hours)
1	09:10:46	09:10:47	09:11:13
2	11:10:46	11:10:48	11:11:13
3	13:10:46	13:10:47	13:11:13

a. Experiment time: 5 hour

The recorded data above will be summarized and stored on the data form below. The above table shows there are three update on the server (Upd). Both of Ajax requests (Req) three times and received three times. It means no request got the old data or Wasting equal with zero. Moreover, because the clients have all data from the server, then value of Un-received (Los) can be filled with zero. Server (Srv) is equal with 3 indicates the server send the signal three times to the Green Ajax client.

Table B.2. Experiment Data Form Low Frequency Update

Data	Green Ajax	Classic Ajax 2 hours
Requests (Req)	3	3
Received (Rcv)	3	3
Wasting (Wst)	0	0
Updates (Upd)	3	3
Un-received (Los)	0	0
Server (Srv)	3	0

a. Experiment time: 5 hour

Based on the scenario, the experiment using Low Frequency Update will be tested uses 30

minutes, 1 hour, 2 hours, 3 hours, and 5 hours as the time to update on the server. The two tables below show the recorded data and the experiment data form using 3 hours as the interval time to update on server and interval time to refresh on the Classic Ajax client.

Table B.3. Recorded Data using Interval Time 3 hours

No	Server Update Time	Refresh Time	
		Green Ajax	Classic Ajax (TTR 3 hours)
1	09:12:05	09:12:06	09:12:39
2	12:12:05	12:12:07	12:12:39

a. Experiment time: 5 hour

Table B.4. Experiment Data Form Low Frequency Update

Data	Green Ajax	Classic Ajax 3 hours
Requests (Req)	2	2
Received (Rcv)	2	2
Wasting (Wst)	0	0
Updates (Upd)	2	2
Un-received (Los)	0	0
Server (Srv)	2	0

a. Experiment time: 5 hour

The data collection based on the scenario will be used to calculate SRP, DLP, and bandwidth consumption. The table below show the SRP result on Low Frequency Update listed on the table 4.18. The formula of SRP and DLP can read on the Chapter 3.

Table B.5. SRP on Low Frequency Update

Scenario[a]	Interval Time				
	30 mins	*1 hr*	*2 hrs*	*3 hrs*	*5 hrs*
Classic Ajax	100.00%	100.00%	100.00%	100.00%	100.00%
Green Ajax	100.00%	100.00%	100.00%	100.00%	100.00%

a. Experiment time: 5 hour

BIBLIOGRAPHY

Blom, S., Book, M., Gruhn, V., Hrushchak, R., & Kohler, A. (2008). Write Once, Run Anywhere A Survey of Mobile Runtime Environments. *Proceeding of the 3rd International Conference on Grid and Pervasive Computing Workshops*, 132-137.

Chen, C.L., & Raman, T.V. (2008). AxsJAX: A Talking Translation Bot Using Google IM : Bringing Web-2.0 Applications to Life. *Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A)*, 54-56.

Converse, T., Park, J., & Morgan, C. (2004). *PHP5 and MySQL Bible*. Indianapolis, Indiana: Wiley Publishing.

Dahlan, A.A., & Nishimura, T. (2008). Implementation of Asynchronous Predictive Fetch to Improve the Performance of Ajax-Enabled Web Applications. *Proceeding of the 10th International Conference on Information Integration and Web-based Applications & Services*, 345-350.

Gal, A., & Mylopoulos, J. (2001). Toward Web-Based Application Management Systems. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 13, No. 4, July/August 2001, 683-702.

Garrett, J.J. (2005). Ajax: A New Approach to Web Applications. Retrieved from http://adaptivepath.com/publications/essays/archives/000385.php

Jablonski, S., Petrov, I., Meiler, C., & Mayer, U. (2004). *Guide to Web Application and Platform Architectures*. New York: NY, Springer-Verlag Berlin Heidelberg.

Jiaqi, W., Jie, L., & Shujuan, W. (2009). The Realization of Google Suggest in Mis System by Using Ajax. *Proceedings of the 2009 International Forum on Computer Science-Technology and Applications*, 02, 93-96.

Kasai, H., & Uchihara, N. (2008). Mobile video Ajax technology for time-directional quick access. *Proceeding of IEEE International Symposium on Wireless Communication Systems 2008*, 144-148.

Kletsch, C., & Volk, D. (2008). Towards an AJAX-based Game Engine. *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, 270-271.

Lin, M.F., Tzu, J.Y., Lin, L., & Lee, H.M. (2009). The IEEE802.11n Capability Analysis Model Based on Mobile Networking Architecture. *Proceeding of IEEE International Conference on Systems, Man and Cybernetics 2009*, 1857-1860.

Lin, Z., Wu, J., Zhang, Q., & Zhou, H. (2008). Research on Web Applications Using Ajax New Technologies. Proceeding of the 2008 International Conference Multimedia and Information Technology, 139-142.

Mahemoff, M. (2006). *Ajax Design Patterns*. Sebastopol, CA: O'Reilly Media, Inc.

Marchetto, A., & Tonella, P. (2009). Search-Based Testing of Ajax Web Applications. Proceedings of the 2009 1st International Symposium on Search Based Software Engineering, 3-12.

Martinez, E.F., (2002). Universal Fuzzy System to Takagi-Sugeno Fuzzy System Compiler. *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems*, 305-309.

Mathworks (2002). *MATLAB Fuzzy Logic Toolbox - User's Guide Version 2*.

McFarland, D. (2005). *Dreamweaver 8: The Missing Manual* (5th ed.). Sebastopol, CA: O'Reilly Media, Inc.

Merrill, C. (2006). Using Ajax to Improve the Bandwidth Performance of Web Applications. Retrieved from http://ajaxian.com/archives/using-ajax-to-improve-the-bandwidth-performance-of-web-applications

Mesbah, A., & van Deursen, A. (2007). Migrating Multi-page Web Applications to Single-page AJAX Interfaces. *Proceedings of the 11th European Conference on Software Maintenance and Reengineering*, 181-190.

Mesbah, A., & van Deursen, A. (2007). An Architectural Style for Ajax. *Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture*, 9-9.

Mouzouris, G.C., & Mendel, J.M. (1997). Dynamic Non-Singleton Fuzzy Logic Systems for Nonlinear Modeling. *IEEE Transactions on Fuzzy Systems*, 5, 199-208.

Pilgrim, C.J. (2008). Improving the Usability of Web 2.0 Applications. *Proceedings of the 19th ACM Conference on Hypertext and Hypermedia*, 239-240.

Redouane, A. (2002). Guidelines for Improving the Development of Web-Based Applications. Proceedings of the Fourth International Workshop on Web Site Evolution (WSE'02), 93.

Sanjaya R. (2011b). Green Ajax Implementation on the Wireless Local Area Network using Randomized Cue Applications. *International Journal of Information and Education Technology (IJIET)*, Vol. 1, No. 1, 63-67.

Sanjaya, R. (2011a). Mobile Traffic Evaluation for Fuzzy-Based Application Using Green Ajax. *Proceeding of IEEE International Conference on Information and Education Technology (ICIET)*, V2.34-41.

Sanjaya, R. (2010b). Trade-off analysis for web application using Green Ajax. *Proceeding of IEEE International Conference on Management of Innovation and Technology (ICMIT)*, 1050-1054.

Sanjaya, R. (2010a). Web Traffic Reduction for Infrequent Update Application Using Green Ajax. Proceeding of the 2nd IEEE International Conference on Information Management and Engineering (ICIME), 170-176.

Sanjaya, R., & Brahmawong, C. (2007). Distance Examination using Ajax to Reduce Web

Server Load and Student's Data Transfer. *International Journal of the Computer, the Internet and Management*, 15 SP3, 24.1-6.

Sanjaya, R., & Sribhadung, P. (2006). Web 2.0 and Its Implementation to eLearning. *International Journal of the Computer, the Internet and Management*, 14 SP1, 47.1-8.

Saritas, I., Etik, N., Allahverdi, N., & Sert, I.U. (2007). Fuzzy Expert System Design For Operating Room Air-Condition Control Systems. *Proceedings of the 2007 International Conference on Computer Systems and Technologies*, IIIA.1-1-8.

Serbinski, A., & Abhari, A. (2007). Improving the Delivery of Multimedia Embedded in Web Pages. Proceedings of the 15th international conference on Multimedia, 779-782.

Smith, K. (2006). Simplifying Ajax-Style Web Development. Computer, 98-101.

Thiesen, P., & Chen, C. (2007). Ajax Live Regions: Chat as a Case Example. *Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility (W4A)*, 7-14.

White, A. (2005). Measuring the Benefits of Ajax. Retrieved from http://www.developer.com/xml/article.php/3554271

Xiao, Y., Tao, Y., & Li, W. (2008). A Dynamic Web Page Adaptation for Mobile Device Based on Web 2.0. *Proceeding of Advanced Software Engineering and Its Applications 2008*, 119-122.

Zakas, N.C., McPeak, J., & Fawcett, J. (2006). *Professional Ajax*. Indianapolis, Indiana: Wiley Publishing.

Zapeda, J.S., & Chapa, S.V. (2007). From Desktop Applications Towards Ajax Web Applications. *Proceedings of the Fourth International Conference on Electrical and Electronics Engineering (ICEEE) 2007*, 193-196.

Zhang, S., & Zhang, S. (2010). Cloud Computing Research and Development Trend . *Proceedings of the Second International Conference on Future Networks (ICFN) 2010*, 93-97.