Asynchronous Communication for Improved Data Transport in The Network

Asynchronous Communication for Improved Data Transport in The Network

S Zepeda* and A Nuñez

Department of Information Technologies, Metropolitan Autonomous University, Division of Communication Sciences and Design, Vasco de Quiroga 4871, Contadero, Cuajimalpa de Morelos, 05370, Mexico City, Mexico

*Email: sergioz@dccd.mx, albanunez@dccd.mx

Abstract. Currently there are two types of interaction in web systems mediated by synchronous and asynchronous communication, each of them with its programming and interaction paradigms. Although intuitively we can say that asynchronous communication is more efficient in data transport, there is not much research on this topic. We present a comparison of these two types of communication by requesting the same data from a server and receiving a client. The requests are through the traditional web forms and an experimental semantic interface, which only needs the data in plain text as a response from the server. The results found are of interest for the problem of excessive traffic on the network. This research can open new lines of investigation or generate new paradigms and programming models to improve communication between client and server.

1. Introduction

The creation of the Internet has been a great revolution in the context of communication in humanity, the evolution of communication on the internet has generated the creation of advanced web systems, which have grown in complexity over time [1]. Nowadays, the creation of a web system needs the interaction and communication between different types of languages and software to obtain the required functionality by final users [2]. The programming of a web system requires tag-based languages such as: HTML, XHTML, XML, HTML5, styles sheet such as CSS, client Script languages such as: JavaScript, Visual Basic Script, Action Script, etc. [3]. Server-side script languages such as: PHP, JSP, ASP, etc., Database Management System (DBMS) such as: PostgreSQL, MySQL, etc. [4]. And web server software such as: Apache, Tomcat, Internet Information System (ISS), etc. [5]. In this way, the complexity of the programming increases dramatically with the complexity of the system because there are many actors that need to be synchronized to work together [6].

A web system has two types of communication to interact in the client-server model: synchronous and asynchronous communication. According to the website Internet Live Stats, currently there are 2 billion websites in the world, where the vast majority of them use synchronous communication as a means for interaction in web systems. This is mainly because web forms are the basic and common way to access databases [7]. Development platforms and programmers continue to use web forms as the main part in the interaction of web systems based on synchronous communication and not taking advantage of the potential of asynchronous communication. This research focuses on showing the differences in network traffic when using these two types of communication.

2. Background

Synchronous communication has been part of the evolution of the web since its creation; the vast majority of web systems use this interaction model. However, since the mid-90s the asynchronous model began to emerge slowly and almost invisibly for purposes of commercial advantage. However, it was not until Jesse James Garret coined the term Ajax that asynchronous communication began to gain notoriety [8]. The term Ajax represents a new way of programming by mixing different languages to provide a rich interaction in web systems [9]. New types of web systems started to emerge mainly based on interfaces oriented to respond to events, so new architecture proposals emerged [10]. This also resulted in improved techniques for manipulating the structure in an HTML document through the Document Object Model (DOM) by means of client script language [11]. Some issues related to communication and data transport emerged to be investigated [12-14]. The interfaces of these web systems with asynchronous communication were designed mainly with an interface to respond to events that required very basic queries to only help in part of the interaction as in the common suggestion boxes. Some proposals emerged to take greater advantage of this type of communication, some proposals based on data semantics, but this did not manage to expand given the nascent notion of giving meaning to data. Nowadays with the emergence of the internet of things, this asynchronous communication gains attention again due to the constant communication that the devices must have with each other [15,16]. On this topic, we work on an experimental semantic interface that builds dynamic queries on the fly automatically for queries on scientific data oriented to microbiology, the interface is capable of performing unplanned and user-generated queries [17]. Our purpose in this research is to compare the data transport from the server to the client side under the traditional method based on web forms and the experimental semantic interface based on asynchronous communication, and with it, we can observe how the performance in the data transport is improved with respect to the traditional method.

3. Methods

Our study is focused on knowing the variations in data transport between the two types: synchronous and asynchronous communications. Synchronous communication is the most widely used, and the vast majority of web systems with database access use it for interaction with web forms.

Now, we show all the processes that occur when interacting with a web form, in order to understand some of the disadvantages of this form of interaction in data traffic. The whole interaction process is described in figure 1.

Interaction process with web forms
The user interacts with the web form.
Client side waits for client writing on the web form.
The user writes data and presses submit button.
Client side sends data to the server.
Server side verifies data and builds query.
Server side gets the data.
Server side creates a new page.
Server adds the new data to page.
Server side sends a complete new page to client side.
Client side gets the new page.
The user views the new page.
The user needs to go to a new web form.
The client side is waiting for a new query from the web form.

Figure 1. The Figure shows the interaction between client and server side, we can see that the server does much of the work. When the user interacts through web forms, it must wait for the answer to continue interacting with the system.

The most important characteristics to highlight in this type of communication are:

- The interaction requires two web pages, the first containing the form to send the data and the second which is obtained from the server with the response.
- The server needs to build a complete page every time it responds to a request.
- The server sends complete pages, which constantly transports repeated data of the page structure in each response.

On the other hand, to test the asynchronous communication, an interface based on semantic queries was created; this interface uses a visual interaction where the elements contained in it contain listening events that trigger queries when the event is pressed on an element of the interface. The client side manages this whole process and sends data to the server to build a query and the server returns the data in plain text. When data is returned the client side controls the process of incorporating it into the interface dynamically by creating tags on the fly for each piece of data, adding and modifying the internal structure of the document. Figure 2 shows this process by means of an interaction behavior:

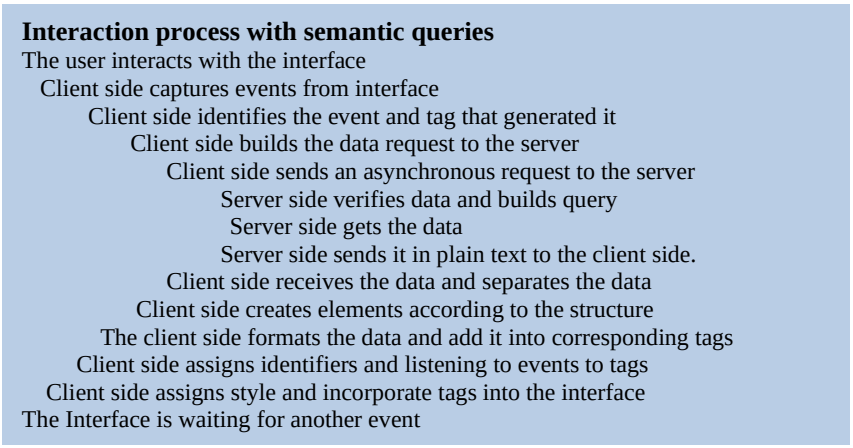

Figure 2. The figure shows how the asynchronous interaction modifies the workload towards the client side. The server does not build a complete page, it only retrieves data in plain text and sends it to the client side, which is responsible for formatting and creating the structure to incorporate it into the structure of the html document dynamically.

The most important characteristics to highlight in this type of communication are:

- The initial page is only sent the first time, and then it is constantly updated given the new data incorporated in each request.
- The largest workload is concentrated on the client side.
- Event capture, request submission, information retrieval, dynamic tag generation, HTML structure update, the client side controls everything.
- The client side only receives plain text minimizing data transport.

Now, with these two types of processes we are going to generate the same query and compare the results obtained. In this research, only the response data sent by the server to the client are of interest, since we are interested in knowing the amount of data transport traffic in each response.

4. Results

Now in this section, we show the results obtained when requests are made with traditional web forms, versus an interface with an experimental semantic interface based on asynchronous communication. The request is the same each time, seven different requests are made with which it is possible to obtain the continuous pattern for a number n of times, and the first results are shown in table 1.

Table 1. The table shows the number of bytes transported in each new request from the client side to the server side.

Request	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Synchronous	4144	4359	4946	5147	7038	9293	10519
Asynchronous	4144	67	132	48	482	504	524

Table 1 shows a first request that contemplates the construction of the initial structure of the page without data, these data do not include images, CSS or Scripts because these vary depending on the design and the main interest in this research is the transport of data in each request. The first request the server sends the complete page so it sends the same amount of bytes, but in the following requests, we can see the difference between the two types of communication. Once we have obtained this data, we can observe in a graph the difference in behavior as shown in figure 3.

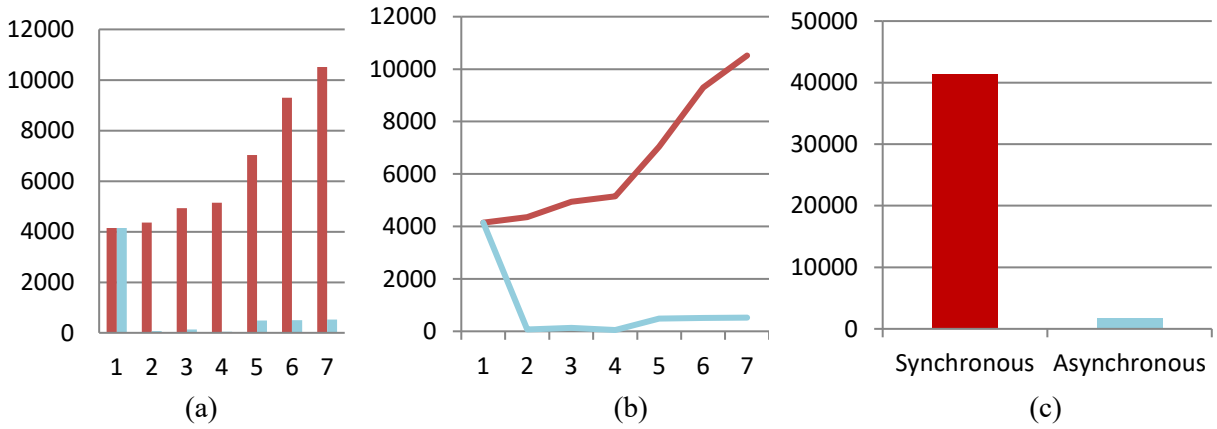

Figure 3. The figure shows the difference of bytes transported in each request, the behavior is interesting, because it starts the same and with every request, there are big differences

Figure 3 shows three graphs that show in red the data transport in synchronous communication and in cyan the data transported in asynchronous communication. Figure 3 (a) shows how when communication is done synchronously the server needs to transport the whole structure again and again of the same page, but with different data. On the other hand, in asynchronous communication, a minimum of data transport can be observed, which is very close to the minimum possible. Figure 3 (b) shows in another visual representation how the synchronous graph tends to be exponential while the asynchronous one tends to be linear. Finally, figure 3 (c) shows the totality of transported data where the first one observes 41292 bytes (41K) in total, while the asynchronous total is 1757 bytes (1 K), this shows a difference of 39535 bytes between one and another to show in an interface the same information. This means that if a page is consulted thousands, or millions of times it continuously transports unnecessary repeated and redundant data of the structure of each page. These graphs show the result for a simple basic consultation, but if we look further imagine for hundreds of millions of pages it is easy to see that due to this situation there is unnecessary and useless traffic on the network.
Although the asynchronous recovery of plain text data allows a lot of efficiency, this can be done in different ways. Data recovery can be done through XML, JSON format or between HTML, XHTML, HTML5 tags. Each format must comply with certain specifications, commonly the data must be enclosed in tags according to the specified recovery format.

Table 2. The table shows the number of characters needed in different formats to send a string in this case: data1.

Type	Data	Total
PlainT	data	**4**
XML	<ref:name>data</ref:name>	25
HTML5	data	17
JSON	{ “name” : “data” }	19
PlainTS	data¬	**5**

Table 2 shows the types of formats used to send from the server to the client, in the first column we see the data to be sent in plain text, in this case only four characters need to be sent. But the server does not send the data this way, because the client side cannot distinguish when several data are sent. Therefore, each data needs a separator or tags that distinguish one from another. The second column shows that if the data is sent in XML format it needs 25 characters. Although this number varies depending on the name assigned in the semantic logic that the programmer assigns to the tags. The third column shows HTML5 format, and this number also varies according to the specific tag used, depending on the type of data sent. The fourth column shows the JSON format commonly used by many programmers and it also varies according to the name assigned to the label.
The fifth column PlainTS (Plain Text-Symbol) which is the format used for this study, it is composed only of the data and a separation character, thus the number of characters carried by data = data + 1 character.

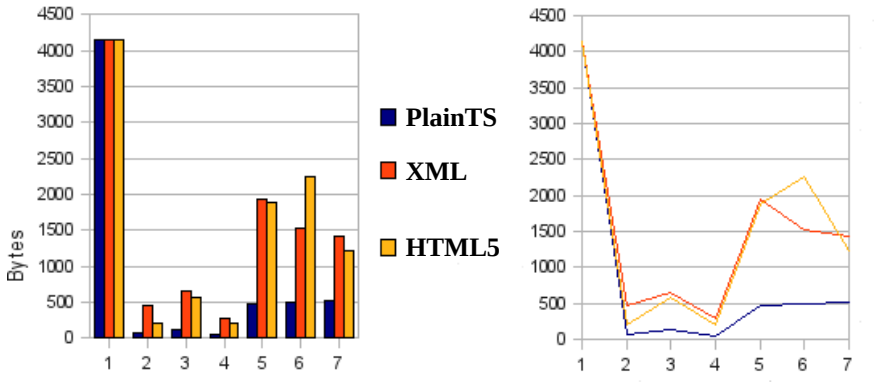

Figure 4. The figure shows the difference in characters transported according to the sending format from the server.

Figure 4 shows how depending on the format used, the number of characters accompanying the data may vary. However, the format used by us PlainTS also offers visible advantages in terms of data transport. This format is very efficient since it is only the data and a separation character. Occupying this format involves generating client script code to perform data separation and can be embedded in the HTML document. The arrival format Plain TS is as follows:: data1¬data2¬data3¬...data$_n$.

For any format such as XML, JSON etc, there are complete libraries which are often unnecessarily large and the programmer depends on their understanding and use of some features. Also, these libraries are composed of thousands and thousands of lines of code completely unnecessary in the vast majority of cases, which brings us back to the problem of transporting completely useless code and causing unnece ssary traffic. Due to this situation we found that having a plain text with only one symbol to differentiate between them, helps greatly to avoid transporting libraries with thousands of lines of unusable code. The PlaintTS format creates a tagless atomic data, and maximizes the efficiency of data transport, because it only moves the necessary data by greatly reducing unnecessary data traffic.

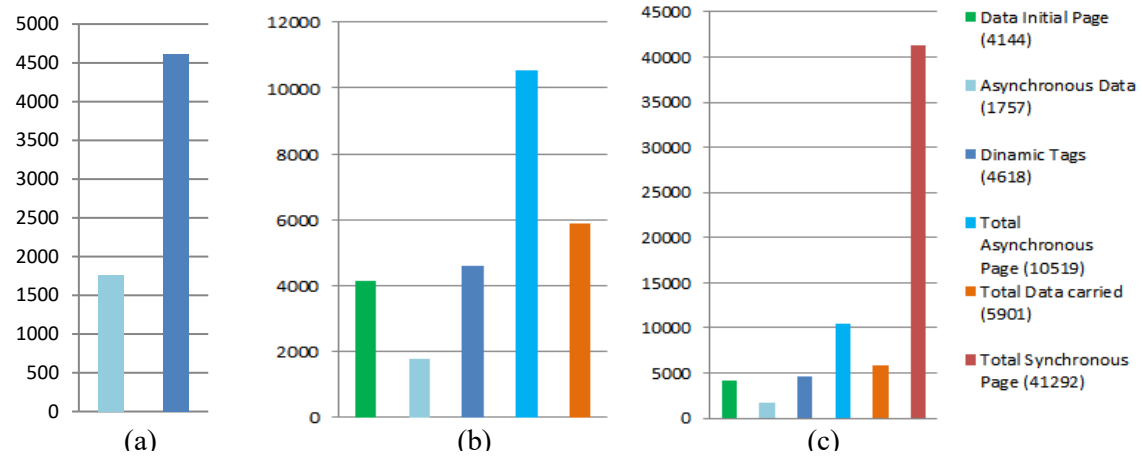

Figure 5. The figure shows the relationship between all parties involved as: data transported in the initial page, asynchronous data transported, dynamically generated tags, and the comparison between data transported between the interaction models.

This means that there must be more programming on the client side, in order to maintain control of the requests and control of the data as it is received. As the data arrives without labels these need to be generated in a dynamic and automatic way, which the use of DOM helps us to achieve this objective, generating the tags on the client side also affects data transport, as can be seen in figure 5 (a) .With all these results we can get an overview of how programming in a synchronous way generates some inefficiency in the communication between client and server. The figure 5(a) shows the amount of bytes generated dynamically and automatically from the client side, which implies that they were not transported. So, from 6385 bytes needed, only 1757 were transported, which multiplied by millions of queries if everyone worked in this way would imply a decrease in network traffic very large.

Figure 5 (b-c) shows very interesting data that allows to generate discussion about the need to evolve the current programming and generate new paradigms based on asynchronous communication. In the figure 5 (b) we can visualize in the first column the bytes transported the first time to transport the initial page with 4144 bytes beginning. The second column shows the number of bytes of the data transported from the queries in total 1757 bytes. The third column shows the total number of bytes generated by the labels generated automatically and dynamically on the client side 4618 bytes. The fourth column shows the total number of bytes contained in the final page at the end of the interaction 10519 bytes with the data included. Here we can see an interesting fact, of the total 100% in bytes of the final page, only 56% was transported from the server to the client and 44% was generated dynamically and automatically on the client side, this means a shared work between client and server. Because, the client side worked that 44% and reduced the server load by processing less data and sending less amount of data, which impacts the overall performance and in the network. The fifth column shows the total data transported in an asynchronous way and represents the 56% mentioned of the total data in the final page.

Figure 5 (c) shows the comparison of the above graph with the total data carried by the synchronous method. Now we can observe an 86% reduction of data transport of the asynchronous vs.

synchronous part, this reduction is very important, because if we multiplied by the millions of existing sites and all used reduced asynchronous communication we would have 86% less network traffic. The inverse relationship shows that 700% more bytes in the use of the traditional method of web form. All these results allow to open the discussion about the future need of new programming techniques, which allow efficiency in data transfer between client and server. We occupied an experimental semantic interface with asynchronous communication for exploration purposes in scientific databases, and created traditional web forms for the same queries to compare data transport showing the same results, but using different programming and interaction techniques. we think that these results can open new lines of research to standardize semantic queries, and create a standard with new and innovative techniques to improve the processing of the client side, and minimize the load on the server and this substantially increase its performance and improving the traffic of the network.

5. Conclusions

We present a study on how the type of communication and programming has an impact on data traffic from the server to the client. New web development paradigms and new asynchronous communication models are needed. In addition, new programming languages to take advantage of this ability to provide maximum performance of both web servers and the network. This topic can open the possibility to the generation of new open, free, generic, and multilingual development platforms can be a route for much more powerful and faster applications than the current ones. Research on new models that allow independence of programming language, software or operating system to connect multiple devices locally without saturating the network is a good possibility.

References

[1] Emmerich W, Parker M, Blakeley J, Szypersky C 2006 Computational Science, Towards 2020 Science *Microsoft Research* pp 14-15
[2] Anquetil R 2019 *Fundamental Concepts for Web Development* (*Ed Wenstreet Learning* France)
[3] Frain B 2020 *Responsive Web Design with HTML5 and CSS* (Packt Publishing Ltd. Third Edition, UK)
[4] Worsley J, Drake J 2002 *Practical PostgreSQL* (O'Reilly, First Edition, USA)
[5] McFredies P 2019 *Web Design Playground: HTML & CSS the Interactive Way (*Manning Publications Co. USA)
[6] Williams N 2014 *Professional Java for Web Applications* (Wiley, Wrox guide, USA)
[7] Dailey L 2005 Building Rich Web Applications with Ajax *Computer* (IEEE Computer Society PressWashington USA) **38**(10) 14-17
[8] Smith K 2006 Simplifying Ajax Style Web Development. *Computer* (IEEE Computer Society PressWashington USA) **39**(5) 98-102
[9] Zepeda S and Chapa S 2007 From Desktop Applications Towards Ajax Web Applications 4*th Int. Conf. on Electrical and Electronics Engineering* (Mexico City) pp 193-196
[10] Arievan A and Deursenb A 2008 Component- and Push-based Architectural Style for Ajax Applications Journal *of Systems and Software* **81**(12) 2194-2209
[11] Edwards J and Adams C. 2006 *Javascript Anthology* (SitePoint, First Edition, USA)
[12] Schneider F, Agarwal S, Alpcan T, Feldmann A 2008 The New Web: Characterizing AJAX Traffic *Int. Con. on Passive and Active Network Measurement* Lecture Notes in Computer Science (Springer, Berlin, Heidelberg) vol 4979 pp 31-40
[13] Zhaoyun S, Xiaobo Z, Li Z 2010 The Web Asynchronous Communication Mechanism Research Based on Ajax *Int. Conf. on Education Technology and Compute*r (Shanghai, China) pp V3-370-V3-372
[14] Marchetto A and Ricca F, Tonella P 2008 A Case Study-based Comparison of Web Testing Techniques Applied to AJAX Web Applications *J. Software Tools for Technology Transfer* **10**(6) 477–492

[15] Hu P, Ning H, Chen L, Daneshmand M 2019 An Open Internet of Things System Architecture Based on Software-defined Device *IEEE Internet of Things Journal* **6**(2) 2583-2592
[16] Zoubeyr F, Ameur Y, Ouederni M, Tari K 2017 A Correct-by-construction Model for Asynchronously Communicating Systems *J. on Software Tools for Technology Transfer* **19**(4) 465-485
[17] Zepeda S, Estrada J, Estrada D 2019 Database Exploration for Microbial Information *3th Int. Conf. on Computer Science and Application Engineering* **99** 1-5