Performance Analysis of Web Systems Based on XMLHttpRequest, Server-Sent Events and WebSocket

Wojciech Słodziak and Ziemowit Nowak

Abstract The aim of the authors of this chapter is to analyze the performance of Web-based systems using XMLHttpRequest, Server-Sent Events and WebSocket. Dedicated tool was presented enabling the performance of research for the above technologies. The diagrams illustrating the method of operation of the tool in the event of the investigation of transmission from the client to the server, from the server to the client and in both directions were presented. The results of the research conducted with the help of dedicated tool in the local network and on the Internet for two web servers and four browsers. The results obtained were discussed and appropriate conclusions drawn. The directions for further research were pinpointed.

Keywords Web systems · Performance analysis · XMLHttpRequest · Server-sent events · Websocket

1 Introduction

The performance of web-based systems may be improved in a number of ways, one of which may be through the use of the existing solutions that differ in terms of application use. In the article [1] the authors address the issue of improving the operation of applications supporting remote collaboration through practices used in creating online games. Problems such as packet loss, poor coverage in the case of mobile and wireless networks, and limited transmission bandwidth are solved by the creators of multiplayer games and their experience can be used e.g. in order to build systems supporting remote collaboration.

Many of the problems associated with the reduced network speed and large size of transmitted messages can be solved by means of data compression. The article [2] describes, and includes studies in GMC (General Message Compressor) compression technology. The developers of the technology prove the effectiveness of

W. Słodziak · Z. Nowak (✉)
Wrocław University of Technology, Wrocław, Poland
e-mail: ziemowit.nowak@pwr.edu.pl

© Springer International Publishing Switzerland 2016

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_6

their creation through the research contained in the article. The use of the tool enables the reduction in the size of the common message formats up to 20 % of the original size for messages based on simple text, and up to 8 % for XML format. Compression is one of the many areas of the complex process of data transmission, in which improvements can be looked for.

The selection of appropriate data transmission technologies tailored to the needs of the application is very important for developers of Web applications. This issue constitutes the main theme of this chapter and is widely discussed in literature [3, 4, 5]. In article [5] the authors examine the performance of applications built on the basis of the following technologies: XHR, Java Applet and WebSocket. In the tests the number of messages sent per second in the direction from the browser to the server and the other way round was chosen as a performance criterion. According to the research, the authors designated WebSocket as the technology that yields the best results.

In view of the above facts, the study and analysis of the available technologies and tools enable the creation of better and better Web application solutions and the movement away from restrictive desktop applications in favour of efficient and readily available web applications. The performance of Web applications is a multifaceted problem and so the improvements are to be searched for in many areas.

The aim of the authors of this chapter is to analyze the performance of Web-based systems using XMLHttpRequest [6], Server-Sent Events [7] and WebSocket [8, 9]. Another issue raised is the characteristics of strengths and weaknesses of the technology and identification of applications in which the technology performs at its best.

2 The Description of Performance Measurement Tool

For the purposes of the tests dedicated tool was developed to measure the mean time of sending a message or a pair of messages, depending on the technology used, and the testing scenario. The tool consists of the client (on the side of the browser) and the server parts.

The client part is used to generate messages and to control the tests and display the results. To this end, the user interface was created which executes actions on the side of the client and the server. Through the interface, test sequences may be run, it is also possible to read the results and control the parameters that influence the number and type of messages sent to and from the server. The parameters that can be set for each of the investigated technologies include:

- the number of messages sent in a sequence,
- the type of message:
  - fixed size message—the size in bytes to be specified,
  - random size message—the range of minimum and maximum in bytes to be specified,

– growing size message—growth in bytes of each subsequent message to be specified.

The server part is responsible for receiving and sending messages to the browser. In order to transmit messages from the server to the client, the methods of generating messages in the same manner as in the case of the client were implemented.

Tool was designed in such a way as to enable the practical comparison of the technologies surveyed. To this end three modules were implemented to compare the various combinations of the technologies with the same application:

- communication from the client to the server,
- communication from the server to the client,
- bidirectional communication.

In each module two sets of technologies were used meeting the same requirements, but differing in the implementation itself.

2.1 Communication from the Client to the Server

For communication from the client to the server XMLHttpRequest and WebSocket technology was used (Fig. 1). WebSocket also allows sending messages in the opposite direction, which was not used in this combination.

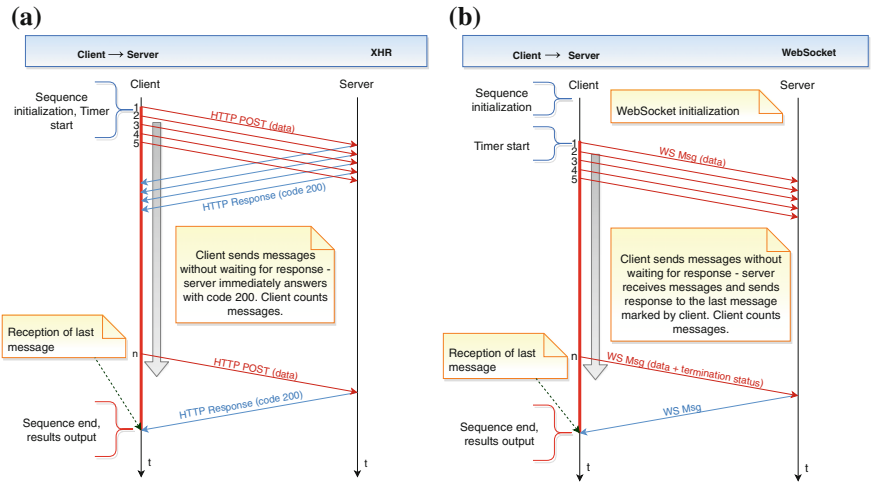

Fig. 1 The diagram of communication from the client to the server: **a** XMLHttpRequest technology, **b** WebSocket technology

In the case of XMLHttpRequest technology it is not necessary to establish a special connection between the browser and the server beforehand, as receiving HTTP protocol messages is a standard Web server functionality.

The situation is different for WebSocket technology, which requires the initialization on the part of the client (the browser). The browser reports its willingness to open TCP connections (via JavaScript), to which the server consents, provided the application, which is running on it, has such functionality implemented. The connection thus formed is maintained throughout the communication.

2.2 *Communication from the Server to the Client*

Server-Sent Events and WebSocket technologies were used for communication from the server to the client (Fig. 2). Both the technologies allow sending messages from the server to the browser (client). WebSocket also allows sending messages in the opposite direction, which was not used in this combination.

Server-Sent Events technology is based on HTTP protocol. The browser initiates the connection by creating (by JavaScript) an EventSource object listing in the URL parameter, to which a Java servlet implementing the SSE is assigned. The server, in order to establish a Keep-Alive connection, returns a partial HTTP response with HTTP Content-Type = text/event-stream header.

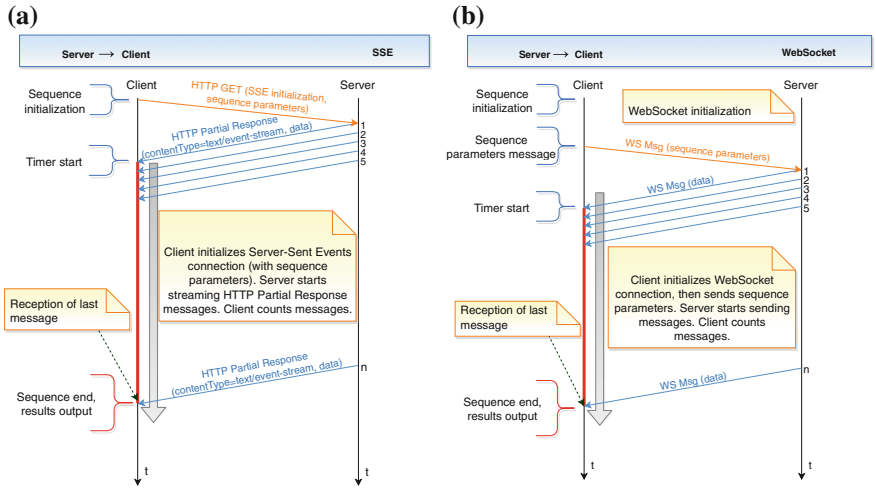

Fig. 2 The diagram of communication from the server to the client: **a** Server-sent events technology, **b** WebSocket technology

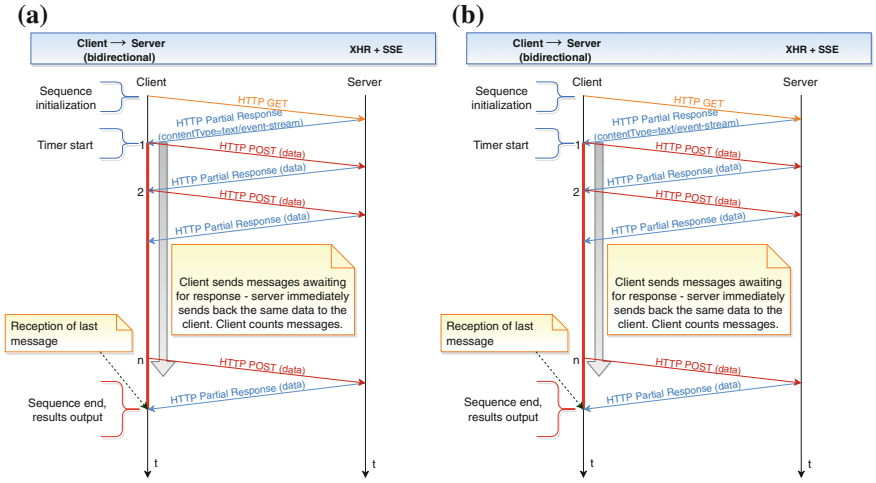

Fig. 3 The diagram of bi-directional communication: **a** Server-sent events + XMLHttpRequest technologies, **b** WebSocket technology

2.3 Bidirectional Communication

Server-Sent Events in conjunction with the XMLHttpRequest and WebSocket technology was used for the purposes of bidirectional communication (Fig. 3). The first set of technologies enables to sequentially send messages from the server to the client and from the client to the server, which, in combination, makes bidirectional communication possible. WebSocket is a technology, which itself ensures such communication.

The combination of Server-Sent Events and XMLHttpRequest technologies is a solution fully based on the HTTP protocol. The initialization of such communication requires, as before, that JavaScript and a special Java servlet that implements SSE be used. To transmit messages from the client to the server does not require an additional implementation except retrieving HTTP Post messages.

3 Performance Tests

The study involved two servers, Apache Tomcat and Glassfish which have an in-built module for enabling the Web container to run Java servlets. The servers differ from each other in that Tomcat is merely a container for servlets and Glassfish is also a container for Java EE, which extends its functionality. It was possible to use other, more popular servers, but it would have required a Web-based container

module to be installed. The study was carried out using two computers, one of which had application servers installed and running for testing, and the other the current versions of the four popular web browsers:

- Chrome—version 43,
- Firefox—version 37,
- Opera—version 29,
- Internet Explorer—version 11.

The computers used were characterized by the following parameters:

- The Server
  - Operating System—Windows 7 Pro x64,
  - Processor—Intel Core 2 Duo T6600,
  - Memory—4 GB RAM,
  - Hard drive—300 GB.
- The Client
  - Operating System—Windows 8.1 x64,
  - Processor—Intel Core i7-4710HQ 2.5 GHz,
  - Memory—8 GB RAM,
  - Hard drive—128 GB SSD.

The studies were conducted using a LAN and a WAN network. During the test involving the LAN network, the connectivity between the client and the server was provided by a WiFi access point, functioning simultaneously as a router. During the test involving a WAN, the client was connected to the Internet through a WiFi access point of the Eduroam network of the Wrocław University of Technology, and the server via a WiFi access point of the Neostrada network (ADSL).

Before starting the target tests, the initial simulation was conducted to determine the number of messages sent in the test sequence, which would give satisfactory results. To do so, the client-to-server module and XHR technology a number of tests were carried out with a variable number of messages sent in a test sequence. The selected amount was 1000, due to the small difference in the standard deviation from the number of 2000, and due to the need to shorten the test time, directly proportional to the number of messages per sequence. It should be noted that the greater the number of messages in a testing sequence the better the test results as the error associated with the current noise arising due to additional processes performed on both machines and accidental network load is reduced.

The target research was carried out for the three communication modules implemented in tool.

The result of the test sequence was the average time from the transmission of a thousand of individual or pairs of messages (in the case of bilateral communication). The time was measured, as shown in the drawings describing the

communication within the tool (Figs. 1, 2 and 3). The research conducted involved the following parameters that were adjusted:

- client (browser),
- server,
- type of network (LAN/WAN)
- number of messages in a sequence.

Due to the extent of the studies only selected results will be presented.

3.1 Communication from the Client to the Server

The first studied type of message transmission was the communication from the client to the server. In this case tool lets one use XMLHttpRequest and WebSocket technologies.

In Fig. 4 you can see a general tendency spread of results between the local and the global network. It is logical that the selected technology achieves better results for the local network than for the global one, as in the local network communication is almost instantaneous, due to the short way there is for packages to travel. In addition, the communication is accompanied by lower noise associated with the network load, which generates more stable results.

For Tomcat server XMLHttpRequest technology obtains worse results than WebSocket for both the local and global networks.

Figure 5 shows the comparison of the performance of a given technology on Tomcat and Glassfish servers. it shows that when it comes to the Tomcat server XMLHttpRequest technology is always slower than in the case of WebSocket. When it comes to the Glassfish server both the technologies have similar times and

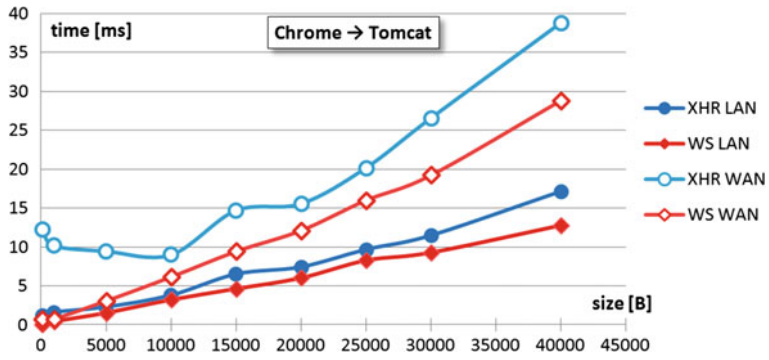

Fig. 4 The time of message transmission depending on its size for Chrome browser and Tomcat server

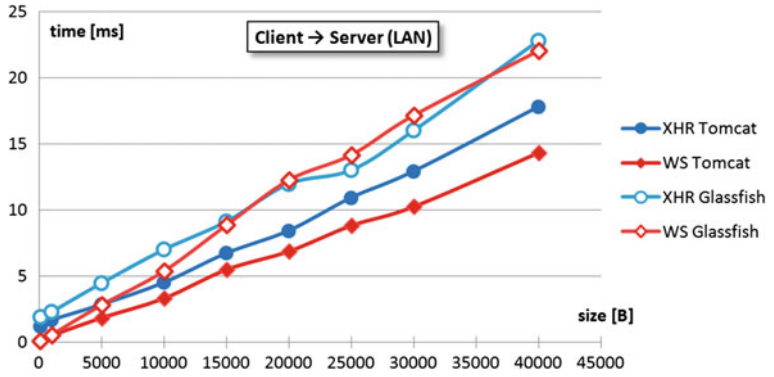

Fig. 5 The time of message transmission on the local network depending on message size (arithmetic average of the performance of individual web browsers)

intertwine in the two places of the graph. WebSocket has only a big advantage for small sized messages.

From the diagram it can be concluded that the Tomcat server is more effective when it comes to the communication from the client to the server. In any case, the same technology copes better on the Tomcat server. What is more, for the majority of the researched sizes of messages XHR technology on the Tomcat achieves shorter transmission times than WebSocket on the Glassfish server. The opposite result was obtained only in the case of message sizes smaller than 5000 bytes.

3.2 Communication from the Server to the Client

The communication module from the server to the client of tool is used to compare Server-Sent Events and WebSocket technologies. As Internet Explorer does not support SSE technology, the former was excluded from further study.

Figure 6 shows the average time for the transmission of messages from the Tomcat server to Opera browser depending on the size of the message. The careful observer will notice that WebSocket technology achieves better results for larger sizes of messages for the global network, than for the local network. This is an unexpected result, since LAN is not accompanied by a delay in contrast to the global network. This result may have been caused by random phenomena, such as e.g. accidental load on the network or CPU. It does not occur with every browser tested.

Another observation, which applies also to the other browsers, is a significant increase in the average transmission time of sending messages for Server-Sent Events technology over WAN for the message sizes larger than 7500 bytes. This phenomenon was not observed in the tests performed using the Glassfish server, as can be seen in Fig. 7.

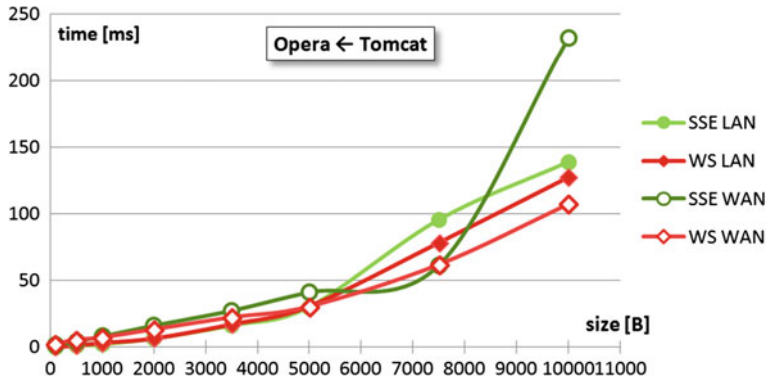

Fig. 6 The time of message transmission depending on its size for Opera browser and the Tomcat server

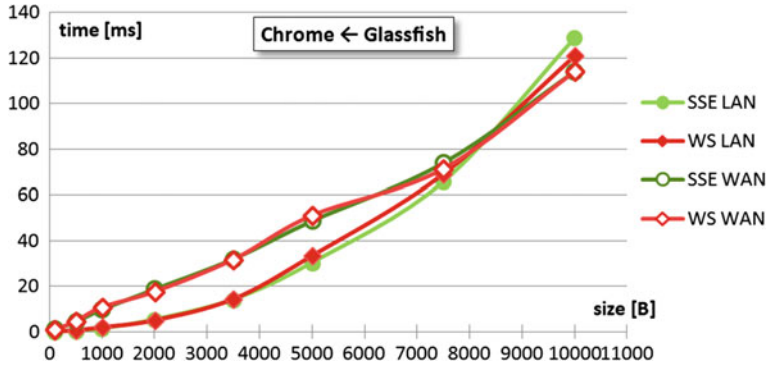

Fig. 7 The time of message transmission depending on its size for Chrome browser and the Glassfish server

In the case of the Glassfish server WebSocket and SSE technologies obtain very similar results. For small sizes of messages transmission sequences in LAN achieve shorter times than in WAN. For sizes 7500 ÷ 10,000 bytes the global network begins to gain an advantage, regardless the browser is used. Such behaviour may be caused by the fact that both the client and the server in LAN use the same WiFi access point (the device must thus perform a double job).

3.3 Bidirectional Communication

Bidirectional communication module of performance measurement tool is used to compare a set of Server-Sent Events + XMLHttpeRequest technologies and WebSocket technology. In the first stage of research the technologies were tested

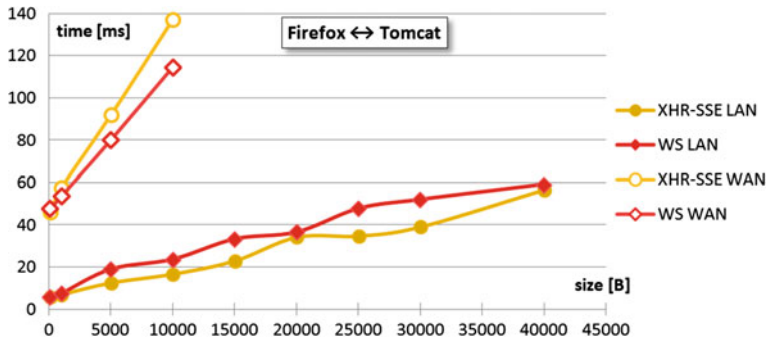

Fig. 8 The time of message pair transmission depending on its size for Firefox browser and the Tomcat server

using the local area network. In the second phase, during the tests which required the use of the global network, there was a problem with the cumulative delay in the transmission of pairs of messages between endpoints. For example, the sequence lasting 16 s in the local network lasted 137 in the global network. For these reasons, the research on the WAN was done only for short messages (the first four of the nine sizes of messages).

Figure 8 presents the comparison of the average message transmission time between Firefox browser and the Tomcat server for the technologies tested. It can be seen that the global network transmissions exhibit similar results but shorter transmission times of message sequences are obtained by WebSocket technology. Such a result was also observed in the case of the other browsers. Since the trend is of linear character, the gain in transmission time for the sequence of messages with an increase in the size of the message can be determined. For WebSocket technology the gain is approximately 6.8 ms per 1000 bytes and for SSE + XHR 8.8 ms.

In the case of a local network it can be seen that for small message sizes (100 or 1000 bytes) HTTP header size in XHR technology has no significant effect on message time transmission, as occurred in the case of transmission from the client to the server. Considering messages of 100 bytes and the average results for the tested browsers, a set of SSE + XHR technologies results in about 15 % longer transmission time sequence of messages in relation to the WebSocket technology. In contrast, in the case of communication from the client to the server, XHR times were 25 times longer as compared to WebSocket for messages of 100 bytes.

Figure 9 shows the comparison of the efficiency of transmission between Web browsers that use WebSocket technology and the Tomcat server. It can be seen that the shortest times for the majority of the message sizes surveyed are reached by Opera. The last on the list is Firefox browser, which, in almost every aspect, is slower than Opera.

In the case of SSE + XHR technologies the results are opposite to WebSocket technology. Firefox, for SSE + XHR technology achieves the shortest time for the majority of message sizes surveyed.

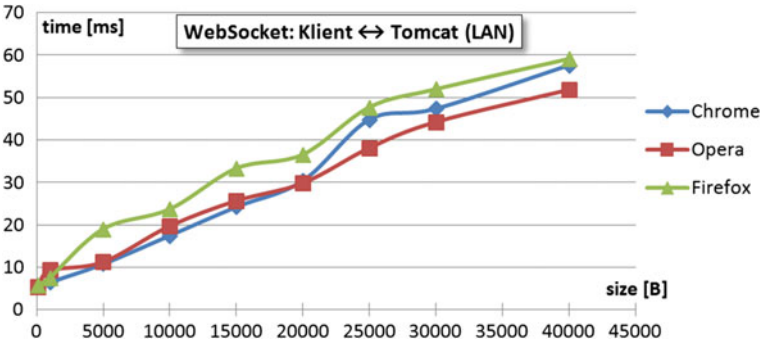

Fig. 9 Message pair transmission time via WebSocket technology, depending on its size for each browser tested, LAN, Tomcat server

3.4 Discussion of Research Results

The problem of technology selection taking into account message sending times only is not trivial. There are many factors influencing the performance of the technology and the ability to use it in a web application. The developers of an application may influence some of them, such as the selection of a server and network configuration, but not all the factors are dependent on them. For this reason, the performance of studies of a similar nature, as those performed in this section, has a positive influence on the decisions people who design applications make and the perception of application users.

In the tests that were carried out the differences in the performance of the technologies could be observed repeatedly depending on the browser used in the studies. It is likely that browsers have different mechanisms implemented to support the operation of web applications. For example, at the expense of optimization of request types that were not analyzed in the study, such as an HTTP request using the cache, or encrypted connections, browsers may worsen the times of simple requests, such as those carried out during the tests. There are many planes of web browsers' activity that may have direct impact on the transmission times achieved and which are not affected by application developers. Such elements may include, for example, JavaScript code interpreter or network sockets manager.

In the course of the research, certain issues which should be considered before performing similar studies were noticed. The first one is hardware. The computer, which worked as a server, turned out to be too weak to generate random messages of such large size, so the results of the tests for communication from the server to the client were disrupted. The second issue concerns the two-way communication module. While the module performs correctly when communicating on LAN, the times obtained in the case of the global network are disrupted by the accumulating communication delay, as a result of which both the server and the client browser

have downtimes at the time of data transmission over the network. All the tested technologies were used in one-way communication modules, so a two-way module may be considered redundant and abandoned in the future.

4 Summary

This chapter was concerned with the analysis of the performance of Web-based systems using XMLHttpRequest, Server-Sent Events and WebSocket. The problem of the performance and capabilities of web technologies is not an exhausted subject, as the above survey, as well as those described in the literature, are of great importance to the development and the future of web applications.

The important conclusion from the study is that the issue of web technology selection is complicated and must not be reduced merely to the choice of the fastest technology. There are many factors that web application developers should consider before choosing a communication technology. On the basis of performance tests, it may be concluded that the best technology in terms of speed in bidirectional communication is WebSocket. If a Web application uses only the communication from the client to the server, then, in terms of speed, XMLHttpRequest technology is slower than WebSocket, but XHR technology advantages associated with the HTTP protocol can be essential for the application. In the case of communication from the server to the client SSE and WebSocket technologies show similar performance. In this case, other factors such as network configuration and browser support should be considered.

To thoroughly examine the technologies listed, such improvements as research time, test points, the other server applications or hardware should be considered. All the tested technologies are constantly improved by the developers of web browsers and the organizations responsible for creating characteristics, so the results obtained from similar studies may become obsolete over time.

Among the directions for further research the following may be listed, for example: enhanced performance testing, testing with other criteria, following the development of current technologies and finding new solutions.

References

1. Dyck, J., Gutwin, C., Graham, N., Pinelle, D.: Beyond the LAN: Techniques from network games for improving groupware performance. In: Proceedings of the ACM Conference on Organizational Computing and Groupware Technologies, pp. 291–300 (2007)
2. Gutwin, C.A., Fedak, C., Watson, M., Dyck, J., Bell, T.: Improving network efficiency in real-time groupware with general message compression. In: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work CSCW'06, pp. 119–128 (2006)
3. Cook, D.: Data Push Apps with HTML5 SSE. O'Reilly Media, Sebastopol (2014)

4. Grigorik, I.: High-Performance Browser Networking. O'Reilly Media, Sebastopol (2013)
5. Gutwin C.A., Lippold M., Graham N.: Real-time groupware in the browser: Testing the performance of web-based networking. In: Proceedings of the 2011 ACM Conference on Computer Supported Cooperative Work CSCW'11, pp. 167–176 (2011)
6. XMLHttpRequest Level 1: W3C working draft 30 Jan 2014. http://www.w3.org/TR/XMLHttpRequest/
7. Server-Sent Events: W3C working draft 20 Oct 2011. http://www.w3.org/TR/2011/WD-eventsource-20111020/
8. The WebSocket Protocol: Request for comments: 6455. Dec 2011. https://tools.ietf.org/html/rfc6455
9. The Web Sockets API: W3C working draft 22 Dec 2009. http://www.w3.org/TR/2009/WD-websockets-20091222/