On the Use of Websockets to Maintain Temporal States in Stateless Applications

Josefa Gómez, Abdelhamid Tayebi, Juan Casado
Computer Science Department.
University of Alcalá
28871 Alcalá de Henares. Madrid.
Email: josefa.gomezp@uah.es

ACCSE 2020 International Conference on Advances in Computation, Communications and Services
September 27, 2020 to October 01, 2020 – Lisbon, Portugal

Presenter

Josefa Gómez Pérez was born in 1984. She received the BS and MS in Telecommunications Engineering from the University Polytechnic of Cartagena, Spain, in 2005 and 2007, respectively, and the PhD in Telecommunications Engineering from the University of Alcala, Spain, in 2011. She has worked as an assistant professor since 2012 at the University of Alcalá. She also worked as a faculty researcher at the Hong Kong University in 2011 and at the Instituto de Telecomunicaçoes of Lisbon in 2014. She has participated in 37 research projects with Spanish and European companies. She has published 23 papers in peer-reviewed journals, a book, two book chapters, and more than 40 conference contributions at national and international symposia. Her research interests are optimization and analysis of antennas, design of graphical user interfaces and the study of propagation for mobile communications or wireless networks in both outdoor and indoor environments.

Outline

1. Introduction.
2. Experimental results.
3. Practical use of websockets in a web-based application.
4. Conclusions.

Introduction

- This paper studies the use of Websockets to maintain temporal states in stateless applications.
- Concretely, it is used in a web-based application that calculates the propagation loss in outdoor environments.
- The reasons why Websockets are used and its limitations are discussed.
- A comparison with other similar technologies is also included.

Introduction

- Current web applications require fast communication between the server and the clients to produce close to real time updates on the web interface.
- If feedback is not received about the progress of the computations performed by the server, the user experience breaks apart.
- This also is applicable to the waiting time between a user action and the webpage displaying the requested information.
- REST (REpresentational State Transfer) and WebSockets will be analyzed, theoretically and experimentally, as the possible communication components to solve this problem.

REST

- Nowadays, lots of web applications and web services are based on the REST architectural style.
- REST has become very popular due to its simplicity and the fact that it builds upon the HTTP
- However, it has some disadvantages such as the lack of saving the stateful information between request-response cycles.
- This implies that it is hard to implement any type of services where the server updates the client without the use of client-side polling of the server or some other type of web hook.
- Consequently, any state management tasks must be performed or initiated by the client.

REST

- The disadvantages of using REST to maintain temporal states are mainly three.
  1. The server must use several TCP connections for each client: one for the client to send an initial request to the server where the operation to perform is described and a new one for each message that contains the progress or fraction of performed work.
  2. The wire protocol has a high overhead, with each client-to-server message having an HTTP header.
  3. The server is forced to maintain a mapping from the outgoing connections to the initial computation request to track which progress information must be sent to each client.

Websockets

- WebSockets is a new protocol that uses a single TCP connection for traffic in both directions that allows a bidirectional, full-duplex, persistent socket connection between a web page and a remote server. It is supported by all major web browsers.

- Based on the two-way communication connection, the server can receive and process data, and can also send data back to the browser.

- Also, communication is more efficient than using HTTP if we focus on the size of the message and on the speed, especially for large messages, since in HTTP, for example, you have to send the headers in each request

Experimental results

- To empirically test the performance differences between WebSockets and REST communication protocols, a demonstrative application was built.

- The design of this application aims to make the comparison as fair as possible, to let us inspect the strengths of both protocols.

- The server and the client communicate with REST or WebSockets allowing us to catch and dump the communication traces and inspect the transmitted packages with applications like Tcpdump or WhireShark.

Experimental results

- Performing the same computations, the following data have been collected with REST and WebSockets as communication protocols:

	REST	WebSockets
Packets	135.098	44.976
Transmitted data	27.634.885 bytes	4.723.849 bytes
Communications	22.481	22.480
Mean Time	86s 778ms	35s 226ms

Experimental results

- In average, using WebSockets, there are two packets exchange between the client and the server, which leaves a total of 16 packets for stablishing the connection (8 packets) and closing it (8 packets).

- With REST, there are an average of six packets exchange between the server and the client per communication. This quantity should have been eight packets, but the libraries used try to reuse TCP connections by not always sending ‘FIN’ packets in order to save resources.

- WebSockets is able to use a smaller header because the header is sent once, when the connection is stablished, and since the connection is never closed there is no need to resend it on every data transmission along that same connection.

Experimental results

- The benefits of WebSockets not only can be seen on the amounts of data and packets transmitted but also in the time taken. Using REST, the communication will take more than double the time than with WebSockets.

- REST is a more efficient protocol for single sporadic data transmissions than WebSockets.

- On the other hand, WebSockets is more efficient for multiple communications even if their number is small.

Practical use of websockets in a web-based application

- Authentication is also simplified by using Websockets.
- When using WebSocket, authentication is performed when the connection is established, so future requests under the same channel do not need to be authenticated again.
- This method greatly simplifies the authentication process.
- Therefore, Websockets improves the security of the system because there is no need of passing user credential in every request.

Conclusions

- WebSockets is a great protocol that solves three communication problems:

1. Sending multiple packets of data between the server and the client with a single communication negotiation required.

2. Creating a channel between a client and a server through which the client can receive notification from the server without polling.

3. Granting a stable connection between a client and a single instance of a replicated server that is behind a load balancer.

Thank you for your attention!

Josefa.gomezp@uah.es