Research document

Low-Latency Client-Server Communication

Thomas van der Molen
IPS3-DB03

Project Information	
Project members	Thomas van der Molen (4168003)
Project name	Text Adventure
Version	1.2

Table of Contents

Version History

Version	Date	Change
1.0	11-01-2022	Created File
1.1	13-01-2022	Added Introduction, Problem, Web sockets, SignalR, Conclusion and Sources
1.2	17-01-2022	Grammar check

Introduction

Within my text adventure game, it is very important that the communication between the user and the game service is as fast as possible. Having a low latency connection will enable the user to have the most responsive and best experience possible while playing the game.

Because there will be constant connection calls being made between the user and the game service it is very important these calls can be made fast and efficient as to not hurt performance.

Problem

In the past I have relied on HTTP requests and AJAX with a request-response cycle to allow for connections between my frontend and backend, this however is not a good solution for a project that relies heavily on constant communication. The overhead from making constant HTTP requests and having the connection be unidirectional makes the delay between a user inputting a command and the action being handled too long.

Example

Inside of the text adventure game, a user will have a command line like interface with an area displaying text and an input field to enter commands.

In the scenario of using a request-response cycle system, the user will send a command to the server, this command will then be handled, and a response message will be sent back (which will be displayed in the command line interface).

At the same time another user might want to chat with other players, this user will send a command to send a message to other players. As before this message will be sent to the backend, but now not only the initiator of the request should get a response but all users that should receive this message should get a response too.

These kinds of commands will be sent many times during a user's play session and the overhead coming from a regular HTTP connection request will add up drastically. For this reason, using a solution like AJAX will not be good enough to keep a good user experience.

Web sockets

Web sockets are a communication protocol that allows the connection between a client and server to be bi-directional and full duplex.

These connections work by opening a TCP socket which will stay open for the duration of the connection's lifetime, from this open socket messages can be sent between client and server freely.

Below is an image showing a more in-depth explanation of how a web socket connection is opened and maintained.

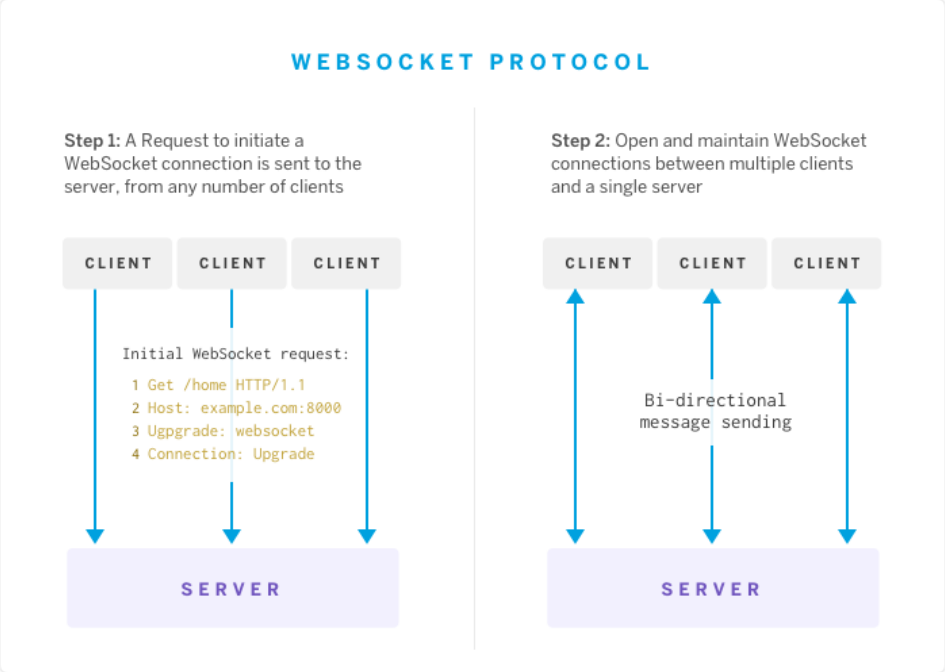

Figure 1 Short explanation web socket connection

Example

As shown in the image above the web socket connection is created by first making an upgrade request to upgrade to connection to a web socket connection, afterwards the connection will be kept open to send messages between the client and server.

Shown below are the two networking requests made inside of the text adventure's client to first negotiate the version and receive needed information such as the connection ID. After this the web socket upgrade will happen and the connection will stay open with the ability for both the client and server to send messages back and forth.

Figure 2 Networking requests made in real time

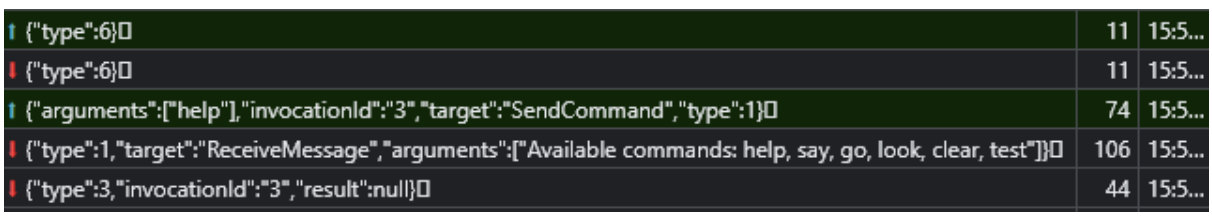

Figure 3 WebSocket connection with a keepalive, and genuine user input message

Benefits

The benefits from using a web socket seems to perfectly fit the needs of a text adventure web application. A web socket connection might have more overhead than a normal REST API because of the initial setup, but the overhead per message is far lower with a web socket than with an HTTP request.

On top of this, because a web socket connection is bi-directional, having rapid communication between client and server will be significantly easier. Furthermore, requests will not have to be stateless because web sockets will stay open for the whole lifetime of a play session, this makes keeping track of users easier and allows for less data needed to be sent back and forth.

SignalR

My project will be made in C# using the ASP .NET Core framework.

This means that I will be able to use Microsoft's SignalR library, at a base level this framework uses and works a lot like a web socket connection, however SignalR allows developers to implement certain functionality a lot easier because of useful procedures such as RPC and connection grouping.

SignalR also handles the establishing and maintaining of real-time connections automatically, this removes a lot of boilerplate from needing to be written and SignalR has fallback implementations for when a web socket connection could not be established.

Conclusion

As explained, the text adventure application relies heavily on low-latency rapid communication between clients and server, this makes the more standard ways of communication that use the request-response cycle less optimal.

Thankfully there are other established ways of creating communication between client and server with the most used and newer one being web sockets. Web sockets allow for bi-directional full duplex communication between clients and server. This works by establishing a web socket connection between client and server and sending messages back and forth with this open connection, this greatly reduces the latency and overhead on a per request basis.

Lastly Microsoft's SignalR library is a very useful and powerful addition to a project that has a use case as mine that relies on constant client-server communication.

Sources

Web sockets:

- https://medium.com/@nerdplusdog/websocket-simultaneous-bi-directional-client-server-communication-e7948203054b
- https://blog.heroku.com/real_time_rails_implementing_websockets_in_rails_5_with_action_cable
- https://www.educba.com/websocket-vs-rest/
- https://blogs.windows.com/windowsdeveloper/2016/03/14/when-to-use-a-http-call-instead-of-a-websocket-or-http-2-0/#:~:text=WebSockets%20allow%20for%20a%20higher,each%20message%20sent%20and%20received.&text=When%20a%20client%20wants%20ongoing,are%20generally%20a%20good%20fit.

Bi-directional communication:

- https://stackoverflow.com/questions/54940099/confusion-regarding-bidirectional-and-full-duplex-in-articles-about-http-2#:~:text=Bidirectional%20means%20you%20can%20send,one%20reading%20data%2C%20executing%20concurrently.

SignalR

- https://docs.microsoft.com/en-us/javascript/api/@microsoft/signalr/messagetype?view=signalr-js-latest
- https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr