HTTP Push

In this lesson, you will learn about the HTTP PUSH mechanism.

We'll cover the following

Time To Live (TTL)#

In the regular client-server communication, which is *HTTP PULL*, there is a *Time to Live (TTL)* for every request. It could be 30 secs to 60 secs, varying from browser to browser.

If the client doesn’t receive a response from the server within the TTL, the browser kills the connection and the client has to re-send the request hoping it receives the data from the server before the TTL ends again.

Open connections consume resources, and there is a limit to the number of open connections a server can handle at once. If the connections don’t close and new ones are being introduced, over time, the server will run out of memory. Hence, the TTL is used in client-server communication.

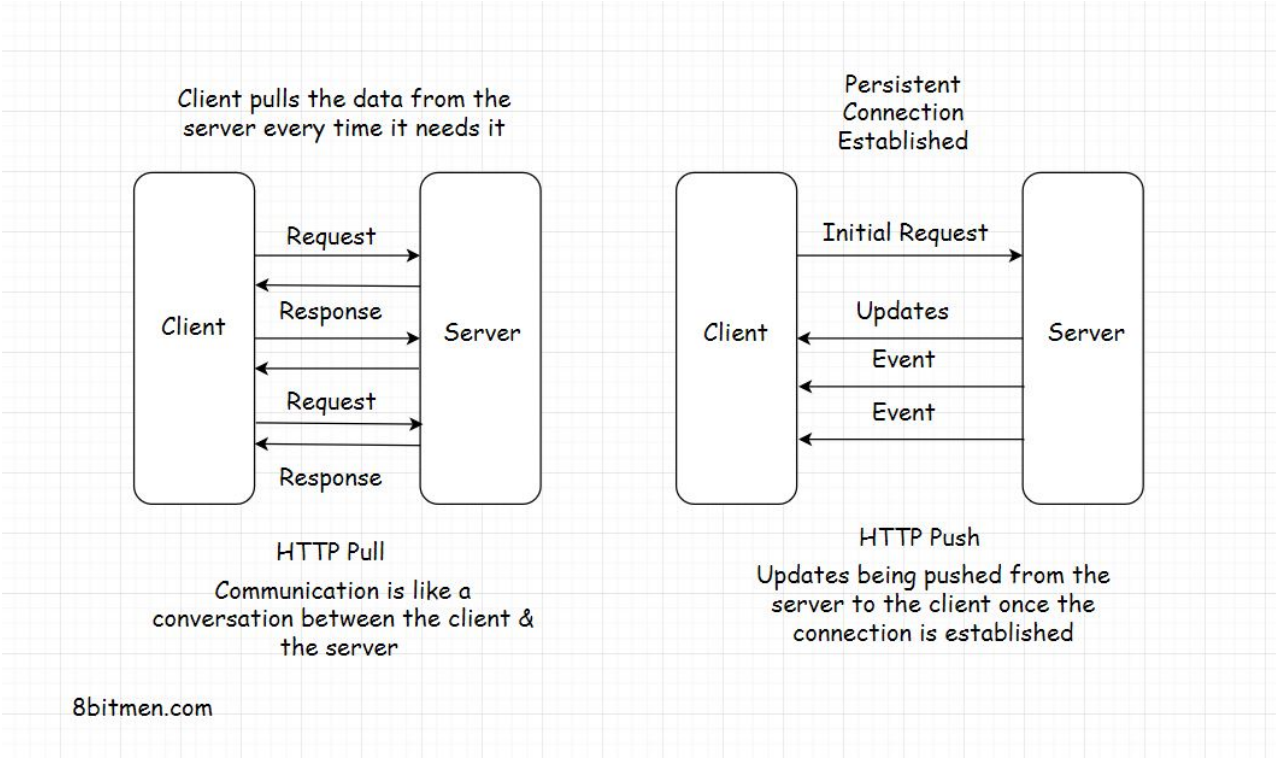
But what if we are certain that the response will take more time than the TTL set by the browser?

Persistent connection#

In this case, we need a *persistent connection* between the client and the server.

> A persistent connection is a network connection between the client and the server that remains open for further requests and responses, as opposed to being closed after a single communication.

This facilitates HTTP PUSH-based communication between the client and the server.



Heartbeat interceptors#

Now you might be wondering how is a persistent connection possible if the browser kills the open connections to the server every X seconds?

The connection between the client and the server stays open with the help

The connection between the client and the server stays open with the help of *Heartbeat Interceptors*.

> These are just blank request responses between the client and the server to prevent the browser from killing the connection.

Isn't this resource-intensive?

Resource intensive#

Yes, it is. Persistent connections consume a lot of resources compared to the HTTP PULL behavior. However, there are use cases where establishing a persistent connection is vital to an application's feature.

For instance, a browser-based multiplayer game has a pretty large amount of request-response activity within a certain time compared to a regular web application.

It would be apt to establish a persistent connection between the client and the server from a user experience standpoint.

Long opened connections can be implemented by multiple techniques such as *AJAX Long Polling*, *Web Sockets*, *Server-Sent Events*, etc.

Let's take a look into each of these methods.

← Back

HTTP Pull - Polling With AJAX

Next →

HTTP Push-Based Technologies

Completed

Report an Issue