Project (CS-307-F20) Report submitted in fulfillment of

Semester Project

in

Analysis of HTTP\2

by

SYED ASAD ZAMAN
18P-00034

Under the supervision of

Eng. Mashal Khan

DEPARTMENT OF COMPUTER SCIENCE

FAST NATIONAL UNIVERSITY OF COMPUTING AND EMERGING SCIENCES

Abstract

We all know how the client server communication is done. A client(let us say Browser) send a request(get and post) and the server respond to that request. These requests are handled in different ways by different protocols. In case of browsers this is done using HTTP. In this paper we will talk about specifically about the working of HTTP 2.0.

Contents

Chapter 1

Introduction

We all had faced the issue of **Timeout** (when a server is taking too long to reply to a data request made from another device). The solution to that is many one of them is **HTTP/2**. Unlikely **HTTP**, **HTTP/2** create a single **TCP** connection for request-respond and the beautiful thing it uses the multiplexing in order to achieve that where each request is sent to server in the form of **Streams**.

HTTP/2 will make our applications faster, simpler, and more robust — a rare combination — by allowing us to undo many of the **HTTP/1.1** workarounds previously done within our applications and address these concerns within the transport layer itself. Even better, it also opens up a number of entirely new opportunities to optimize our applications and improve performance!

The primary goals for **HTTP/2** are to ***reduce latency*** by enabling full request and response multiplexing, minimize protocol overhead via efficient compression of HTTP header fields, and add support for request prioritization and server push.

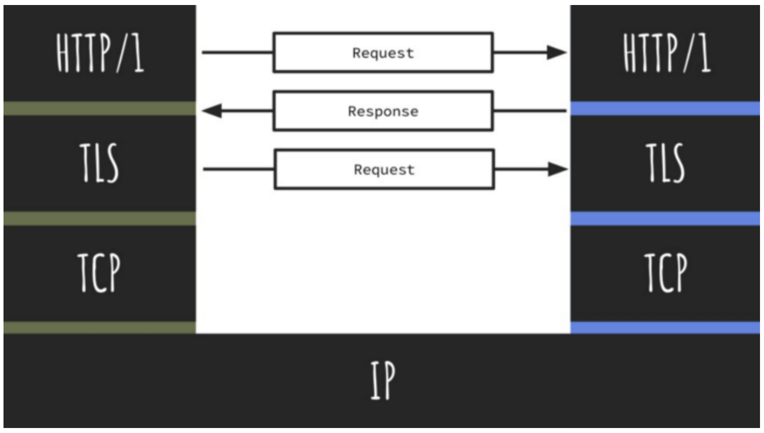
Chapter 2

Working

2.1 HTTP 1.1 Working

To understand the working of HTTP/2 will we first understand how the HTTP/1 and later works.

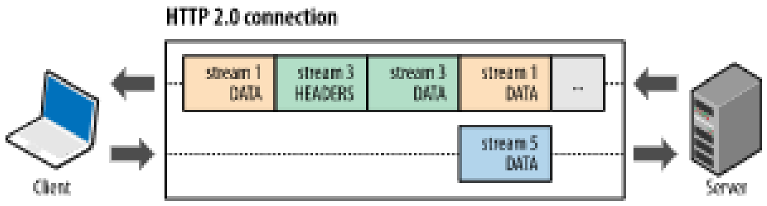
In HTTP, the client-server communication is done through a channel(TCP) in which the client(browser) send a request and the server respond to that request one by one. This sounds good but what actually happen during this communication the whole TCP channel is block means no other request is sent until the previous request is full fill by the server. In which results in wastage of resources. Because a single TCP is much more capable than just of handling a single request. In order to overcome this issue. The client made TCP connections up to 6(not more than that). But the issue remains the same. Because it is possible that client made connections more than 6.



2.2 HTTP 2.0 Working

So now we know how HTTP/1 works lets talk about the working of HTTP 2.0 working.

In HTTP/2, a single TCP connection is made in which request is made to the server simultaneously unlikely HTTP/1. But the question is how it is possible, means how the server would know in response of which request I would give the corresponding response i.e. intermixing is possible. The answer to this question is that HTTP/2 do ***Multiplexing*** which in turn do ***Streaming*** of the corresponding request i.e. it assign the **stream ID** to each request made. By compressing the logical data and Header. This is how the server doesn't intermixes the requests made by the client.



Chapter 3

Advantages & Disadvantages

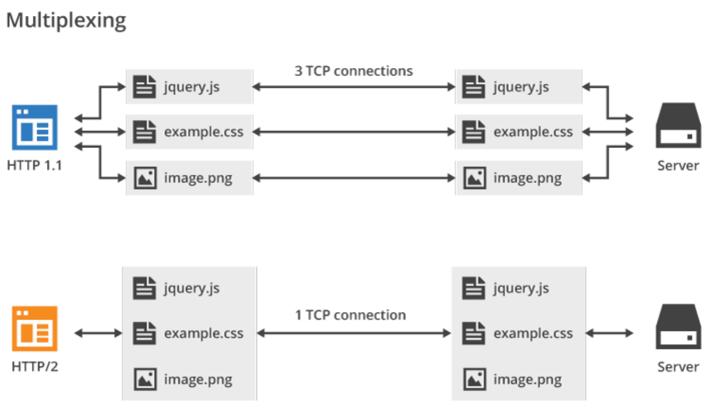
3.1 Advantages

3.1.1 Single TCP Connection

In order to perform the request-response protocol. HTTP/2 make a single TCP connection which results in low latency.

3.1.2 Multiplexing

Multiplexing is a method in HTTP/2 by which multiple HTTP requests can be sent and responses can be received asynchronously via a single TCP connection. Multiplexing is the heart of HTTP/2 protocol.



3.1.3 Server Push

Server push allows you to send site assets to the user before they've even asked for them. It's an elegant way to achieve the performance benefits of HTTP/1 optimization practices such as in lining, but without the drawbacks that come with that practice.

3.2 Disadvantages

3.2.1 Server Push can be abused

One of the main of the HTTP/2 is that if client requests for a certain data and the server give the whole data related to that. In that case the **server abused** is possible. Because it is possible that TCP connection or the client doesn't have the capacity that much amount of requests that can cause problems.

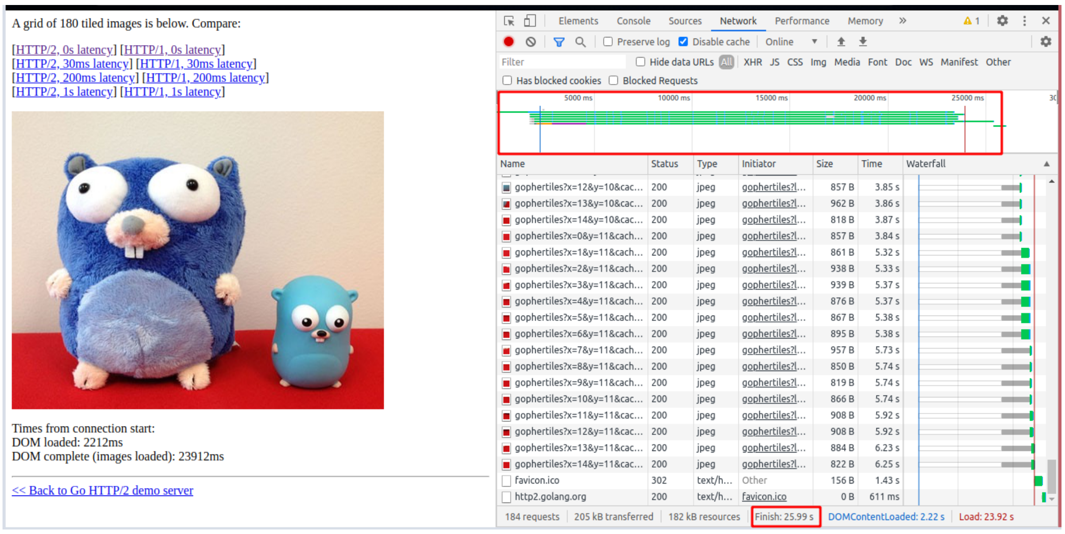
3.2.2 Can be Slower

Another disadvantage of HTTP/2 is that it can become slower if the back-end supports the the HTTP/2 & on the other hand front-end is lacking the support of HTTP/2. So in that case resources will get waste which can results in the slowness in the perfomance.

Chapter 4

Results

We perform our test we find something very interesting. The site which contain 184 small pictures we loaded with HTTP/1 run almost take half(or even less)time on the HTTP/2. Besides, in HTTP/1 it made 6 connections while on the other side HTTP/2 only did by establishing only one connection. For the same 184 requests



A grid of 180 tiled images is below. Compare:

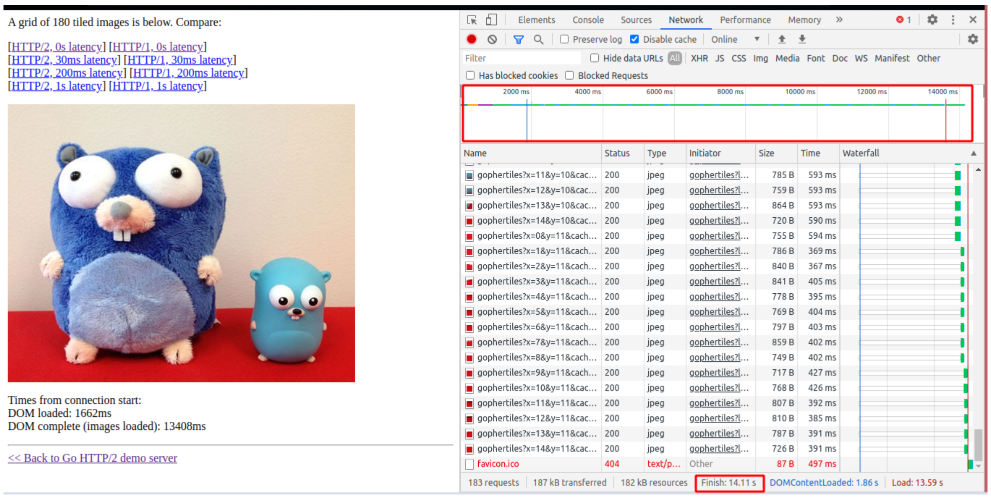
[HTTP/2, 0s latency] [HTTP/1, 0s latency]
[HTTP/2, 30ms latency] [HTTP/1, 30ms latency]
[HTTP/2, 200ms latency] [HTTP/1, 200ms latency]
[HTTP/2, 1s latency] [HTTP/1, 1s latency]

Times from connection start:
DOM loaded: 2212ms
DOM complete (images loaded): 23912ms

<< Back to Go HTTP/2 demo server

Name	Status	Type	Initiator	Size	Time
gophertiles?x=12&y=10&cac...	200	jpeg	gophertiles?l...	857 B	3.85 s
gophertiles?x=13&y=10&cac...	200	jpeg	gophertiles?l...	962 B	3.86 s
gophertiles?x=14&y=10&cac...	200	jpeg	gophertiles?l...	818 B	3.87 s
gophertiles?x=0&y=11&cach...	200	jpeg	gophertiles?l...	857 B	3.84 s
gophertiles?x=1&y=11&cach...	200	jpeg	gophertiles?l...	861 B	5.32 s
gophertiles?x=2&y=11&cach...	200	jpeg	gophertiles?l...	938 B	5.33 s
gophertiles?x=3&y=11&cach...	200	jpeg	gophertiles?l...	939 B	5.37 s
gophertiles?x=4&y=11&cach...	200	jpeg	gophertiles?l...	876 B	5.37 s
gophertiles?x=5&y=11&cach...	200	jpeg	gophertiles?l...	867 B	5.38 s
gophertiles?x=6&y=11&cach...	200	jpeg	gophertiles?l...	895 B	5.38 s
gophertiles?x=7&y=11&cach...	200	jpeg	gophertiles?l...	957 B	5.73 s
gophertiles?x=8&y=11&cach...	200	jpeg	gophertiles?l...	850 B	5.74 s
gophertiles?x=9&y=11&cach...	200	jpeg	gophertiles?l...	819 B	5.74 s
gophertiles?x=10&y=11&cac...	200	jpeg	gophertiles?l...	866 B	5.74 s
gophertiles?x=11&y=11&cac...	200	jpeg	gophertiles?l...	908 B	5.92 s
gophertiles?x=12&y=11&cac...	200	jpeg	gophertiles?l...	908 B	5.92 s
gophertiles?x=13&y=11&cac...	200	jpeg	gophertiles?l...	884 B	6.23 s
gophertiles?x=14&y=11&cac...	200	jpeg	gophertiles?l...	822 B	6.25 s
favicon.ico	302	text/h...	Other	156 B	1.43 s
http2.golang.org	200	text/h...	favicon.ico	0 B	611 ms

184 requests | 205 kB transferred | 182 kB resources | Finish: 25.99 s | DOMContentLoaded: 2.22 s | Load: 23.92 s



A grid of 180 tiled images is below. Compare:

[HTTP/2, 0s latency] [HTTP/1, 0s latency]
[HTTP/2, 30ms latency] [HTTP/1, 30ms latency]
[HTTP/2, 200ms latency] [HTTP/1, 200ms latency]
[HTTP/2, 1s latency] [HTTP/1, 1s latency]

Times from connection start:
DOM loaded: 1662ms
DOM complete (images loaded): 13408ms

<< Back to Go HTTP/2 demo server

Name	Status	Type	Initiator	Size	Time
gophertiles?x=11&y=10&cac...	200	jpeg	gophertiles?l...	785 B	593 ms
gophertiles?x=12&y=10&cac...	200	jpeg	gophertiles?l...	759 B	593 ms
gophertiles?x=13&y=10&cac...	200	jpeg	gophertiles?l...	864 B	593 ms
gophertiles?x=14&y=10&cac...	200	jpeg	gophertiles?l...	720 B	590 ms
gophertiles?x=0&y=11&cach...	200	jpeg	gophertiles?l...	755 B	594 ms
gophertiles?x=1&y=11&cach...	200	jpeg	gophertiles?l...	786 B	369 ms
gophertiles?x=2&y=11&cach...	200	jpeg	gophertiles?l...	840 B	367 ms
gophertiles?x=3&y=11&cach...	200	jpeg	gophertiles?l...	841 B	405 ms
gophertiles?x=4&y=11&cach...	200	jpeg	gophertiles?l...	778 B	395 ms
gophertiles?x=5&y=11&cach...	200	jpeg	gophertiles?l...	769 B	404 ms
gophertiles?x=6&y=11&cach...	200	jpeg	gophertiles?l...	797 B	403 ms
gophertiles?x=7&y=11&cach...	200	jpeg	gophertiles?l...	859 B	402 ms
gophertiles?x=8&y=11&cach...	200	jpeg	gophertiles?l...	749 B	402 ms
gophertiles?x=9&y=11&cach...	200	jpeg	gophertiles?l...	717 B	427 ms
gophertiles?x=10&y=11&cac...	200	jpeg	gophertiles?l...	768 B	426 ms
gophertiles?x=11&y=11&cac...	200	jpeg	gophertiles?l...	807 B	392 ms
gophertiles?x=12&y=11&cac...	200	jpeg	gophertiles?l...	810 B	385 ms
gophertiles?x=13&y=11&cac...	200	jpeg	gophertiles?l...	787 B	391 ms
gophertiles?x=14&y=11&cac...	200	jpeg	gophertiles?l...	726 B	391 ms
favicon.ico	404	text/p...	Other	87 B	497 ms

183 requests | 187 kB transferred | 182 kB resources | Finish: 14.11 s | DOMContentLoaded: 1.86 s | Load: 13.59 s

4.1 Statistics

Here are the statistics running both the protocol on different speeds.

Protocol	Online	Fast 3G	Slow 3G
HTTP 1.1	23.95 s	20.33 s	1.2 min
HTTP/2	12.85 s	7.94 s	22.96 s

From the results above we noticed that HTTP/2 took almost half time when we test it **Online** and less than quater time when run our test on **Fast 3G** & **slow 3G**.

Bibliography

[1] https://developers.google.com/web/fundamentals/performance/http2

[2] https://tools.ietf.org/html/rfc7540