INVESTIGATING HTTP/2 AT BBC SPORT

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2015

By
Bowen Liang
School of Computer Science

Contents

Word Count: 17801

List of Tables

List of Figures

Abstract

INVESTIGATING HTTP/2
AT BBC SPORT
Bowen Liang
A dissertation submitted to the University of Manchester
for the degree of Master of Science, 2015

The aim of the thesis is to investigate HTTP/2 at the BBC. HTTP/2, the next version of HTTP, is recently finalized as an Internet standard. BBC Sport is interested the impacts and benefits that HTTP/2 brings to the user experiences for web page browsing.

This thesis investigates HTTP/2 for BBC Sport from different aspects covering the feasibility and the performance. The investigation of web browsers shows whether enough proportion of BBC Sport visitors is ready for HTTP/2. Following the investigation in server side implementation suggests the current available way enabling HTTP/2 including web frameworks and reverse proxy servers. Also the obstacles and chances for BBC Sport migrating to HTTP/2 are identified.

Performance tests cover page loading and visual changes which is relatively essential to users. A software named Sport-proxy is developed along with the design of a testing system. With the testing system proxying BBC Sport pages, HTTP/2 in load benchmarks and page browsing are tested and compared with HTTP/2. The network conditions of landline and mobile data are considered and tested to inspect how much HTTP/2 benefits user's web page browsing. The tests for server push, the new feature in HTTP/2, is also conducted to investigate the potential of HTTP/2.

Despite some deviations could be caused by using different tools for benchmark tool in benchmark tests, the result of repeated tests is still able prove the conclusions and provide a general review of HTTP/2 compared with HTTP/1.1.

Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to thank the BBC Sport Digital department for supporting this project and providing a perfect environment and sufficient resources for the investigation.

I would also like to thank Matthew Clark and Keith Mitchell for their support. You have contributed a lot to the success of this project.

My special thank you goes to my supervisor, Dr. Duncan Hull. I am deeply grateful for all your help throughout the entire project. I am convinced that without your assistance the project would not have been as successful.

I also want to thank my parents Zongyao Liang and Jianping Guo. The love and support from my family is always encouraging my life in Manchester.

Thanks again to who helped me during my MSc project.

Chapter 1

Introduction

1.1 Project background and involved parties

This project was initiated and co-supervised by the University of Manchester and the BBC Sport department. Its aim was to investigate the possibility of enabling HTTP/2 protocol for BBC Sport and to evaluate the changes and impacts in website page browsing from page load performance to user visual effect performance with new features provided by HTTP/2.

BBC Digital Sport is the team responsible for producing the BBC's sport website, and mobile & tablet apps. It is part of BBC Digital, the division responsible for all of the BBC's online output, including BBC News and BBC iPlayer.

BBC Digital Sport offered several different varieties of resources to support this project. Matthew Clark, the senior technical architect from BBC Sport, actively supervised the complete period of this project from initial topics to substantial progress and final report. As for the computational resources required by the tests, BBC Sport provides two virtual compute nodes on Amazon web services, which are heavily involved in testing and deploying the experimental proxy.

1.1.1 Project description

The subject of this project is investigating HTTP/2, which is the next major version of Hypertext Transfer Protocol. This project is based on BBC Sport's support and current situation by inspecting the website system architecture and reusing the page content and structures of BBC Sport.

The investigation in HTTP/2 is conducted into two major parts, one for discussing

necessary adaption to HTTP/2 with support from user side and server side, while the other part focusing page load performance. A lightweight proxy software is created for enabling and controlling server push.

First part starts with the investigation in necessary support from both client side and server software, since HTTP/2 is a new major version with changes in underlying structure like binary framing and multiplexing which requires a complete new stack of implementation. The trends and share of BBC Sport website visitors with default support for HTTP/2 will be discussed and concluded as part of the reason for applying HTTP/2 in practices.

The investigation continues in the discussion in possible obstacles in the migration to HTTP/2 with current website pages and architectures of BBC Sport. The changes in page contents and enabling secure transfer layer with proper cipher suites are listed. It is also discussed the dual band strategy for foreseeable future architectures supporting both HTTP/2 and current HTTP/1.1.

The second part focus on the testing and evaluation of the client side performance from page load performance to user visual effect improvement by comparing HTTP/2 and HTTP/1.1.

A lightweight proxy implementation, which is enabling different server push rules and strategy, is designed and developed for testing HTTP/2 in real world situations. To reveal the impacts and improvement in client side performance, a series of comparison between HTTP/2 and HTTP/1.1 will be conducted in several aspects from waterfall and connection view to network bandwidth utilization.

1.1.2 Motivation

There are several reasons for this project. The most obvious interests in HTTP/2 from BBC Sport aspect is to evaluate the benefits the new protocol bring to end-users as user experience is critical for BBC Sport website.

BBC Sport is eager to collect useful conclusion in this project for further deployment decision of HTTP/2, particularly in the browser support from its website visitors and the performance HTTP/2 brings to users' browsing in real practices.

And theres is a general interest in HTTP/2 outside of BBC. Considering HTTP/2 has been recently standardized in 2015, the research for the performance comparison between HTTP/2 and HTTP/1.1 with the web pages from real websites is rare or convincing for migrating to HTTP/2.

1.1.3 Research questions

The research goal of this project was to evaluate the feasibility of enabling support HTTP/2 for BBC and the performance improvement in user browsing under real world deployment circumstances.

The research first concerns what is the support from the web browsers and how much of BBC Sport users can be benefit from enabling HTTP/2 with default support. The trends and portion of HTTP/2 browser support is a critical references for enabling HTTP/2 in BBC Sport.

The research focus the user experiences, as BBC Sport is a public accessed website toward end-users. The questions about how much improvement for BBC Sport pages that HTTP/2 brings, from page loading performances and visual performances, are answered by the this research. Especially, as BBC serves users over different types of network connections, the impact of network situation in HTTP/2 performances is also considered. The research also answers the improvement with server push as a new feature that HTTP/2 brings to HTTP.

No available published data researches satisfied the questions for full inspected coverage for applying HTTP/2 from user side to applying new features, especially targeting the situations and web pages of BBC Sport website.

1.1.4 Structure

This dissertation thesis is constructed in the following way:

Chapter 2 introduces the detailed information and importances of HTTP/2 to current worldwide web world and BBC Sport's interested in HTTP/2 as the foundation of this project. The used tools and measurement matrixes are also listed in this chapter.

Chapter 3 and Chapter 4 respectively investigates the support from client side browsers and server side implementations as the pre-requirements for enabling HTTP/2. The following Chapter 5 discussed the possible obstacles and chances for BBC to enable HTTP/2 support.

Chapter 6 introduced the testing system and the proxy software called Sport-proxy for the use of further test. Chapter 7 is the most important part of the thesis, as the a series tests and related evaluation focused on the page load performances and visual performances of HTTP/2 are conducted and concluded under different network situations and with server push features.

The thesis is ended with the conclusions that have been drawn and an outlook into

future work that came up during this project.

Chapter 2

Background

The first part of the background comes up with the origin of the Hypertext Transfer Protocol, which enables and stipulates the basic rules for the World Wide Web. Since the last major version of HTTP was standardized 19 years ago, the trends of World Wide Web are revealing several critical problems bought by HTTP/1. And it is shown that the main objectives of the attempt addressing these problems introduced by the Internet industries and how they are standardized into Internet standard protocol HTTP/2.

The next half of background introduces the experiments in HTTP/2 conducted by BBC research departments. However, the recent experiments and results are mainly focused in streaming media distribution instead of page load time or end-user experience, which is more concerned by BBC Sport.

2.1 Introduction of Hypertext Transfer Protocol

HTTP originally comes with World Wide Web (WWW) and became the foundation of data communication of WWW. Tim Berners-Lee and his team are credited with inventing the original HTTP along with HTML and the associated technology for a web server and a text-based web browser. Berners-Lee first proposed the ”World Wide Web” project in 1989 — now known as the World Wide Web. World Wide Web idea requires a new protocol for file transfer functionality via the request to an index search of a hypertext archive.

In 1991, to prove theory in action, Tim Berners-Lee implemented a simple prototype of the protocol [BL91] . This is a subset of the full HTTP protocol, and is known

as HTTP 0.9. The reason for the unofficial acquired version code is the protocol version 0.9 is backward compatible by the successor versions including 1.0 and 1.1.

In HTTP version 0.9, the protocol uses the normal internet-style telnet protocol style on a TCP-IP link. The following describes how a client acquires a (hypertext) document from an HTTP server, given an HTTP document address. The request is a single ASCII character string listed with method name, document path and the server port and it is terminated by a carriage return (CRLF). The response is also ASCII character encoded stream with in hypertext markup language (HTML).

HTTP/1.0, which was released in May 1996 by HTTP Working Group (HTTP-WG) in RFC (Request for Comment) 1945 [TBL96], enhanced HTTP/0.9 to be more flexible, extensible, and reliable. To distinguish between the two protocol versions and provide backward compatibility, an HTTP version number was added to each HTTP command. One of the major shortcomings of HTTP/0.9 was the inability to include meta information in HTTP requests and replies. This lack of information made it difficult for clients to determine what type of media was being received. HTTP/1.0 overcame this problem by extending the format of requests and replies to something very similar to that of the Format for ARPA Internet Text Messages and the Multipurpose Internet Mail Extensions. Using this format, servers can include content-type information in replies to better describe the content being sent to clients. Along the same lines, clients can include additional information in requests to potentially allow the server to better serve the client [Boy99]. So far, until HTTP/1.0, the protocol standard remained an unofficial state with definition in RFCs.

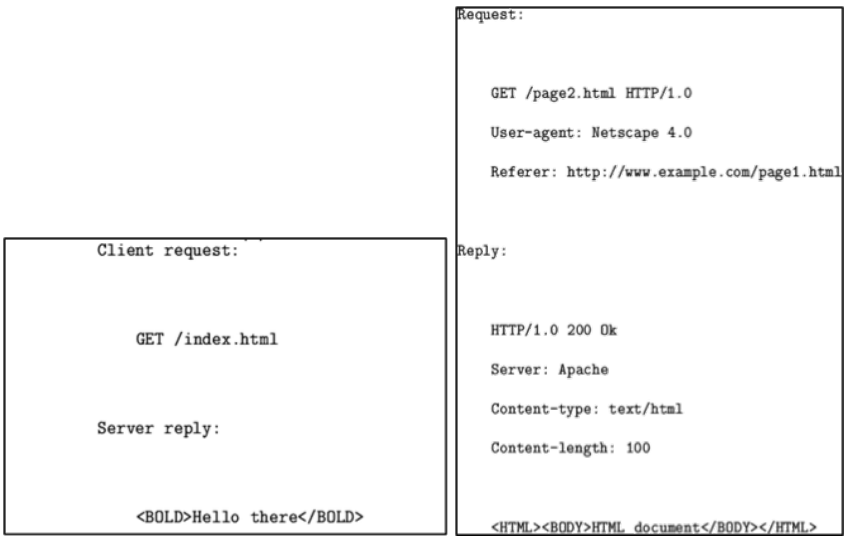


Figure 2.1: Examples of the request and reply of HTTP/0.9 and HTTP/1.0

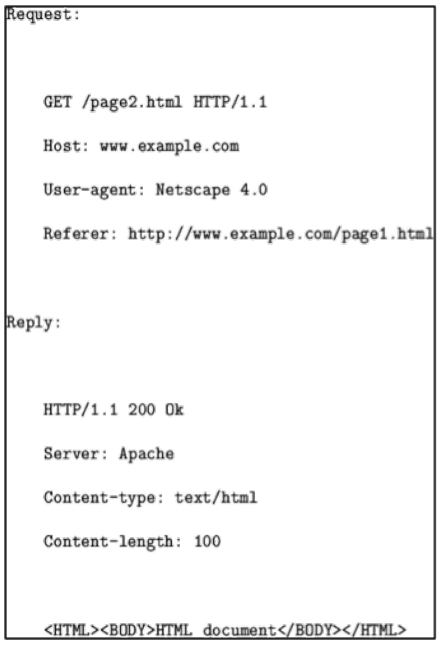


Figure 2.2: Example of the request and reply of HTTP/1.1

Figure 2.1 shows an example of legacy request and response of HTTP/0.9 and HTTP/1.1. And Figure 2.2 is for the HTTP/1.1. URI, methods, status code and core style of requests and responses of HTTP/1.1 will be continued to use in HTTP/2.

The HTTP 1.1 standard changed the situation and become the first official IETF (The Internet Engineering Task Force) released standard that was defined in RFC 2068 and published in January 1997 roughly six months after the publication of version 1.0.

HTTP/1.1 resolved a lot of ambiguities found in earlier versions and introduced a number of critical performance optimizations including from keep alive connections, chunked encoding transfers, byte-range requests to additional caching mechanisms.

2.2 Problems with HTTP/1.1

The trends form the current state of the modern web application helps to realize the usage and potential problems of real world. The wider application of WWW may lead to a growing number in web resources requests and the page load time (PLT), and moreover, the previously existing underlying HTTP is not good at handling the current situation with many resources to load and many connections to initial and hold.

The project HTTP Archive (http://httparchive.org) is tracking how the websites is built by crawling the most popular websites and over 448923 URLs analyzed by April

1, 2015. It is recording and providing the analytics on the key factors showing the structure of website including the number of out-linked domains, the number and types of requested resources and other meta information data for each analyzed destination.

As of the latest analysis report in April 1st, 2015, an averaged web application is composed of 93 requests to 16 hosts with 1,950KB total transfer size from the following resources and requests:

- HTML: 9.3 requests, 58KB
- CSS (Cascading Style Sheets): 6.2 requests, 60 KB
- JavaScript: 18 requests, 303 KB
- Images: 51 requests, 1249 KB
- Fonts: 2.1 requests, 87 KB
- Other: 6.7 requests, 126 KB

As for the view of whole trend of web application, the number of total requests and the total transfer size are growing up in the period of recent four years from Apr 2011 to Apr 2015. While the number of total requests increased by approximately 14.8% from 81 requests to 93 requests on average, the average total transfer size is almost doubled, from 1,018 KB in Apr 2012 to 1,950 KB in April 2015 [Arc15] (see Figure 2.3).

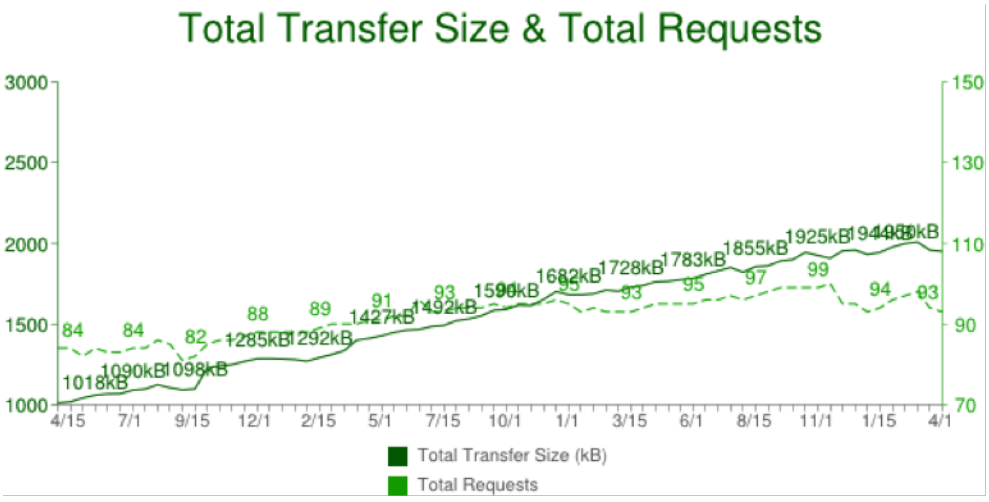


Figure 2.3: Trend of total transfer size & total requests

The global average internet bandwidth is growing by 20% from 3.6Mbps at the end of 2013 to 4.5 Mbps, according to Akamai report [Aka14b]. However, with the overall increase in total requests and total transfer size, the page load time on average is actually increasing. It is curious to find out the contrary the growing page load time with larger bandwidth at the same time.

One of the reason for this contrary situation is that network latency resulted in RTT (or Round-Trip Time) is becoming the bottleneck of web page loading performance, rather than bandwidth, according to Mike Belshe's paper called *More Bandwidth Doesn't Matter (Much)* [Bel10]. See the comparison of page load time with different bandwidth and roundtrip trip time in Figure 2.4 below.

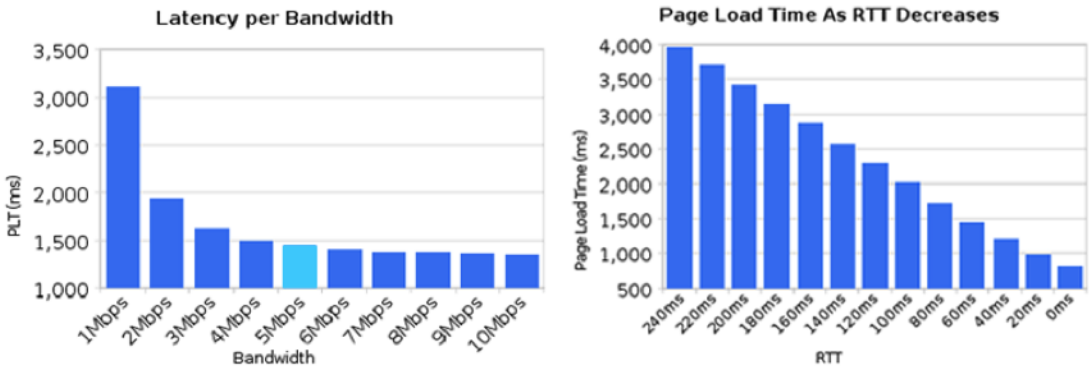


Figure 2.4: Page load time with different bandwidth and Round Trip Time

HTTP 1.1, which is the underlying protocol carrying all the requests and responses, is a highly network latency sensitive protocol. As part of the reason for this is that HTTP is running on the top of TCP (The Transmission Control Protocol), which indicates each request on HTTP has to finish the three-step handshake to setup a TCP connection. And it takes at least two round trip time to retrieve a resource in a http request. In the following case as below, the time cost by TCP connection and round trip time in transferring object is taking the major part of the whole response time. Figure 2.5 explains the time cost in two round trip for an http request.

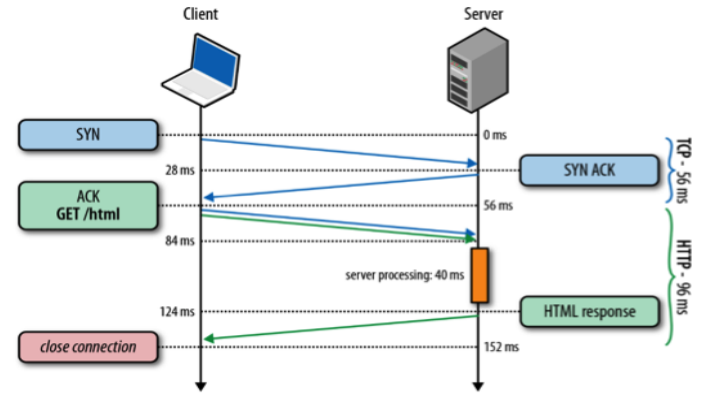


Figure 2.5: Two round trip time for an http request

Loading a web page is more resource intensive than ever (see the HTTP Archive's page size statistics), and loading all of those assets efficiently is difficult, because HTTP practically only allows one outstanding request per TCP connection. It comes with a problem of 'head-of-line blocking'.

HTTP/1.x has the problem where effectively only one request can be outstanding on a connection at a time. In specification of HTTP/1.1, pipelining feature is added to solve the problem, which later has been found not fully address the head-of-line blocking problem and difficult to deploy in practice. Also in HTTP/1.1, the keep-alive option is added into request headers, to make it possible to reuse an established TCP connection avoiding create a unique connection for each request. Figure 2.6 shows requests over HTTP/1.1 using keep-alive option.

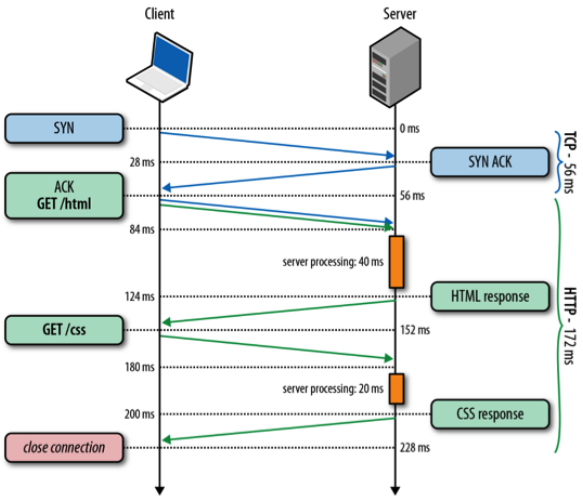


Figure 2.6: HTTP 1.1 requests reusing connection with keep-alive option

However, with keep-alive option enabled, the later requests reusing the existing connection still have to wait for the pervious requests to receive responses where the blocking problem occurs. In a worse situation, a very slow response for the earlier dispatched request may block the other requests for long time and they are never dispatched and doing nothing to wait for timeout for next dispatching which will cost another RTT. Lack of full multiplexing, the clients under HTTP 1.1 has to set up more than one connection to send the requests in parallel. In practice, popular browsers like Chrome will set up 6 connections to each domain at maximum. However, there are limits to this; if too many connections are used, it's both counter-productive. IETF mentioned that 'TCP congestion control is effectively negated, leading to congestion events that hurt performance and the network.' [IET15a]

2.3 HTTP/2 and SPDY

HTTP/2 is the second version of HTTP protocol and it is based on SPDY. SPDY (pronounced speedy) is an open network transfer protocol with an initial purpose of addressing some major problems of HTTP/1.1 but without changes for development.

Google Inc. develops SPDY and first published SPDY in 2009 as an experimental protocol for faster web. The high level goal of SPDY is announced [Goo09]:

1. To target a 50% reduction in page load time. Our preliminary results have come close to this target (see below).

2. To minimize deployment complexity. SPDY uses TCP as the underlying transport layer, so requires no changes to existing networking infrastructure.

3. To avoid the need for any changes to content by website authors. The only changes required to support SPDY are in the client user agent and web server applications.

4. To bring together like-minded parties interested in exploring protocols as a way of solving the latency problem. We hope to develop this new protocol in partnership with the open-source community and industry specialists.

The IETF set up Hypertext Transfer Protocol working group for next generation of HTTP. The httpbis-working group considered Google's SPDY protocol, Microsoft's

HTTP Speed+Mobility proposal (SPDY based), and Network-Friendly HTTP Upgrade. And the httpbis decided to choose SPDY as HTTP/2's base and the standardization effort comes to SPDY.

To address the multiplex and other inadequate usage problem of HTTP/1.1, HTTP/2 has several major features to distinguish with HTTP/1.1 by providing:

1. Binary framing layer, which change the way how data is exchanged between client and server using binary framing instead of text encoding streaming,

2. Streams and frames, as the prerequisite for multiplexing, now the binary framed data can be transferred in streams and frames rather than waterfall sequence,

3. Multiplexing in requests and responses, only one connection per origin yet handling different requests and responses to maximize the usage of one TCP connection,

4. header compression, using an index header table on both sides and incrementally record the changes in header fields and values, that can dramatically decrease header sizes especially in the context of http session.

Moreover, HTTP/2 provides other advanced features for better network performance. One of powerful new features is server push, granting the ability of server to send replies for a single request from client even without requests for additional resources. And with requests prioritization, the client now can suggest the priority of requests in order to have more important resources delivered like CSS files are often prior to JavaScript files for earlier page rendering.

The relationship between HTTP/2 and HTTP/1.1 is forward compatibilities. HTTP/2 provides new features without changes for header fields, status codes, HTTP methods and URI structures. For web development, modifications in application layer are not need to migrate to HTTP/2.

2.4 BBC R&D's HTTP/2 experiment

The recent evolution of next generation HTTP draws the attention of another department of BBC, the BBC Research and Development (BBC R&D), to follow up the latest trends and technology improvements. They noticed the HTTP version 2 was developed progressively in drafts and they are interested to evaluate the potential benefits and drawbacks of HTTP/2 in streaming media distribution.

To run a series of tests in streaming distribution over HTTP/2, BBC R&D has started an experiment on HTTP/2 released by Lucas Pardue in December 2014 [Par14]. As part of the Adaptive Bitrate Technologies project, BBC R&D is already looking at media delivery over HTTP-based formats. The main purpose is to designed a public trial that presents a public-facing HTTP/2 server providing access to a number of test MPEG-DASH streams. The reverse proxy (see Chapter 4 for introduction of reverse proxy pattern) is using the application called nghttpx, a proxy server implementation in nghttp2 project, to serve HTTP/2 request from internet and translate them into HTTP/1.1 to the origin backend. At the moment when the tests are conducted, HTTP/2 is still in the experimental state with latest draft version 14.

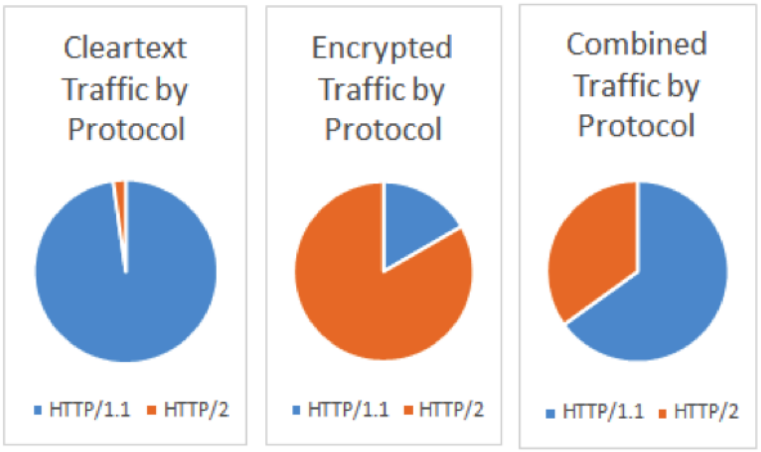


Figure 2.7: Breakdown of traffic on HTTP/2 and HTTP/1.1 in BBC Adaptive Media Streaming over HTTP/2 Trial Test

Recently in Jan 2015, some initial results have been revealed in a web blog article on BBC R&D website [Par15]. See the breakdown of traffic on HTTP/2 and HTTP/1.1 of the experiment in Figure 2.7. For the HTTP/2 activities, the HTTP/2 over clear text traffic access(h2c) is taking rather small share of all, while encrypted HTTP fared better, with 83% of requests being made with h2. The result is lead by the lacking support to h2c from most web browser.

From the recent public accessible initial results of this HTTP/2 on the department blog, it is not able to conclude any evidences or conclusions about the performances with HTTP/2 in streaming media distribution.

2.5 BBC Sport's Interest in HTTP/2

Compared to streaming media distribution, the page load performance and user browsing experience are more concerned by BBC Sport

BBC Sport is a department of the BBC North division providing national sports coverage for BBC television, radio and online, reporting the sports related news from popular sports like football and tennis to skiing and other sports. BBC Sport development team supports a series of products for BBC Sport users including BBC Sport websites and BBC Sport mobile applications including Android and iOS applications.

The website pages performance is affecting most BBC Sport products. One of the reason is that the mobile applications is actually displaying the web page contents directly from BBC Sport websites rather than using native system based constructed content.

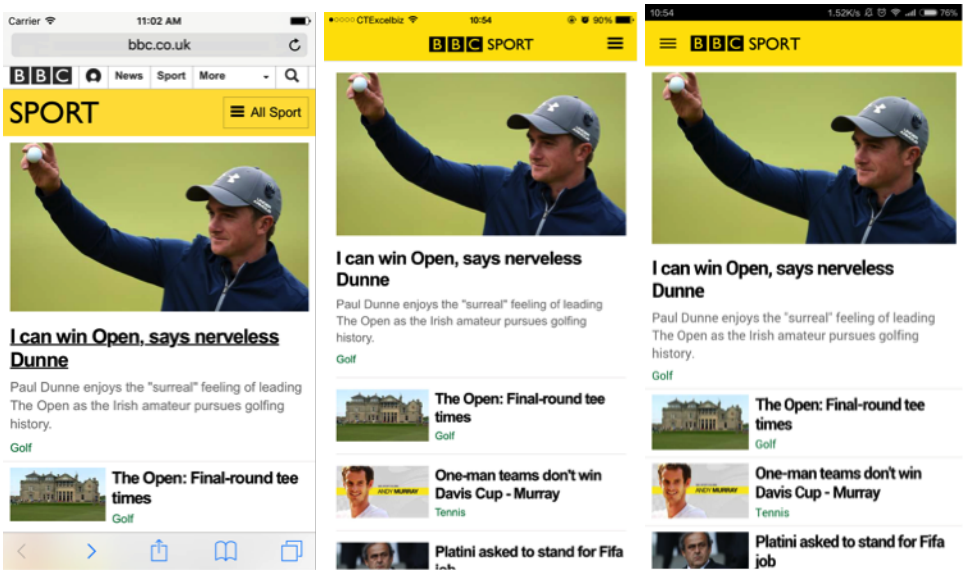


Figure 2.8: Homepage screenshots of BBC Sport website, BBC Sport app for iOS and BBC Sport app for Android

The Figure 2.8 provides an overview of the homepage screenshots of how the website shares the page contents with mobile applications on Android and iOS system. In the mobile apps, the latest contents from BBC Sport website are loaded and displayed inside the webview widget. Before the web page in webview is fully loaded, the mobile apps keep showing the waiting indicator for users. Each time launching the mobile apps to homepage or switching the pages (form football page to specific news article

page), it requires loading an entire web page from BBC website. Or it can be concluded any page performance change will affect the total waiting time and experiences for the users.

BBC Sport is eager to collect useful conclusion in this project for further deployment decision of HTTP/2, particularly in the browser support of user side and the performance HTTP/2 brings to users' browsing in real practices.

2.6 Related research methods and tools

In the scope of project, the research mainly covers two main parts including the statistics in client-side browsers and the web performance. For the investigation in user browsers, the analysis tools for statistics are used for fetching and generating reports. For the web performance, the quantity base method and a series of testing tools are used and applied to inspect the details of web page loading waterfall and the comparison of load test benchmark.

2.6.1 Research methodologies

Literature review The HTTP/2 related material publications, articles and data are collected from varying ways including school libraries and Internet sources. Since HTTP/2 is the standard for web application protocol, the specification and related applications are officially hosted on the Internet. As the sequence, the supporting data for the trends and compatibilities of HTTP/2 are collected from the Internet reports. Other earlier related research on SPDY and HTTP/2 will be gathered from publications and books.

Interview with technical specialists The information and architecture details of current BBC websites are directly gathered from BBC engineers to ensure the investigation suitable for BBC real situation and environment and its applicability for future migration to HTTP/2 deployment.

Statistical analysis tools Digital Analytix application, or app, measurement provides you with an in-depth view of interactions and conversion including detailed information about usage by device and type [Com15]. In this project, Digital Analytix is an analysis tool for collecting user browsing statistics and generating reports for querying request. BBC R&D uses it for web performance analysis and improvement.

The statistic and the trend about end-users' browsers, platforms and geolocation will be collected from this tool for showing how many users will be benefit from HTTP/2.

Network performance analysis tools WebPageTest is an open-source project and a free web service that provides a system for testing the performance of web pages from multiple locations around the world[Web15]. The browser runs within a virtual machine and can be configured and scripted with a variety of connection and browser-oriented settings. Following the test, the results are then available through a web interface.

2.6.2 Key measurements

The key measurements include load events and Speed Index. Load events are measured with PhantomJS or browsers via WebPageTest. The data source of Speed Index is WebPageTest as the speed index itself is the concept initialed and used by WebPageTest.

Load events DOMContentLoaded and load events are two major milestone timings to determine how fast or slow web pages were. Once the browser engine has finished parsing the main documents of the web application, the *DOMContentLoaded* event fires and the state of the page become ready. Moreover, the load event is fired when all page's resources are retrieved. The load event is easy to measure both in a lab environment and in the real world with the network performance tools, like Chrome Developer Tool or WebPageTest listed above.

Speed Index Speed Index is another similar indicator but used to measure how quickly the page contents are visually populated. It is first introduced into WebPageTest in April, 2012 [Web12]. Different from the load events, which are not very good indicators for end-user experience, the main focus of Speed Index takes the visual progress of the visible page loading and computes an overall score for how quickly the content painted.

$$\text{Speed Index} = \int_0^{end} 1 - \frac{VC}{100}$$

end = end time in milliseconds
VC = % visually complete

To do this, first is to calculate how "complete" the page is at various points in time during the page load and it is done in WebPageTest by capturing a video of the page

loading in the browser and inspecting each video frame. Second is to use the algorithm to calculate the Speed Index with the completeness.

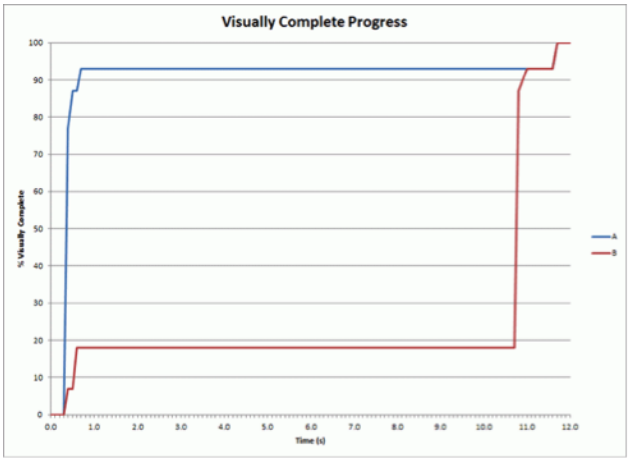


Figure 2.9: Visually complete progress in two cases

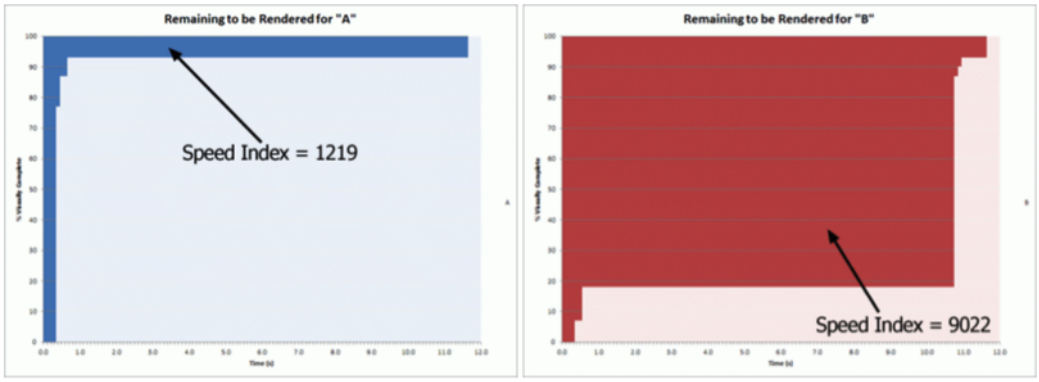


Figure 2.10: Speed Index for comparing two cases

Figure 2.9 and Figure 2.10 respectively shows the visual complete progress and explains the relationship between Speed Index and the visual progress of example case A and B. The lower in Speed Index is better which means faster the content is visually usable for end-users. Given the example case A and B below in Figure. As earlier the visually complete progress is shown in case A than case B, the Speed Index of A is much higher than B.

Chapter 3

Investigation of web browser support

3.1 Browser compatibilities of SPDY and HTTP/2

To enable the new major version of HTTP protocol and benefit from the features it provides, it requires the HTTP/2 support from the web browsers. For end-users, web browsers are the most commonly used applications in different devices for retrieving, presenting and traversing information resources on the World Wide Web over HTTP protocol.

According to the global usage share statistics data from StatCounter GlobalStats [Glo15], the most popular desktop browsers from July 2008 to Apr 2015 are Chrome, Internet Explorer, Firefox and Safari. For mobile and tablet devices, top used browsers is Chrome followed with Safari on iOS, Android webview based on WebKit, UC browser and Opera. Figure 3.1 indicates the trends of global share of global desktop browsers in recent 5 years while Figure 3.2 shows the trends of browsers on mobile and tablet devices. The support of the top browsers on either desktop or mobile platforms is the emphasis of the compatibilities investigation.

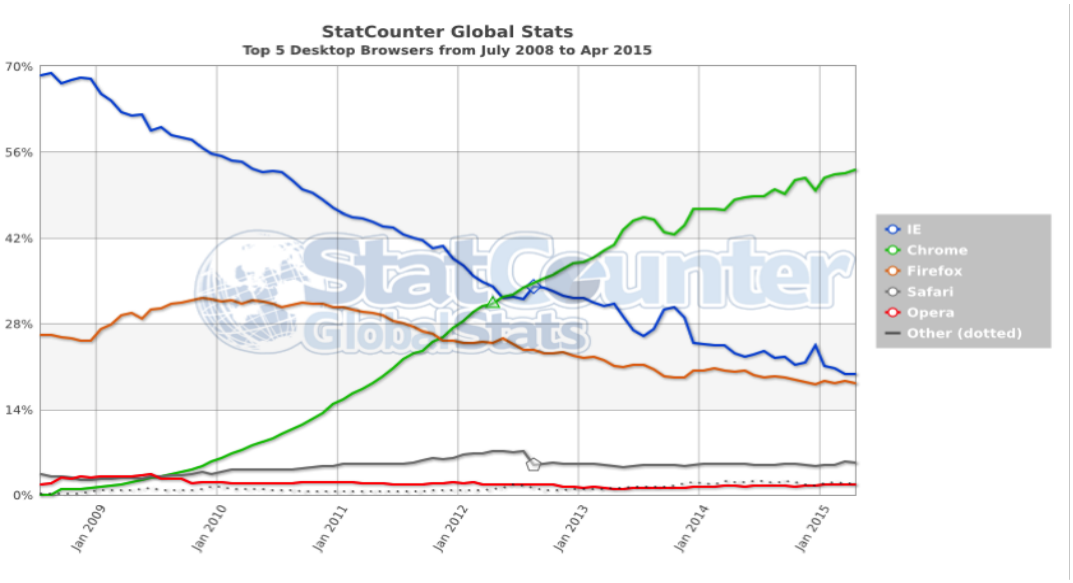


Figure 3.1: Global desktop browser share statistics

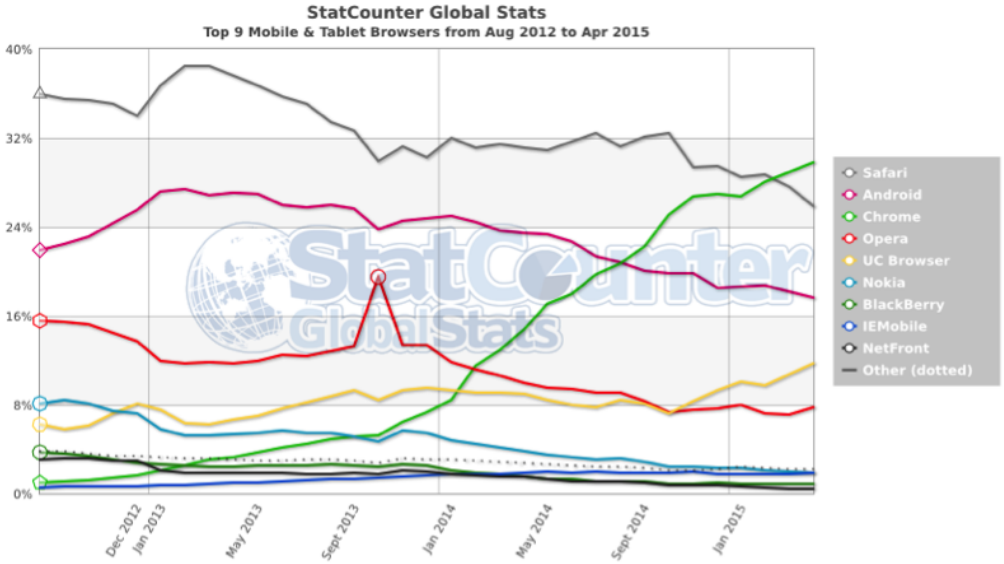


Figure 3.2: Global mobile and tablet browsers share statistics

A series of mainstream web browsers has released their support for SPDY protocol since 2010 and the HTTP/2 with the implementation for binary framing, prioritized requests and other features. Since HTTP/2 is actually based on SPDY with draft 0 chosen as the start point for as the working base for its specification draft and editing, the support for both SPDY and HTTP/2 will be discussed in this part.

Chrome is the first browser supporting SPDY protocol. In September, 2010, Google

releases SPDY in Chrome for all versions of Chrome 6 [Saf11]. Starting with version 40.x in Feb 2015 Chrome has already dropped support for SPDY/3 and only supports SPDY/3.1 going forward. As IESG has formally approved the HTTP/2 and HPACK specifications, the support for HTTP/2 is default to be enabled in Chrome 40.x and remove support for SPDY in early 2016, and to also remove support for the TLS extension named NPN in favor of ALPN in Chrome at the same time [CBB15].

Firefox started to support SPDY from version 11.x but it is not enabled by default until version 13. Moreover, HTTP/2 draft support is included in routine builds of Firefox, enabled by default in Firefox 34 and later. Firefox provides switch option in advanced settings via 'about:config', with option 'network.http.spdy.enabled.http2' for enabling or disabling HTTP/2 support and option 'security.ssl.enable_alpn' for ALPN.

Safari, the default browser on OS X operation system, is embracing HTTP/2 and started to support it from version 8 with OS X 10.x Yosemite. And Opera start to support for HTTP/2 since stable release of version 27.

Internet Explorer (IE) has support for HTTP/2 in version 11 and the support is available in Windows 10, which means users commonly use on windows previous version like Windows 7 and XP cannot benefit from HTTP/2. For further migration to HTTP/2, the latest news from Microsoft is that Windows 10 will be shipped with two browsers of IE 11 and its successor browser project Spartan, which of both are supporting HTTP/2. And Microsoft is planning to announce one-year free-of-charge migration to Windows 10 in Autumn 2015 which may encourage more users migrate to Windows 10 in second half of 2015 and a later period. Nevertheless, considering Windows 10 is still in beta stage and Windows 7 and XP would continue take a large share of PC users revealed by historical data, the fact that HTTP/2 is not supported by older version of IE will limit the general application of HTTP/2 in desktop browsing usage.

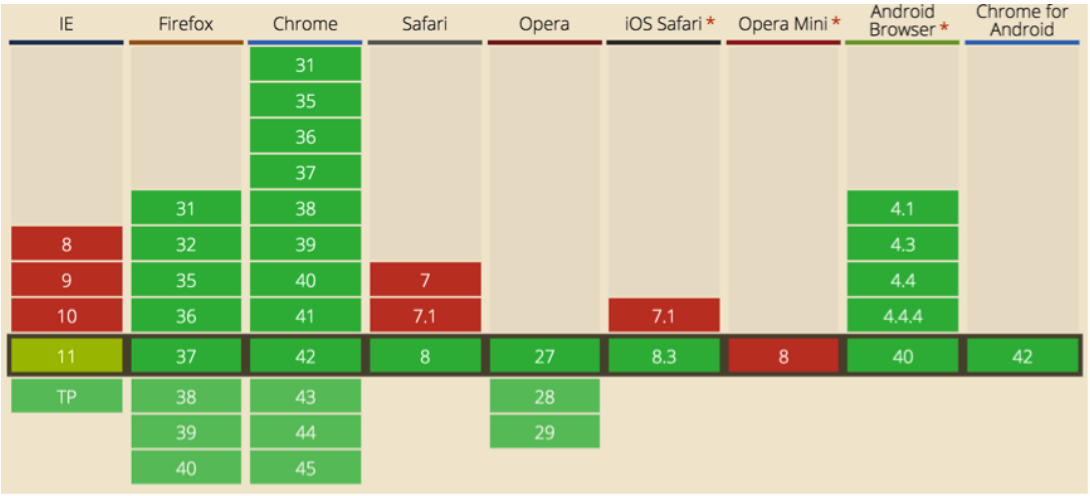


Figure 3.3: Compatibilities of SPDY and HTTP/2 on desktop and mobile browser aligned to current version

The compatibilities of SPDY and HTTP/2 in popular desktop and mobile browsers are listed for quick referencing in Figure 3.3.

For mobile devices, the support for SPDY and HTTP/2 in Chrome can be separated into Chrome for Android, which start to support HTTP/2 since version 41, and Chrome for iOS. The latter one and Safari for iOS initiate the support from Safari 8.x with version 8.x in iOS operation system. For Android browsers, means the web view and default embedded browser inside Android, started to support HTTP/2 after the release of Android 4.0 with Android browser 4.x based on Chrome core engine.

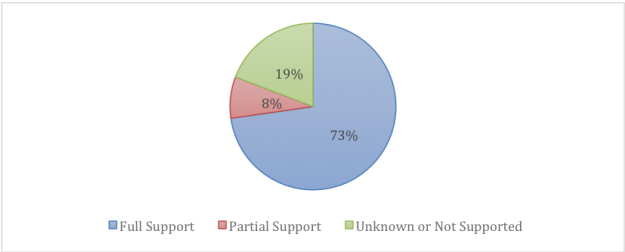


Figure 3.4: Global browser support for HTTP/2 and SPDY

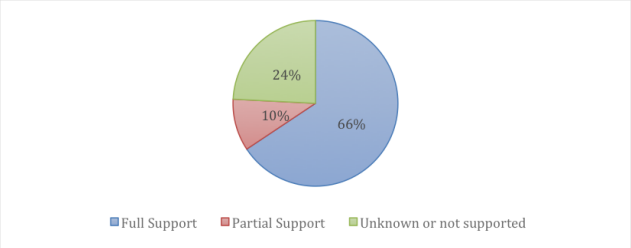


Figure 3.5: Browser support for HTTP/2 and SPDY in UK

The browser with support for HTTP/2 and SPDY are compared in Figure 3.4 and Figure 3.5. Based on the browser version usage share statistics based on data from StatCounter GlobalStats for March, 2015 [Dev15], the global usage with browser support for HTTP/2 and SPDY is 80.51% in overview, with 72.47% full support and 8.04% partial support. For UK users' usage, the overall support for HTTP/2 and SPDY is 75.8%, including 65.68% full support and 10.13% partial support. The partial support manly comes from Internet Explorer 11 without system version information to determine the support for HTTP/2.

3.2 HTTP/2 Browser support of BBC Sport visitors

Before deploying HTTP/2 to BBC websites, it is interesting to collect the browser information by accessing browser header logging and figure out how many visitors will be benefit from the features of HTTP/2.

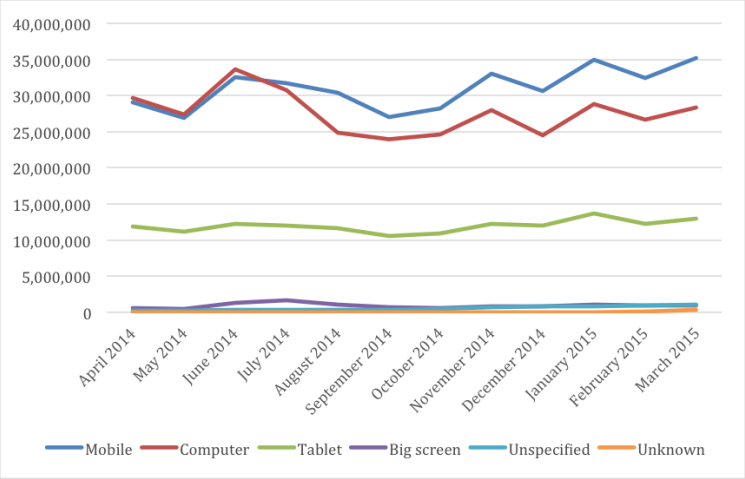


Figure 3.6: Browser platform share statistics of BBC Sport website from April 2014 to March 2015

With the statistics collected with Digital Analytix tools, the browser visits from April 2014 to March 2015 shows a trend that mobile devices (mobile + laptop) are taking a larger share from 57.33% to 61.23% with 17.58% annual growth in number of total visits, with the growing trends shown in Figure 3.6.

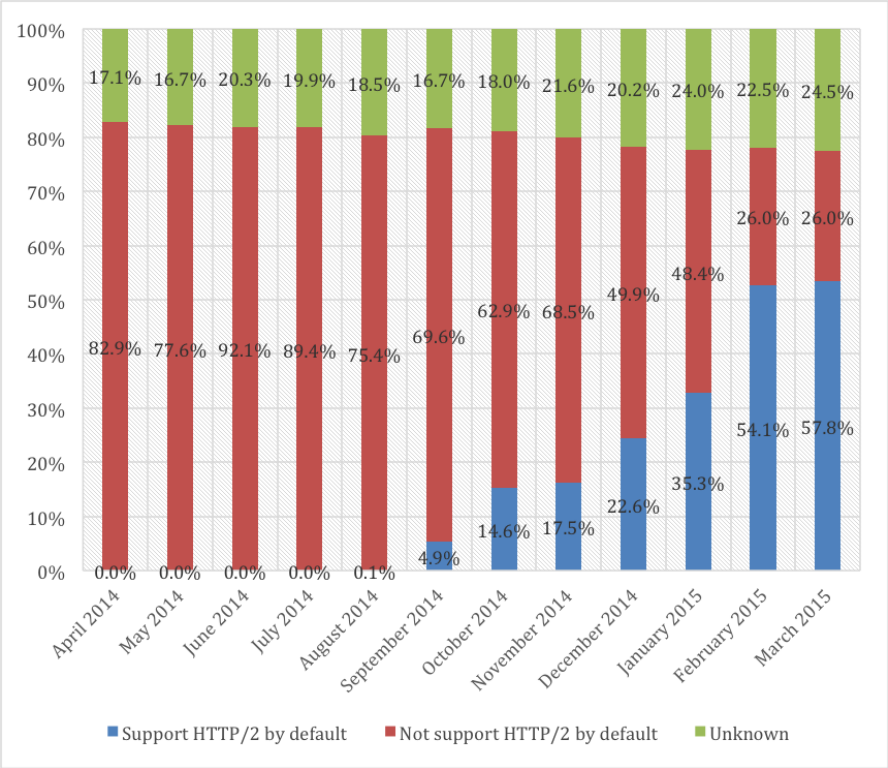


Figure 3.7: Stacked percentages bar chart of browsers with default support for HTTP/2 among BBC Sport website visitors from April 2014 to March 2015

Among all the visits, the graph in Figure 3.7 about percentage of major version of browsers is revealing the possible percentages with support for HTTP/2.

The class called 'Support by default' contains the major version of browser supporting HTTP/2 (excluding SPDY) by default without user's explicit configuration or option setting. It includes Chrome 31 and above, Firefox 34 and above, and Safari 8.x. And 'Not Support by default' includes IE 8 &9, Chrome 30 and below, Firefox 33 and below and Safari. As so far IE 11 can either be or not be running on Windows 10 with default support for HTTP/2, it is accounted into 'Unknown' class together with sole 'unknown' browser User Agent header and the sole 'Safari' which there is no way to determine the major version of it.

To conclude the browser support for HTTP/2 investigated in Chapter 3, almost all the mainstream web browsers are now with HTTP/2 supported by default. And the proportion of BBC Sport users with default support for HTTP/2 and SPDY is growing from 0 to 57.8% in the last year till March 2015, taking a dominant position. And with more users updated to latest version of web browsers and more mobile devices and laptops sold to the users, it is optimistic to see the rate will continue to climb up in recent years.

Chapter 4

Investigation of server side implementations

The focus of this part of investigation is HTTP/2 support in server-side implementations, including the reverse proxy server or intermediary and the web application framework based on different programming languages.

For most websites, web servers prevents users not directly accessed via the Internet connections. The requests from the Internet are often handled by the reverse proxy server and forwarded to web servers inside the internal network.

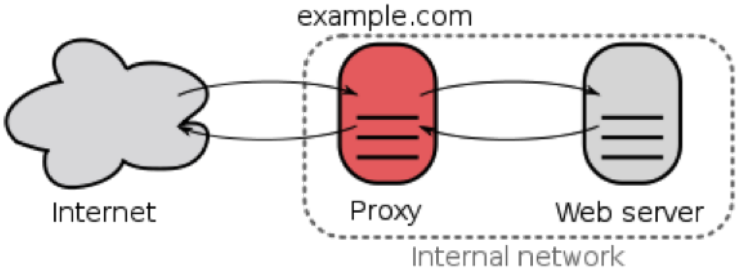


Figure 4.1: A reverse proxy taking requests from the Internet and forwarding them to servers in an internal network.

Common reverse proxy pattern is shown as Figure 4.1. With such pattern of practice, it takes of the advantages of reverse proxy server for static web resources caching, maintaining large number of connections and load balancing high throughput of data requests, while using web servers for generating dynamic contents with logics with different application programming languages.

For BBC websites to handling high throughput http requests, the same strategy is applied. Similar basic architecture of BBC websites is shown as Figure 4.2. When the

users requests reach data centre, they are distributed to different Varnish servers by the load balancers to reduce the load pressure. The Apache and Varnish servers as serving together as two-level proxy servers and intermediary for content caching and further requests distribution. The actual web pages are generated inside the PHP application using Zend framework.

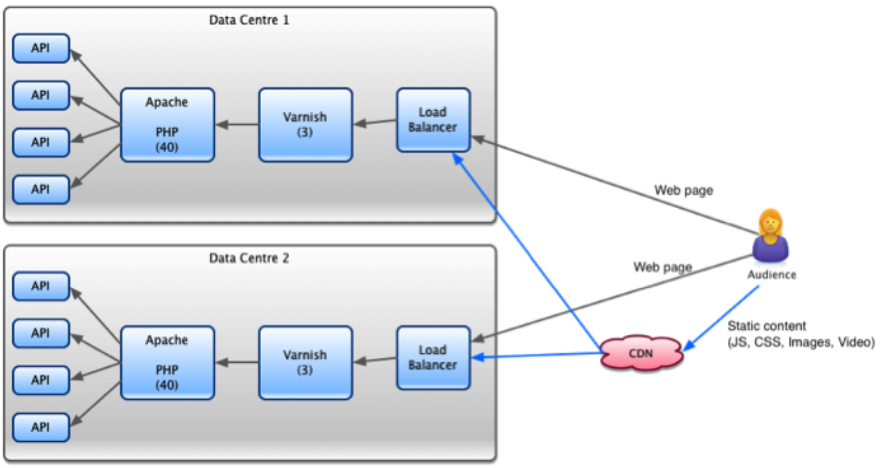


Figure 4.2: Basic architecture of BBC websites

To apply HTTP/2 support in BBC websites, a proper server side strategy has to be made. First is to find out the most suitable HTTP/2 solution of reverse proxy or intermediary among popular production-use implementations. And optionally, migrate the web application to enable HTTP/2 within programming languages or related frameworks.

4.1 HTTP/2 support of web servers

By the end of April 2015, the most widely used http server application is Apache with 57.5% of all web servers tracked on W3techs [W3T15b]. And it followed Nginx and Microsoft's IIS (Internet Information Services) with the share of 23.9% and 13.2%. Since the other minor web server applications are only holding the rest 6.2%, the top three web servers are the focus of HTTP/2 support. Figure 4.3 shows the trends and the usage of popular web servers used by worldwide websites [W3T15b].

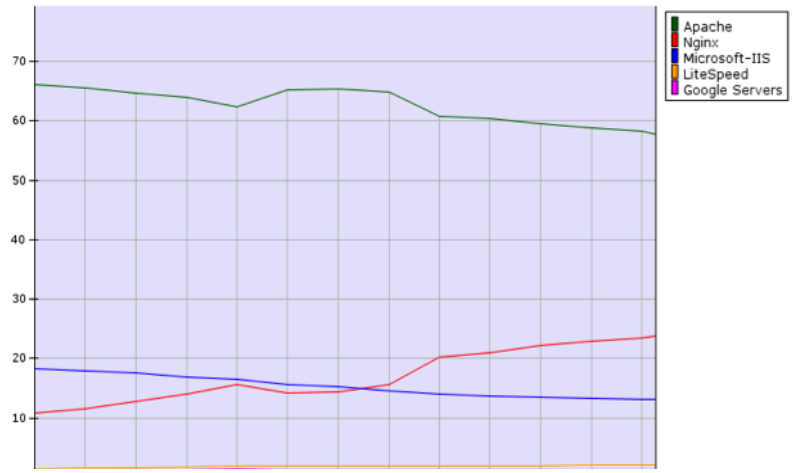


Figure 4.3: Usage of web servers of websites by April 2015

As what Figure 4.4 indicates, since SPDY was released publicly as the predecessor of HTTP/2 in July 2012, the usage of SPDY is growing up from zero to 3.9% recently. The reason for the earlier decrease in 2013 is SPDY is mainly used by Google websites. But in the later period, with the mainstream popular websites join to use SPDY support like Facebook and Twitter, more websites enabled the SPDY support.

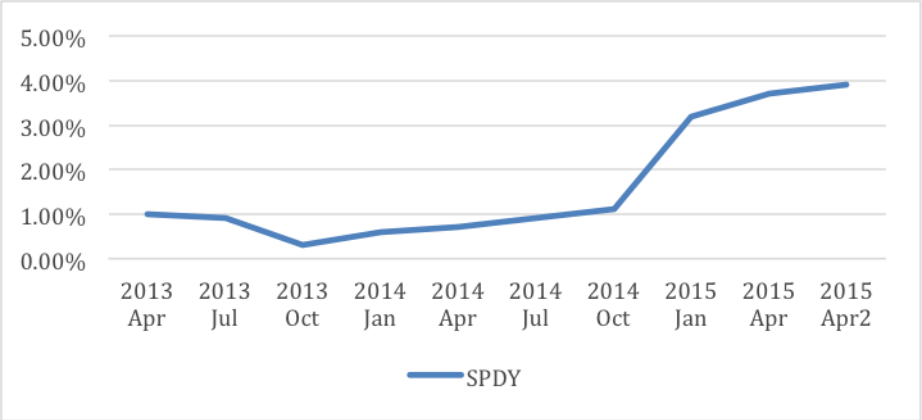


Figure 4.4: Historical trends of SPDY support of all websites since its release

Among all the SPDY-enabled websites, Nginx takes the absolute largest share with 95.6%.(see Figure 4.5) The other two generally popular web servers Apache and IIS are listed in top 4 with NodeJS in the third position. It reveals more aggressive attitude to SPDY and the interest in HTTP/2 among Nginx server and NodeJS server users.

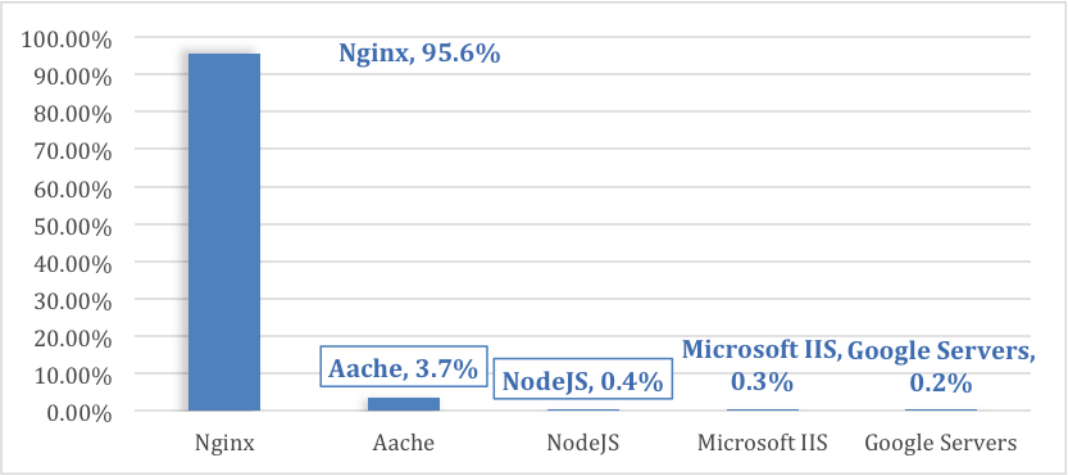


Figure 4.5: web servers share of SPDY-enabled websites

Nginx is currently unable to work with HTTP/2 but it is plan to release versions of both NGINX and NGINX Plus by the end of 2015 that will include support for HTTP/2 [Gar15]. Nginx is officially maintained a module called ngx_http_spdy_module to support SPDY draft 3.1. The module is shipped with distributed version but not built by default. SPDY support can be can be built along with Nginx using the –with-http_spdy_module configuration parameter and easily enabled inside configuration within only one line of code.

As for Apache httpd server, a module called mod_spdy brought SPDY support in 2014 by Google. It was donated in 2014 to become an Apache project integrated with http server distribution. Apache's mod_spdy is an open sourced module that allows requests over HTTPS to be served using the SPDY protocol.

The module called mod_h2 (https://icing.github.io/mod_h2/) can be another option for Apache httpd server. It contains Apache httpd module implementing the HTTP2 protocol. It uses Nghttp2 (https://nghttp2.org) as base engine and connects it with the Apache infrastructure. But it is maintained by individual without Apache official support.

For Microsoft IIS, the HTTP/2 support comes with latest technical release of Windows 10, which will be publicly released in mid-2015.

As conclusion, to enable HTTP/2 but prevent change web server implementation to avoid possible technical risk, the best strategy is to continue using Apache and apply mod_spdy once it starts to support HTTP/2.

4.2 HTTP/2 support of web frameworks

By April 2015, the top 3 popular server programming languages among all websites are PHP(82.5%), ASP.NET(17%) and Java (2.9%) [W3T15a]. PHP is also the common programing language used by BBC websites for web pages.

On the wiki of IETF's HTTP/2 official websites, the list page [IET15b] keep tracks on tracks known implementations of HTTP/2.

As for PHP websites, no tracked HTTP/2 implementation is available. Zend framework, which is used in BBC websites, is not currently working with HTTP/2. And there is no coming news of planned support for HTTP/2 from Zend.

For Java language family based websites, the popular Java web servers are ready for HTTP/2 with Jetty, Netty and Undertow.

For BBC websites, it is not wise to change programming language from PHP to any language else. As to serve HTTP/1.1 and HTTP/2 at the same time for future certain period, it is better to enable HTTP/2 in the intermediary and reverse proxy server.

The possible solution in a greater chance comes with the standalone server implementation solutions including H2O server and Nghttp2 server.

Nghttp2 is an implementation of HTTP/2 in C [Tsu15]. It provides a full stack HTTP/2 implementation form server, client, proxy server to the load benchmark tool. It also provides a series API for C++ and Python. It is well documented and easy to deploy. Nghttp2 is also used by some of other implementations as fundamental base library, by mod_h2 for Apache httpd server, Trusterd for ruby server and libhttp2 for D language.

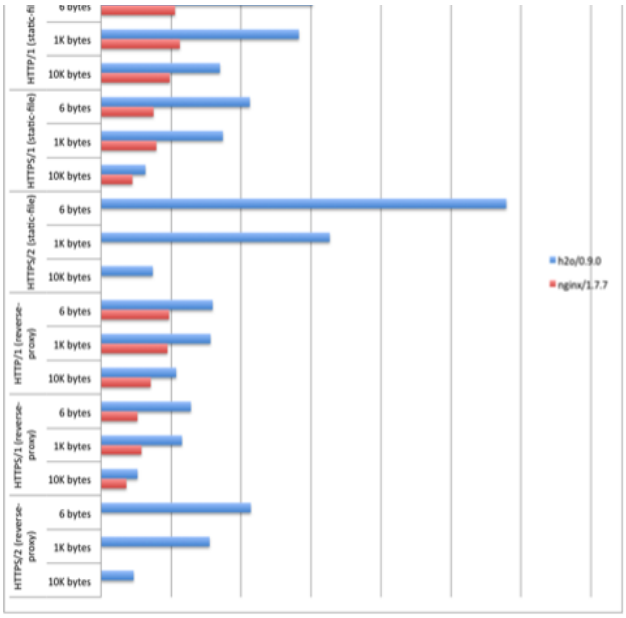


Figure 4.6: HTTP server benchmark comparison of H2O and Nginx

The other candidate is H2O, a popular open source HTTP/2 server project on Github.com. H2O is a very fast HTTP server written in C, serving both HTTP/1.1 and HTTP/2 [Oku15a]. In a series of load benchmark with results concluded in Figure 4.6, the scores were tested on Amazon EC2 running two c3.8xlarge instances (server and client) on a single network placement. Serving under HTTP/1.1, H2O scored twice much than Nginx. H2O has overwhelmingly score in different size of responses for requests.

The main conclusion for the investigation of server side implementations is that the best way to enable HTTP/2 for current websites in near future is adding an mature HTTP/2 reverse proxy server, for instance Nghttpx or H2O. And currently no web frameworks nor mainstreaming web servers fully ready for HTTP/2 support, but HTTP/2 feature is planned to added to the web servers within the future releases.

Chapter 5

Enabling HTTP/2 support for BBC Sport website

This section focuses the possible changes adapting current BBC Sport website to support HTTP/2 protocol. It starts with the review of current architecture and page contents of BBC Sport website. To satisfy requirements of HTTP/2, the new dual band architecture severing HTTP/1.1 and HTTP/2 is then suggested. Other changes in pages and the evaluation of cipher suites for enabling secure transfer layer are discussed.

Recently BBC Sport website is available over HTTP/1.1 without HTTPS support over secure transfer layer, which is an essential precondition of most HTTP/2 browsers for common end-users. Moreover, although HTTP/2 provides an optimized transport for HTTP semantics and supports all the core features of HTTP/1.1, HTTP/2 has some differences with HTTP/1.1 with new server push feature and multiplexing on one single connection.

Considering the reasons listed above, it is necessary to identify the possible changes in current BBC Sport website to enabling HTTP/2 support and give some feasibility proposals in setting up secure transfer layer and server push strategies for next version of BBC Sport.

5.1 Current BBC Sport website

Currently BBC Sport website is maintaining both two versions of website page contents. One is desktop version for desktop user browsing, while the other one is mobile version for mobile browsing and in-app browsing inside BBC Sport apps on Android and iOS devices. There are several differences in details of technologies used.

The desktop version website of BBC Sport is based on the domain of www.bbc.co.uk with homepage with URL as http://www.bbc.co.uk/sport/0/. The '/0/' is added to every BBC public websites, since BBC enable alternative web page design by adding the appendix in order to redirect all the requests to the alternative servers (shown as Figure). Every time requesting the homepage of desktop version, the single HTML response is returned with 200 status code and it is plain text rather then gzip encoding which can be sized over 169KB. And the whole size for BBC Sport desktop homepage can reach 1.1MB(varying depends on the size of images used).

On the other hand, the mobile version is a completely re-designed version. For URL, it is using the m.bbc.co.uk and hosting the homepage at http://m.bbc.co.uk/sport, which is comparatively shorter than desktop version. For transferring data, It is now gzip compression which dramatically reduces response size. For page design, responsive design is applied and make the website page much friendlier to users on devices with different screen sizes. For page content, the most dependencies from JavaScript libraries to images shown on the pages are now generated in runtime by inlining JavaScript scripts instead of using hardcoded HTML tag for referencing.

The mobile version of BBC Sport website is the subject as our targeted version in further discussion in this project, as the mobile version is being progressively replacing the legacy desktop version on both BBC news websites and BBC Sport websites during the progress of this project.

The general architecture of BBC Sport website. Both versions of the website is served by Apache server software inside the server clusters. The requests from Internet is served by one of the Apache servers and the original pages is generated by PHP servers with the support of other related supporting API services inside the BBC Sport. Varnish servers are used in front most for resources caching and handling high numbers of concurrent TCP connections.

5.2 Chances of BBC Sport website for HTTP/2

To display a web page, the browser is performing a set of iteration from fetching HTML file, discovering dependencies, fetching resources and then parsing them and rendering to the display for users. HTTP serves as the critical network protocol performing resources retrieving and interacting with servers directly. The problems led by the HTTP/1.1 can be resolved or improved by HTTP/2 at some degree. These could be chances for HTTP/2 as listed below:

1. Too many TCP connections opened for browsing a single page over HTTP/1.1

In a common example of loading the homepage of BBC Sport (in Figure 5.1), the browser opens 24 connections to 11 different domains over HTTP/1.1. More established connection cost more additional resources in memory and computation resources. With multiplexing over single TCP connection provided by HTTP/2, the number of total connections are reduced to 11 connections as 'one connection per domain'.

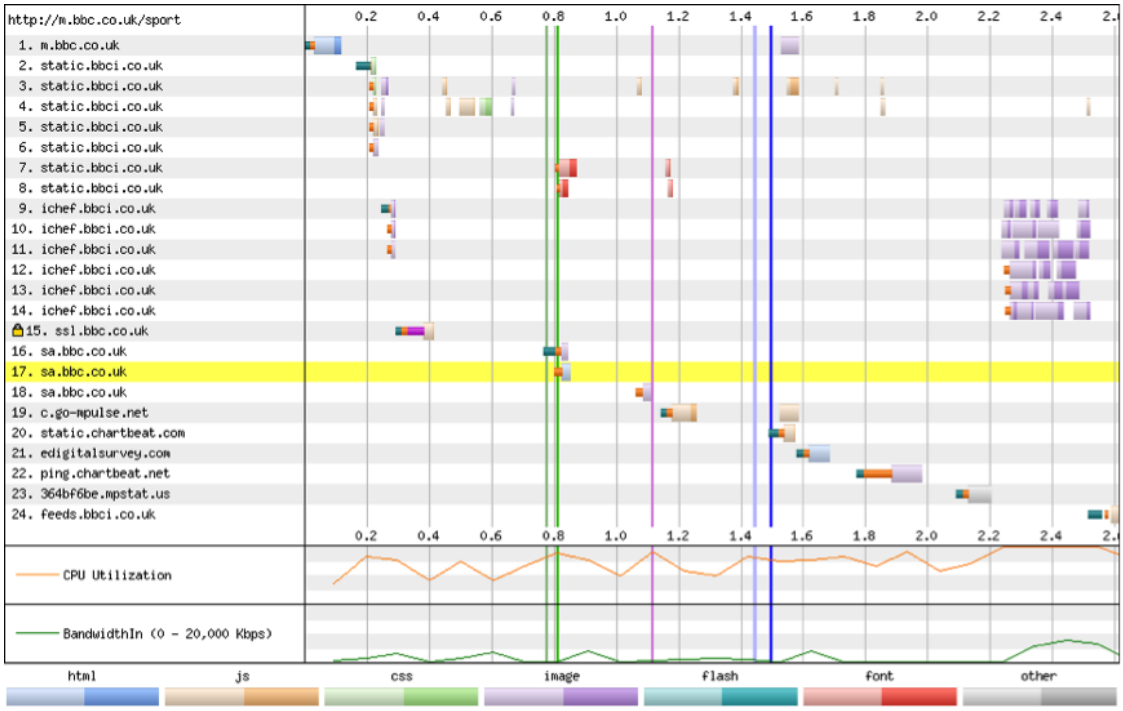


Figure 5.1: Connection view of waterfall and bandwidth utilization of loading mobile homepage on http://m.bbc.co.uk/sport

1. Inadequate bandwidth utilization

   The bandwidth utilization graph in Figure shows inadequate utilization in incoming bandwidth over HTTP/1.1. Even with 20,000kbps maximum bandwidth, the peak of utilization only reaches 1/4 before the document complete event fires. The increase in bandwidth probably cannot accelerate faster bowering as it is not the bottleneck of performance. But with HTTP/2, the multiplexing and server push features will enable faster resources loaded or pre-fetched.

2. Network latency has more impact in page loading time

HTTP/1.1 without multiplexing leads a problem called 'head-of-line blocking'. It makes latency in network layer a major factor for page loading. Take the snapshot in 2.2-2.4 seconds loading images as example, every request has to be sent after previous ones. In HTTP/2, the resources can be requested to server once they are discovered which overcome the similar problem occurs in HTTP/1.1.

3. Inlining for HTTP/1.1 can be replaced by server push in HTTP/2.

   Inlining is applied commonly on BBC Sport web pages as one of optimization technology used for HTTP/1.1. HTTP/2 can use server push to fulfill same content delivery, and it does decouple the inlined resources and the html file itself simplifying the production flow on server side and the workflow of page development.

5.3 Challenges of BBC Sport website for HTTP/2

While HTTP/2 has potential benefits in most aspect of networking for web page browsing, there are also several challenges for enabling HTTP/2 support for BBC Sport websites.

First, an obvious challenges comes from the current situation of lacking secure transfer support in BBC Sport web pages. HTTP/2 prepares two identifiers, where "h2" identifies the protocol where HTTP/2 uses TLS and "h2c" identifies the protocol where HTTP/2 is run over cleartext TCP. But in the real world situation, the market dominant browsers including Chrome and Firefox are only supporting HTTP/2 over TLS.

Second, subdomains of BBC Sport are not support HTTPS as well. The browsers loading a web page over TLS is requesting all depending resources over secure transfer layer. Otherwise, a warning will be shown and it stops further parsing the scripts from domains via cleartext transfer protocol. Several main subdomains used for BBC Sport, including static.bbci.co.uk, c.files.bbci.co.uk, and ichef.bbci.co.uk, are both powered by Akamai as the CDN services provide worldwide. Akamai claims to be ready for HTTP/2 support but it is still in limited beta status [Aka14a]. To enable HTTP/2 for BBC Sport, it is necessary to enable HTTP/2 support for CDN services with updated contract with Akamai.

Third, the current web pages are using the hardcoded path of scripts and CSS style files and resources over cleartext HTTP with 'http: //' prefix. One possible solution is

to simply replace the relevant link with prefix ‘https: //’. Another way to use relative reference starting with prefix ‘//’ to let the browser decide available protocol either HTTP or HTTPS to fetch the resource. This relative style network path reference is defined in RFC3986 [TBL05] and being accepted for all main steaming browsers.

Fourth, BBC Sport has to support both HTTP/1.1 and HTTP/2 at the same time for many years in foreseeable future since HTTP/2 support would be added. Although the previous investigation shows more and more users has default support for HTTP/2, there are still users cannot access HTTP/2 only websites for legacy reasons. BBC Sport has to prevent over use or anti-patterns of HTTP/2 meanwhile maintaining the compatibility for all users and.

Chapter 6

Design and Implementation

To compare the performance of HTTP/1.1 and HTTP/2, a testing system is designed and deployed. The web page contents from BBC Sport website will be tested on the system over both HTTP/2 and HTTP/1.1. The objectives and assumptions of the testing system will be identified and then the system architecture and technologies used are introduced with other technical details.

Inside the testing system, a proxy is created and developed which is designed for controlling and enabling server push strategies based on the requirements of testing system. Several different modes for server push rules and content parsing are enabled. Plus, the overview of API (the application programing interfaces) is listed for switching server push rules or modes at runtime.

6.1 Objectives and assumptions of the testing system

The testing server system is a foundation part of this project, since the tests for page loading performance and benchmarks are running with this server system. The performance data and detailed comparison will help the decisions of real deployment in BBC websites.

The design of server system has to reflect and satisfy the objectives and subjects of the investigation. The main objectives of the testing server system includes:

1. The testing server system is designed for comparing the performance differences of HTTP/1.1 and HTTP/2. The testing server system is able to handle and response the different requests respectively over HTTP/1.1 or HTTP/2. For HTTP/1.1, the testing server system is running both HTTP and HTTPS.

2. User side performance including page loading time and efficiency of resources fetching is the main target for tests. A several levels of test for this subjective can be carried on this testing server system.

3. Collecting the details of page loading waterfall or benchmarking to quantify the performance differences of HTTP/1.1 and HTTP/2 for further comparison and analysis.

4. The BBC Sport web pages are used for testing the performance loading current BBC Sport websites under different version of HTTP. The responses of the server system remain untouched as much as possible, with the original details from the structure and sequence of web page content to visual display effects of BBC Sport website.

5. Support for server push feature by HTTP/2 is available and tested with this system. Different rules and strategies of server push for different pages URL can be applied and changed during runtime.

Assumptions for the system are also important and considered in the design of system. From previous investigations in client web browser and server-side implementations, several assumptions for demonstrating HTTP/2 of BBC websites are listed:

1. It is ready to deploy HTTP/2 for BBC websites, since over half of the browser visits now has HTTP/2 support by default by the end of 2014 and the ratio will continue to increase in the coming future.

2. The support for both HTTP/2 and HTTP/1.1 has to be maintained as the same time. The old browsers or users on systems without updates will continue use HTTP/1.1 to connect BBC Websites.

3. Use HTTP/2 over TLS (as using ‘h2’ identifier) rather than over clear text TCP (with ‘h2c’ identifier). In the specification of HTTP/2, it defines both ‘h2’ and ‘h2c’ for HTTP/2 over TLS and clear text TCP. However, for the mainstream browsers, Google Chrome and Mozilla Firefox announced that they only implement h2 for constrained security and integrity.

4. For HTTP/1.1, the system uses the Apache server which is currently used by BBC Sport website to demonstrate endpoint server for incoming request. For HTTP/2, since recently no available plugin enabling HTTP/2 that is suitable for

Apache server, the system will use the current available implementations for demonstrating HTTP/2 server.

5. To maximize the effect of multiplexing over connection layer, the dependent resources of the web pages will be loaded from the test server as much as possible. Current CDN based resources will also proxy by the system. Some possible changes in page content will be applied.

6. To control the factors in secure transfer layer, the same cipher suite and certificates used for testing HTTPS over HTTP/1.1 and HTTP/2.

6.2 System architecture

The testing system includes two general major parts demonstrating the behavior of servers and browsers. One part is severing as the proxy server for BBC Sport website and it is the place where both server implementations of HTTP/2 and HTTP/1.1 installed. The other part is performing client-side browsing behaviors, where the related browsers and benchmark tools are ready for collecting performance details.

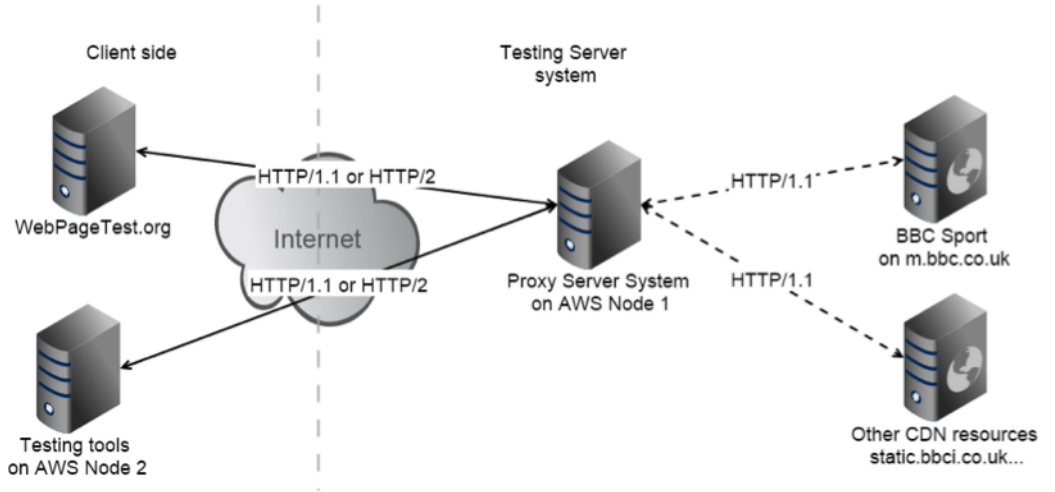


Figure 6.1: General architecture of testing system

The figure 6.1 above shows the general architecture of the testing system which is following the reverse proxy pattern. The reverse proxy pattern is introduced as the

basic architecture of the testing server system in order to satisfy the previous objectives and conform the assumptions of the testing server system, especially ensuring the responses from the testing server system have no differences in content and the results collected on the client side are comparable.

BBC Sport provides two computation nodes on the Amazon Web Services (AWS) for the uses of this project. AWS No.1 is used for setting up the proxy server system, while AWS No.2 is served as part of client side with benchmark tools installed. They are both public accessible via Internet.

6.2.1 Client side

Client side is designed for collecting two important measurement matrixes of end user experiences is the benchmarking for network resources retrieving and the waterfall with performance details for web page browsing.

WebPageTest, which is an open source web page performance tool, is used for collecting details of page browsing with the mainstream browsers, including the waterfall, a series of page load event times and other raw performance details. As WebPageTest has different test nodes worldwide, making use of them to access the test server via Internet could make the results more reliable and close to real-world practices.

For benchmarking the server performances, a set of benchmark tools is installed on AWS No.2 machine. Since no current benchmark tools support both HTTP/2 and HTTP/2, separate tool is used for either protocol. Apache HTTP server benchmarking tool, or well known as ‘ab’, is used for benchmark HTTP/1.1 performance while h2load with similar features and command line usages tests the performance of HTTP/2. Nghttp will be also used to demonstrate the HTTP/2 headless client for comparing resources loading details with or without server push enabled.

6.2.2 Testing server system

On the server side, it mainly contains the proxy server system with main logics and configurations for the tests.

The proxy server system is handling the requests from Internet over HTTP/1.1 or HTTP/2 on different ports. The request will be parsed and passed to real resources on BBC Sport website and other related CDN websites over HTTP/1.1.The structure of testing seserverver is shown in Figure 6.2.

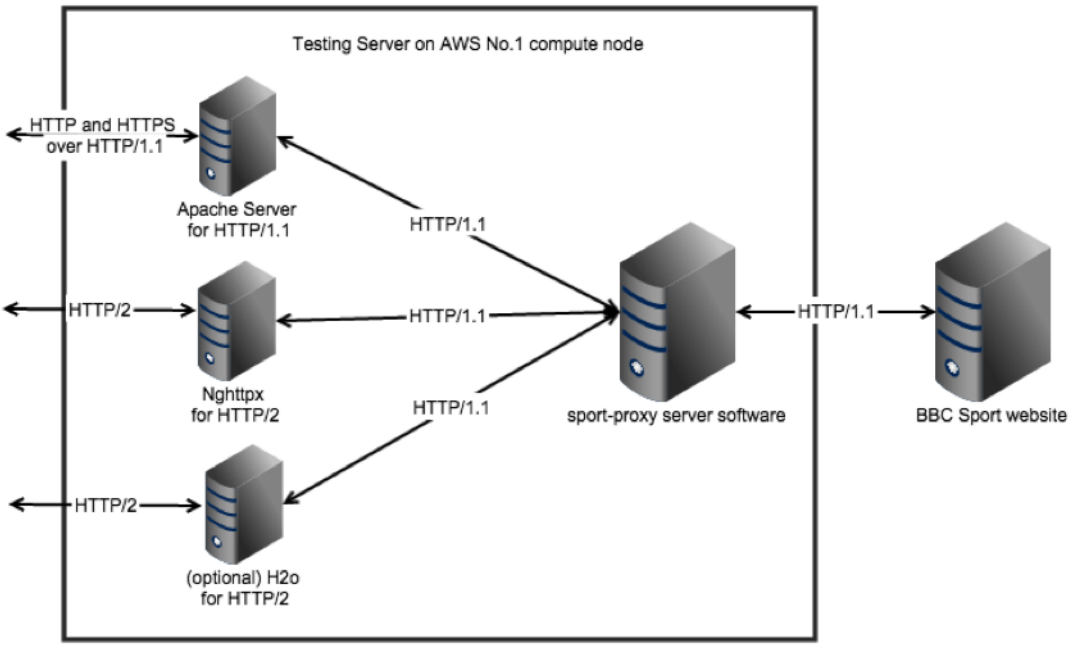


Figure 6.2: Inner structure of testing server on AWS No.1 compute node

Inside the testing server, a server program called 'Sport-proxy' will fetch the real resources on BBC Sport website for all the proxy server runtime instances The Sport-proxy is a lightweight proxy server crated and developed during this project and the details of Sport-proxy will be discussed in the following part of this chapter.

For the incoming requests, the Apache server is in charge of dealing the HTTP/1.1 requests including the ones over clear text HTTP and secure transfer layer on HTTPS.

And Nghttpx and H2O are handling HTTP/2. Nghttp2 is now one of the most mature HTTP/2 implementations. Nghttpx is the reverse proxy server of Nghttp2 project, which is written in C and commonly used as the foundational core by many projects for other language HTTP/2 server. The unofficial HTTP/2 plugin for Apache server called 'mod_h2' (open sourced on https://github.com/icing/mod_h2) is also based on Nghttp2.

H2O is optionally deployed to serve HTTP/2 requests as well. Setting up an H2O server for HTTP/2 in parallel as a reference, since H2O has a potential higher benchmark performance than Nghttp2 and it probably has the highest performance in current HTTP/2 server implementations.

Either of Apache, Nghttpx and H2O uses the same certificate for TLS/SSL secure

connection, which is of RSA encrypted and has 2048 bit in length. The default cipher suite used with Chrome is using ECDHE_RSA for key exchange mechanism and AES_128_GCM for encryption and authorization with SHA-256 hashed.

6.3 Detailed information of technologies used

Amazon Web Services This testing system is using two compute nodes on Amazon Web Services, which are the instances of type m4.large and equipped with 2 virtual CPU cores and 6 gigabytes memory. Details and system information of the instances:

- 2.4 GHz Intel Xeon E5-2676 v3 (Haswell) processors
- 8 gigabytes memory
- SSD based storage
- Ubuntu 14.04.2 LTS (Trusty Tahr, long term support version)
- Located in EU-Ireland region
- Support for Enhanced Networking
- Providing a balance of compute, memory, and network resources for common applications

Apache HTTP server Apache HTTP server is an open-source HTTP server for modern operating systems including UNIX and Windows NT. Apache HTTP server is installed via package manager in Ubuntu 14.04 and version 2.4.7 is the latest available version.

Simplified configuration for setting up proxy server on Apache HTTP Server:
(default configuration file path in system: /etc/apache2/sites-available/000-default.conf)

```
...

<VirtualHost *:80>

# passing the requests to port 8888 on localhost

        ProxyPass / http://127.0.0.1:8888/
```

```
</VirtualHost>

<VirtualHost *:8000>

#passing the requests to port 8888 on localhost

        ProxyPass / http://127.0.0.1:8888/

        SSLEngine on

# path to certificates and private keys for TLS/SSL

        SSLCertificateFile /home/ubuntu/cert/server.crt

        SSLCertificateKeyFile /home/ubuntu/cert/server.key

</VirtualHost>

...
```

Nghttp2 with nghttpx and nghttp Nghttp2 is an open sourced implementation of the Hypertext Transfer Protocol version 2 in C language. It consists of a full stack of HTTP/2 implementation including nghttp for HTTP/2 client, nghttpd for HTTP/2 server, nghttpx for simple reverse server and h2load for HTTP/2 benchmark tool.

This project is using the latest version 1.1.2. Nghttpx is acting HTTP/2 server in testing server system, while h2load and nghttp are utilized on client side.

Configuration file in default path redirecting the requests to port 8888:

```
...
front-end=0.0.0.0,8001 # listening port 8001

backend=127.0.0.1,8888 # pass requests to port 8888


# path to certificates and private keys for TLS/SSL
```

```
private-key-file=/home/ubuntu/cert/server.key

certificate-file=/home/ubuntu/cert/server.crt


# remove the default 'via' header to maintain original headers

no-via=yes


# running http in daemon mode

pid-file=/home/ubuntu/nghttpx.pid daemon=yes

...
```

H2O H2O is a new generation HTTP server providing quicker response to users when compared to older generation of web servers. Written in C, it can be used as a library. H2O is also open sourced and written by Kazuho Oku [Oku15b]. H2O version 1.4.2 satisfies the final draft of HTTP/2 and is applied in this server system.

Configuration for H2O server:

```
error-log: /home/ubuntu/H2O-error.log

listen:

# listening on port 8002

  port: 8002


# path to certificates and private keys for TLS/SSL

  ssl:
```

```
    certificate-file: /home/ubuntu/cert/server.crt

    key-file: /home/ubuntu/cert/server.key

hosts:

  "http2test.org:8002":

     paths:

/:

# passing requests to port 8888

      proxy.reverse.url: "http://127.0.0.1:8888/"
```

H2load and Apache Benchmark Tool As no benchmark tool supporting both HTTP/1.1 and HTTP/2, different benchmark tools are used in the system for either protocol. Apache Benchmark Tool (or 'ab') is for HTTP/1.1 while H2load from nghttp2 project is for HTTP/2.

They have similar usage on command line terminal. Two critical parameters for every test are the number of requests with '-n' parameter and the number of concurrent clients passed with '-c' parameter.

Node.js and Express web framework The Sport-proxy software is written in JavaScript language and running on the Node.js engine. Node.js is a platform built on V8 engine for easily building fast, scalable network applications [Fou15]. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Express is a fast and minimalist web framework for Node.js. The Sport-proxy software uses Express as the basic web framework to accelerate developing progress and apply best practices of web development for common web operations.

6.4 Sport-proxy server software

Architecture of Sport-proxy software Sport-proxy software is a lightweight web server program particularly designed for satisfying the requirements of a series HTTP/2 tests in this project for BBC Sport. It is written in JavaScript language and running on Node.js platform. Two major requirements for the Sport-proxy are clear: One is to pass the request to real sources on BBC Sport and send back the responses, with some modifications in HTML responses changing the URI of the resources from rely on the remote BBC Sport to the test server.

The second main requirement for the Sport-proxy is to implement correct mechanisms for the server push modes and the auto-parse modes. They are separated logics for controlling server push when sending back the responses and generating server push rules automatic when no default rules are applied in auto-parse modes.

Sport-proxy uses the default modular mechanisms in Node.js to decouple features and encapsulate the modules. For the external module dependencies, they are installed via npm which is package manager in Node.js for JavaScript. Dependencies are used for ifferent purposes:

1. ExpressJS (http://expressjs.com/):

   Used for basic structure with Express.js web framework for handling request and responses and other http methods.

2. request (https://github.com/request/request): For requesting target resources on the remote location.

3. bluebird (https://github.com/petkaantonov/bluebird)

   a fully featured promise library used with focus on innovative features and performance for accelerating Sport-proxy development

4. compression (https://github.com/expressjs/compression):

   an node.js compression middleware , applied with express framework enabling gzip and deflate compression for the responses to reverse servers.

5. htmlparser2 (https://github.com/fb55/htmlparser2):

   an HTML/XML parser implementation in JavaScript, used inside Sport-proxy for auto-parse modes generating server rules

The architecture of Sport-proxy software is constructed as shown in figure below in Figure 6.3:

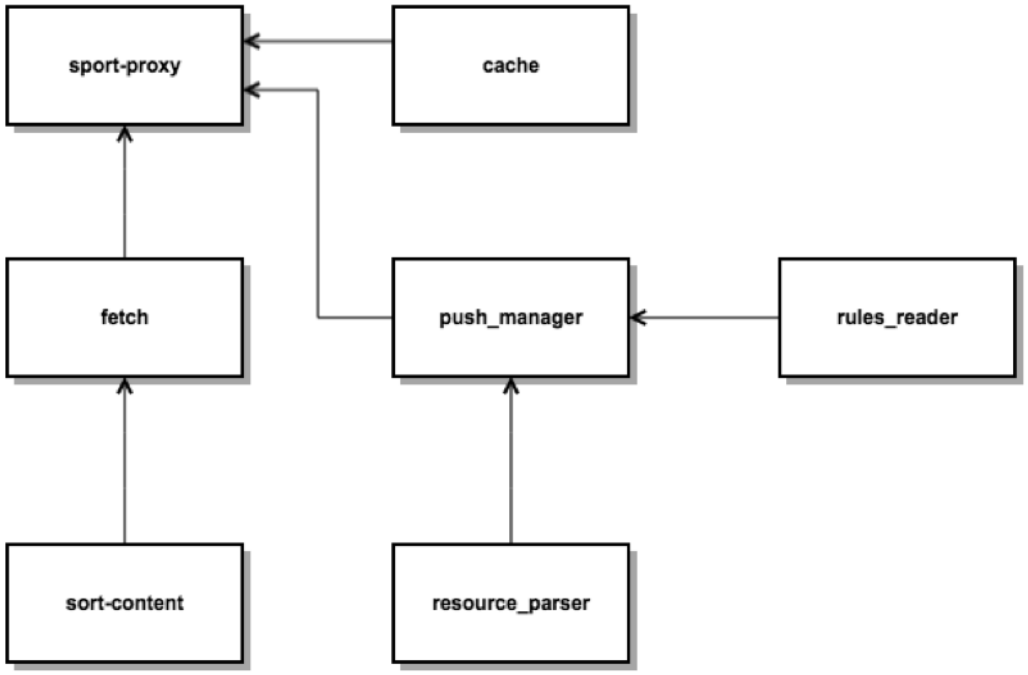


Figure 6.3: Modular architecture in Sport-proxy software

The Sport-proxy module is the entry point for starting the proxy server. It contains the main logic for setting up Express web framework and initiating instances of cache and push_manager module at runtime.

Cache module provides caching service and it is assigned for caching the body and content types for requests. It is where the cache will be store in memory and no advanced caching strategy applied.

Fetch module is for requesting the real resources on the remote BBC Sport related websites. Sort-content module then will be used in Fetch module for inspecting content of HTML files and apply some replacement modification making all the links relative and disabling the unnecessary noise requests which probably annoy and disturb the page loading waterfall and performances. Code snippets:

```
// STEP 1: make all links relative
```

```
body = body.replace(/http:(\\?)\/(\\?)\//g, /)
body = body.replace(/https:(\\?)\/(\\?)\//g, /)

// STEP 2; Disable page tracking that causes unnecessary noise:

body = body.replace("oqsSurveyManager(window, 'ON')",
"oqsSurveyManager(window, 'OFF')")

body = body.replace('loadChartbeat();', '')

body = body.replace("istats.isEnabled() ", 'false', 'g')

body = body.replace(" window.bbcFlagpoles\_istats = 'ON'",
" window.bbcFlagpoles\_istats = 'OFF'")
```

The important features of Sport-proxy for server push modes and server push rules control is fulfilled together by push_manager, rules_reader module and resources_parser.

The push_manager module will control the server push hints by adding Link header according the server push rules. Server push rules are generated at startup by rules_reader module from static JSON format file or auto parsed at runtime from HTML tags by resources_parser module. They can be also added or removed at runtime via web APIs provided by Sport-proxy and eventually added into server push rules in push_manager module.

Server push modes and auto-parse modes in proxy software

Server push modes One of the important jobs assigned to Sport-proxy software in this project is to control server feature by adding proper hints to the nghttpx and H2O.

For controlling the server push, the nghttpx and the H2O proxy server both support HTTP/2 server push feature by default. And they are using similar strategy, which is based on the Link header field(RFC5988). The reverse proxies inspect the responses from backend server and look for Link header and extracts URI-reference with parameter 'rel=preload' and push those URI to front-end client. The sample of Link header field for initiating server push:

```
Link:</static.bbci.co.uk/frameworks/barlesque/2.83.10/orb/4/style/
orb.css>; rel=preload

Link:</static.bbci.co.uk/frameworks/barlesque/2.83.10/orb/4/script/
orb/api.min.js>; rel=preload

Link:</static.bbci.co.uk/gelstyles/0.10.0/style/core.css>; rel=
preload
```

To enable and explore possible usage of server push for BBC, Sport-proxy provides 3 different modes server push for HTML page requests. The server push modes include No-server-push mode, Default mode, CSS mode and CSS+JS mode.

With No-server-push mode, the Sport-proxy will disable the server push feature by preventing adding any Link header before sending back to the HTTP/2 reverse server.

With Default mode, the HTML requests will be mapped to the same URI with existing server push rules and the Sport-proxy will add all the related resources into Link header one by one according to the rules.

CSS mode provides a filter for push rules where only the resources with URI ended with '.css' will be added into the Link header as the server hints.

Similarly with CSS+JS mode, the proxy will solely asked the reverse proxy to push the JavaScript and CSS resources.

Auto-parse modes Rules for server push can be set in either of following two ways. The one is to manually set the rules for specific URIs, either preloaded at launch phase from the JSON format file or set up at runtime.

Sport-proxy also provides the auto-parse modes for generating server push rules. Only HTML responses, which have 'text/html' in content type, will be parsed under auto-parse modes. Two modes including 'css' and 'css+js' are provided in auto-parse modes for facilitate server push generation.

As what the mode's name implies, in 'css' mode only the hardcoded Link to CSS resources will be generated for server push rules while in 'css+js' mode the references of CSS and JavaScript file.

Sport-proxy is based on htmlparser2 library module for html content parsing. Htmlparser2 (open sourced on https://github.com/fb55/htmlparser2) and has a generally better performance than other implementations written on Node.js. Benchmark results for html parsers on Node.js [Böh13]:

gumbo-parser : 34.9208 ms/file ± 21.4238
html-parser : 24.8224 ms/file ± 15.8703
html5 : 419.597 ms/file ± 264.265
htmlparser : 60.0722 ms/file ± 384.844
htmlparser2-dom: 12.0749 ms/file ± 6.49474
htmlparser2 : 7.49130 ms/file ± 5.74368
hubbub : 30.4980 ms/file ± 16.4682
libxmljs : 14.1338 ms/file ± 18.6541
parse5 : 22.0439 ms/file ± 15.3743
sax : 49.6513 ms/file ± 26.6032

By applying different event handlers for 'css' and 'css+js' modes, the correct tags are parsed and generated into server push rules. The logical codes below showing how auto parsing modes applied in getting push rules:

```
...
function getPushRules(url, body, filterlist) {
  if (push rules[url]) { return push rules[url].pushResources; }
  switch (rules gen mode) {
  case 'manual':
    return [];
    break;
  case 'css+js':
    resources parser.parseHtml(url, body, push rules);
        return filterRules(push rules[url].pushResources
, filterlist);
        break;
  case 'css':
    ...
  }
}
...
```

Getting the rules for resources with specific URL first is to look up existing server push rules. And then if no existing rules are found the responses from BBC Sport will be parsed under auto-parse modes besides the 'manual' mode. Inside the parse HTML function, the content type will be detected and only HTML will be parsed.

Chapter 7

Testing and Evaluation of HTTP/2

The evaluation of HTTP/2 is an important part of this project. This chapter provides the information about evaluation process. As HTTP/1.1 is the previous version for HTTP/2, HTTP/1.1 will be the reference object for HTTP/2 and it is involved in all evaluation of HTTP/2. The testing server system discussed in previous chapter sets up a base benchmark environment for both HTTP/2 and HTTP/1. A series of tests are running with the testing server system.

The evaluation of HTTP/2 will be focus on the user side performance, as HTTP/2 promises multiplexing, header compression and server push to improve web resources loading time. This project conducted three major parts of evaluation by comparing HTTP/2's performance with HTTP/1.1.

First part is the benchmark performance over HTTP/2 comparing to HTTP/1.1, which of the results showing the capabilities of web servers and it will affect the efficiency for responding to requests from clients.

The second part is conducted with browsers to inspect the changes in waterfall of web browsing via HTTP/2. Full-scale inspection covers from load time for whole web page and key dependencies resources to visual effect improvements and computation resources utilization.

The third part evaluates the performances with making use of the server push feature on HTTP/2 reverse proxy servers. To control server push rules and mechanism, server push rules and mechanisms will controlled via the functions of Sport-proxy software (see Chapter 7).

7.1 Load benchmark tests with static files

The performance of the web server is one of the primary concerns for BBC Sport. BBC Sport has a large user base with the website and the mobile applications of different platforms serve millions of user requests every day. Over 50 million views daily on average were initialed to BBC Sport website and the peak of single day reaches over twice of the average during the period from Aug 2014 to Jul 2015. Once the HTTP/2 is deployed in BBC Sport website, the performances over HTTP/2 server will affect considerable proportion of users using HTTP/2 enabled users.

The performance benchmark tests provides an overview of server side performances. To observe the benchmark results by comparing HTTP/2 and HTTP/1.1, the worry about the possible performance loss with switching to HTTP/2 could be explained.

In benchmark tests, static files are used as the target contents. The reason for this is to simulate the real world situation where caching all the requested files on the distribution servers with minimum performance loss. Static files in different sizes are tested to capture the differences between different scales of throughput capacities.

The test are included remote benchmark and local benchmark.

7.1.1 Local benchmarks

Local benchmark tests are designed for exhausting the performance with no public bandwidth limit and minimum networking loss. The tests are operated on the same machine as the testing server system. Using localhost as the target domain name to prevent cost of DNS.

Sizes of the response bodies varies in 4 different levels from 6bytes to 100KB to simulate real situations. Within each benchmark tests, the benchmark tool simulates 500 concurrent connections to generate 50000 requests in total with other default parameters.

For HTTP/1.1, Apache benchmark tool is used as the benchmark tool for both HTTP and HTTP requests. And h2load is the alternative for HTTP/2.

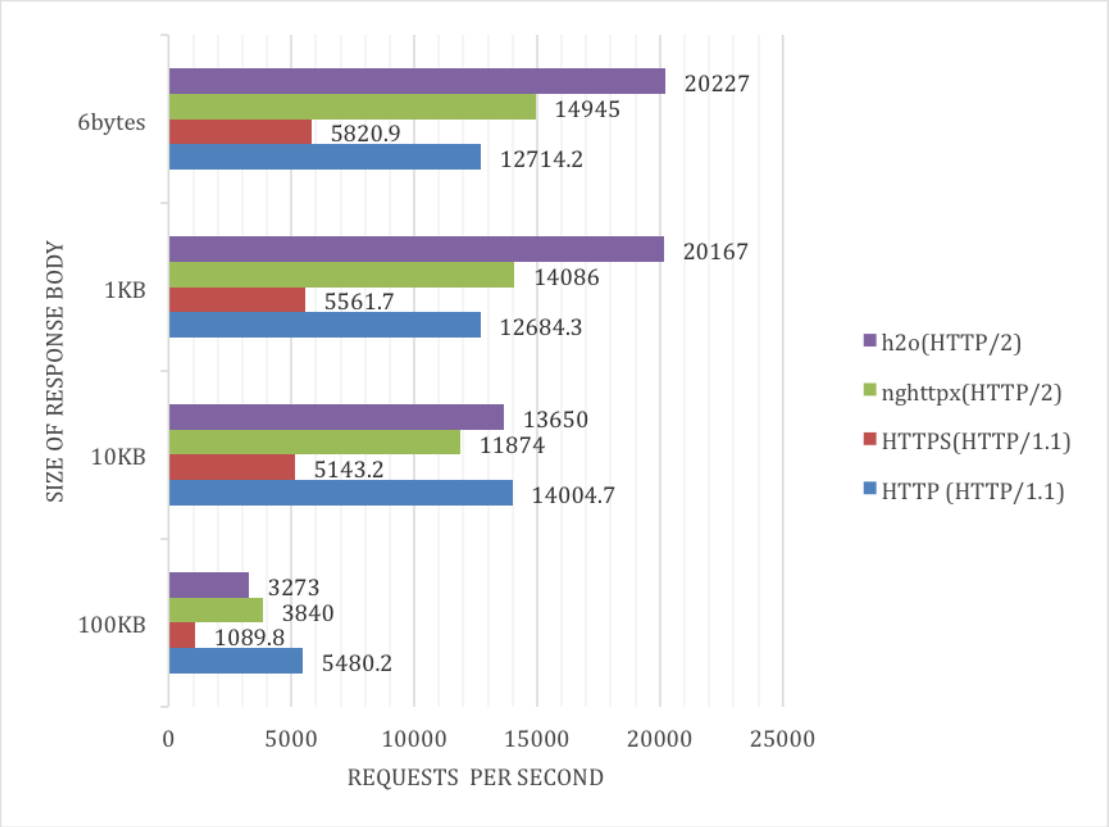


Figure 7.1: Local load benchmark scores of H2O, nghttpx over HTTP/2 and HTTPS and HTTP over HTTP/1.1

Figure 7.1 above shows the local benchmark scores of the candidate servers over either HTTP/2 or HTTP/1.1. In all the tests, the HTTPS over HTTP/1.1 by Apache server has the worst performance among all, with a significant difference behind others that the gap is growing from 1/2 of HTTP in 6 bytes test to 1/5 in 100KB tests.

For the small objects with size of 6bytes and 1KB, both of HTTP/2 candidates have better performances than HTTP/1.1 by Apache. The H2O is the best server candidate in the test of 6 bytes with 26% more scores than the next position nghttpx server and takes 37% more scores comparing to HTTP by Apache. Then in 1KB tests, it shows similar results that HTTP/2 servers take the lead ahead of HTTP/1.1.

But as the size of body increases multiple, the HTTP/1.1 service by Apache reverts to be top candidate better than other two HTTTP severs. And with the growth in sizes, cleartext HTTP/1.1 has more scores comparing to H2O and nginx.

7.1.2 Remote benchmarks

Accordingly, remote load benchmarks are designed for evaluating the performance differences between HTTP/2 and HTTP/2 in real-world environment and under bandwidth cap. The tests are proceed with same test tools and environment but on the other m4.large type machine in the same region of AWS.

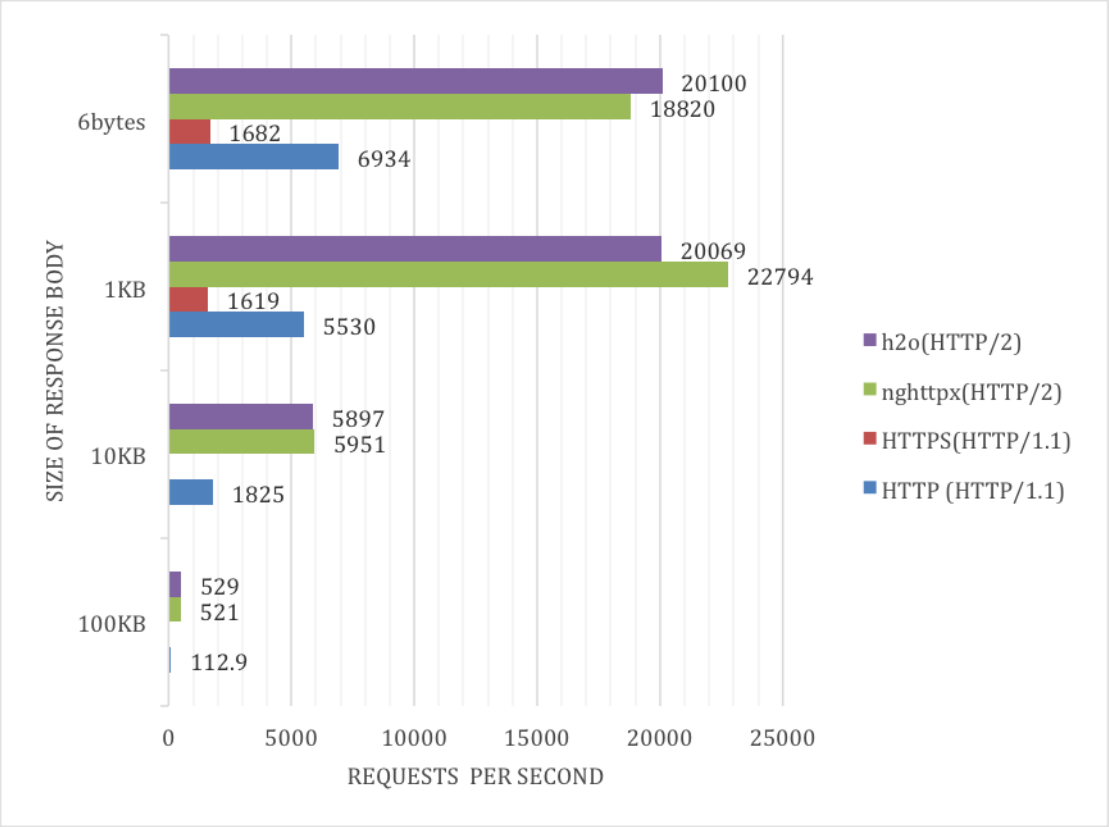


Figure 7.2: Remote load benchmark scores of H2O, nghttpx over HTTP/2 and HTTPS and HTTP over HTTP/1.1

From the results listed in the figure 7.2, it reveals that HTTP/2 servers have an obvious advantages in load performance, even considering the growth in body sizes. Furthermore, H2O and nghttpx have similar performances in all scale of tests. The reason that gap between H2O and nghttpx narrows in tests of 10KB and 100KB is the limit of maximum bandwidth by reaching approximately 50MB per second.

On contrary, the HTTP over HTTP/1.1 have weak performances comparing to HTTP/2 candidates. And the tests with larger size objects of HTTPS over HTTP/1.1 failed finish. One possible explanation for this phenomenon can be the limit of benchmark tool's performance of Apache benchmark tool.

7.1.3 Summary of load benchmark tests

To summarize the load performance benchmarks on local environment and remote nodes, enabling HTTP/2 will not lead to performance loss. In most cases of the tests, especially with small size objects, HTTP/2 server implementations have a relative higher performance than via HTTP/1.1 clear text channels. Moreover. HTTPS over HTTP/1.1 has worst performance in all tests.

Considering different benchmark tools are used for HTTP/2 and HTTP/1.1, it may cause the possible difference in benchmark scores. However, it would not deny the conclusions above, as there is considerable evidence in performance gaps to support it.

7.2 Page browsing performance over HTTP/2

The users of BBC Sport browse web pages with browsers. Page browsing is a set of continuous iterations performed in web browsers including fetching web pages, parsing and executing contents, discovering and requesting dependent resources over HTTP. HTTP serves as the network protocol between browsers and servers and defines the behaviors and standards of their interaction. The ultimate goal of HTTP is fetching the depending resources properly to assist browsers to display fully loaded web pages and perform all the visual effects.

To simulate users browsing behaviors and capture the details of web browsing, the page browsing tests are performed over the actual web browser via WebPageTest tool. WebPageTest helps record and compare the details of load event timing and the resources loading waterfalls.

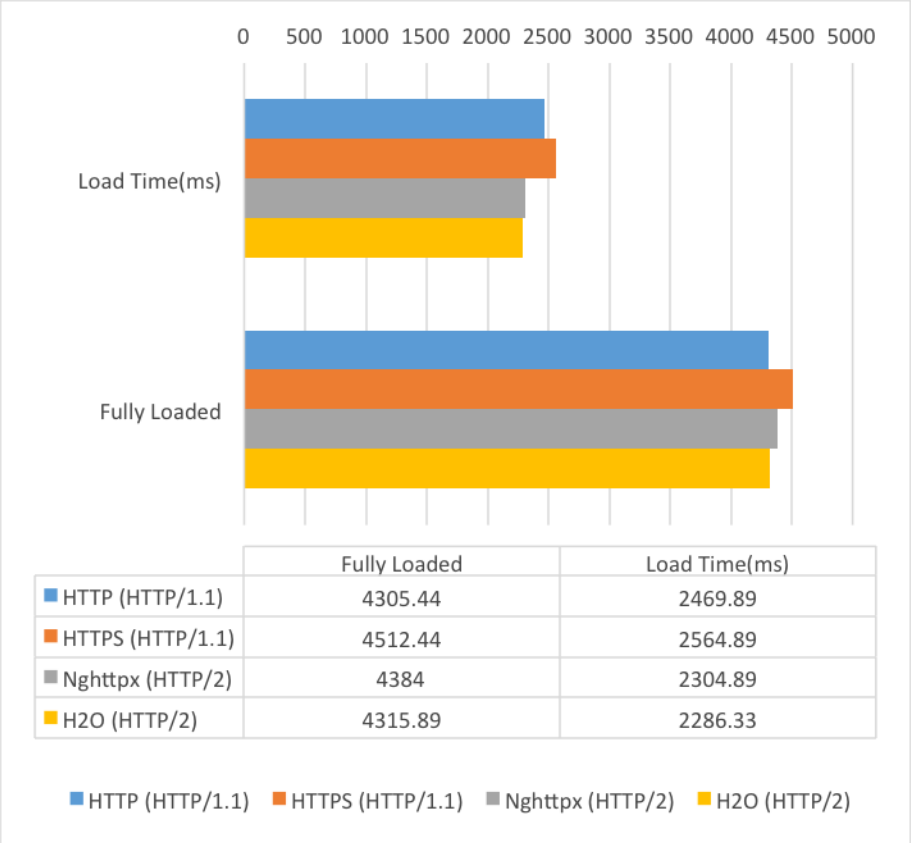
Among all the web pages of BBC Sport website, the homepage of BBC Sport (http://m.bbc.co.uk/sport) is chosen as the target test page for the page browsing tests. The representativeness of the home page come from two reasons: First is that home page is the most frequently visited and updated page of all BBC Sport web pages. Every visit to BBC Sport website or Sport mobile applications triggers one or more browses to the homepage. Second, the home page has a similar page structure with other pages on BBC Sport website. They are commonly consisted of dynamic content generated by Sport server, the One-sport framework for BBC Sport common components and ORB framework for BBC webpage templates.

Four server candidates are tested in this section. Two of them are running by Apache over HTTP/1.1 for clear text HTTP and HTTPS over TLS. The other two for HTTP/2 over TLS as well, respectively by Nghttpx server and H2O server.

The tests over each server contain 9 test runs which are repeated as the first view without content caching nor reusing the connections to the servers. The connection shaping is cable type which has 5MB/1MB (download/upload) bandwidth with 28 round trip time.

7.2.1 Timings of load events

Usually, Load time and fully loaded time are two of important measurements for web page browsing performance. They both describe how quickly the page is loaded onto the browsers. Load time is recorded when page and all of its contents are ready for interaction with users. And the fully loaded time describes when the last response ends.



	Fully Loaded	Load Time(ms)
HTTP (HTTP/1.1)	4305.44	2469.89
HTTPS (HTTP/1.1)	4512.44	2564.89
Nghttpx (HTTP/2)	4384	2304.89
H2O (HTTP/2)	4315.89	2286.33

Figure 7.3: Average load times and fully loaded times of HTTP/2 and HTTP/1.1

The comparison of the mean load times and mean fully loaded times between HTTP/2 and HTTP/1.1 for the home page is shown in Figure 7.3. HTTP/2 cost less time than HTTP/1.1 in load time and fully loaded time on average. Comparing to

HTTP (4305.44ms), H2O costs 7.4% less time and Nghttpx costs 6.6 percentages less. Although both HTTPS and HTTP/2 are transferred over TLS secure transfer layer, HTTPS over HTTP/1.1 performs worse than any HTTP/2 candidate for load time and fully loaded time. HTTPS over HTTP/1.1 consume 10.8% more time in load time and 4.3% more in fully loaded time, comparing to H2O with HTTP/2. It is also worth to mention HTTP over HTTP/1.1 takes the least time in fully loaded time of all. Fully load time is not as significant as it is for other web pages, since BBC Sport pages is using some analysis and assistant tools from third parties. The time cost by loading scripts for these services is summed up into fully load time and make it less convincing.

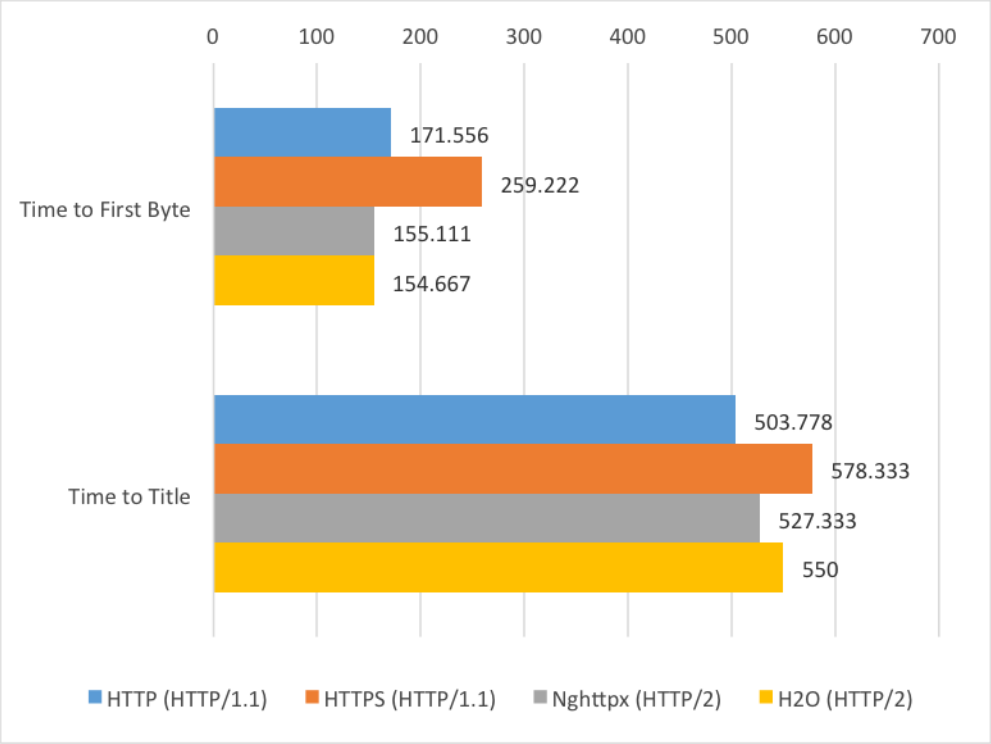


Figure 7.4: Average timings for time to first byte and time to title of HTTP/2 and HTTP/1.1

Two timings for browsing are time to first byte and time to title. They describe how quickly when the connection ready for application data transferring and when the title of HTML are parsed with partial content of HTML file is available.

Figure 7.4 is for the time to first byte and time to title of HTTP/2 and HTTP/1.1. On average according to the test results, the HTTP/2 servers cost over 9.5% less than HTTP and 40% less than HTTPS. However, in the time to title, HTTP over HTTP/1.1 has better than HTTP/2 and HTTPS with the lead up to 8.4% in advance compared to

H2O over HTTP/2.

7.2.2 Visual performances

To address the limitations of load times and satisfying the end users' need, the visual performances are considered and introduced into investigating HTTP/2 performances.

Although timings of load events are important indicators for web performance, they are not reflecting all the aspects of page loading for the end users. Loading times are not that accurate and studies in visual performances can improve the evaluation:

1. Delayed progressive loading strategy: BBC Sport websites generally delays the dependent resources from critical scripts to images without hard coded into HTML file, but load them at runtime progressively instead, which means important requests usually initial after the page load event triggered. Visual performance depends on changes in visual effects rather than loading waterfall to avoid this problem.

2. Not satisfying users' needs: users usually care more about what they see on the page screen rather than details of network loading waterfall. During the page is loading, visual changes help evaluating how the page contents delivered to the users screen.

Speed Index is a measurement for evaluating visual changes. It is the average time at which visible parts of the page are displayed. It is expressed in milliseconds and dependent on size of the view port. For Speed Index, smaller is better, which means the quicker the page contents are visually delivered and displayed to users' screen.

The figure below shows the mean scores of Speed Index for the HTTP/2 and HTTP/1.1 test server candidates.

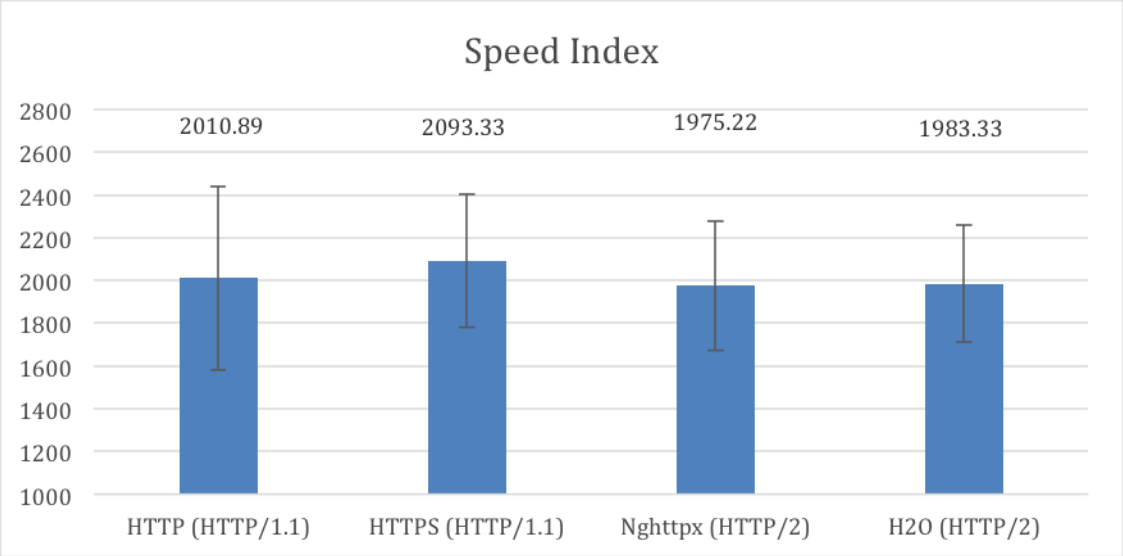


Figure 7.5: Average Speed Index of HTTP/2 and HTTP/1.1

Among all the test samples in Speed Index tests, HTTPS has the worst average performance, while HTTP/2 candidates record similar scores which are relatively lower than HTTP and HTTPS over HTTP/1.1 on average. It is not surprising to find HTTPS over HTTP/1.1 falling behind HTTP, considering HTTPS spends time creating secure transfer connection that delays visual delivery at a degree. However, the tests for HTTP/2 also running on the TLS connection. For the best HTTP/2 candidate Nghttpx, it shows a 1.77 percents lower than HTTP and 5.64 percents lower than HTTPS. This result shows the initial evidence for that HTTP/2 is improving visual performances by multiplexing for better utilization and one connection policy saving time by avoiding time loss in multiple connections initialization.

To have a detailed investigation in visual performances of HTTP/2 comparing to HTTP/1.1, particular test cases are need for the comparison. From all nine test runs of each server candidates, the ones with median page load time are selected. All test runs initialed same amount of requests including 2 requests for HTML files, 4 for CSS files, 18 for scripts, 54 for images and 4 for fonts.

Figure 7.6: Screenshot filmstrips comparison of HTTP/2 and HTTP/1.1

The screenshot filmstrips above in Figure 7.6 are captured with 1 second intervals from blank screen until the page is fully loaded. From the screenshot filmstrips, it is obvious that both HTTP/2 candidates have better visual performance than HTTP/1.1. The first visual changes with Nghttpx and H2O happened during 1.0s to 2.0s and finished last visual change in 2.0s to 3.0s. In comparison, HTTP and HTTPS over HTTP/1.1 delay rendering page in 2.0s to 3.0s and last visual change for HTTPS occurs even later during 5.0s to 6.0s.

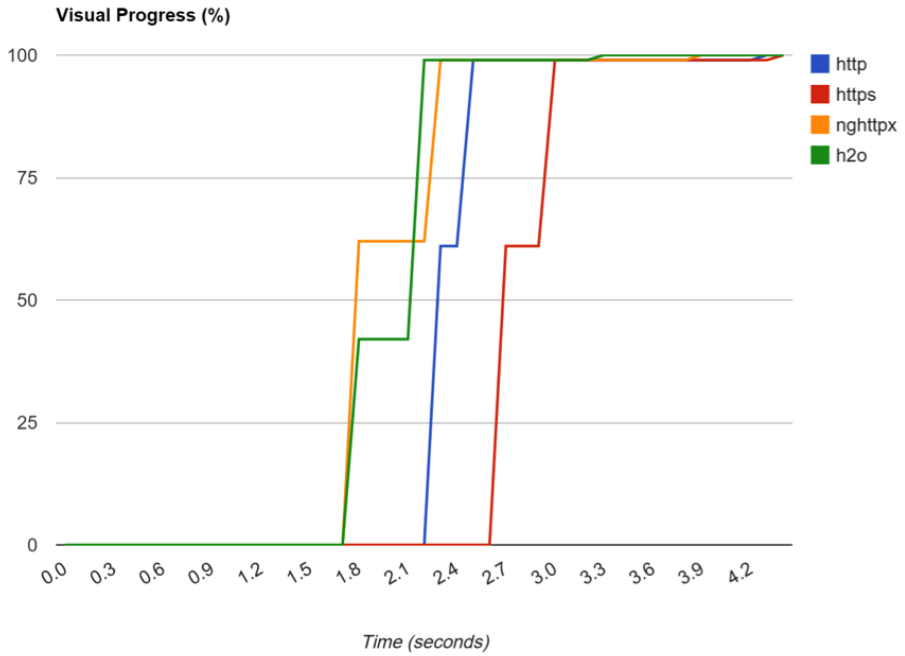


Figure 7.7: Visual progress comparison for HTTP/2 and HTTP/1.1

To understand the details of visual performance, the visual progress captured during page loading shows how the page is render to user screen progressively and what the progress is at the certain time. From Figure 7.7, it reveals that each candidate shows similar visual progress pattern: First is from zero to first rendering time spent on retrieving and related CSS files and the screen painting starts once they are ready. Second is the small scale of platform time for parsing and running the key scripts and requesting font files until they are ready to trigger next phrase of painting. Third progress is continuously loading related images until the page is fully loaded.

Taking the result of visual progress comparison considered, it is clear that HTTP/2 has advantages in visual delivery as it shift the whole rendering progress ahead of HTTP and HTTPS. In these test runs, both Nghttpx and H2O started to render the page at the time of 1.7s and reach 99% visual progress at 2.3s when HTTP just started its displaying progress. And HTTPS started even later at 2.7s the moment when all the other candidates achieved 99% visual progress.

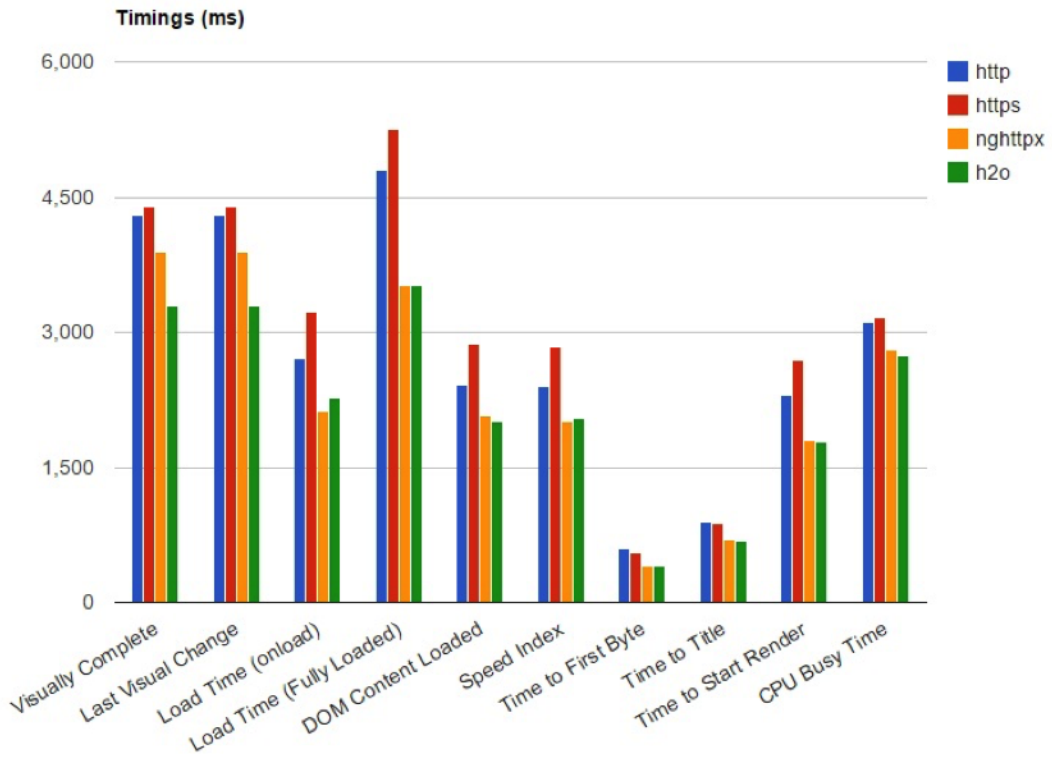


Figure 7.8: Timings for visual performances of HTTP/2 and HTTP/1.1

To conclude the visual performances with the timings showing above in Figure 7.8, HTTP/2 test runs of Nghttpx and H2O take the lead all listed target indicators. Take Nghttpx as the representative for HTTP/2. It is 27% quicker to start rendering than

HTTP and 33% than HTTPS. And it took 3900 ms to complete the visual rendering which is 9.30% less than HTTPS (4300ms) and 11.36% than HTTPS (4400ms).

7.2.3 Performance under different network connections

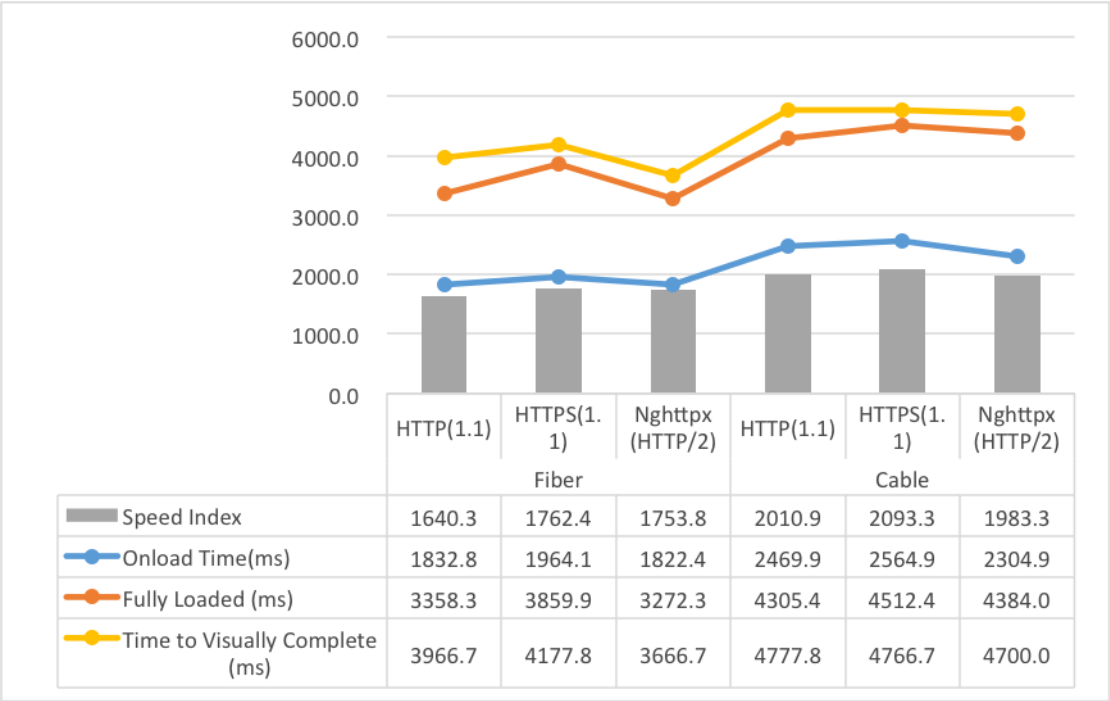
BBC Sport has users across the world with different devices and network connections. To investigate HTTP/2's performance under various network conditions, we simulate 4 representative types of network connections, which includes Fiber, Cable, 3G and 2G Edge.

Fiber and Cable connection types demonstrate the high-end and common network used by desktop users or in-home WIFI sport users, while 3G and 2G Edge connections are for the mobile users accessing the website via cellar data network. The details of traffic shaping are listed as below in the chart.

	Download Bandwidth	Upload Bandwidth	Round Trip Time (ms)
Fiber	20 Mbps	5 Mbps	4 ms
Cable	5 Mbps	1 Mbps	28 ms
3G	1.6 Mbps	768 Kbps	300 ms
2G Edge	240 Kbps	200 Kbps	840 ms

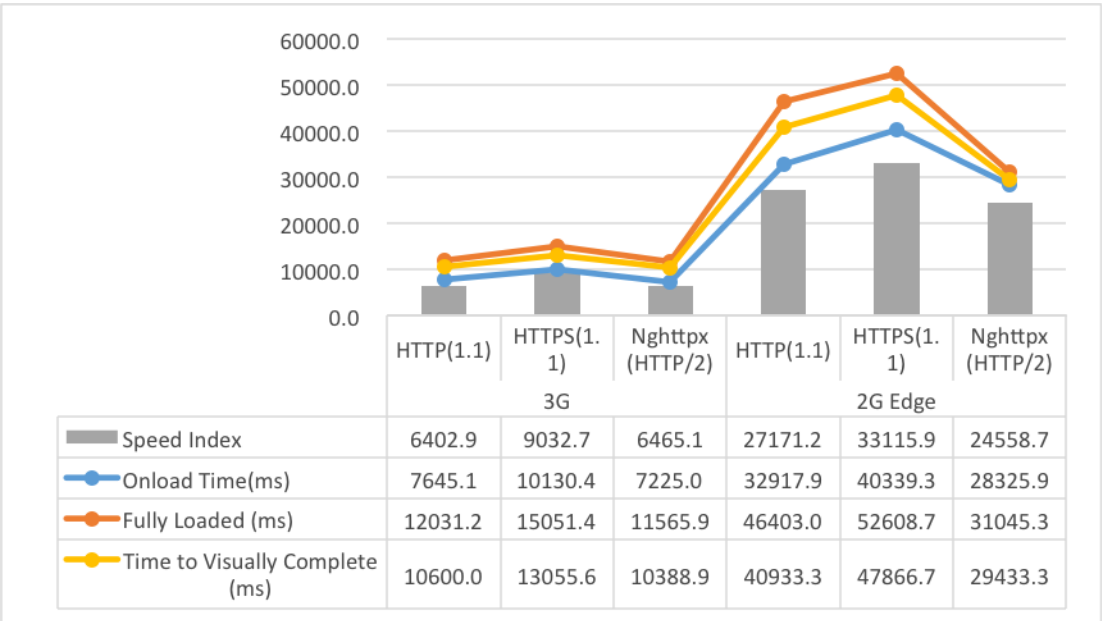
Table 7.1: Traffic shaping details of testing connections

To compare the performances of page browsing over HTTP/2 and HTTP/1.1, four measurement indicators within two types are chosen for the tests, on-load time and fully loaded time for page loading performance while Speed Index and the time to visually complete for measuring user visual experiences. 9 test runs are conducted on each network connection condition and the average value will be counted into analytics.



	HTTP(1.1)	HTTPS(1.1)	Nghttpx (HTTP/2)	HTTP(1.1)	HTTPS(1.1)	Nghttpx (HTTP/2)
	Fiber			Cable		
Speed Index	1640.3	1762.4	1753.8	2010.9	2093.3	1983.3
Onload Time(ms)	1832.8	1964.1	1822.4	2469.9	2564.9	2304.9
Fully Loaded (ms)	3358.3	3859.9	3272.3	4305.4	4512.4	4384.0
Time to Visually Complete (ms)	3966.7	4177.8	3666.7	4777.8	4766.7	4700.0

Figure 7.9: Performance comparison of HTTP/2 and HTTP/1.1 over fiber and cable connections



	HTTP(1.1)	HTTPS(1.1)	Nghttpx (HTTP/2)	HTTP(1.1)	HTTPS(1.1)	Nghttpx (HTTP/2)
	3G			2G Edge		
Speed Index	6402.9	9032.7	6465.1	27171.2	33115.9	24558.7
Onload Time(ms)	7645.1	10130.4	7225.0	32917.9	40339.3	28325.9
Fully Loaded (ms)	12031.2	15051.4	11565.9	46403.0	52608.7	31045.3
Time to Visually Complete (ms)	10600.0	13055.6	10388.9	40933.3	47866.7	29433.3

Figure 7.10: Performance comparison of HTTP/2 and HTTP/1.1 over 3G and 2G edge connections

The performance statistics of test results over landline and mobile networks are respectively shown in Figure 7.9 and Figure 7.10. The bar charts in both figures represents Speed Index scores and the broken lines are for the timing events in terms of milliseconds.

In general, HTTP/2 takes a lead in page load timings and visual performances for both landline and mobile data network connections. HTTP/2 spent less on-load time than HTTP over all network connections. For the fully loaded time, except a minor gap over the cable connection, HTTP/2 improves the performance in other network conditions. The gap is only 1.8% behind HTTP and can be ignored.

For the Speed Index, HTTP/2 only takes a lead in cable and 2G Edge connections comparing to HTTP, which means in general HTTP/2 is not improving the visual display performance that much especially in the situations with higher speed and lower round trip time. However, HTTP/2 effects the time to visually complete with a considerable improvements which helps promoting user experiences.

On the other hand, HTTP/2 has a huge advantages in all four indicators than HTTPS over HTTP 1.1 which means HTTP/2 is always improving page performances and visual experiences for the users compared to HTTPS.

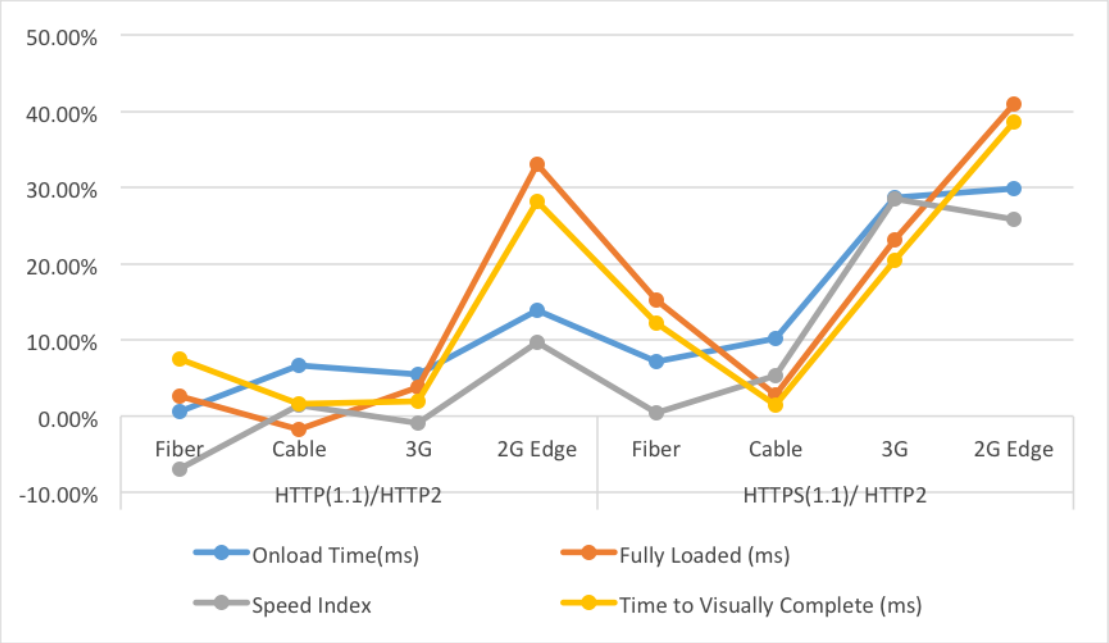


Figure 7.11: Performance improvement of HTTP/2 compared to HTTP (HTTP/1.1) and HTTPS (HTTP/1.1)

To conclude and have an intuitive understanding the performance improvements that HTTP/2 brings, the percentages of each indicators under different connections are shown in Figure 7.11.

For mobile users with 3G and 2G Edge connection comparatively with the lower bandwidth and higher round trip times, HTTP/2 brings more performances improvements. HTTP/2 reduced 33.1% fully loaded time than HTTP and 41.0% than HTTPS with 2G Edge connection, while it reduced only 2.6% and 15.2%, comparing to HTTP. It also shows similar trend in visual performances. HTTP/2 increased the Speed Index by 1.4% than HTTP with cable connection and 9.6% with 2G Edge.

7.3 Network traffic and utilization

This part of the evaluation focus on the network utilization of HTTP/2. With the results collected from test runs in real world Internet environment, it provides an overview of the impact and changes in network traffic.

Header compression and multiplexing are two important features that HTTP/2 promises to improve network utilization. In HTTP/2, a new header compression standard is used for minimizing header data transferred on the network which is clear text without compression in HTTP/1.1. And for connections, HTTP/2 introduces multiplexing to avoid head of line problem and save time loss for connection initialization.

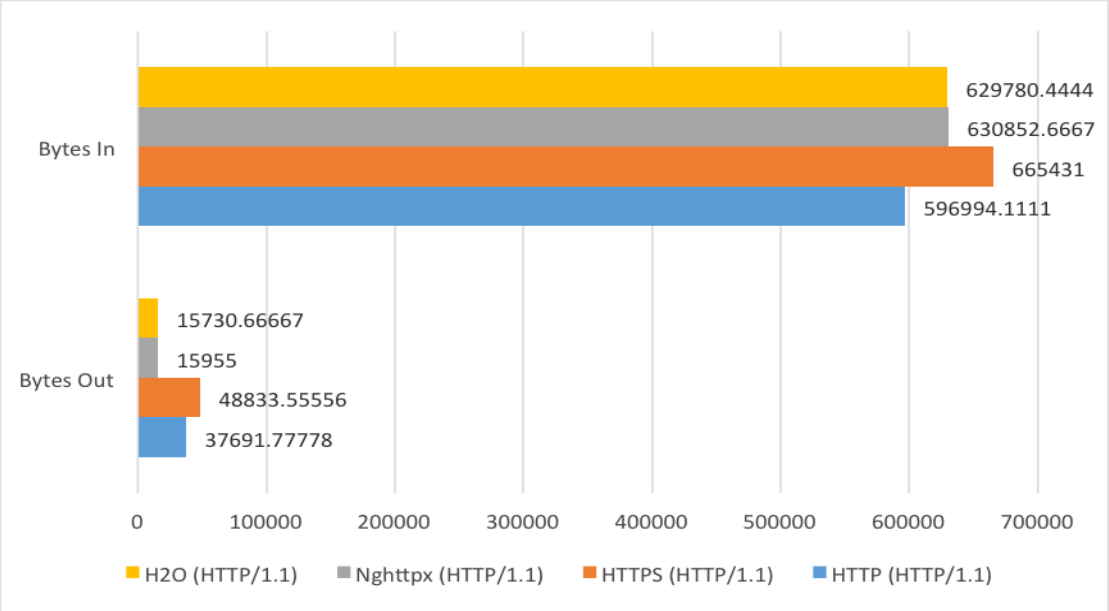


Figure 7.12: Bytes In and Bytes out of HTTP/2 and HTTP/1.1

With the same test sets with 9 test runs for each candidates, the figure 7.12 shows average sizes of the incoming and outgoing data transferred over the network. It reveals that HTTP/2 with header compression reduces the traffic data comparing to HTTP/1.1. As for the bytes sent out to the server from browsers, where the requests are mainly consist of headers, Nghttpx of HTTP/2 sent approximately 58.26% less data than HTTP and 67.78% than HTTPS over HTTP/1.1 without header compression, which means HTTP/2 saves about 267 bytes in every requests on average.

As the incoming data, HTTP/2 does not have obvious advantages in data size. On the contrary, HTTP over HTTP/1.1 receive the least data to load the complete page among all the candidates. Considering both HTTPS and HTTP/2 candidates transferring data over secure transfer layer, the additional bytes transferred can be taken as the relearnt cost for it. Even though, both HTTP/2 candidates with header compression still have the advantage in incoming traffic reduction comparing to HTTPS.

In the tests, HTTP/2 opened 6 connections to 6 different domains following the rule one connection per domain. On the other hand, HTTP and HTTPS over HTTP/1.1 used 11 connections to the same numbers of domains.

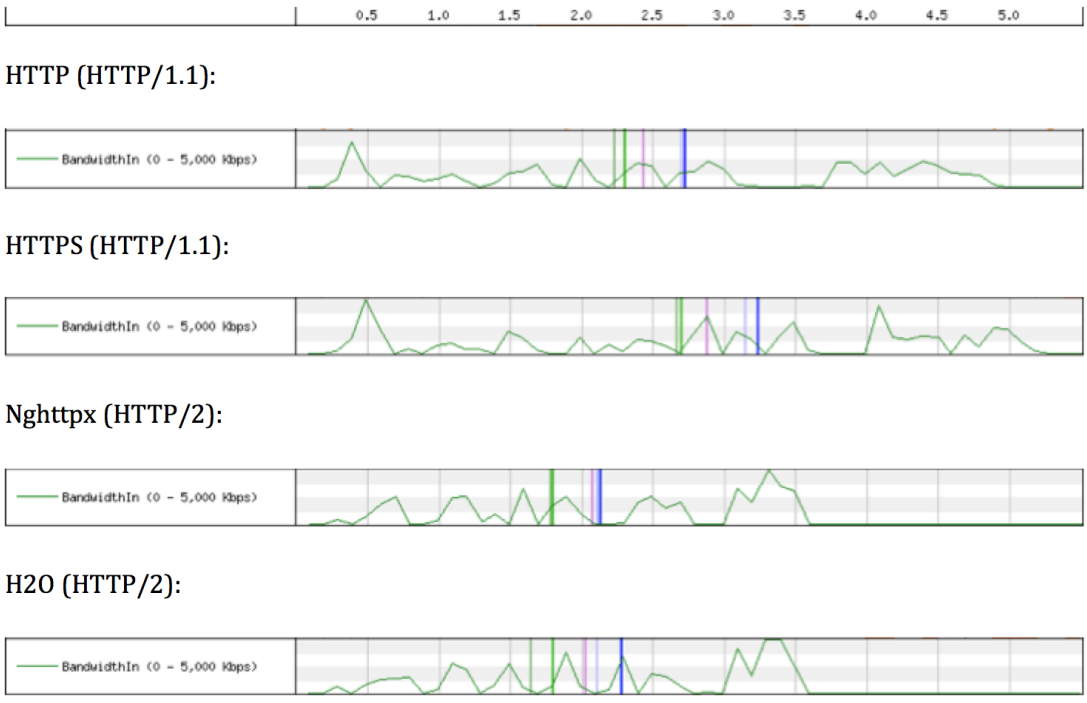


Figure 7.13: Bandwidth utilization of HTTP/2 and HTTP/1.1 loading demo BBC Sport homepage

The graphic chart in Figure 7.13 demonstrates the differences in bandwidth utilization of HTTP/2 and HTTP/1.1. In the early stage from 0 to 1.0, HTTP and HTTP/1.1 have a higher traffic peak earlier than HTTP/2 candidates. However, in reality HTTP and HTTPS only finished loading the HTML file while HTTP/2 already loaded several dependent CSS and scripts.

Another significant difference is noticed the different ways loading body content images. On BBC Sport homepage, the news images on the page are requested after the page on loaded. From 3.5s to 5.0s, the requests of images over HTTP/1.1 have to wait in queue for the available connection. And the bandwidth usage is lower than half of maximum. On the other hand, HTTP/2 has better bandwidth utilization by multiplexing the image requests and responses on the only connection to the test server. Therefore the requests are dispatched and known by the server immediately (at around 3.0s) without waiting and finished data retrieving in half a second. During this progress, the average bandwidth is comparatively higher than HTTP/1.1 and the peak reached the maximum connection speed.

7.4 Applying server push

Server push is a new powerful feature that HTTP/2 brings to HTTP, which allows a server to proactively send things to the client's cache for future use. It requires a new mechanism with proper push strategies to apply server push for BBC Sport websites.

For the first time ever, the resources can be pushed from server side over HTTP before the client requests them. In other way, server push can be used for reduce the runtime delay on the client side.

The figure 7.14 below, which is an example of network waterfall loading the BBC Sport home page over HTTP/2 with the test system, help identifying the possible runtime delays. It is clear that the whole loading process can be divided into three phrases including different groups of resources requests.

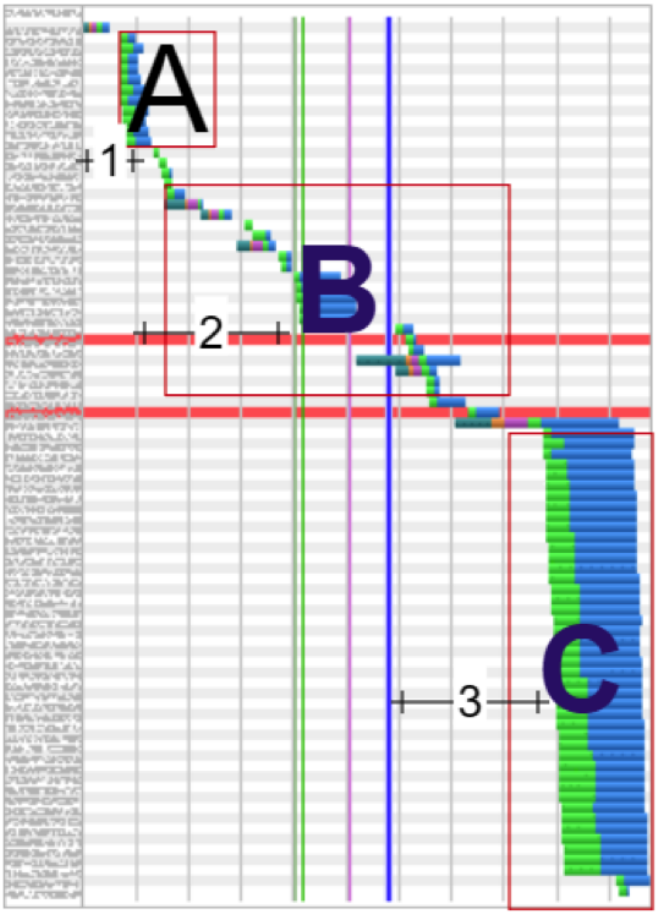


Figure 7.14: An example network waterfall loading demo BBC Sport homepage over HTTP/2 without server push

Group A is the directly dependent resources on HTML file including several base JavaScript and CSS files. Group B consists of CSS and JavaScript files required by the page structure and framework dependencies in executing process during runtime. Finally it comes with the Group C which is mainly consisted by the requests of the news images.

Accordingly, the runtime delays from stage 1 to 3 are identified (as also shown in the figure) and they together contributes to the total delay of page loading.

Server push in HTTP/2 can be applied to help resolving these runtime delays:

1. For the stage 1 delay, the information of dependencies comes from the HTML

tags and transformed into server push rules. Then the Group A's resources can be pushed with HTML response to save the time for processing and parsing HTML file.

2. For the delay in stage 2 and stage 3, the dependencies of Group B and Group C can be manually set up on the server. By manually setting the push rules, the network transfer of the dependent requests are sent to client in advance. And once they are discovered by the client, they can be fetched from the cached server push responses.

The Sport-proxy in this project satisfied the need for these requirements. For the server push rules, it support auto-parse mode and the manual mode. The previous one automates the process from parsing from HTML file to save to push rules. The latter one enables update server push rules manually at runtime.

To evaluate and exhaust the performance of HTTP/2 with server push, the tests satisfy the listed preconditions:

1. The target test runs are loading demo BBC Sport homepage over HTTP/2 either with or without server push, since server push is an optional optimization on the HTTP/2.

2. To exhaust the effect of server push, the tests will perform 'fully push' mode which is pushing all the resources required by the HTML page or at the runtime.

3. Block all requests not from the same domains, only reserving the ones from the test server in order to eliminate unrelated noise requests for statistics analysis and evaluate the performance with only one connection. By doing these, the test blocks the listed domains: c.go-mpulse.net, tpc.googlesyndication.com, partner.googleadservices.com, www.googletagservices.com, static.chartbeat.com and ping.chartbeat.net.

In every test run, 74 resources will be pushed together with the response of HTML page in the same order of the playback captured by the browser.

Nghttpx is used as the HTTP/2 server in the tests, which serves as the reverse proxy with default server push feature pushing the resources only with status code 200 of the response, according to the hints in the link header from the upstream sever, the Sport-proxy in this project.

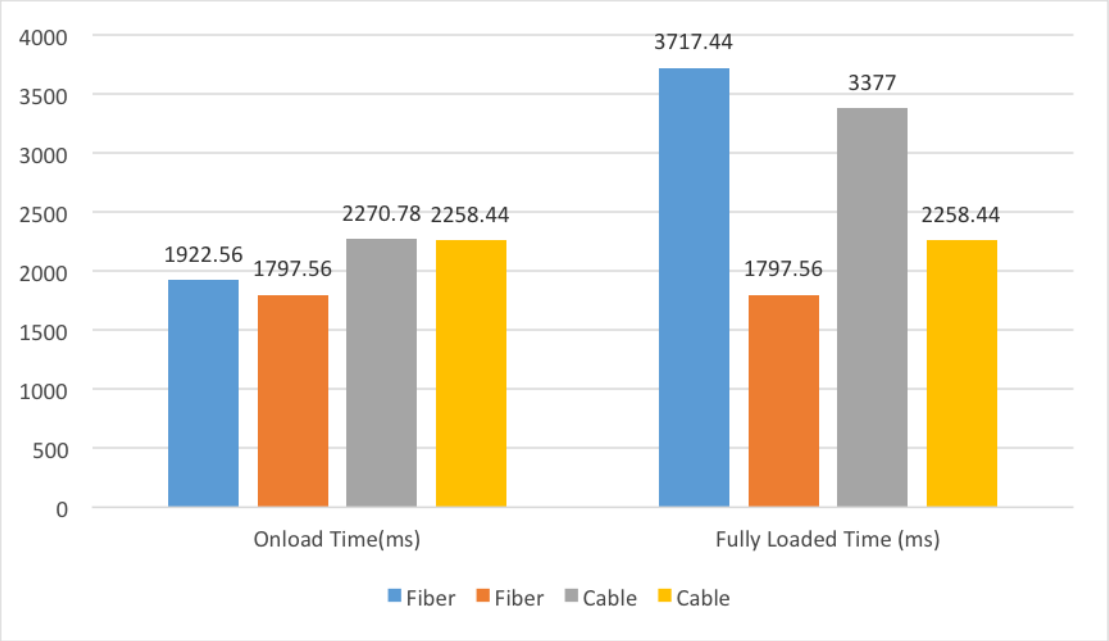


Figure 7.15: Timing comparison of on load time and fully loaded time loading BBC Sport home page over HTTP/2 with and without server push

From the timings comparison shown in Figure 7.15, several obvious phenomenons are noticed:

First, the difference in on load time between HTTP/2 with and without server push is not significant as much as the situations in fully loaded time. The explanation for this is that certain relative time is required in both conditions for the browser to parse the HTML page while more requests spread during loading process which delays fully loaded time.

Second, while HTTP/2 without server push takes 1.48 to 2 times on load time to reach fully loaded status, the one with server push finishes fully loaded immediately as soon as the on load event triggers.

Third, comparing the page loading performances under fiber and cable network connections, HTTP/2 with server push significantly benefits from the increase in bandwidth.

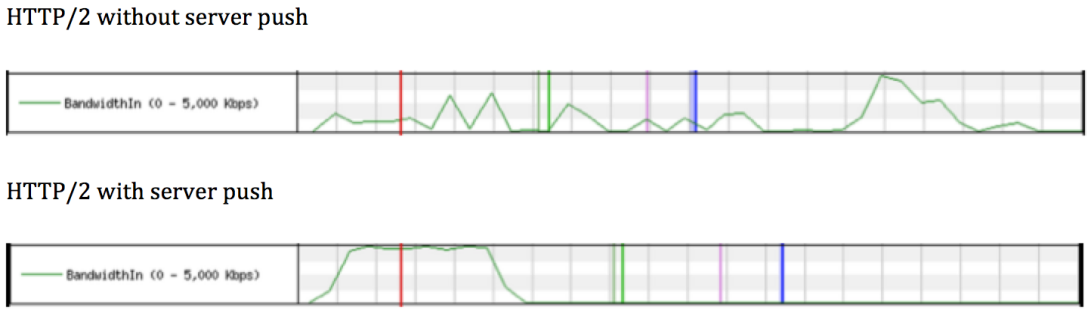


Figure 7.16: An example of bandwidth utilization comparison between loading BBC Sport home page with cable connection over HTTP/2 with and without server push

The bandwidth utilization graphs help explaining the phenomenons. With proactively pushed resources right after the response of HTML file, the bandwidth over HTTP/2 with server push is consumed and the usage reaches and maintains the maximum bandwidth until all the pushed resources are loaded. When the page is parsed, all the required resources are ready for use to display the web page without sending more requests. By contrast, HTTP/2 without server push, the browser has to wait for the requests sent during the whole loading process which delays the fully loaded time.

It also explains why fiber connection improve the page loading performance with the bandwidth increase when applying server push, comparing to cable connection. Multiplexing over the connection in HTTP/2 encourages the better bandwidth utilization, as long as the enough requests and responses are initialed. As server push putting all the responses initialed as many as it can, depending on the server push rules, it prioritize the HTTP networking with higher significances to make the most of HTTP/2's benefits.

In conclusion, by applying server push, it benefits the page loading performances especially in fully loaded time which is important for user page browsing with more quicker news images loaded and displayed. And it also reveals that the higher maximum bandwidth is, the greater improvement impact in performance and bandwidth utilization that server push brings into page browsing.

Chapter 8

Conclusions and Future

The following chapter summaries the key aspects of the dissertation. The conclusions of this project are summarized in Section 8.1. The next section 8.2 l an outline of possible future projects and describes research questions and problems that arose during the project.

8.1 Conclusions

For the first time in recent 16 years, HTTP welcomes its major upgrade and HTTP/2 has been standardized with RFC 7540 by the IETF HTTP Working Group. BBC Sport is interested in the chances and challenges that HTTP/2 would bring to the websites, especially the effect to user browsing experiences. This project contributes to investigating HTTP/2 for BBC Sport in two major parts. One part is for the feasibility of enabling HTTP/2, investigating the support from client side browsers and the server side implementations. In the second part, a test system for HTTP/2 is developed and deployed and a series of related tests are conducted for comparing the performances of HTTP/2 and HTTP/1.1. As user experience is the priority in this investigation, the comparison of visual performances is considered and introduced as well.

For the client side support for HTTP/2, the browser survey shows that most main stream browsers like Chrome and Firefox have added HTTP/2 support and enabled it by default in the year of 2014. According to the analytical statistics inside BBC Sport, the website visitors with default enabled HTTP/2 support climbs up from 0% in August 2014 to over 57.8% in March 2015. Furthermore, with growing sales of the mobile devices with latest version web browsers, and considering the newly released dominant desktop operation system Windows 10 equipped with Edge browser, it is

expected that the rate of default HTTP/2 support will continue to increase rapidly in 2015 and become a dominant position among BBC website visitors.

The investigation in server side implementations shows that currently the web frameworks for most popular programing language, including PHP language which used widely in BBC Sport, is the lack of HTTP/2 support. Also for the web servers, Apache, Nginx and other traditional popular servers does not support HTTP/2 at the time but they are welcoming HTTP/2 in next year or later. In the near future, the feasible way to apply HTTP/2 is to add a HTTP/2 reverse server along with existing HTTP/1.1 servers. Among all the HTTP/2 server implementation, Nghttp2 project is the most mature stack for HTTP/2 from server implementation, reverser server, headless client and programming interfaces, while H2O is another mature HTTP/2 server candidate with relatively higher benchmark performance.

A test server system is designed and deployed into public Internet for the following performance tests. It supports HTTP/2 with Nghttp2 and H2O separately and it sets up Apache server for HTTP/1.1 services as references. A software called Sport-proxy written in JavaScript running on NodeJS platform is developed to control server push hits for the reverse servers. With Sport-proxy, the server push rules can be either manual set or auto parsed from HTML, which indicates a new possible way to apply server push.

In the load benchmark tests, HTTP/2 servers have a better performance than HTTP and HTTPS over HTTP/1.1. Considering different benchmark tools used for separate HTTP versions, it is still reasonable that enabling HTTP/2 is not trading off the load performances.

Generally, HTTP/2 reduces page load time and visual experience indicators by approximately 10%. The results of page browsing tests reveal that HTTP/2 has a relatively better performance with BBC Sport homepage, not only in the page loading but also in the visual displaying progress. HTTP/2 has a better improvement in slower mobile data networks than in high speed landline networks. In all the tests, HTTPS has the worst performance among all the candidates. Comparing to HTTP over HTTP/1.1, HTTP/2 takes a greater lead than in HTTPS from page loading to visual experience.

For network traffic, with newly introduced feature in HTTP/2 of header compression, Nghttpx of HTTP/2 sends out approximately 58.26% less data than HTTP and 67.78% than HTTPS over HTTP/1.1 in the test, when loading the demo BBC Sport homepage from the test servers. For incoming data, HTTP/2 cost 5.49% most traffic transferred than HTTP over HTTP/1.1 but still 5.35% less data than HTTPS again

thanks to header compression. Multiplexing over connection in HTTP/2 also benefit the network utilization by avoiding opening additional connections to same domain and multiplexing the requests and responses in streams and frames.

Server push is the last feature discussed to minimize the runtime delays which is a main bottleneck constrains in the performance of loading current BBC Sport page. With the extreme situation pushing all the required resources with the HTML responses, server push reduced 33.12% page fully loaded time over cable connection comparing to running over HTTP/2 without server push. And furthermore, server push takes more advantage in higher bandwidth, with 51.64% less page fully loaded time over fibre connection, as server push proactively pushes promised responses to consume the bandwidth.

8.2 Limitations and Future

The tests and the results has limitations from several aspects:

1. Different load benchmark tools used for the load tests. The reason for this is that no current benchmark tool supports both HTTP/1.1 and HTTP/2. The difference in inner mechanism of benchmark tools may cause different benchmark result.

2. The Sport-proxy software, which serves as the cached proxy running on the JavaScript platform which potentially limit the performance. Implement the Sport-proxy in other language like C or C++ may help to get close to server performance in real world, as most fast popular server are written in C/C++.

3. The real page loading performance is difficult to measure with single measurement matrix. Any network and computation factors impact in measuring the timings of page loading and visual displaying. On the other hand, allowable error could also affect the performance results that varies in every test runs.

For the future works, several aspects are worth to have a further investigation or development:

- The problem of TCP low start may have a negative impact on the performances of HTTP/2, considering web browser over HTTP/2 opens only one connection

per remote domain. To test the performances under network situations with different number in packet loss rate, it could provide a more clear overview of the problem effects HTTP/2.

- The HTTP/2 testing solution for BBC Sport website can be also applied for other websites, proxying the targeted remote website with auto url link replacing and server push features. What is more, the domain name *http2test.org*, which is bought to facilitate the tests, could be reused again for the further general purpose comparing HTTP/2 and HTTP/1.1.

- It is desirable to develop a new benchmark tool or add HTTP/2 support to current benchmark tool in order to ensure same benchmark mechanism and proper similar usages when comparing the benchmark performances of HTTP/2 and HTTP/1.1. Recently the public available benchmark tool is out of date for HTTP/2 and it increases the difficulties and unreliability to the comparison results.

- Partial performances of HTTP/2 and the real world engineering problems rely on the industrial support or implementation. For the server implementations, if Apache or Nginx releases the official support for HTTP/2, the performances tests could be repeated for further advanced references. For the third parties services, for instance Akamai CDN, the only way to have proactive feedback of improvements or risks is to get on the trials for HTTP/2 and test the performances with real BBC Sport web files.

Bibliography

[Aka14a] Akamai. Http/2 is the future of the web, and it is already here! 2014. https://http2.akamai.com/.

[Aka14b] Akamai. The state of the internet of 4rd quarter 2014, 2014. https://www.akamai.com/us/en/our-thinking/state-of-the-internet-report/.

[Arc15] HTTP Archive. Http archive trends of tracked websites from apr 1 2012 to apr 1 2015, 2015. http://httparchive.org/trends.php?s=All&minlabel=Apr+1+2012&maxlabel=Apr+1+2015.

[Bel10] Mike Belshe. More bandwidth doesn't matter (much), 2010. https://www.belshe.com/2010/05/24/more-bandwidth-doesnt-matter-much/.

[BL91] Tim Berners-Lee. The original http as defined in 1991, 1991. http://www.w3.org/Protocols/HTTP/AsImplemented.html.

[Böh13] Felix Böhm. htmlparser-benchmark by andreasmadsen, 2013. https://travis-ci.org/AndreasMadsen/htmlparser-benchmark/builds/10805007.

[Boy99] Mark R. Boyns. Http - hypertext transport http/0.9, 1999. http://muffin.doit.org/thesis/html/node13.html.

[CBB15] Multiplexing Manager Chris Bentzel and Bence Béky. Hello http/2, goodbye spdy, 2015. http://blog.chromium.org/2015/02/hello-http2-goodbye-spdy-http-is_9.html.

[Com15] Ltd. ComScore. About digital analytix, 2015. http://www.comscore.com/Products/Enterprise-Analytics/Digital-Analytix-Enterprise.

[Dev15] Alexis Deveria. Browser usage table in march 2015, 2015. http://caniuse.com/usage_table.php.

[Fou15] Node.js Foundation. About node.js, 2015. https://nodejs.org/about/.

[Gar15] Owen Garrett. How nginx plans to support http/2, 2015. http://nginx.com/blog/how-nginx-plans-to-support-http2/.

[Glo15] StatCounter GlobalStats. Global usage share statistics, 2015. http://gs.statcounter.com/#browser-ww-monthly-201501-201503.

[Goo09] Google. Spdy: An experimental protocol for a faster web, 2009. http://dev.chromium.org/spdy/spdy-whitepaper.

[IET15a] IETF. Http/2 frequently asked questions. 2015. https://http2.github.io/faq/.

[IET15b] IETF. Known implementations of http/2. 2015. https://github.com/http2/http2-spec/wiki/Implementations.

[Oku15a] Kazuho Oku. H2o - the optimized http/1.x, http/2 server. 2015. http://h2o.github.io/.

[Oku15b] Kazuho Oku. H2o, the new http server goes version 1.0.0 as http/2 gets finalized, 2015. http://blog.kazuhooku.com/2015/02/h2o-new-http-server-goes-version-100-as.html.

[Par14] Lucas Pardue. Adaptive media streaming over http/2 trial, 2014. http://www.bbc.co.uk/rd/blog/2014/12/adaptive-media-streaming-over-http-2-trial.

[Par15] Lucas Pardue. Initial results from the adaptive media streaming over http/2 trial, 2015. http://www.bbc.co.uk/rd/blog/2015/01/initial-results-from-the-adaptive-media-streaming-over-http-2-trial.

[Saf11] Ido Safruti. From fast to spdy, 2011. http://www.slideshare.net/ido-cotendo/from-fast-to-spdy-velocity-2011.

[TBL96] R. Fielding T. Berners-Lee, MIT/LCS. Request for comments: 1945: Hypertext transfer protocol – http/1.0, 1996. http://tools.ietf.org/html/rfc1945.

[TBL05] R. Fielding T. Berners-Lee, W3C/MIT. Rfc3986 uniform resource identifier (uri): Generic syntax. 2005. http://tools.ietf.org/html/rfc3986#section-4.2.

[Tsu15] Tatsuhiro Tsujikawa. nghttp2 - http/2 c library. 2015. https://nghttp2.org/documentation/package_README.html.

[W3T15a] W3Techs. Usage of server-side programming languages for websites, 2015. http://w3techs.com/technologies/overview/programming_language/all.

[W3T15b] W3Techs. Usage of web severs for websites by 27 april 2015, 2015. http://w3techs.com/technologies/overview/web_server/all.

[Web12] WebPageTest. Speed index, 2012. https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/metrics/speed-index.

[Web15] WebPageTest. About webpagetest.org. 2015. http://www.webpagetest.org/about.

Appendix A

Review Comments from BBC

From Matthew Clark, Senior Technical Architect, BBC Online:
27th August, 2015

On behalf of the BBC, I'd like to thank Bowen and the University of Manchester for the opportunity to research HTTP/2 with the BBC Sport website. The findings of this research will definitely help us to deploy the technology sooner. This, in turn, will allow us to offer a faster, more accessible website to our audience both in the UK and worldwide.

We proposed HTTP/2 as a research project for two reasons. Firstly, we needed to know more about the technology and what it meant for the BBC website. The costs and benefits of the technology need to be understood to justify and plan its use across our large site. In the first part of this thesis, Bowen has helped the BBC by explaining both the advantages of HTTP/2 and its current support. It's useful to see, for example, that over half of users of the BBC Sport site (and growing) can support HTTP/2 (section 3.2).

Our second reason for an interest in HTTP/2 is performance. There is a well-known correlation between page load times and user retention. In other words, the faster the BBC's site is, the more it will be used and so the more its users will value the BBC. So we have a strong interest in any technology that can reduce page load times.

The significant performance testing described in this thesis is enlightening. We see that HTTP/2 can offer a performance improvement in the majority of cases, and in some cases that improvement can be significant. For example, our site is often accessed over a slow (2G) network, and it is encouraging to see in section 7.2 that page load times in these circumstances could be reduce by about a third. Similarly, an

increasing proportion of users have very fast connections, and the potential halving of load times for this, shown by the use of server push in section 7.4, is a very exciting prospect.

Together, the understanding and performance evidence explained in this thesis makes a strong case for why BBC Online should adopt HTTP/2 soon. The BBC will use this research internally to create a business case and plan for supporting HTTP/2 across all of its websites. I'd be hopeful that this could be achieved and in use within the next twelve months.

Thanks again Bowen for your hard work and friendly approach to collaboration.