Native Web Communication Protocols and Their Effects on the Performance of Web Services and Systems

Nitin Naik[1], Paul Jenkins[1], Philip Davies[2] and David Newell[2]

[1]Defence School of Communications and Information Systems, Ministry of Defence, United Kingdom
[2]Department of Computing and Informatics, Bournemouth University, United Kingdom
Email: {nitin.naik100, paul.jenkins683}@mod.uk, daviesp@bournemouth.ac.uk, dnewell@bournemouth.ac.uk

***Abstract*—Native Web communication protocols are the pivotal components of Web services, applications and systems. In particular, HTTP is a de facto protocol standard used in almost all Web services and systems. Consequently, it is one of the crucial protocols responsible for the performance of Web services and systems. HTTP/1.1 has been successfully deployed in Web services and systems for the last two decades. However, one of the most significant issues with HTTP/1.1 is the Round Trip Time and Web latency. To resolve this issue, two successor protocols SPDY and HTTP/2 have been developed recently, with some studies suggesting that SPDY improved the performance of Web services and systems, whilst some did not find significant improvements in the performance. HTTP/2 is a relatively new protocol and has yet to be tested with any rigour. Therefore, it is important to investigate the effects of these two enhanced protocols SPDY and HTTP/2 on the performance of Web services and systems. This paper conducts a number of practical investigations to evaluate the performance of Web services and systems with and without the support of SPDY and HTTP/2 protocols at the client and server. This study investigates the impact of SPDY and HTTP/2 on the overall performance of Web services and systems from the end-user's perspective.**

Keywords*—*Web Protocols, Web Services, Web Applications, Web Systems, Web Latency, Round Trip Time, HTTP, SPDY, HTTP/2, SOAP, REST

I. Introduction

The Web is the lifeline of modern society and Web services, applications and systems are its integral components. A Web service is predominantly a method of communication between applications/machines over the Web (commonly via HTTP) [1]. It is a collection of open standards and protocols used to exchange data between applications/machines [2]. A Web application is a piece of software that is designed to help the user in achieving a function or task. A Web application could use multiple web services to achieve its goal or result. At a lower level, both Web applications and Web services are similar in function. However, at a higher level, Web applications are intended for users and Web services are intended for applications/machines. A Web-based system or Web system is one that employs Web services and Web applications to carry out its operations, such as Web Portals, Websites, Messengers and Skype. In Web services, applications and systems, perceived latency is the most crucial performance criterion for any Web users, which is the amount of time, a user sees between making a page request and it being rendered. One of the most important factors, which affect the perceived latency, is the Web communication protocol that decides how the client and server communicate over the wire. In particular, HTTP is a de facto protocol standard used in almost all Web services, applications and systems. Therefore, it is one of the critical protocols responsible for the performance of Web services, applications and systems.

One of HTTP's biggest challenges as a Web communication protocol is reducing the Round Trip Time (RTT) and Web latency to improve the performance of systems [3]. Therefore, this paper conducts several practical investigations of native Web communication protocols HTTP/1.1, SPDY and HTTP/2; and their impact on the performance of Web services and systems. Six different experiments are performed to examine the effects of SPDY and HTTP/2 in reducing the RTT and Web latency and improving the performance of Web services and systems. It should be noted that these experiments have analysed the overall performance of Web services and system, rather than an individual protocol. The experimental results demonstrate the wide variation and delay in the load time for websites utilising SPDY and HTTP/2. It indicates that the support of SPDY and HTTP/2 at the client side does not make any significant improvement in the overall performance of any websites. Therefore, this study suggests that the overall performance of Web services and systems is heavily dependent on other parameters such as contents of the website (number of DOM elements/requests and their size), location of the server, transmission media, data transfer rate, number of intermediate nodes, traffic density, priority of event/request, and processing delay. Finally, this experimental analysis shows the effects of Web protocols SPDY and HTTP/2 on the overall performance of Web services and systems is insignificant as compared to the factors stated above.

The remainder of this paper is organised as follows: Section II explains the theoretical background of Web services, applications and systems, Round Trip Time (RTT) and Web latency; Section III describes the evolution of native Web communication protocols HTTP/1.1, SPDY and HTTP/2 and their characteristics; Section IV illustrates the six practical investigations to evaluate the performance of Web services and systems with and without the support of SPDY and HTTP/2 protocols at the client and server; Section V presents the performance analysis of SPDY and HTTP/2 for Web services and systems; Finally, Section VI concludes the paper and

suggests some future areas of extension.

II. Theoretical Background

This section presents the theoretical background of Web services, applications and systems, Round Trip Time (RTT) and Web Latency.

A. Web Services, Web Applications and Web Systems

Web services, Web applications and Web systems are interrelated terminologies. A Web service is mainly a method of communication between applications/machines over the Web (commonly via HTTP) [1]. It is a collection of open standards and protocols used to exchange data between applications/machines [2]. Web services are designed to allow applications developed using different technologies to communicate with each other without any interoperability issues [4]. A Web application is a piece of functional software that is designed to help the user in achieving a task. A Web application could use multiple Web services to achieve its goal or result. At a lower level, both Web applications and Web services are similar. However, at a higher level, Web applications are intended for users and Web services are intended for applications/machines. Web applications generally present data in HTML which is easily readable by the user and Web services generally present data in XML/JSON, which is easy to parse by other applications. Web services are the application/machine-readable equivalent to the Web applications. A Web-based system or Web system is one that employs Web services and Web applications to perform its operations, such as Web Portals, Websites, Messengers and Skype.

The two most popular Web services are Simple Object Access Protocol (SOAP) and REpresentational State Transfer (REST). These Web services use XML and JSON data formats as most client and server frameworks are designed around using these formats. Similarly, the native Web communication protocol for SOAP and REST is HTTP, and both are currently tied to HTTP; whilst, other protocols could be used. HTTP is a request-response protocol based on the client-server model (i.e. a Web client and a Web server in a Web System as shown in Fig. 1). A Web client is a service consumer that sends service requests to a Web server which provides service responses to the Web client.

B. Round Trip Time (RTT) and Web Latency

The primary focus of this investigation is the native Web communication protocol HTTP (i.e. HTTP/1.1) and its successor SPDY and HTTP/2. HTTP is an application-layer protocol providing basic request-response semantics for transporting content over the Web. Fig. 2 shows the process of communication between the HTTP client and HTTP server using TCP. TCP is a reliable transport layer protocol supporting HTTP for all underlying services on the Web such as guaranteed delivery, duplicate suppression, in-order delivery, flow control and congestion avoidance [5]. In the Web environment, RTT is the time between a request from Web client (or Web browser) and its complete response from the Web server. RTT depends on a number of factors such as physical distance between the client and server, transmission media, data transfer rate, number of intermediate nodes and traffic density. Fig. 3 shows

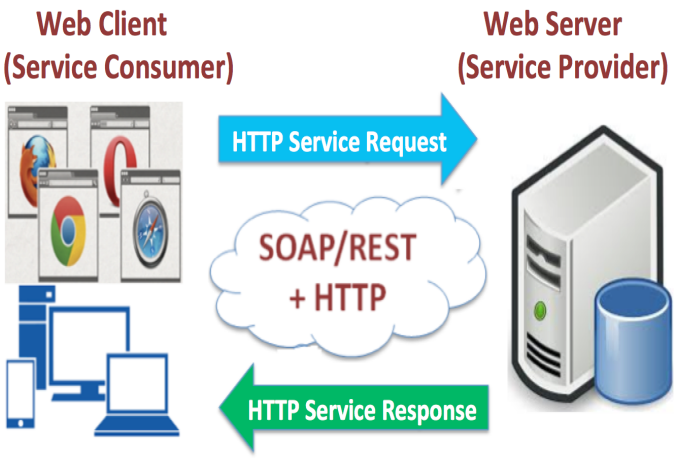

Fig. 1. Employment of HTTP in Web Services, Applications and Systems

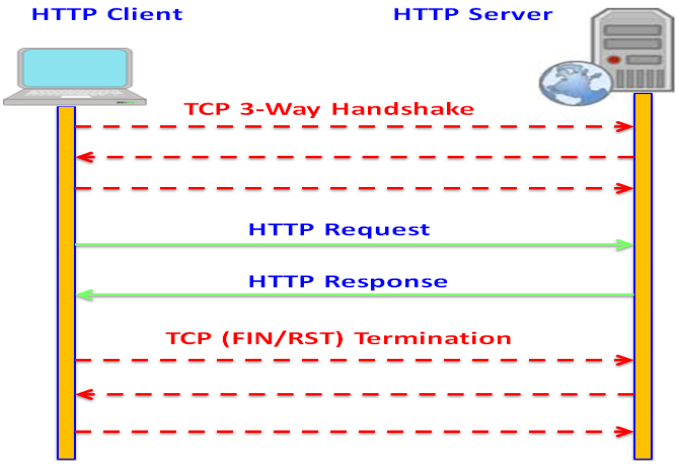

Fig. 2. Communication between HTTP Client and HTTP Server using TCP Connection

the RTT concept in HTTP-based communication, where it demonstrates three RTTs: one RTT for TCP connection setup, one for TCP connection termination, and minimum one RTT for the actual HTTP request and response. However, it does not include the other RTTs such as RTT for DNS name resolution.

Web latency is the time it takes for the Web server to receive and process a request (e.g., for a page object) from the Web client (or Web browser). It also depends on a number of factors, such as priority of event/request, processing delay, Web communication protocols and Web applications. An average page is composed of many small resources, which requires a number of RTTs. Web performance and a page load time can be improved in two ways: reducing the RTT and total number of round trips, even for higher bandwidth. Both HTTP and TCP protocols are responsible for the RTT and Web latency; unfortunately, neither protocols were particularly designed with latency in mind [6]. One of the most significant issues with HTTP is that it incurs many more round trips than necessary to retrieve the Web objects [3]. Furthermore, TCP is not helpful in reducing the RTT and Web latency because of its retransmission of every lost packet with the three-way handshake needed to open a connection (and also closing it), for every HTTP round trip. Additionally, TCP does not fully

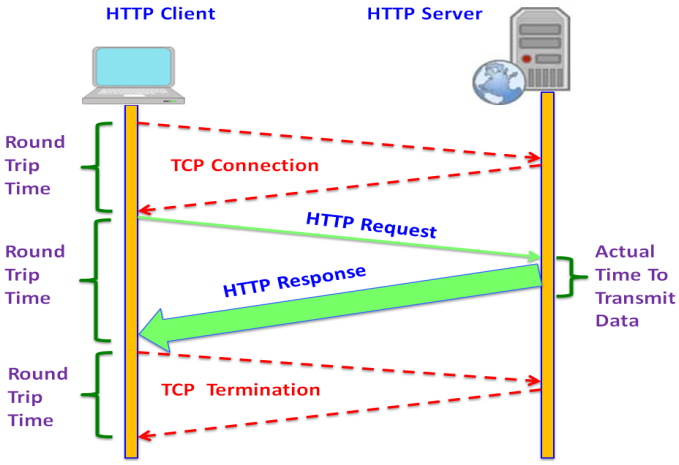

Fig. 3. HTTP Round Trip Time

utilize the available network bandwidth for the first few round trips of a connection because of its "slow start technique", which is used to avoid network congestion. Consequently, the current and successful combination of Web protocols HTTP and TCP is a major bottleneck of the RTT and Web latency. This problem may be one the largest impediments leading to further improvements in HTTP (i.e. HTTP/1.1).

III. Evolution of Native Web Communication Protocols and Their Characteristics

As mentioned earlier, native Web communication protocols are the crucial component of Web services, applications and systems, which affects their performance. Therefore, this sub-section will discuss the native Web communication protocol HTTP/1.1 and the emergence of two new alternative protocols SPDY and HTTP/2 to improve the performance of Web services, applications and systems. The development stages of these Web protocols are shown in Fig. 4.

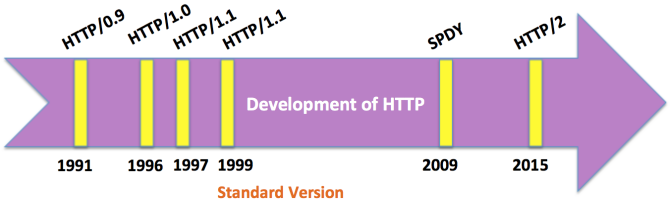

Fig. 4. Development of native Web communication protocols

A. HTTP or HTTP/1.1 Protocol

HTTP/1.1 was first published by the IETF in 1997 and later standardised in 1999. Since its inception in 1999, it has worked successfully for around two decades. The HTTP 1.1 Working Group has improved the performance and reduced the RTT and Web latency of HTTP/1.1 with the introduction of persistent connections, request pipelining, and chunked transfer encoding [7]. However, these features of HTTP/1.1 such as request pipelining have effectively failed due to the lack of support and deployment challenges; while some browsers today support pipelining as an optional feature, which forces strict request queuing on the client [7]. Figs. 5 and 6 show these two

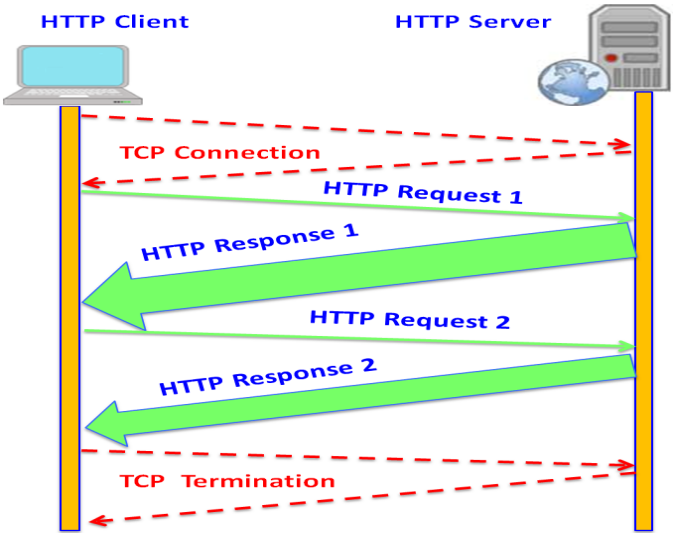

Fig. 5. HTTP/1.1 persistent connection for improving its performance

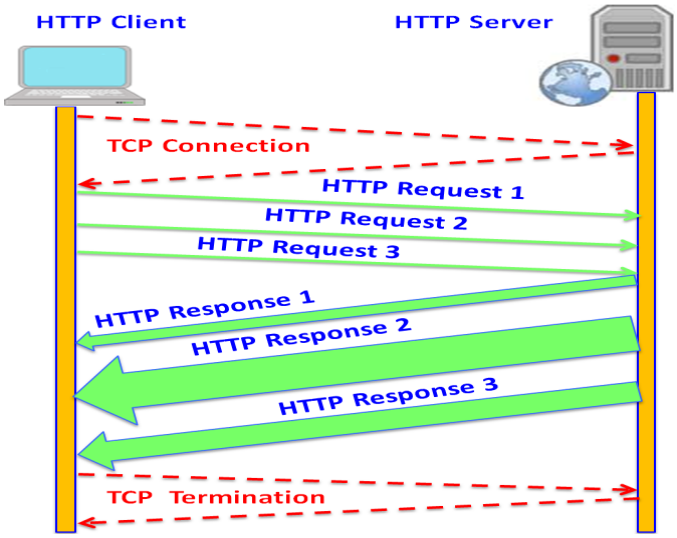

Fig. 6. HTTP/1.1 pipelined connection for improving its performance

provisions of persistent connections and pipelined connections to improve the performance of HTTP/1.1.

B. SPDY Protocol

SPDY is an application-layer protocol for transporting content over the Web. It is an experimental protocol developed by Google in 2009. The main aim of designing SPDY was to minimise Web latency by up to 50% [5]. It does not replace HTTP protocol instead it augments it by adding a number of features that increase the speed of Web transactions. Akin to HTTP, SPDY also uses TCP as the underlying transport layer, therefore it does not demand any change in the existing networking infrastructure. The practical requirement of SPDY is the use of TLS/SSL, but it is not compulsory. Therefore, the end-to-end encrypted TLS/SSL tunnel allows the client and the server to exchange SPDY frames without intervention of intermediate nodes. Consequently, how does the client and server know to use SPDY once the TLS/SSL tunnel is opened? For this, a new protocol Next Protocol Negotiation (NPN) is

used. NPN is a TLS/SSL extension, which allows the client and the server to negotiate the application protocol as part of the TLS/SSL handshake. Moreover, it eliminates the additional round trip to negotiate the application protocol. This is one of its main advantages as compared to the current WebSocket handshake, which imposes another round trip of latency in addition to the SSL negotiation. The development of SPDY has continued up to the version SPDY3.x with SPDY4 not being released as a separate specification rather becoming an alias for the new HTTP/2 standard. In HTTP/2, NPN has now been deprecated. SPDY's design requirement of TLS/SSL has limited its actual adoption.

C. HTTP/2 Protocol

HTTP/2 was recently introduced in 2015, nearly two decades after its predecessor HTTP/1.1. Developed by the IETF HTTP Working Group, mainly based on Google's experimental SPDY protocol. It enables a more efficient use of network and server resources, and a reduced perception of latency by introducing header field compression and allowing multiple concurrent exchanges on the single connection from a browser to a website [8], [9]. HTTP/2 also introduces unsolicited push of representations from servers to clients [10]. HTTP/2 replaces HTTP/1.1 on the wire only but maintains the HTTP/1.1 message syntax. Therefore, all HTTP methods, status codes and semantics are the same, and it is possible to use the same APIs as HTTP/1.1 with some alterations to represent the new version [11], [12]. Similar to SPDY NPN protocol, HTTP/2 employs the TLS/SSL extension protocol, called Application-Layer Protocol Negotiation (ALPN) within the TLS/SSL handshake. In this case multiple application protocols are supported on the same TCP or UDP port. It also allows the application layer to negotiate which application protocol will be used within the TLS/SSL connection.

HTTP/2 communication is based on the two new concepts *frames* and *streams* as shown in Fig. 7. Frames are the basic unit of communication in HTTP/2, which replace the well-known header and body format of HTTP/1.1 requests-responses. Each fragment of communication between the client and server is packed up into a binary frame before it is sent over the connection [13]. HTTP/2 supports several types of frames such as header frames and data frames. Streams can be considered as one logical request-response communication containing several frames. It is a bi-directional sequence of frames with a common frame identifier. Streams enable the multiplexing of frames from multiple streams together, which allows for true multiplexed communication over a single connection as shown in Fig. 7.

IV. Practical Investigations of the Performance of Native Web Communication Protocols HTTP/1.1, SPDY and HTTP/2

RTT and Web latency can be improved at the application and transport layers. However, this investigation is only focused on the application layer protocols HTTP/1.1, SPDY and HTTP/2. Various research studies have been conducted on the performance of SPDY (as HTTP/2 has been introduced recently), some of them have claimed that SPDY has reduced the Web latency [14], [15], however, some of them have even presented the adverse effect [16]. These past studies are crucial

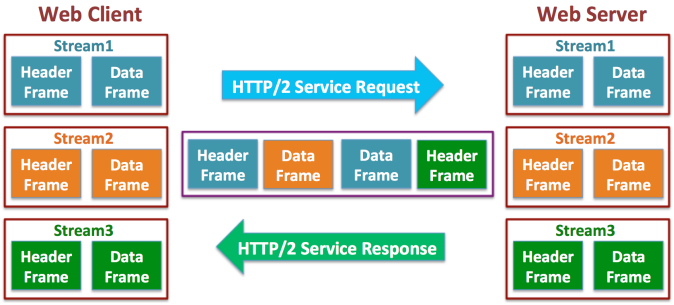

Fig. 7. Streams are multiplexed by splitting communication into frames in HTTP/2

as SPDY forms the basis of the new HTTP/2 protocol. This practical investigation incorporates both SPDY and HTTP/2 protocols including the existing HTTP/1.1. Here, six different experiments are conducted to investigate the performance of SPDY and HTTP/2 over on HTTP/1.1. The investigation includes all the possible situations: when client and server both support SPDY and HTTP/2 protocols, when only one supports and when both do not support these protocols. The two client browsers chosen are Google Chrome (a version that supports SPDY and HTTP/2) and Mozilla Firefox (a version that does not support SPDY and HTTP/2). Similarly, to access the server resources, two types of websites are chosen one that supports SPDY and HTTP/2 and the other which does not support SPDY and HTTP/2. It should be noted that this was the status of these browsers and websites at the time of experiment; however, the whole process is in perpetual change and the current status may be completely different. The other investigation tools used in these experiments are HAR (HTTP Archive) Analyser, Wireshark and several other websites to identify the current status of browsers and websites. This investigation has focused on the three parameters: average load time, number of requests and amount of bytes in for the first document view and repeat document view to gain insight from the user's point of view only. HTTP performance logs including these parameters are recorded in the HAR files. HAR (HTTP Archive) is a file format used by several HTTP session tools to track all the interactions of the Web browser with a website.

A. Experiment-1: HTTP/1.1-Enabled Website using HTTP/1.1-Enabled Browser

In the first experiment, six random websites are chosen that do not support either SPDY or HTTP/2 protocols at the time of experiment. They are probed through a version of Mozilla Firefox browser that does not support either SPDY or HTTP/2. Therefore, this experiment is based on only the existing HTTP/1.1 protocol and to check its performance and Web latency. To avoid errors and other side effects in the final result, this experiment is repeated 50 times and, finally, the average value of each parameter is calculated as shown in Table I.

B. Experiment-2: HTTP/1.1-Enabled Website using SPDY-HTTP/2-Enabled Browser

In the second experiment, the same six websites are chosen that do not support either SPDY or HTTP/2 protocols at the time of experiment. However, they are probed through the

TABLE I. PROBING HTTP/1.1-ENABLED WEBSITE USING HTTP/1.1-ENABLED BROWSER (MOZILLA FIREFOX VERSION THAT DOES NOT SUPPORT SPDY AND HTTP/2)

HTTP/1.1-Enabled	First View			Repeat View		
Website	Time (S)	Request	Bytes (KB)	Time (S)	Request	Bytes (KB)
ieee.org	7.022	204	2028	5.023	66	367
one.com	5.646	57	1110	3.548	20	198
godaddy.com	2.246	13	361	1.573	2	96
names.co.uk	5.219	69	1468	4.051	31	104
bbc.co.uk	7.795	113	1328	5.44	19	148
skynews.com	5.326	72	1121	4.58	15	41

Google Chrome browser that supports SPDY and HTTP/2 protocols at the time of experiment. Therefore, this experiment is based on the partial support of SPDY and HTTP/2 and to check its performance and Web latency. Similar to the first experiment, for avoiding errors and other side effects in the final result, this experiment is repeated 50 times and, finally, the average value of each parameter is calculated as shown in Table II.

TABLE II. PROBING HTTP/1.1-ENABLED WEBSITE USING SPDY-HTTP/2-ENABLED BROWSER (GOOGLE CHROME VERSION THAT SUPPORTS SPDY AND HTTP/2)

HTTP/1.1-Enabled	First View			Repeat View		
Website	Time (S)	Request	Bytes (KB)	Time (S)	Request	Bytes (KB)
ieee.org	8.319	204	2030	6.598	75	381
one.com	4.126	55	1114	3.245	20	195
godaddy.com	2.821	13	360	1.881	2	96
names.co.uk	4.697	69	1468	3.182	31	105
bbc.co.uk	8.191	111	1319	5.173	17	142
skynews.com	5.873	77	1128	5.241	18	58

The first and second experiments are carried out on the same websites that do not support either SPDY or HTTP/2, however, one browser supports and other does not support SPDY and HTTP/2. One of the interesting findings of the comparative analysis of the results is that the client browser support of SPDY and HTTP/2 does not affect the overall performance and Web latency significantly. This is confirmed by the results where the HTTP/1.1 supported browser has performed better than SPDY and HTTP/2 supported browser. Perhaps, this indicates the effects of other factors on the performance of Web services and systems. However, Web protocols are one of the major factors responsible for reducing the Web latency, the variation and delay in the load time for similar types of websites suggests that the Web latency is heavily dependent on the contents of the website (number of DOM elements/requests and their size) and location of the server.

C. Experiment-3: SPDY-Enabled Website using HTTP/1.1-Enabled Browser

In the third experiment, six random websites are chosen that support SPDY but not HTTP/2 protocol at the time of experiment. However, they are probed through a version of Mozilla Firefox browser that does not support either SPDY or HTTP/2. Therefore, this experiment is based on the partial support of SPDY and HTTP/2 and to check their performance and Web latency. Similar to the previous experiments, for avoiding errors and other side effects in the final result, this experiment is also repeated 50 times and, finally, the average value of each parameter is calculated as shown in Table III.

TABLE III. PROBING SPDY-ENABLED WEBSITE USING HTTP/1.1-ENABLED BROWSER (MOZILLA FIREFOX VERSION THAT DOES NOT SUPPORT SPDY AND HTTP/2)

SPDY-Enabled	First View			Repeat View		
Website	Time (S)	Request	Bytes (KB)	Time (S)	Request	Bytes (KB)
reelseo.com	7.621	139	1725	6.877	31	158
yelp.com	6.272	74	1205	4.577	13	47
shareasale.com	2.703	37	1045	1.702	1	7
people.com.cn	11.484	237	3012	7.307	223	252
addthis.com	5.337	47	580	2.776	33	36
taobao.com	12.739	116	1367	8.046	13	84

D. Experiment-4: SPDY-Enabled Website using SPDY-HTTP/2-Enabled Browser

In the fourth experiment, the same six websites are chosen that support SPDY but not HTTP/2 protocol at the time of experiment. They are probed through Google Chrome browser that supports SPDY and HTTP/2 protocols. Therefore, this experiment is based on the full support for SPDY and for checking its performance and Web latency. Similar to the previous experiments, for avoiding errors and other side effects in the final result, this experiment is also repeated 50 times and, finally, the average value of each parameter is calculated as shown in Table IV.

TABLE IV. PROBING SPDY-ENABLED WEBSITE USING SPDY-HTTP/2-ENABLED BROWSER (GOOGLE CHROME VERSION THAT SUPPORTS SPDY AND HTTP/2)

SPDY-Enabled	First View			Repeat View		
Website	Time (S)	Request	Bytes (KB)	Time (S)	Request	Bytes (KB)
reelseo.com	6.432	140	1732	4.786	16	144
yelp.com	8.322	74	1201	5.798	13	46
shareasale.com	2.835	37	1096	2.071	2	39
people.com.cn	12.567	234	2945	8.132	75	137
addthis.com	3.782	48	581	2.757	29	33
taobao.com	11.219	116	1336	6.55	14	84

The third and fourth experiments are carried out on the same websites that support SPDY but not HTTP/2, however, one browser supports and other does not support SPDY and HTTP/2. The comparative analysis of the results of the two different client browsers shows very minor changes in the overall performance and similar patterns as the first two experiments. Therefore, they do not affect the overall performance and Web latency significantly. This is confirmed by the results where the HTTP/1.1 supported browser has performed better than SPDY and HTTP/2 supported browser. Some of the results are most surprising, where the average load time is much higher than the first two HTTP/1.1 results. Again, this indicates the effects of other parameters on the performance and Web latency such as contents of the website (number of DOM elements/requests and their size) and location of the server. The support of SPDY and HTTP/2 at the webserver (website) may be crucial for the success of SPDY and HTTP/2 and can perhaps reduce the overall latency but requires further in-depth analysis. Overall, these two experimental results reveal that the support of SPDY and HTTP/2 may not be sufficient to improve the overall performance and reduce the overall web latency.

E. Experiment-5: HTTP/2-Enabled Website using HTTP/1.1-Enabled Browser

In the fifth experiment, six random websites are chosen that support HTTP/2 but not SPDY protocol at the time of

experiment. However, they are probed through a version of Mozilla Firefox browser that does not support either SPDY or HTTP/2. Therefore, this experiment is based on the partial support of SPDY and HTTP/2 and to check their performance and Web latency. Similar to the previous experiments, for avoiding errors and other side effects in the final result, this experiment is also repeated 50 times and, finally, the average value of each parameter is calculated as shown in Table V.

TABLE V. PROBING HTTP/2-ENABLED WEBSITE USING HTTP/1.1-ENABLED BROWSER (MOZILLA FIREFOX VERSION THAT DOES NOT SUPPORT SPDY AND HTTP/2)

HTTP/2-Enabled	First View			Repeat View		
Website	Time (S)	Request	Bytes (KB)	Time (S)	Request	Bytes (KB)
wordpress.com	2.202	11	202	1.025	2	24
emarketdesign.com	2.490	30	411	1.799	3	17
wix.com	8.124	81	3679	4.296	24	56
sohu.com	16.4461	397	1860	12.906	294	659
indiewebcamp.com	3.265	63	779	1.694	2	58
detik.com	24.629	257	2868	22.053	94	428

F. Experiment-6: HTTP/2-Enabled Website using SPDY-HTTP/2-Enabled Browser

In the sixth experiment, the same six websites are chosen that support HTTP/2 but not SPDY protocol at the time of experiment. They are probed through Google Chrome browser that also supports SPDY and HTTP/2 protocols. Therefore, this experiment is based on the full support for HTTP/2 and for checking their performance and Web latency. Similar to the previous experiments, for avoiding errors and other side effects in the final result, this experiment is also repeated 50 times and, finally, the average value of each parameter is calculated as shown in Table VI.

TABLE VI. PROBING HTTP/2-ENABLED WEBSITE USING SPDY-HTTP/2-ENABLED BROWSER (GOOGLE CHROME VERSION THAT SUPPORTS SPDY AND HTTP/2)

HTTP/2-Enabled	First View			Repeat View		
Website	Time (S)	Request	Bytes (KB)	Time (S)	Request	Bytes (KB)
wordpress.com	1.911	11	208	0.739	2	53
emarketdesign.com	2.680	30	411	1.636	3	17
wix.com	6.352	80	4153	2.903	24	60
sohu.com	14.117	380	1775	11.891	283	800
indiewebcamp.com	3.599	63	799	1.819	2	58
detik.com	22.668	257	2852	20.460	89	354

The fifth and sixth experiments are carried out on the same websites that support HTTP/2 but not SPDY, however, one browser supports and other does not support SPDY and HTTP/2. Similar to the comparative results of the third and fourth experiments, this comparative analysis of the results of the two different client browsers show very minor changes in the performance and similar patterns as the previous experiments. Therefore, they do not affect the overall performance and Web latency greatly. This is confirmed by the results where the HTTP/1.1 supported browser has performed better than SPDY and HTTP/2 supported browser. Again, some of the results are most surprising, where the average load time is much higher than the previous results. As explained earlier, this indicates the effects of other parameters on the performance and web latency such as contents of the website (number of DOM elements/requests and their size) and location of the server. Similar to the third and fourth experiments, the support of SPDY and HTTP/2 at the webserver (website) may be crucial for the success of SPDY and HTTP/2 and can perhaps reduce the overall latency but requires further in-depth analysis. Again, these two experimental results reveal that the support of SPDY and HTTP/2 may not be sufficient to improve the overall performance and reduce the overall Web latency.

V. PERFORMANCE ANALYSIS OF SPDY AND HTTP/2 FOR WEB SERVICES AND SYSTEMS

In modern Web systems and enterprise Web applications, the page load time and response time are crucial for the productivity and success of an enterprise. All real-time and safety-critical systems require real-time responses on the Web. Even for those Web applications that do not require instantaneous responsivity, they still must respond in close to real-time for providing a better user experience and service [17]. In real-time and safety-critical systems, data must be collected and processed continuously with controlled latency for safety and security. It emphasises the requirement for real-time operations in all real-time and safety-critical Web systems with no place for batch processing models. Based on the experimental results obtained in the previous section, the minimum response time is around 2 seconds even when using the enhanced Web protocols SPDY and HTTP/2. This is a much higher response time for many enterprise Web systems and certainly not suitable for the real-time and safety-critical systems. However, the results also suggest that the effect of the Web protocol HTTP on the performance of Web services and systems is trivial as compare to other factors. For example, if the client is further away from the Web server or an average page is composed of many large size resources, the longer it takes to get a response back from the Web server [18]. However, distance-related network latency can be reduced by pushing data and processing closer to the Web server where possible, but Web services and systems will remain susceptible to poor routing decisions and network congestion [17]. Similarly, compressed and smaller size resources can be used to improve the size-related latency. In essence, HTTP has less impact on the overall performance of Web services and systems from the end-user's perspective.

The main concern of SPDY and HTTP/2 web protocols is the pace of acceptance in the Web community itself. According to the W3techs.com website [19], [20], the recent usage statistics of SPDY and HTTP/2 on the Web are 7.0% and 7.4% in May 2016 as shown in Figs. 8 and 9. Both graphs show the historical trend of the percentage of websites using SPDY and HTTP/2 up to May 2016. Perhaps, one of the biggest obstacles in the adoption of SPDY and HTTP/2 is the practical compulsion of TLS/SSL. However, SPDY is submerged into HTTP/2 and HTTP/2 was only launched recently, therefore, it will take time for these protocols to be adopted by the majority of the Web community. Simultaneously, it is crucial to investigate the accepted latency for real-time and safety-critical Web systems. Finally, based on the experiments conducted here and current recognition of HTTP/2 indicates that HTTP/2 requires additional review in the near future, or could be replaced with extra lightweight protocols such as XMPP, MQTT, CoAP and AMQP in some particular types of Web services and systems.

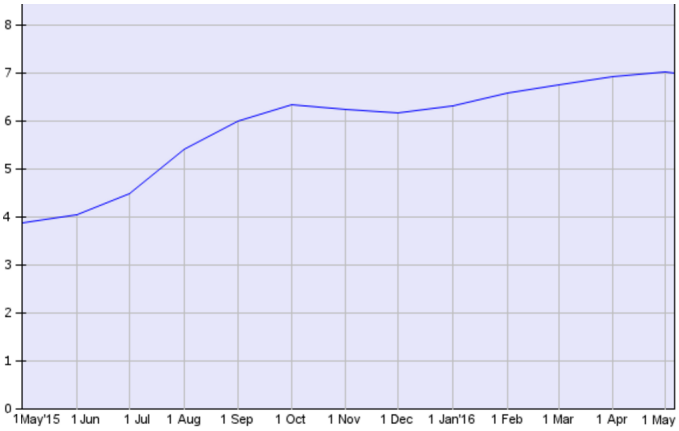

Fig. 8. Usage of SPDY for Websites [19]

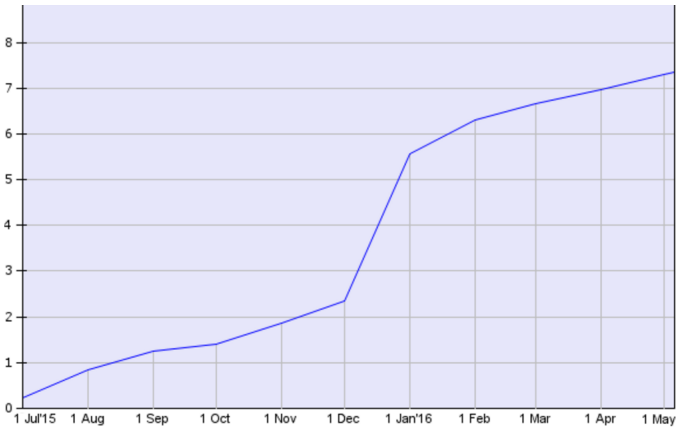

Fig. 9. Usage of HTTP/2 for Websites [20]

VI. Conclusion

This paper has presented a practical investigation to evaluate the performance of Web services and systems with and without the support of Web protocols SPDY and HTTP/2 at the client and server. Web communication protocols are key protocols, which are responsible for the performance of Web services and systems and affect the RTT and Web latency. Without an effective Web communication protocol, improvement in performance of Web services and systems would not be possible. However, RTT and Web latency are one of the main issues in using the Web communication protocol HTTP. Consequently, this paper conducted several experiments to determine whether the successor of HTTP/1.1 (i.e., web communication protocols SPDY and HTTP/2) can improve the performance of Web services and systems as compared to the existing HTTP/1.1. The experimental results have suggested that Web communication protocols SPDY and HTTP/2 have minimal effects on the overall performance of Web services and systems as compared to other crucial factors such as contents of the website (number of DOM elements/requests and their size), location of the server, transmission media, data transfer rate, number of intermediate nodes, traffic density, priority of event/request and processing delay. However, the implementation of these communication protocols at the server side may perhaps improve the performance moderately and reduce the RTT and Web latency, but this requires further in-depth analysis. Finally, the experimental results and current acceptance level of Web communication protocols SPDY and HTTP/2 suggest that in the near future HTTP/2 requires another improvement or perhaps replacement with equivalent lightweight and low latency communication protocols such as XMPP, MQTT, CoAP and AMQP in some particular types of Web services and systems. In the future, it may be interesting to investigate the implementation of HTTP/2 protocol in different kinds of Web services and systems.

References

[1] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. big'web services: making the right architectural decision," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 805–814.

[2] L. Richardson and S. Ruby, *RESTful web services*. O'Reilly Media, Inc., 2008.

[3] Z. Shelby, "Embedded web services," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 52–57, 2010.

[4] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 795–804.

[5] G. Baker and E. Arvidsson, "Let's make the web faster," 2010.

[6] N. Naik and P. Jenkins, "Web protocols and challenges of web latency in the web of things," in *Eight International Conference on Ubiquitous and Future Networks (ICUFN)*, 2016.

[7] I. Grigorik, "Making the web faster with http 2.0," *Communications of the ACM*, vol. 56, no. 12, pp. 42–49, 2013.

[8] R. Peon and H. Ruellan, "HPACK: Header compression for HTTP/2," Tech. Rep., 2015. [Online]. Available: chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/https://www.rfc-editor.org/rfc/pdfrfc/rfc7541.txt.pdf

[9] Akamai.com. (2015) Turn-on HTTP/2 today! [Online]. Available: https://http2.akamai.com/

[10] W. Cherif, Y. Fablet, E. Nassor, J. Taquet, and Y. Fujimori, "Dash fast start using HTTP/2," in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2015, pp. 25–30.

[11] M. Belshe, R. Peon, and M. Thomson. (2015, May) Hypertext Transfer Protocol Version 2 (HTTP/2). [Online]. Available: https://tools.ietf.org/html/rfc7540

[12] Github.io. (2015) What is HTTP/2? [Online]. Available: https://http2.github.io/

[13] W. Reilly. (2014, November 7) HTTP/2: A quick look. [Online]. Available: http://blog.scottlogic.com/2014/11/07/http-2-a-quick-look.html

[14] Chromium.org. (2010) Spdy: An experimental protocol for a faster web. [Online]. Available: https://www.chromium.org/spdy/spdy-whitepaper

[15] B. Thomas, R. Jurdak, and I. Atkinson, "Spdying up the web," *Communications of the ACM*, vol. 55, no. 12, pp. 64–73, 2012.

[16] J. Erman, V. Gopalakrishnan, R. Jana, and K. K. Ramakrishnan, "Towards a spdyier mobile web?" *Networking, IEEE/ACM Transactions on*, vol. 23, no. 6, pp. 2010–2023, 2015.

[17] F. Toomey. (2015, April 15) Why latency management will decide the future of the IoT. [Online]. Available: http://www.wirelessweek.com/article/2015/04/why-latency-management-will-decide-future-iot

[18] J. Parkinson. (2015, April 8) Forecasting the future of the Internet of Things. [Online]. Available: http://ww2.cfo.com/forecasting/2015/04/forecasting-future-internet-of-things/

[19] W3techs.com. (2016) Usage of SPDY for websites. [Online]. Available: http://w3techs.com/technologies/details/ce-spdy/all/all

[20] ——. (2016) Usage of HTTP/2 for websites. [Online]. Available: http://w3techs.com/technologies/details/ce-http2/all/all