UNM | Electrical & Computer Engineering

ECE540 Advance Networking

Final Project

SPDY Protocol

Alejandro Flores
Mustafa Al Mashhadani
Husain Al Yusuf

Supervised by: Dr. Wennie Shu

Table of Contents

1.0 Abstract 3
2.0 Introduction 3
2.1 HTTP Browsing protocols 3
2.1 SPDY Browsing protocols 4
3.0 Experiment/Lab 5
3.1 SPDY Server on Amazon EC2 5
3.2 SPDY Server on Virtual Environment 6
3.3 SPDY on Real World Servers and Websites 6
4.0 Results 7
4.1 SPDY Server on Amazon EC2 7
4.2 SPDY Server on Virtual Environment 8
4.3 SPDY on Real World Servers and Websites 9
5.0 Conclusion 10
6.0 Works Cited 11
7.0 Appendix A – HTML Code Used in Scenarios 1 & 2 12
7.1 Appendix B – SPDY Server on Amazon EC2 14
7.2 Appendix C – SPDY Server on Virtual Environment Results 15
7.3 Appendix D – SPDY On Real World Server and Websites Results 16
7.4 Appendix E – Full Size Graphs 17

1.0 Abstract

The 21st century has brought with it new innovations in webpage programming composed of dynamic and multi-domain content. These new pages make use of the current HTTP protocol, which has failed to evolve and meet the demands of the new generation of web users. For this reason, we decided to test and analyze SPDY, pronounce speedy, the recently developed Google protocol for transporting web content. Google claims that SPDY will improve the web browsing experience for users by reducing the number of TCP connections, packet prioritization, and performing HTTP header compression. In this paper, we evaluate these claims by conducting our own tests via various simulated scenarios. To be specific, we focused on comparing the TCP connection and load times of both HTTP and SPDY over SSL. As you will see, we found a reduction in both webpage load times and TCP connections. Finally, we attempt to make a quick comparison with results published by Google, taking into consideration that the same methodology was not used. With this paper we hope to provide additional evidence that can help determine if Google's SPDY protocol is a true contender to replacing the current HTTP protocol.

2.0 Introduction

Internet was first used for mail exchange purposes in military and research laboratories but the link concept was not known until the 1960's when Ted Nelson, a researcher, popularizes the hypertext concept (Stewart). The concept of a hyperlink was then materialized by Douglas Engelbart; the developer of the first working hypertext system. By the 1980's the Web itself was invented by Tim Berners-Lee and Robert Cailliau in Europe. The concept of the web rapidly spread around the world over the *Internet* in the 1990's by Marc Andreessen and the National Center for Supercomputer Applications (NCSA) team [Stewart]. Once the commercial Internet became available, the need for a web browser became apparent. Marc Anderson and his team at the NCSA designed the first web browser code named Mosaic. The Mosaic browser helped spread use and knowledge of the web across the world (Stewart). Soon after, alternatives to Mosaic were invented, as the use of Internet had become part of everyday life. Each browser built to run using the HyperText Transfer Protocol (HTTP) to transfer webpages between Internet clients and the web servers. Recently work from Google has revealed a new protocol for web browsing named SPDY which claims to deliver faster webpages loading time and make more efficient use of network resources via the use of less TCP connections.

2.1 HTTP Browsing protocols

HTTP is an application layer protocol and a set of standards that allow users of the World Wide Web to exchange information found on webpages. When wanting to access any webpage, http:// is used in front of the web address, instructing the browser to communicate over HTTP. Today's modern browsers no longer require http:// in front of the URL since it is now the default method of communication. HTTP's first version was release in 1991 under HTTP/0.9 but then improved in 1996 under HTTP/1.0. Today, we use the version that was release in 1997 and specified in RFC-2616 as HTTP/1.1. The HTTP protocol mainly uses the ports 80, 8008, and 8080 for server-to-clients communication. HTTP is equipped with a secure method of communication in which data is encrypted before it is sent called HyperText Transfer Protocol Secure (HTTPS). This secure protocol runs on port 443 and it encrypts sensitive data that requires private communication but slows down the data transfer (Computer Hope). However, the recent increase in bandwidth offered by Internet Service Providers has diminished the impact of slow data rates during encryption (Bryce).

2.1 SPDY Browsing protocols

Recently, Google has developed a proposed improvement to HTTP called SPDY. This alternative is an experimental protocols designed by Google to make web browsing faster. The protocol designed for low-latency transport of content over the World Wide Web introduces two layers of protocol. The lower layer is a general purpose framing layer which can be used on top of a reliable transport (likely TCP) for multiplexed, prioritized, and compressed data communication of many concurrent streams. The upper layer of the protocol provides HTTP-like RFC2616 semantics for compatibility with existing HTTP application servers (Google Developers). The technical goals specified by Google for this protocol are: (Chromium Project)

1- To allow many concurrent HTTP requests to run across a single TCP session.
2- To reduce the bandwidth currently used by HTTP by compressing headers and eliminating unnecessary headers.
3- To define a protocol that is easy to implement and server-efficient. Reduce the complexity of HTTP by cutting down on edge cases and defining easily parsed message formats.
4- To make SSL the underlying transport protocol, for better security and compatibility with existing network infrastructure. Although SSL does introduce a latency penalty, Google believe that the long-term future of the web depends on a secure network connection. In addition, the use of SSL is necessary to ensure that communication across existing proxies is not broken.
5- To enable the server to initiate communications with the client and push data to the client whenever possible.

In our experiment, we will try to validate some of the claims made by Google with this innovative protocol.

3.0 Experiment/Lab

In the following sections each experiment's scenario will be presented with its associated experimental design. Three scenarios will be examined on our study and all experiments are designed to focus on observing the performance of SPDY and HTTP over SSL(port 443). As previously stated, SPDY is strictly deployed over SSL to avoid issues with the many HTTP proxies (Bryce). We will focus on measuring webpage downloading time and number of TCP connections established for each session.

3.1 SPDY Server on Amazon EC2

In this first scenario, a Ubuntu web server was built and configured within the Amazon cloud computing service. This server, would host a specially constructed webpages consisting of html-frames that would simulate multiple domain-originated websites more details about the HTML code shown in the Appendix. With this setup, we had more control over the amount of traffic on the server and client side and therefore being able to capture a more accurate reading of SPDY performance over HTTP.

<u>Network Diagram</u>

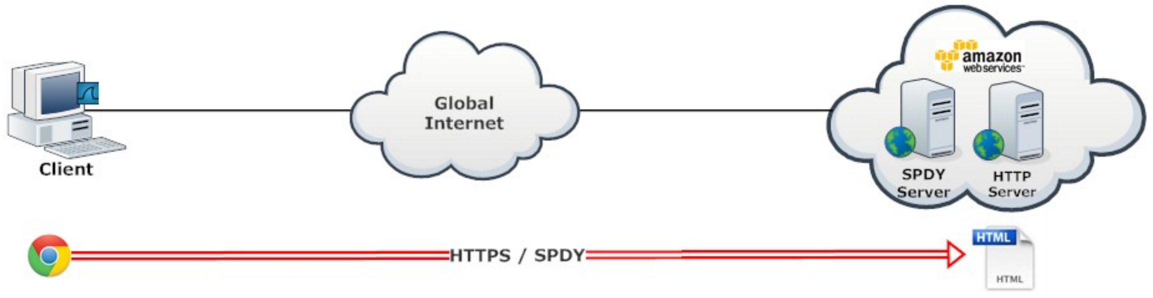


Figure 1: SPDY on Amazon EC2

<u>Experiment Process</u>

The process for this part of the experiment is as follows:

1- Configure an Amazon hosted server with SPDY enabled and another with HTTP (both over SSL).
2- A SPDY supported browser such as Chrome or Firefox was used to access both Amazon hosted servers.
3- Wireshark was installed on the client computer to capture all network traffic from the time the website was launched to when it finished loading.
4- Once the website finished fully downloading, all traffic capturing was stopped and saved to an output file for future analysis.
5- The browser cache was then cleared and the process was repeated for twenty times for both SPDY and HTTP .

3.2 SPDY Server on Virtual Environment

The second scenario was built as a virtual lab environment, where VirtualBox was used as the platform to host a Ubuntu virtual servers, router and client computer as seen in figure 2. As simulated in the first scenario, the website developed in the webserver contained a reference to another webpage that is external to the webserver, this causing a second TCP connection. Conducting this scenario allowed us to study the behavior of SPDY from theoretical point of view and examine SPDY's performance in a more meticulous manner.

Network Diagram

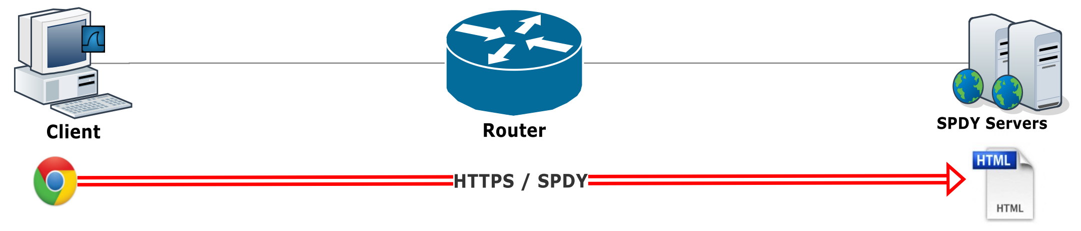


Figure 2: SPDY Server on Virtual Environment

Experiment Process
The process to perform this experiment was exactly the same as second scenario experiment's procedure.

3.3 SPDY on Real World Servers and Websites

In an effort to adhere our experiment to a real world scenario, we decided to do one last simulation. This last scenario was designed to study the SPDY effect in real life SPDY enabled web servers such as Google.com and Amazon.com. Furthermore, this last scenario is somewhat unique to the first two in that we decided to stop clearing the browser cache after the first 10 simulations. The idea behind this was to capture the different reading or webpage load time and TCP connections when browser cache was not cleared.

Network Diagram

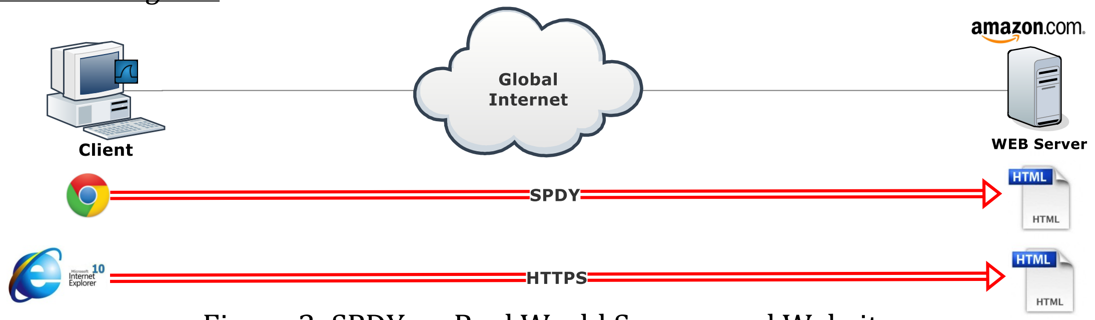


Figure 3: SPDY on Real World Servers and Websites

Experiment Process

The process for this experiment is as follows:

1- As the web server configuration is not controlled by us, Chrome was used to communicate using the SPDY protocol and Internet-Explore for HTTP communications as IE version 10 and older doesn't support SPDY
2- Wireshark was installed on the client computer to capture all network traffic from the time the website was launched to when it finished loading.
3- Once the website finished fully downloading all traffic capturing was stopped and saved to an output file for future analysis.
4- The above process was performed for 20 times with loading the Amazon page from the SPDY enabled browser and from a non-SPDY supported browser such as IE v.10 or older.
5- The browser cache was only cleared for the first 10 sessions.

4.0 Results

In this section, we will go over the results obtained for each scenario and compare them to those published by Google in terms of webpage load times and TCP connections. We do recognize that our result may vary widely from those published by Google since the same methodology for testing was not used. Therefore in this report we will consider Google published results as a baseline. In addition, we want to disclose that the speeds used to conduct each of our experiment varied for each scenario. Refer to the Appendix for detail data and gull size graphs.

4.1 SPDY Server on Amazon EC2

In our analysis of the first of experiment, we found that the SPDY protocol makes a considerable improvement in terms of webpage download time (Fig. 4). On average, there was a decrease of 21% download time when compared to HTTP. This reduction in webpage download time can be attributed to SPDY's efficient use of TCP that cause TCP's fast retransmission to engage rather than basic retransmission (Chrome Project). In addition, we need recall that SPDY multiplexes requests via a single TCP connection without the head-of-line blocking limitation offered in HTTP (Robert).

In regards to TCP connections (Fig. 5), SPDY did an impressive job at suppressing the number of simultaneous TCP connections within different domains. On average, the number of TCP connections dropped by 57% when compared to those in HTTP.

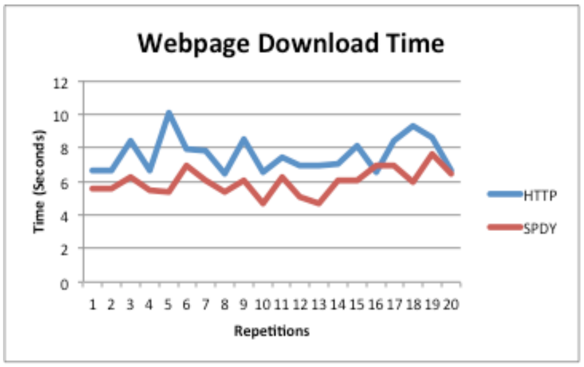


Figure 4

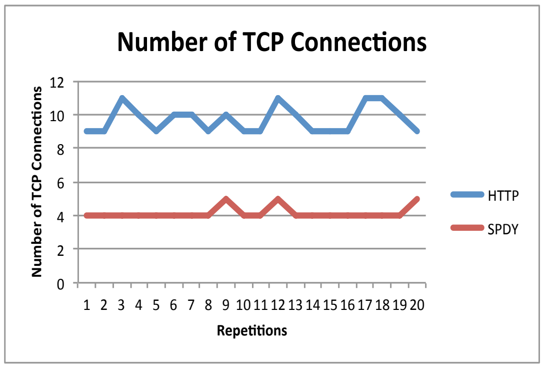


Figure 5

In regards to TCP connections (Fig. 5), SPDY did an impressive job at suppressing the number of simultaneous TCP connections within different domains. On average, the number of TCP connections dropped by 57% when compared to those in HTTP.

As a final step to this scenario, we attempted to compare the results obtained in our experiment to those published by Google. Keeping in mind that results may vary based on the fact that the same testing methodology was not used but as mentioned, Google results are being used as a baseline. In this first scenario, we noticed that our results in webpage load times falls below those advertised by Google of 39%-55% over SSL. Furthermore, our results in TCP connections of 57% exceed those obtained by Google in the range of 23%-43%.

4.2 SPDY Server on Virtual Environment

Taking into consideration that our website is using one extra IP address as referral within its HTML-page, it is obvious that SPDY has only two TCP connections (one for each IP) however HTTP has reached up to seven TCP connections in many attempts (Figure 7). This significant different makes SPDY better than HTTP in terms of resources utilization, as it use lower network resources to achieve the same final result which is downloading the webpage.

An interesting behavior for SPDY was observed during performing this scenario. HTTP server sends rest packet to close the session once browser completes downloading the webpage. In contrast, SPDY server keep the connection open for further communication and this feature avoid delay caused by TCP-handshake process for further communication between the same end hosts.

As expected, times required to download a webpage from SPDY servers is lower than HTTP server (Figure 6). This can be recognizing easily while performing the experiment. Although, this feature will improve end user's experience, it utilizes more throughput from the actual bandwidth of the link.

As expected, times required to download a webpage from SPDY servers is lower than HTTP server. This can be recognizing easily while performing the experiment.

Although, this feature will improve end user's experience, it utilizes more throughput from the actual bandwidth of the link.

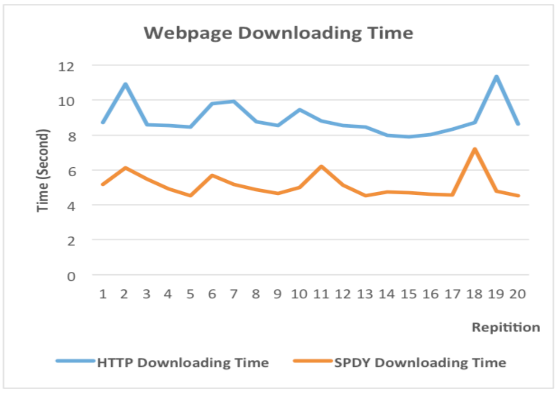


Figure 6

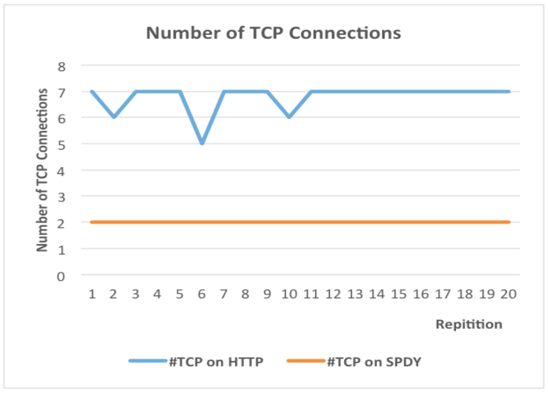


Figure7

Once again, taking into consideration the fact that Google did not make their testing methodology public, we do our best effort to compare out results to what they have published (Chromium Project). In this scenario, we saw a reduction in TCP connections of approximately 70% between SPDY and HTTP. This indicates a significant improvement from the 43% Google obtained. Also, the webpage downloading time almost doubled to that of Google. SPDY's downloading time in our test was 42% better than HTTP, while Google archive only 26.79%.

4.3 SPDY on Real World Servers and Websites

The loading time performance shows good results with SPDY and cache disabled test criteria as we can see that the loading time is near the half of HTTP loading time, but when we have the caching enabled the loading time will be slightly lower for HTTP with the cost of memory overhead with the use of HTTP.

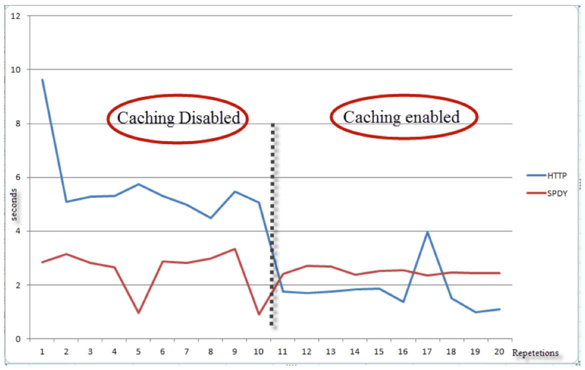
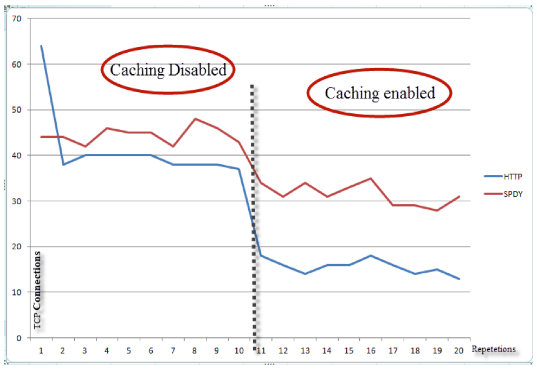


Figure 8: Webpage Downloading Time



9: Number of TCP connections

The performance test for the number of TCP connections shows that HTTP have lower number of TCP connections for both cases, with or without the use of cache, we can note that with the use of cache that number of TCP connections gets lower for both cases with an obvious difference using HTTP protocol.

In this scenario, SPDY fails to deliver in term of reducing download time when caching is enabled. However, it does perform well when it comes to reducing the amount of TCP connections. For this reason, we feel that further testing need to been done in this area of the project as simulating with real life systems is very complex.

5.0 Conclusion

The SPDY protocol presents an improvement over the HTTP protocol in terms of load time and TCP connections. While this new protocol shows potential, this does not mean that Google comes out as the winner. The SPDY protocol is only an option of many others that promise to enhance the performance of web browsing. As mentioned before, our intent was to test the protocol performance and compare it to those results published by Google. Without a doubt, additional test need to be conducted with real live scenarios that can more clearly depict the behavior this new protocol promises to deliver. In addition, more accurate testing scenarios could be built if Google fully published their testing methodology. Overall, based on our experiment we can conclude that SPDY does in fact perform better than HTTP.

6.0 Works Cited

Bryce, Thomas, Raja Jurdak, & Ian Atkinson. (2012). SPDYing Up the Web. Communications of the ACM, Vol. 55, 12.

Chromium Project, (2013) . An Experimental protocol for a faster web. Retrieved from http://www.chromium.org/spdy/

Computer Hope . (2013) HTTP. Retrieved from http://www.computerhope.com/jargon/h/http.htm

Google Developers. (2012) Make the Web Faster. Retrieved from https://developers.google.com/speed/protocols/

Kurose, & Ross.(2013). Computer Networking (6th ed). New York, NY: Pearson Education

Robert Peon, (2012). SPDY It's Here!, https://www.youtube.com/watch?v=zN5MYf8FtN0

Stewart, Bill. (1996). Ted Nelson Discovers Hypertext, The World's First Web Publishing. Retrieved from http://www.livinginternet.com/w/wi_nelson.htm

Stewart, Bill .(1996). TWolrd Wide Web (WWW0 History, The World's First Web Publishing. Retrieved from http://www.livinginternet.com/w/wi_nelson.htm

Stewart, Bill. (1996). Mosaic—The First Global Web Brwser, The World's First Web Publishing. Retrieved from http://www.livinginternet.com/w/wi_nelson.htm

Will Chan, & Roberto Peon .(2011). SPDY Essential, https://www.youtube.com/watch?v=TNBkxA313kk

7.0 Appendix A – HTML Code Used in Scenarios 1 & 2

```
<html>
<body>
<img src="img/soe.gif">
<hr>
<center>
<h1>SPDY Server</h1>
<p>Welcome to our ECE540 SPDY test server!</p>
Link to <a href="https://54.201.125.84">HTTP server</a>
</center>


Enable TCP Connection to Google<br/>
<img src="https://www.google.com/images/srpr/logo11w.png"
width="300">
<br/>

<h3>Introduction</h3>
<p>
WEB applications are dominating the business industry with all
its varieties. Customer satisfaction while surfing any
organizationâ€™s website is one of the most important factors
that companies are looking after. As webpages starts to become
more advance technologists for web surfing have failed to adapt
and meet business requirements. In 2012, Google launched a new
protocol called â€œSPDYâ€  as an upgrade to HTTP 1.1. In this
paper we propose to explore this new protocol and compare its
performance with existing technologies.
</p>
<h3>Objective</h3>
<p>
The objective of this study is to measure the performance of
using â€œSPDYâ€  and comparing it to the previous HTTP1.1
protocol. This project will be a combination of â€œproject and
evaluationâ€  based style as described in the assignmentâ€™s
sheet. Our focal point will be on the web page downloading time,
which is in another words the throughput achieved with SPDY in
contrast with HTTP1.1. Another interesting feature of SPDY that
will be examine in our study is the SPDYâ€™s TCP connections
behavior in comparison with HTTP1.1.
</p>

<H3>Group Members</h3>
<img src="img/a_flores.jpg" width="300"></br>
<b>Alejandro Flores</b></br>
Email alejflor@unm.edu</br>
Degree seeking: MS in CompE, CompNet&Sys</br>
Advisor: TBD</br>
```

```
I am originally from El Paso, TX. I graduated from New Mexico
State University in 2005 with a Bachelorâ€™s Degree in Computer
Engineering Technology. Since then, I have been working for both
UNM and NMSU in the area of network design, server administration
and application development. I am currently pursuing a Masters
Degree in Computer Engineering and in the search of an advisor
that can use my skills to conduct research in the area of
computer engineering.
</br></br>

<img src="img/mustafa.jpg" width="300"><br>
<b>Mustafa Al Mashhadani</b></br>
Email malmashhadani@unm.edu</br>
Degree seeking: MS in CompE, CompNet&Sys</br>
Advisor: Dr. W. Shu</br>
My name is Mustafa I've get my BS in Information Engineering from
Nahrain University (Iraq), I plan to complete my study in Network
Engineering.
</br></br>

<img src ="img/mypic_afs.jpg" width="300"></br>
<b>Husain Al Yusuf</b></br>
Email halyusuf@unm.edu</br>
Degree seeking: MS in CompE, CompNet&Sys</br>
Advisor: Dr. W. Shu</br>
I'm from Bahrain. I have earned my B.Sc. in Computer Engineering
from King Abdul-Aziz University in Jeddah-KSA and graduated in
2006. I have 6+ years of work experience and most of it was in
the IT infrastructure services within the financial services
industry.
</br>

<h3> Load Test</h3>
<center>
<img src="img/nikon.jpg" width="800"></br>
Picture by Nikon!
</center>
</br></br>
<hr>
<center>
Server Powered by Ubuntu
</center>

</body>
</html>
```

7.1 Appendix B – SPDY Server on Amazon EC2

Repetition	HTTP				
	#Packets	Downloading Time	Downloaded Bytes	Throughput(Mbits/s)	#TCP on HTTP
1	8339	6.682552	9726197	11.64369181	9
2	8395	6.635064	9730443	11.73214667	9
3	8405	8.462049	9732155	9.200755042	11
4	8319	6.614158	9724785	11.76238608	10
5	8162	10.070165	9714036	7.717081895	9
6	8366	7.976932	9729269	9.757404476	10
7	8357	7.871864	9728311	9.886665725	10
8	8355	6.461424	9728805	12.04540052	9
9	8387	8.480123	9730729	9.179799868	10
10	8668	6.585104	9754032	11.84981376	9
11	8307	7.407157	9723199	10.50140992	9
12	8379	6.928105	9727439	11.23243831	11
13	8458	6.903054	9738863	11.28643989	10
14	8374	7.00848	9730753	11.10740474	9
15	8364	8.110783	9728934	9.596049111	9
16	8351	6.530395	9727661	11.91678114	9
17	8379	8.420031	9730140	9.244754562	11
18	8388	9.318425	9731822	8.354907187	11
19	8364	8.638991	9729938	9.010254091	10
20	8270	6.6181	9718333	11.74758073	9
Average	**8369.35**	**7.5861478**	**9729292.2**	**10.43865828**	**9.7**

Repetition	SPDY				
	#Packets	Downloading Time	Downloaded Bytes	Throughput(Mbits/s)	#TCP on SPDY
1	8273	5.528038	9792952	14.1720473	4
2	8159	5.60174	9761551	13.94074127	4
3	8274	6.284596	9793873	12.46714729	4
4	8103	5.454791	9755855	14.30794324	4
5	8104	5.401357	9757457	14.45186015	4
6	8331	6.963359	9797261	11.25578733	4
7	8290	6.013799	9795328	13.03046943	4
8	8200	5.33201	9786710	14.6837084	4
9	8337	6.057527	9798515	12.94061421	5
10	8410	4.73131	9815775	16.59713695	4
11	8275	6.225865	9792489	12.58297634	4
12	8190	5.100791	9785899	15.34804935	5
13	8177	4.671292	9785520	16.75856701	4
14	8187	6.042038	9786816	12.95829785	4
15	8180	6.041744	9783410	12.95441846	4
16	8345	6.963252	9800334	11.25949082	4
17	8232	6.962994	9790088	11.24813607	4
18	8183	5.939066	9786265	13.18222764	4
19	8288	7.67592	9794530	10.20805845	4
20	8252	6.453954	9793281	12.13926347	5
Average	**8239.5**	**5.97227215**	**9787695.45**	**13.32434705**	**4.15**

7.2 Appendix C – SPDY Server on Virtual Environment Results

Repetition	HTTP				
	#Packets	Downloading Time	Downloaded Bytes	Throughput (mbits/s)	#TCP on HTTP
1	1237	8.708029	9323156	8.565112496	7
2	1265	10.923309	9324428	6.829013443	6
3	1340	8.599731	9329984	8.679326365	7
4	1274	8.558562	9325640	8.717015779	7
5	1271	8.462417	9325444	8.815868091	7
6	1263	9.795501	9323487	7.614505476	5
7	1329	9.919201	9329246	7.524191515	7
8	1292	8.773668	9326804	8.504360092	7
9	1258	8.523815	9324572	8.751547986	7
10	1211	9.443609	9320778	7.895945713	6
11	1228	8.785404	9322592	8.489164073	7
12	1270	8.553245	9324410	8.721284144	7
13	1264	8.435742	9324956	8.84328231	7
14	1283	7.965119	9326180	9.367021384	7
15	1135	7.873165	9316430	9.46651569	7
16	1253	8.027133	9324242	9.292724563	7
17	1195	8.314962	9320372	8.967326128	7
18	1238	8.712659	9323276	8.560671088	7
19	1222	11.35193	9322184	6.56958526	7
20	1276	8.618497	9325748	8.656495906	7
Average	**1255.2**	**8.9172849**	**9324196.45**	**8.441547875**	**6.8**

Repetition	SPDY				
	#Packets	Downloading Time	Downloaded Bytes	Throughput (mbits/s)	#TCP on SPDY
1	2259	5.183361	9460956	14.60204065	2
2	2165	6.119488	9454722	12.36014778	2
3	2285	5.48345	9462642	13.80538457	2
4	2140	4.917874	9453102	15.09260767	2
5	2157	4.515869	9454224	12.63057739	2
6	2193	5.692672	9456810	1.104252372	2
7	2205	5.171035	9457488	2.314198867	2
8	2234	4.870176	9459352	15.53841504	2
9	2352	4.669991	9467405	1.17213761	2
10	2265	5.005456	9461368	15.12168801	2
11	2126	6.202168	9421740	12.1528343	2
12	2042	5.141831	9446634	7.463561877	2
13	2315	4.501827	9464668	7.9772827	2
14	2131	4.73222	9451830	15.97868231	2
15	2224	4.698316	9458916	1.204321616	2
16	2337	4.587777	9466150	16.50673082	2
17	2251	4.564547	9460474	16.58078929	2
18	2386	7.192253	9469384	6.768527481	2
19	2283	4.800492	9462752	2.256564894	2
20	2179	4.534262	9455916	1.170332477	2
Average	**2226.45**	**5.12925325**	**9457326.65**	**9.590053885**	**2**

7.3 Appendix D – SPDY On Real World Server and Websites Results

Webpage Loading Times

	SPDY									
Cache	Test 1	Test2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10
Cache Disabled	2.834118	3.135537	2.826154	2.667771	0.961368	2.865283	2.815795	2.997078	3.346629	0.896259
Cache Enabled	2.420956	2.700922	2.691155	2.370933	2.522045	2.536273	2.366712	2.450488	2.439385	2.442189

	HTTP									
Cache	Test 1	Test2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10
Cache Disabled	9.624393	5.075998	5.289511	5.313426	5.756009	5.30933	4.980125	4.482583	5.48583	5.07536
Cache Enabled	1.751177	1.697038	1.76351	1.834875	1.849248	1.382889	3.960572	1.51279	0.994619	1.110107

TCP Connections

	SPDY									
Cache	Test 1	Test2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10
Cache Disabled	44	44	42	46	45	45	42	48	46	43
Cache Enabled	34	31	34	31	33	35	29	29	28	31

	HTTP									
Cache	Test 1	Test2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10
Cache Disabled	64	38	40	40	40	40	38	38	38	37
Cache Enabled	18	16	14	16	16	18	16	14	15	13

7.4 Appendix E – Full Size Graphs

SPDY Server on Amazon EC2

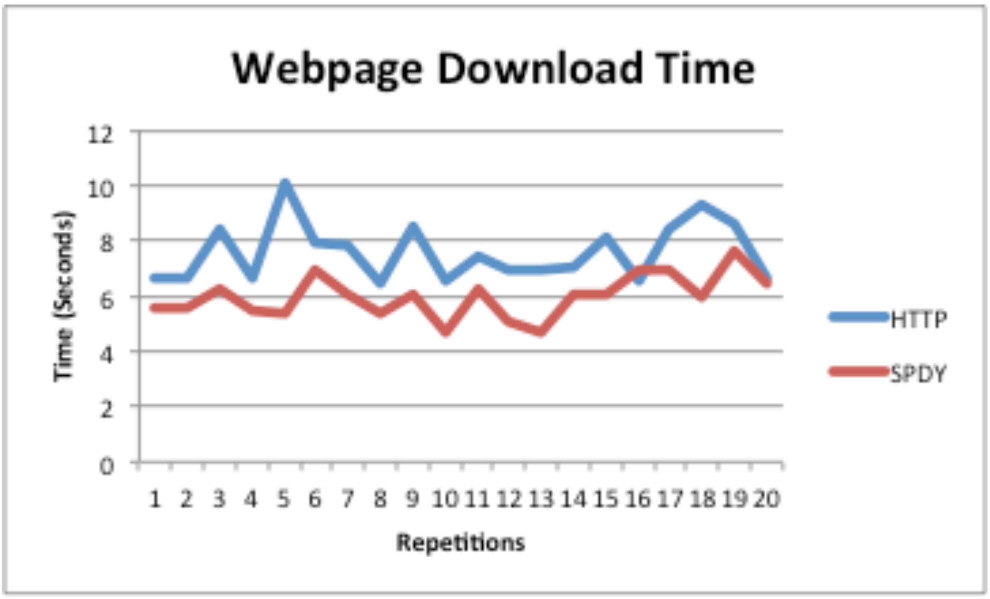


Figure 4

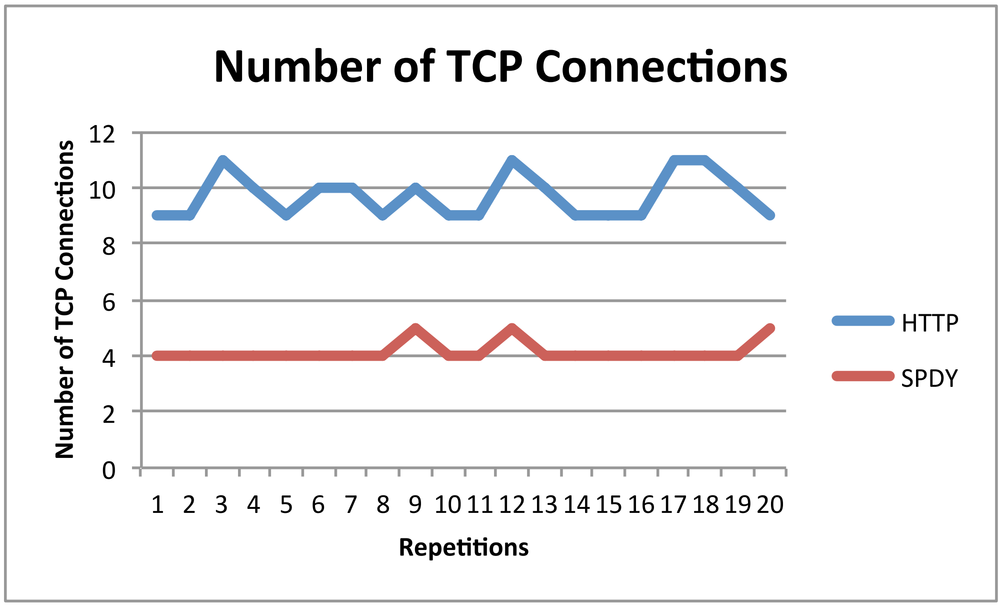


Figure 5

SPDY Server on Amazon EC2

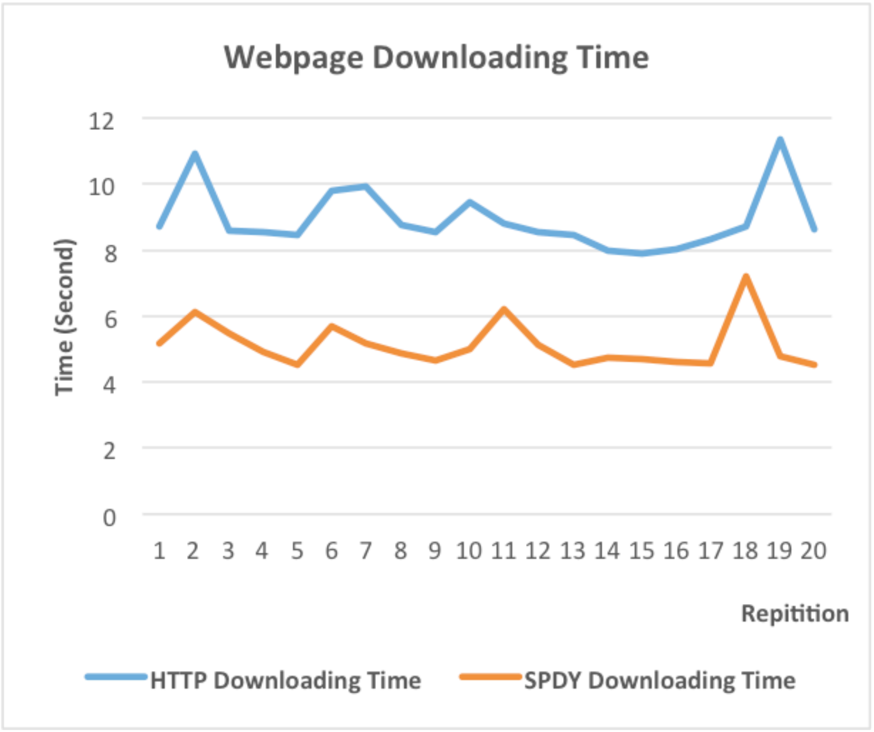


Figure 6

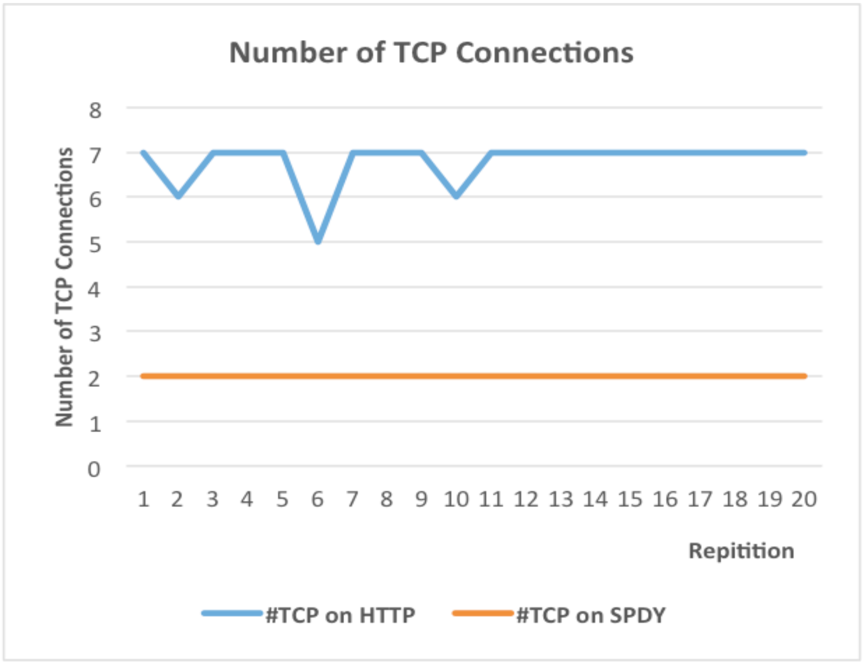


Figure 7

SPDY on Real World Servers ad Websites

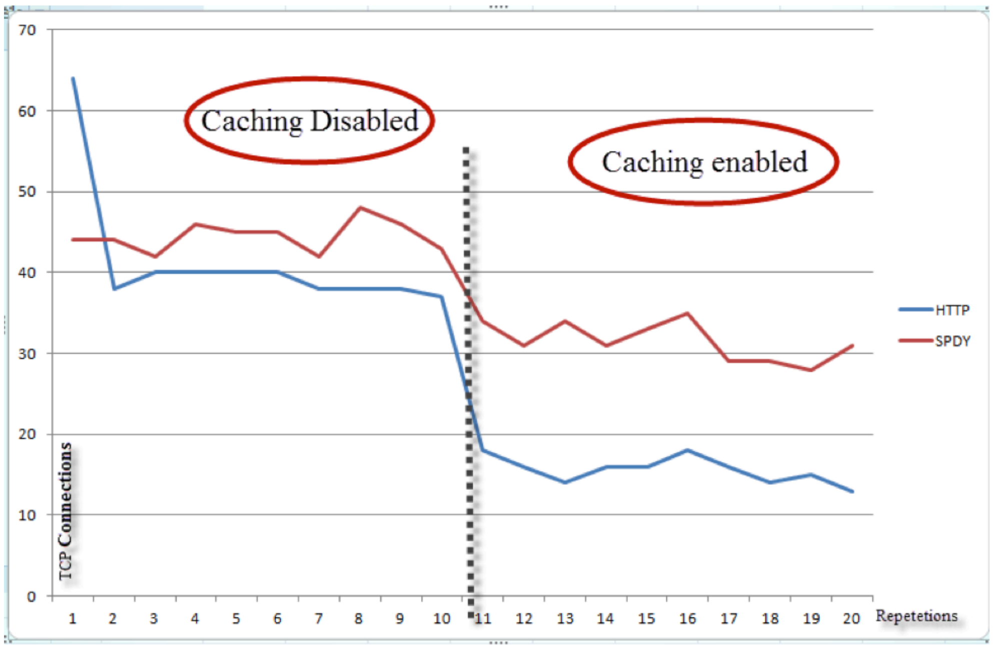


Figure 8: Webpage Downloading Time

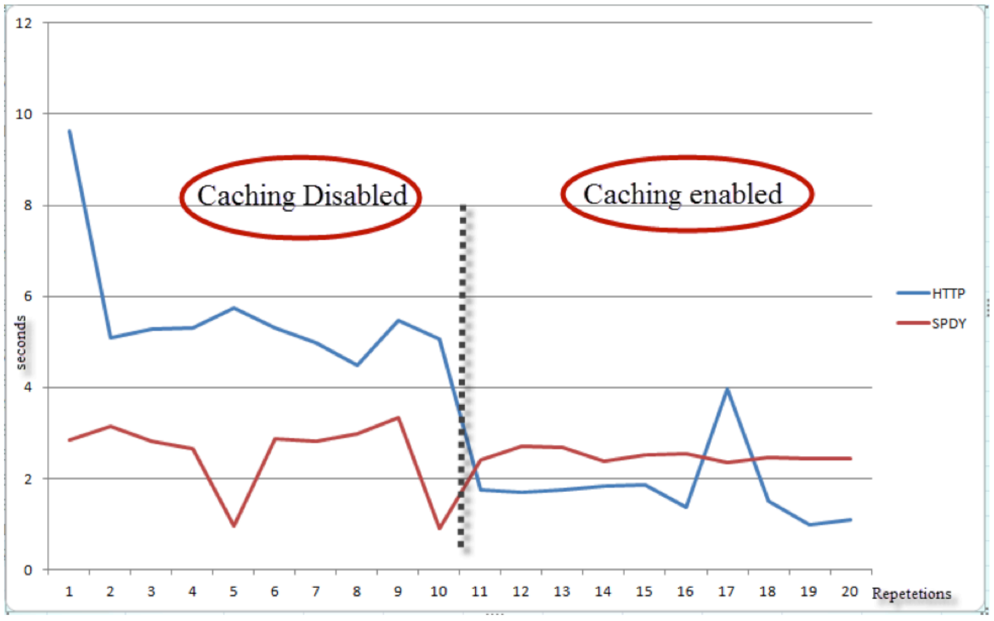


Figure 9: Number of TCP connections