Jari Kallio

4G Mobile Network Efficiency Improvement Using Small Object Caching and Speedy Protocol

Helsinki Metropolia University of Applied Sciences

Master's Degree

Information Technology

Master's Thesis

1 May 2016

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Jari Kallio 4G Mobile Network Efficiency Improvement Using Small Object Caching and Speedy Protocol 73 pages 1 May 2016
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Jouko Kurki, Principal Lecturer

The Hypertext Transfer Protocol (HTTP) is the most widely used application layer protocol today. It is easy to realise because every time a Web connection is opened, HTTP protocol is used over Transmission Control Protocol / Internet Protocol (TCP/IP). It is a so called request – response protocol, where the web browser sends a request to the web server which returns the web content.

HTTP has a long history, starting already in the early nineties with the version 0.9 to the latest version 2. Version 1.1 is still the most common today but it is not suitable to serve the new web page structure which means a very large number of small objects per a single web page. Speedy (SPDY) is Google's solution trying to handle the HTTP protocol more effectively. SPDY version 2 was chosen as the basis for HTTP version 2.

This study concentrates on comparing HTTP version 1.1 and SPDY performance in a laboratory environment. A 4G network was reserved only for this test period. The web pages were downloaded through HTTP or SPDY proxies and the main topic of interest was the page download time. Web pages include a different number and size of small objects. Another subject of the research was the impact of caching. Different servers were used with caching allowed.

The two most critical issues affecting web page download time are latency and bandwidth. During the test period it was possible to add some delay and change the bandwidth. Because the network was reserved only for this study it was difficult to detect the effect of network load. Limiting the number of physical resource blocks in the base station was used as means to simulate a high usage of the base station.

As a result the tests showed that the SPDY protocol reduces the web page load time when the number of small objects per page is more than 250. The positive impact is even better when delay is increased. Caching with SPDY also performs better than HTTP version 1.1 with caching when the number of small objects per page is more than 250.

Keywords	HTTP, SPDY, Web page download time, Energy efficiency, 4G mobile network

Contents

List of Figures

List of Tables

Abbreviations/Acronyms

2G networks	Second generation mobile network
3G networks	Third generation mobile network
4G networks	Fourth generation mobile network, also called Long Term Evolution (LTE) network
ADB	Android Debug Bridge
AH	Authentication Header
ALPN	Application Layer Protocol Negotiation
APM	Application Performance Management
CDN	Content Delivery Networks
CSS	Cascading Style Sheet
DL	Downlink
eNB	Evolved Node B, 4G network base station
EPC	Evolved Packet Core
ESP	Encapsulating Security Payload
GSA	Global mobile Suppliers Association
HOL	Head Of Line
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HTTP-GW	HTTP Working Group
ICT	Information and Communication Technology
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPsec	Internet Protocol Security Architecture
KPI	Key Performance Indicator
L1	Layer 1
L2	Layer 2
LTE	Long Term Evolution
MIMO	Multiple-Input and Multiple-Output
NCSA	National Centre of Supercomputing Applications
NPN	Next Protocol Negotiation
PAC	Proxy Auto-Configuration
PKI	Public Key Infrastructure
PRB	Physical Resource Block
PLT	Page Load Time

QoS	Quality of Service
QoE	Quality of Experience
RFC	Request for Comments
RTT	Round Trip Time
SD card	Secure Digital card
SDK	Software Development Kit
SSH	Secure Shell
SSL	Secure Sockets Layer
SPDY	An experimental protocol for a faster web
TLS	Transport Layer Security
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
USB	Universal Serial Bus
UE	User Equipment
URL	Uniform Resource Locator
VPN	Virtual Private Network
W3C	World Wide Web Consortium
W3Techs	World Wide Web Technology Surveys
WPA2	Wi-Fi Protected Access II
WWW	World Wide Web

1 Introduction

Hypertext Transfer Protocol (HTTP) traffic has grown heavily in recent years. This is easy to understand because most of today's popular applications are using HTTP protocol, e.g. online shopping, Google, Facebook, YouTube and Twitter just to name a few. If thinking of Internet usage HTTP is probably the most popular application layer protocol. The main reasons why Internet usage has grown so rapidly are new and fast mobile networks, such as Long Term Evolution networks (LTE), and new smart phones and tablets. The clear trend of increasing mobile device usage can be seen in Figure 1 [1].

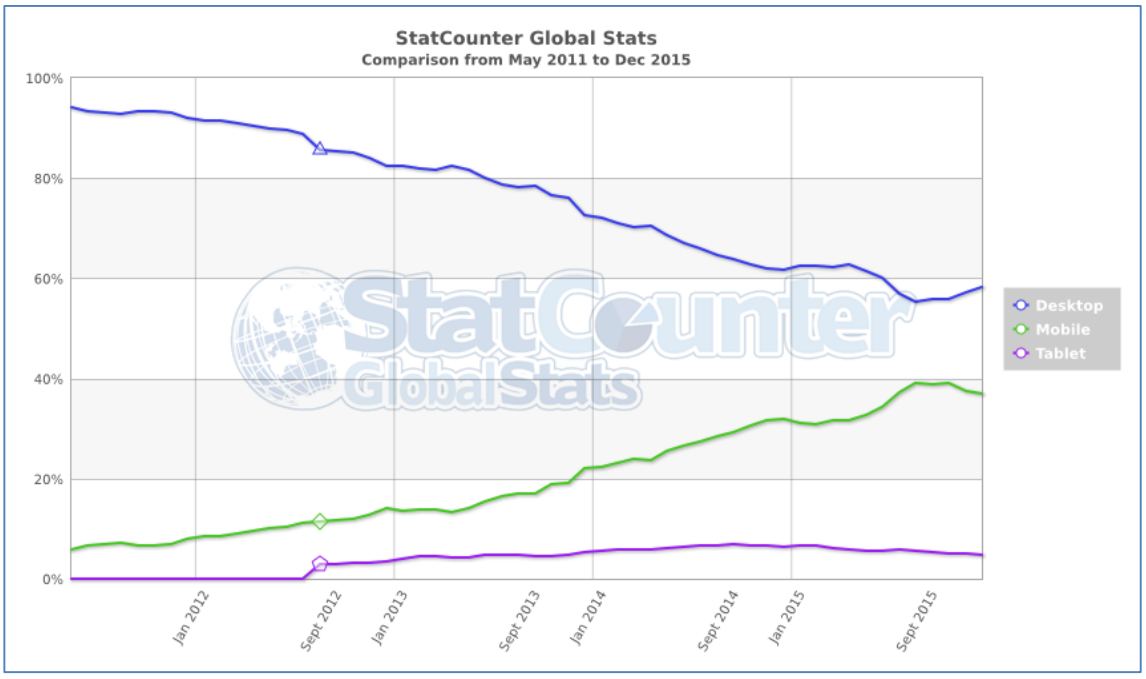

Figure 1. Internet web access worldwide between May 2011 to Dec 2015. Copied from StatCounter Global Stats (2015) [1]

Another example is from the US. Figure 2 shows the time people spent with Internet. In early 2014, Internet usage on mobile devices exceeded PC usage (minutes per month) [2].

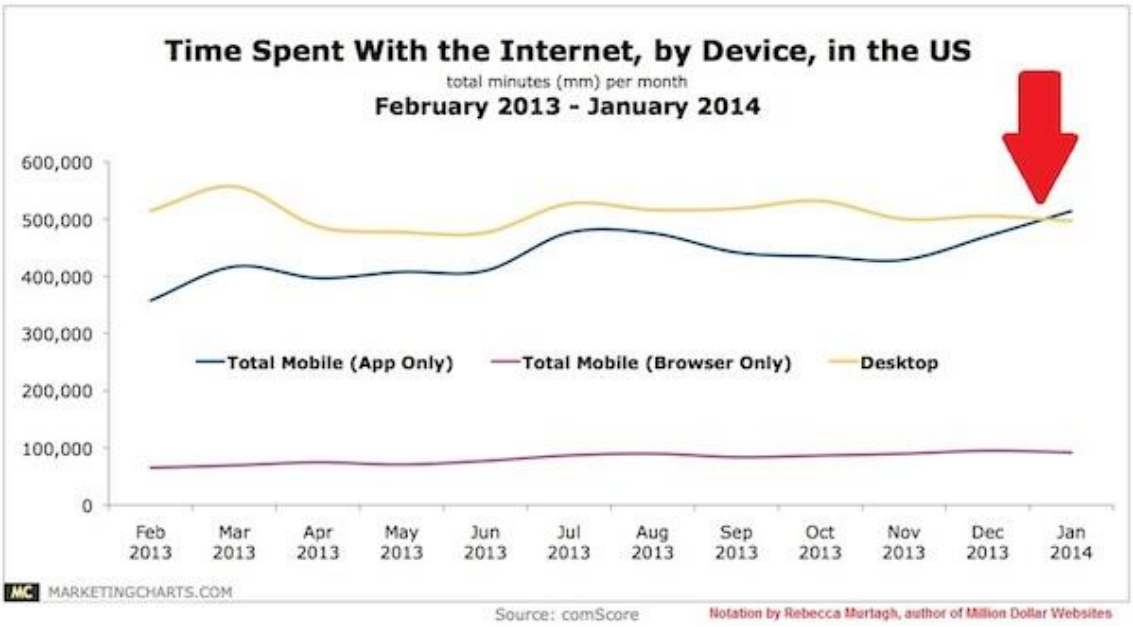

Figure 2. Time spent with the Internet in US between Feb 2013 to Jan 2014. Copied from Search Engine Watch (2014) [2]

Thanks to the new networks and powerful smart phones and tablets the data traffic has exceeded many times the traditional speech traffic. That means more and more HTTP requests which is the first application layer message that the terminal is sending when establishing Internet connection.

According to the Global mobile Suppliers Association (GSA) LTE user devices growth has been very aggressive during the past few years as seen in Figure 3 [3]. Internet access with mobile devices became available in 1991 with second generation (2G) networks but only now with LTE networks and new smart phones we can talk a real mobile broadband connection. The old, common statement "the Internet in your pocket" is now usable.

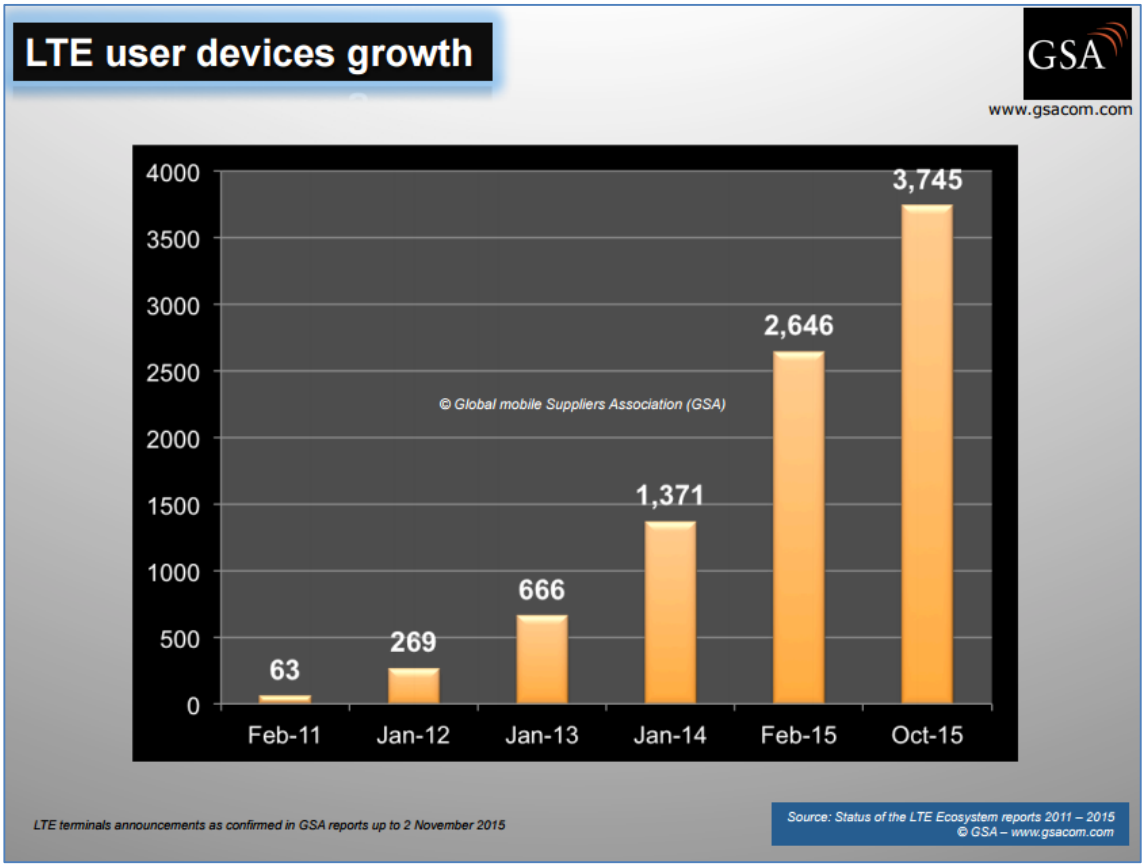

Figure 3. LTE user devices growth between Feb 2011 to Oct 2015. Copied from GSA, LTE user devices growth (2015) [3]

LTE ecosystem has already now shown its success among the other mobile network technologies. According to GSA's forecast LTE will have 45% of global mobile subscriptions in 2020.

The present study focuses on the web browsing performance improvement comparing HTTP and Speedy protocols. The most interesting variables were the number and the size of small objects per web page.

First the study provides some theoretical background information concerning the protocols and caching method. The second part concentrates on the test methodology and the lab setup continuing to different test cases. The third part presents the measurement results. The last two parts summarise the study and provide few ideas for a follow-up testing which could be the new HTTP/2 features and the terminal battery consumption.

2 Overview of HTTP Browsing

HTTP protocol has long history as can be seen also in Figure 4.

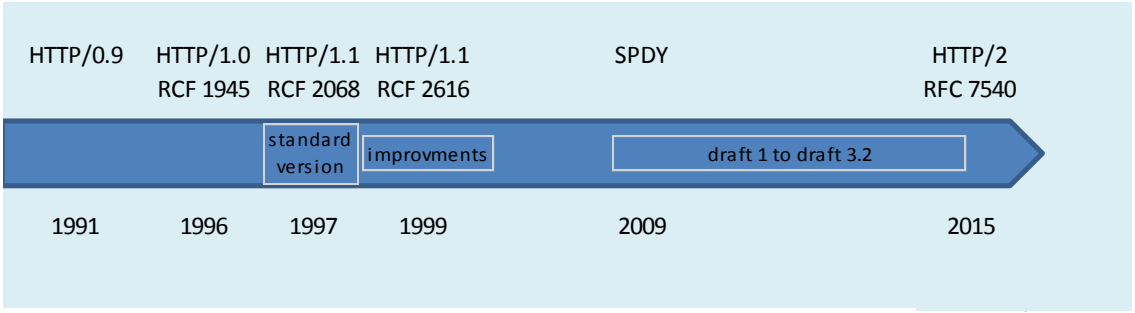

Figure 4. HTTP history

The Hypertext Transfer Protocol (HTTP) is the common language between clients and servers. It is a network protocol which is used to deliver virtually files and other data such as images on the World Wide Web (WWW). HTTP traffic takes place through Transmission Control Protocol/Internet Protocol (TCP/IP) sockets. The history of HTTP began already in 1991 and the first version was HTTP/0.9.

Today the mostly used HTTP version is 1.1 which is already about 18 years old. During that time the web content has changed dramatically. In the early phase the content was more or less pure text, today one web page could easily contain more than 100 objects [4]. So the page content has come more and more complex because of the page size and the number of separate objects.

2.1 HTTP/0.9

Timothy Berners–Lee also known as the developer of WWW outlined the protocol in 1991. The first version was quite simple for example the client request was only one line, a single ASCII string and the server response was a single hypertext document. Only one method (GET) was supported. [5, Chapter 9.] The main features of HTTP/0.9 are shown in Table 1.

Table 1. The main features of HTTP/0.9. Data gathered from Grigorik IL [5, Chapter 9]

The main features of HTTP/0.9
Client-server, request-response protocol
ASCII protocol, running over a TCP/IP link
Designed to transfer hypertext documents
The connection between server and client is closed after every request

After the first proper browsers the popularity of HTTP or Web start to increase rapidly and a protocol which was supporting just hypertext documents was not anymore enough. There are still some web servers, e.g. Apache, which are supporting HTTP/0.9.

2.2 HTTP/1.0

Because of the growing Web usage and the limitations of HTTP/0.9 some improvements were needed. HTTP Working Group (HTTP-WG) published a Request for Comments (RFC) number 1945 in May 1996 [6]. That was so called informational RFC and it was not a real Internet standard.

It is quite interesting that about at the same time NCSA Mosaic, the first popular web browser, was born. [5.] Later the browser was better known as Netscape Navigator. Also about the same time the World Wide Web Consortium (W3C) was established. Both of the groups, HTTP-WG and W3C, are taking care the evolution of the Web. Due to the new protocol the response object could be also something else than just a hypertext document. HTML files, plain text files, images, or any other content type were supported. HTTP/1.0 protocol was supporting GET, POST and HEAD methods. The main features of HTTP/1.0 are shown in Table 2. [5, Chapter 9.]

Table 2. The main features of HTTP/1.0. Data gathered from Grigorik IL [5, Chapter 9]

The main features of HTTP/1.0
Request may consist of multiple newline separated header fields
Response object is prefixed with a response status line
Response object has its own set of newline separated header fields
Response object is not limited to hypertext.
The connection between server and client is closed after every request

Even though HTTP/1.0 was not a real Internet standard almost every web server is still able to support it. It might be useful for example when writing certain test scenarios.

2.3 HTTP/1.1

The biggest problem when using HTTP/1.0 was the extra delay which was caused by the need to open a new TCP connection for every HTTP request. When using HTTP/1.1 the browser creates one TCP connection and uses that same connection to send many requests, because of keep-alive connections.

In January 1997 a new standard for HTTP/1.1 was defined in RFC 2068 [7]. That was the first official Internet standard and was supporting four new methods; OPTIONS, PUT, DELETE and TRACE. The first version was updated to RFC 2616 in June 1999 [8]. The updated version was supporting one new method; CONNECT.

The new version introduced a number of critical performance optimizations such as keep alive connections, chunked encoding transfers, byte-range requests, additional caching mechanisms, transfer encodings, and request pipelining. [5.]

2.4 Network Security Protocols

There are several network security protocols such as IP Security Architecture (IPsec), Secure Socket Layer (SSL), Transport Layer Security (TLS) and Secure Shell (SSH). All of those offer a standard set of encryption and hashing algorithms. The main difference is the OSI layer where the algorithms perform.

Encryption algorithm could be symmetric or asymmetric. Symmetric (secret key) cryptography uses the same secret key to encrypt and decrypt the packet. Asymmetric (public key) cryptography uses two keys (public and private). [9.]

2.4.1 Secure Shell SSH

SSH is an application layer solution which typically uses TCP. It is a protocol for secure remote login over an insecure network. It uses public key cryptography to prove the authenticity of the remote user. It also provides also some extensible features such as port forwarding and secure tunnelling. SSH is a simple and easy solution for shell access and file transfer. [10.]

2.4.2 Secure Shell SSH and Transport Layer Security TLS

TLS is a successor of SSL which was proprietary to Netscape and it is based on SSLv3. TLS offers privacy and secure data connection between two communicating applications. It consist of two layers; the TLS record protocol and the handshake protocol. It runs directly on top of TCP and it is a layer 4 protocol. TLS is using Public Key Infrastructure (PKI) to provide user authentication. TLS is used for example to secure HTTP sessions, online banking and for Wi-Fi Protected Access II (WPA2) wireless security authentication. [10.]

2.4.3 IP Security Architecture IPsec

IPsec is a set of protocols and algorithms used to secure IP data at the network layer (Layer 3). It defines two main protocols to use; Authentication Header (AH) and Encapsulating Security Payload (ESP). IPsec offers confidentiality (encrypting data), integrity and authentication and it supports transport or tunnel mode. Today most Virtual Private Networks (VPN) are using IPsec. [10.]

2.4.4 Hypertext Transfer Protocol Secure HTTPS

HTTPS is used as a security version of HTTP. It offers encrypted traffic between the browser and the server. There are two cryptographic protocols which can be used for the implementation; Secure Socket Layer (SSL) and Transport Layer Security (TLS). Today HTTPS is commonly used to authenticate web pages and to provide secure client server connection [11, 282].

TLS client initiates a connection to the server using a certain port number and sends the TLS ClientHello message to begin the TLS handshake. After finishing the handshake the client is able to initiate the first HTTP request message [12].

2.5 Main Difference between HTTP/1.0 and HTTP/1.1

In most cases HTTP uses TCP as its transport protocol. In HTTP/1.0 the client needs to establish a new TCP connection for every request. The objects which need to be downloaded from the server are often quite small and the number of objects is quite high. So it means that the client needs to establish and terminate a huge number of short TCP connections which wastes time and that means longer page load time. Because there is a so called slow start effect with TCP connection it means that it is nearly impossible to utilize the whole link capacity and it means HTTP/1.0 is quite ineffective especially if the downloadable page consist many small objects. [13.]

Figure 5 below illustrates the main differences between HTTP/1.0 and HTTP/1.1 concerning TCP connection [14].

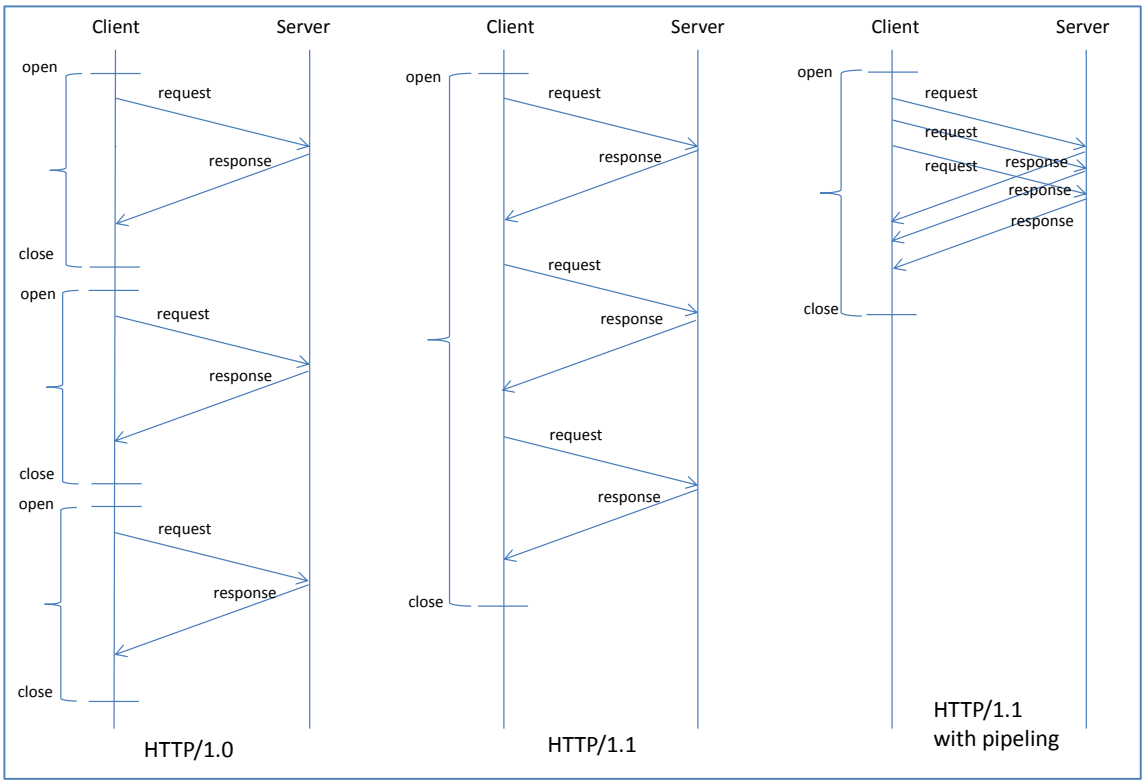

Figure 5. Main difference between HTTP/1.0 and HTTP/1.1. Modified from Difference between HTTP pipeling and HTTP multiplexing with SPDY [14]

From the page download time point of view HTTP/1.1 presents two main features, a persistent connection and HTTP pipelining. Persistent connection means that within one TCP connection the client is able to handle many request and response messages. HTTP pipelining means that the client is able to send multiple request without receiving a response. However the server must send responses in the same order as it has received the requests from the client. As shown in Figure 6 this is not an optimal solution in some situations [14.]

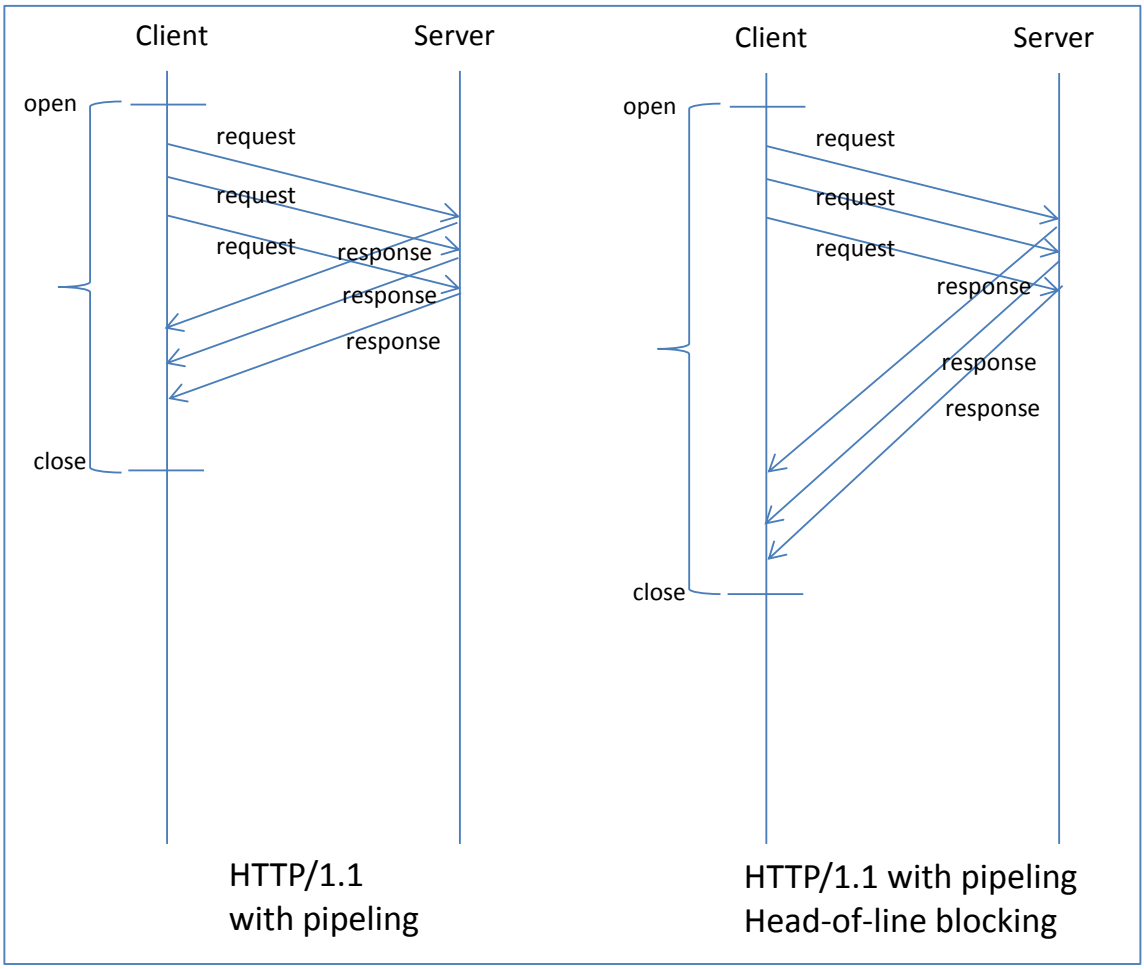

Figure 6. Main difference between HTTP/1.1 and HTTP/1.1 with pipeling. Modified from Difference between HTTP pipeling and HTTP multiplexing with speedy SPDY [14]

The possible problem is called TCP head-of-line (HOL) blocking and it means following. If the client is sending to the server many requests over the same TCP connection and if the first response is huge in content length and the second response is small in content length the problem occurs.

Now because of the HTTP/1.1 protocol implementation the second response must wait for the first response to complete. In other words the second response is head-of-line blocked by the first response as seen in Figure 6.

2.6 Caching

In simple terms caching is a method to temporary store recently used information. For the end user it means faster access when reducing the response time and in general more efficiency network. For the end user the cache/proxy servers are invisible, just the browser needs to know the cache/proxy server IP address.

Especially in wireless networks the operators need to add bandwidth to meet the continuous growing data demand. The new bandwidth is an answer for coverage and capacity problems, but it doesn't help for the low latency. There are no industry standards for web page response time but already seven years ago Akamai published a study which indicates that 47 percent of consumers expect an ecommerce page to load in two seconds or less [15].

With wireline networks the main way to reduce the response time has been the usage of content delivery networks (CDNs) and caching. Both above-mentioned are also possible to use with wireless networks but let us first concentrate on caching. From caching point of view LTE network can be divided into two parts; backhaul network and core network. Backhaul part is calling as Layer 1 (L1) and core part is calling Layer 2 (L2) caching as seen in Figure 7. [16.]

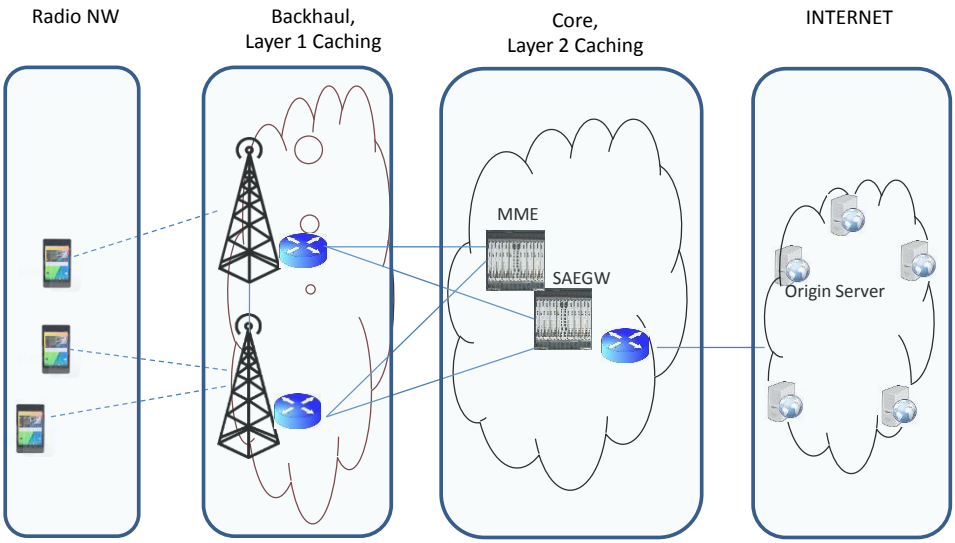

Figure 7 Layer 1 and Layer 3 Caching in wireless network. Modified from Layer 1 and Layer 3 Caching Locations in a Wireless Network [16]

In LTE networks the L2 caching is deployed in the evolved packet core (EPC). These cache servers could offer huge data storage (multi-terabytes). The advantage of centralized caching is reduced Internet latency and remote server response times. Same time also the cost of the Internet connection is lower because of lower bandwidth need. Connections between LTE base station and EPC could be quite costly. If the L1 caching is built in the base stations it will reduce the need of the transport capacity and reduce also the connections cost. L1 and L2 caching together with sophisticated caching algorithms are the solution to reduce the traffic and lower the cost. [16.]

According to Wireless 20|20 Whitepaper [16] the caching in mobile networks can be divided in L1 and L2 caching. The goals of L1 caching are shown in Table 3 and the goals of L2 caching are shown in Table 4.

Table 3. The coals of L1 caching. Data gathered from Sarkissian H [16]

The goals of L1 caching
Bring the popular content closer to the end user, the closer the better
Reduce the overall backhaul traffic
Improve network efficiencies

L1 caching also named as caching at the edge of the network is more effective the closer it will be done the end user. At this point of view it should be done in the base station.

That is also a way to save the backhaul and the transport costs.

Table 4. The goals of L2 caching. Data gathered from Sarkissian H [16]

The goals of L2 caching
More centralized caching meaning also high hit ratios
Reduce the uncertainties of Internet latency
Reduce remote servers response times

Cache hit means that the requested data can be found in the cache and it needs not to be read from a slower data store. The higher the ratio is, the more effective the cache is in improving the performance. [16.]

From the end user point of view it is worth to remember that increasing latency will destroy the benefit of increased bandwidth. So far the caching has been focusing mainly on large HTTP objects. It has a clear impact for bandwidth savings but not so clear impact for the end user QoE. That is because of the current web page structure. Especially with the mobile users the small object caching is improving the end user performance. More detailed information can be found in Chapter 2.13.2 Web Page Structure.

2.7 Speedy Protocol SPDY

Because the web page structure started to be more and more complex and dynamic there was a degrading performance especially with page download time. In 2009 Google introduced a new experimental protocol for faster web. They started to call it speedy (SPDY). Like HTTP, SPDY is a client-server, request-response protocol. Google Chrome was the first SPDY capable browser and the server side code was open source [17].

Google was not the only one to have proposals for improvements. For example Microsoft's Speed+Mobility was one possibility [18]. Clearly Google's SPDY solution has some clear improvements to compare with HTTP/1.1 because it was chosen as a basis of the HTTP/2 technical specification [19]. Continuous SPDY development makes it possible for HTTP/2 to adopt new features quickly and safely. Thus SPDY offers a way to test and evaluate new proposals before adding those in the HTTP/2 standard. There was also some critique against SPDY because it was not an IETF standard and it required Secure Sockets Layer (SSL) which means that SPDY always requires an additional SSL

handshake time. Also at the very beginning only Google Chrome browser was supporting SPDY [20]. The latest SPDY protocol specification is SPDY Protocol - Draft 3.2 [21].

2.7.1 Goals for SPDY

The main idea of the SPDY project was to develop and implement an application-layer web protocol which will reduce the latency. The main targets are shown in Table 5 [17.]

Table 5. The main goals for SPDY. Data gathered from Google [17]

The main goals for SPDY	
Reduce the web page down load time.	The original target was to reduce the page down load time by 50%.
No need to change the existing network infrastructure.	The underlying protocol remains TCP as with HTTP.
Open source protocol.	Bring together developers who has ideas to improve the page load time.
No need to change the existing website content.	The only change will be in the server applications and the client user agent.

Because SPDY was open source protocol it enables to utilize code, test results and possible new ideas also from other developers.

2.7.2 SPDY Features

SPDY features can be divided into basic and advanced features. [17.] [18.]
Table 6 presents the SPDY basic features.

Table 6. SPDY basic features

SPDY basic features	
Feature	Description
Multiplexing	SPDY allows an unlimited number of HTTP requests and responses (SPDY streams) per single TCP connection. Streams are multiplexed as seen in Figure 8.
Request prioritization	Client can specify a priority level for each object to download. Server is using that prioritization when scheduling the object transfer. This feature prevents the pending of high priority request –response, like Java Script.
HTTP header compression	SPDY compresses all header data, it means fewer packets and fewer bytes to transmit.
Universal encryption	SPDY offers better security because it is negotiated over SSL.

It is important to remember that SPDY does not replace HTTP protocol. It is still using the basic request-response structure and the HTTP headers and methods.

In addition to the basic features SPDY has also few advanced features. Server push feature sounds effective but sometimes it can have a negative impact for terminal's battery consumption. Table 7 presents the SPDY advanced features.

Table 7. SPDY advanced features

SPDY advanced features	
Feature	Description
Server push	The SPDY server is pushing a resource to the client before the client has asked for it.
Server hint	The SPDY server is suggesting the client to ask for specific resources. Server is not sending before request from the client has arrived.

SPDY offers also so called server-initiated streams. It means the server is able to offer some content to the client without the client request.

As mentioned in Table 6 SPDY allows an unlimited number of HTTP requests and responses (SPDY streams) per single TCP connection. The new feature is that now the Streams are multiplexed as seen in Figure 8.

As mentioned in Chapter 2.5 TCP head-of-line (HOL) blocking could exist with HTTP/1.1. Because SPDY is multiplexed protocol it means that this type of HOL does not exist. As seen in Figure 8 the second and third small responses could arrive to the client before the first larger response.

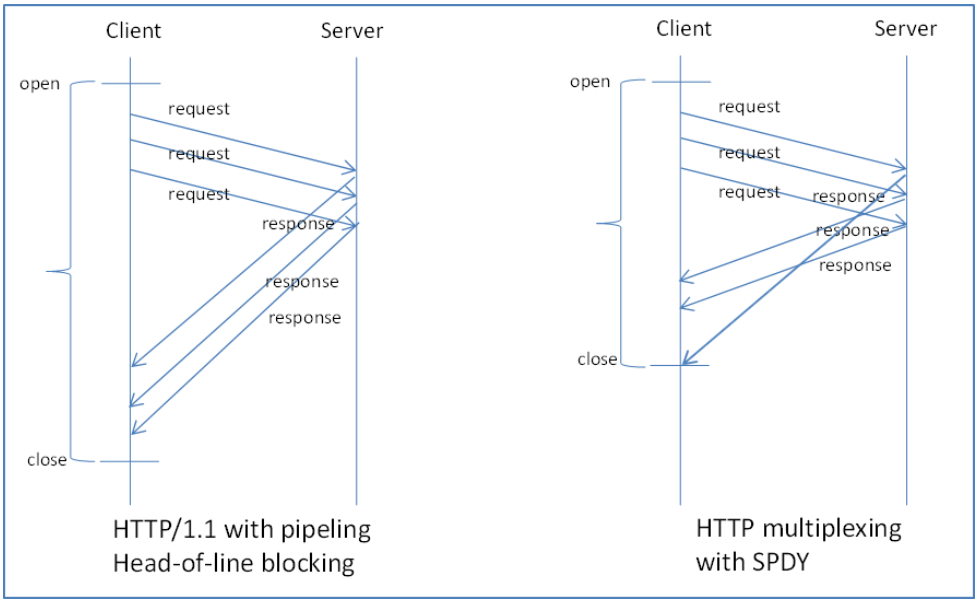

Figure 8. Main difference between HTTP/1.1 and SPDY. Modified from Difference between HTTP pipeling and HTTP multiplexing with SPDY [14].

With SPDY also the feature request prioritization could help to avoid the possible HOL blocking. Client can hint the server which request should have higher priority.

2.7.3 SPDY Protocol Stack

To compare normal HTTP protocol stack to SPDY stack the difference is new SPDY layer which is integrated as a session layer between HTTP and TCP. SPDY is fully compatible with HTTP. SPDY is using SSL connection which is offering better security. Figure 9 illustrates the differences between the HTTP and the SPDY stack [17].

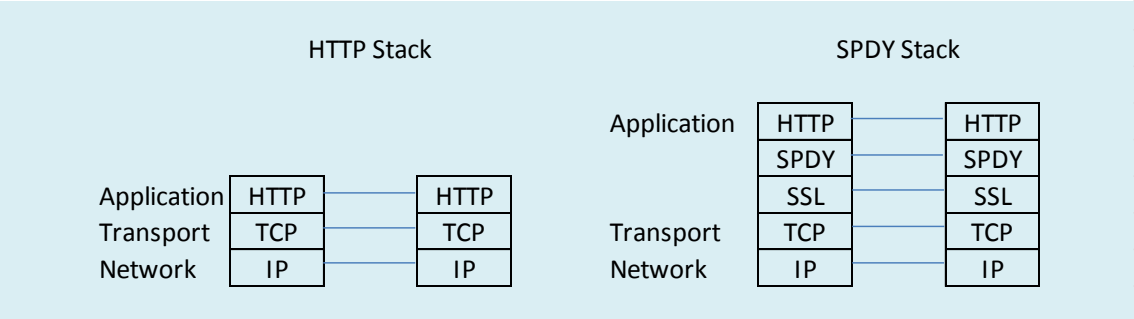

Figure 9. HTTP and SPDY Protocol Stack. Modified from The Chromium Projects, SPDY design and features [17].

As seen in the above figure there is a new SPDY layer which enables multiple concurrent data streams within a single TCP connection.

2.8 Cache/Proxy Server

Today most of the web servers are still supporting mainly HTTP/1.1. It is still possible to utilize SPDY features at least partly. That is the way how for example Google, Facebook and Twitter have used SPDY. The solution is SPDY-capable proxies. [22.] The traffic between the clients and the proxies, often named “last mile”, is using SPDY and the traffic between the proxies and the origin servers is using HTTP/1.x. This kind of a solution is useful for example in mobile environments where the round-trip times (RTTs) are quite high. In some cases also the secure connection between the client and SPDY proxy is important. The basic structure concerning the proxy and the origin servers is shown in Figure 10 [23].

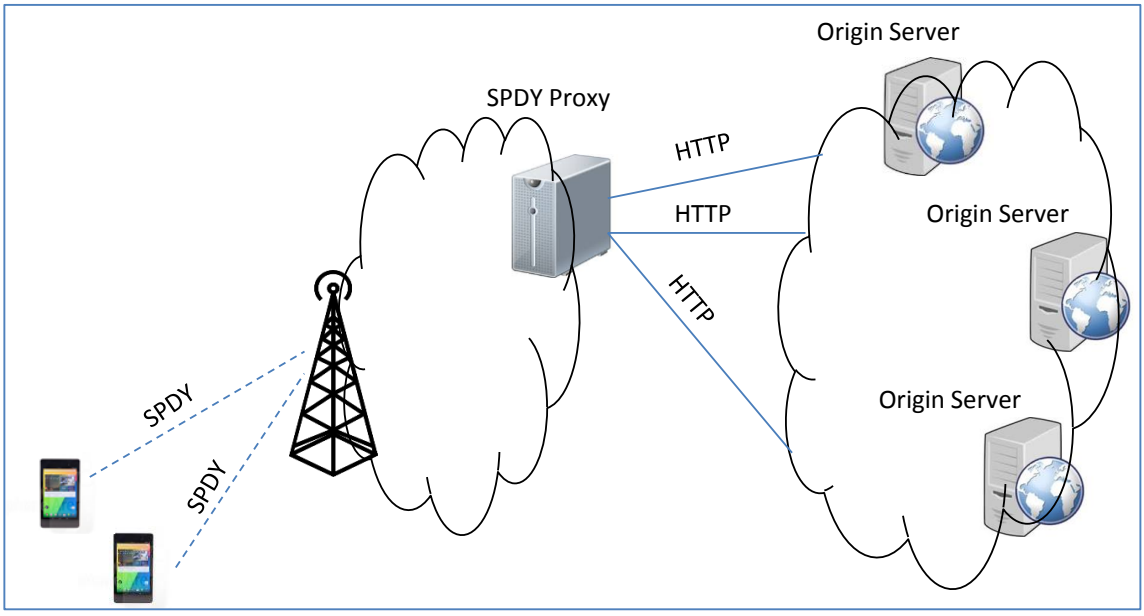

Figure 10. SPDY proxy usage in mobile environment. Modified from improving the performance of SPDY for mobile devices [23].

Configuration as seen in Figure 10 is fast to implement. SPDY support is needed from the client and proxy, rest of the network may remain unchanged.

2.8.1 Implementing SPDY Proxy

There is a need for some changes before it is possible to utilize SPDY proxy. Both the browser and proxy need to support the SPDY protocol. Also support of SSL between the browser and the proxy is needed. Before the browser is able to choose the appropriate proxy server a proxy auto-configuration (PAC) file needs to be modified.
A PAC file is a text file including at least one JavaScript function (FindProxyForURL(url, host). Depending on the value returned by the function the browser will use certain proxy server or direct connection to the content server. [22.]

2.9 HTTP/2

HTTP/2 is the next major revision of the official HTTP network protocol. In May 2015 the new standard was defined in RFC 7540 [24]. It is backwards compatible with HTTP/1.x, so it means that all previous HTTP methods, status codes, header fields, Uniform Resource Identifiers (URIs) and semantics remain unchanged. Actually there was also another new standard published at the same time. It is for header compression and defined in RFC 7541 [25].

As mentioned earlier SPDY was a so called coevolution for HTTP/2. It means that the main new features were tested with SPDY and the implementation of HTTP/2 should be fast and easy. Both protocols support binary framing layer which is the main idea for the performance enhancement. It defines the encapsulation and transferring of HTTP messages between the client and server. [5, Chapter 12.]. One important difference between SPDY and HTTP/2 is header compression. SDPY uses DEFLATE format and HTTP/2 uses HPACK [26]. SPDY requires the use of SSL, but HTTP/2 does not require the use of encryption. Currently all browsers are using HTTP/2 with encryption [27].

2.10 Usage of SPDY and HTTP/2

World Wide Web Technology Surveys (W3Techs) provide different kind of information concerning the web. According to the W3Tech the usage of SPDY or HTTP/2 protocols is not wide yet. According to the survey HTTP/2 is used by 2.9% of all the websites. Respectively SPDY is used by 6.2% of all the websites (situation 7th December) [28]. These statistics do not tell the whole truth. There could be much more transactions using SPDY or HTTP/2 because of the proxy usage, as mentioned in Chapter 2.8. Figure 11 below shows the historical trend in the percentage of SPDY and HTTP/2 usage [29].

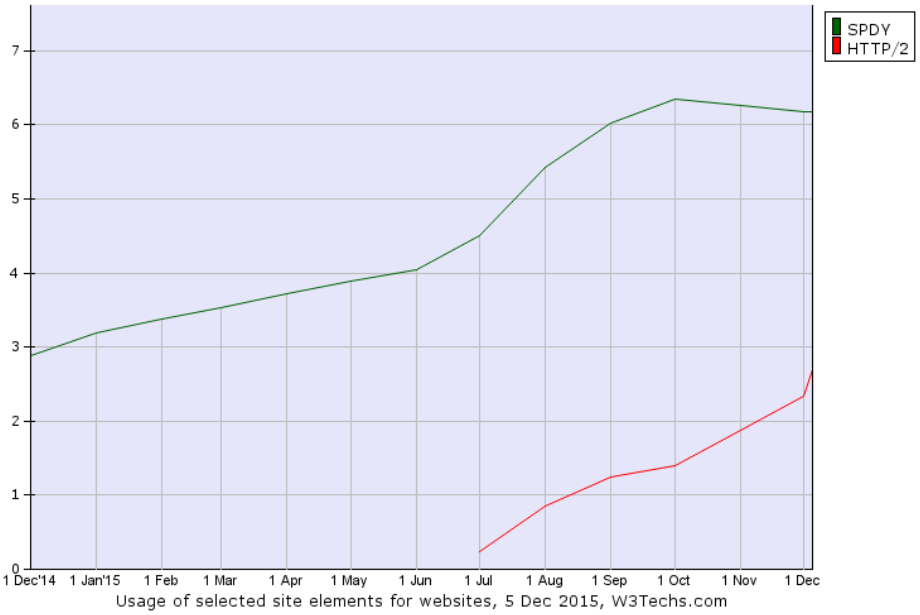

Figure 11. Historical trend in percentage of SPDY and HTTP/2 usage. Copied from W3Techs (2015) [29]

The official HTTP/2 specification was published in May 2015. After the official RFC its usage has start to grown rapidly as seen in Figure 11.

When comparing the usage percentage it seems that SPDY is much more popular with webpages ranked in the top 1 000 000 or lower than HTTP/2 as seen in Figure 12 [29].

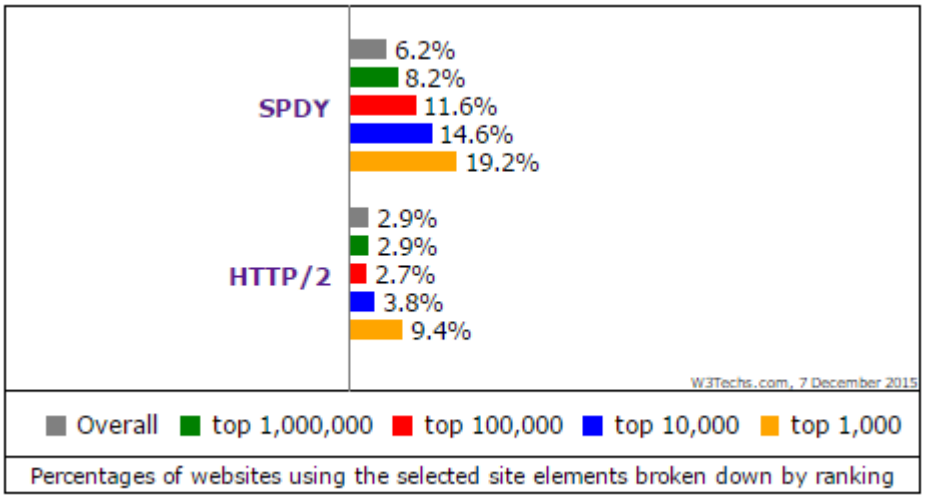

Figure 12. SPDY and HTTP/2 usage broken down by ranking. Copied from W3Techs (2015) [29]

Figure 13 [29] below shows the market position of SPDY and HTTP/2 in terms of popularity and traffic compared to the most popular site elements. As shown in the chart, both protocols are in the area used by high traffic sites. The best place for both protocols would be the upper right-hand side corner.

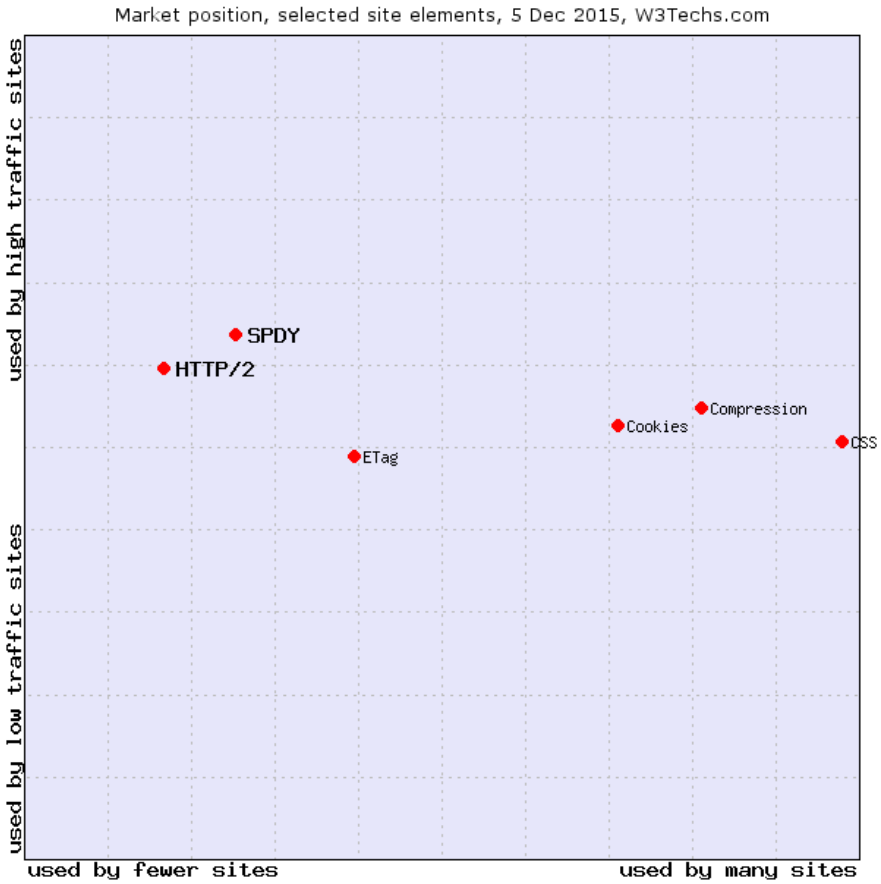

Figure 13. SPDY and HTTP/2 Market position. Copied from W3Techs (2015) [29]

As seen in the above figure Cascading Style Sheet (CSS) is used by more than 90 % of all the web sites. It is middle of the range low traffic and high traffic sites. It is quite obvious because every real web page should include a CSS. SPDY and HTTP/2 are little bit above the average traffic site limit. That is also obvious because SPDY and HTTP/2 should give the biggest benefit when the site traffic is high. The more complicated the site is the more traffic is generated. Thus the place of SPDY and HTTP/2 should be the upper right-hand corner.

2.11 Energy Efficiency

If ten years ago it was enough to charge the mobile device once per week today the need is at least once per day, due to the new features in modern smart devices. So also from the energy efficiency point of view it is very important how effective the mobile web is. When download times are decreasing it will increase also the end user Quality of Experience (QoE). New smart phones include many sensors and different I/O components, such as camera, GPS, SD card, etc. This means that the phones have to offer more computational power to be able to run the needed applications fast enough. Together all this means increased power consumption and Internet access with a smart phone is one of the most energy consumed use case [20]. Long battery life seems to be one of the most important features when people are buying a new smart phone [30].

According to Chowdhury et al. [20.] HTTP/2 is more effective in energy savings as compared to HTTP/1.1 when RTT between the client and server is above 30ms and TSL is in use. The study was made comparing HTTP/2 and HTTP/1.1 but because HTTP/2 is based on SPDY it should also be more energy efficient than HTTP/1.1. The main reason is the large number of TCP connections when using HTTP/1.1. The difference is clearer when RTT and the number of objects per page is increasing. The study also observes that the mobile clients have an increasing energy consumption when using HTTPS. On the other hand HTTPS is offering user's privacy and security through encryption.
The above-mentioned study does not take a position on the server site power consumption which will be more and more important within operators with big data centres.

2.12 Future of Network Energy

Energy consumption in general has heavily increased since the industrial revolution and it will continue to increase. This trend also involves the information and communications technologies (ICT) sector. As mentioned earlier the rapid adoption of smart phones and tablets is increasing the internet usage heavily.

Bell Labs has estimated that the bytes per day traffic in mobiles networks will increase more than a hundred times in the time period from 2015 to 2025. Even the conservative estimate predicts about 60 times higher traffic volumes in 2025 compared to 2015 [31]. The traffic growth forecast prepared by Bell Labs is shown in Figure 14.

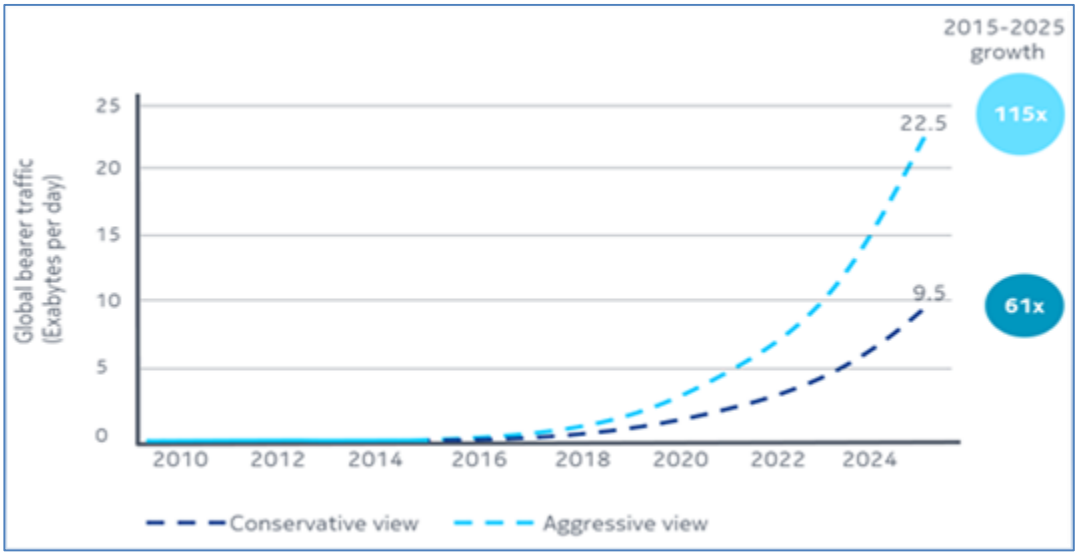

Figure 14. Bell Labs estimated traffic growth from 2015 to 2025. Copied from [31]

Also according to the Bell Labs's forecast the Internet traffic will increase up to 85 times by 2017, as compared to 2010 [31, 442].

Figure 15 [31, 443] illustrates the trend of the annual energy cost of the global telecommunication networks between 2011 and 2025. It looks quite alarming, the global energy cost seems to increase from $40B to $343B annually.

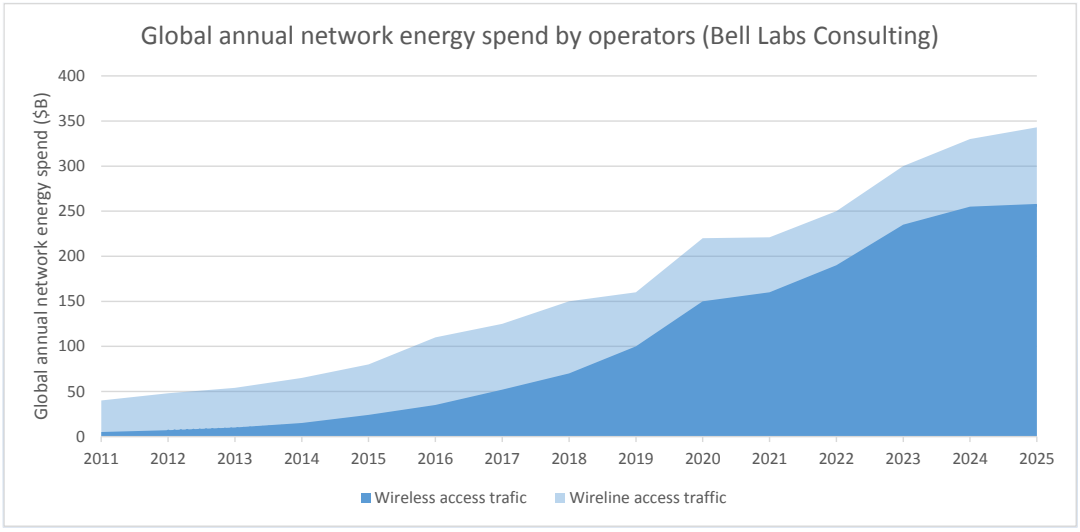

Figure 15 Global annual network energy spend by operators (Bell Labs Consulting). Copied from [31, 443]

Investigating the energy efficiency as such was not, however, within the scope of this study.

2.13 Impacting Items on Web Page Load Time

This chapter introduces the different features affecting the web page load time. There are two main aspects, the content type and the page structure.

2.13.1 Web Page Content Type

As mentioned earlier in Chapter 2, Overview of HTTP Browsing, the web page structure has changed radically. Earlier the web page included more or less pure text but today the page consists of also many other content types. According to the latest information (from December 2015, average bytes of all Uniform Resource Locators (URLs)) the average web page size is about 2.2 MB.

Figure 16 illustrates the different content types and their share in the case of all web pages.
Figure 17 does so respectively in the case of top 100 web pages. [32.]

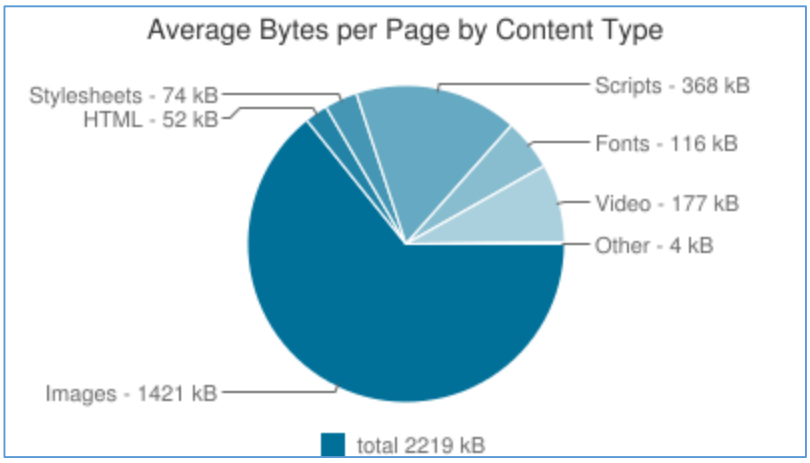

Figure 16. Average bytes per page by content type. Copied from HTTP archive, URL=All (2015) [32]

The share of images, also called small objects, is about 65% of the page whole size. It is by far the biggest part. [32.]

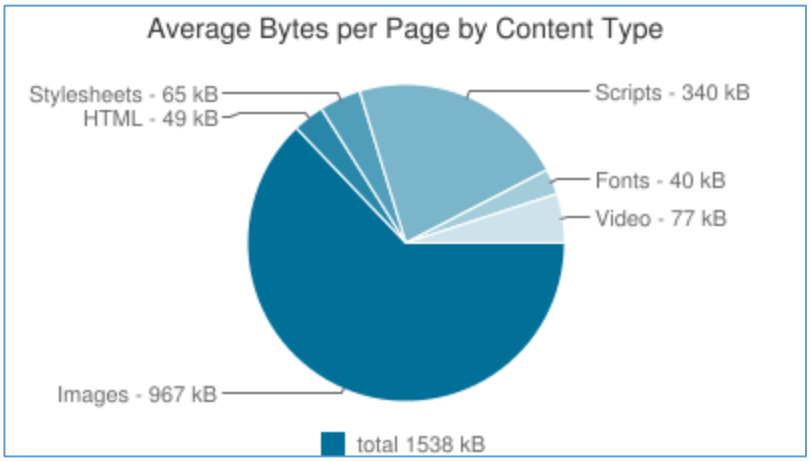

Figure 17. Average bytes per page by content type. Copied from HTTP archive, URL=Top 100 (2015) [32]

Also in this case the share of images is about 65% of the page whole size. Unfortunately HTTP/1.x was not particularly designed to handle big numbers of different objects. Table 8 presents some of the limitations [33]

Table 8. Some of the HTTP/1.x limitations

Some HTTP/1.x limitations
Browser open too many TCP connections
No request prioritization
No header compression to reduce overhead

During the period from July 2009 to July 2014, the size of the average web page (top 1000 pages) has roughly tripled, and the number of external objects has roughly doubled. The average web page size has grown from 600 KB to 1622 KB. During the same period, the number of objects per web page has grown from 70 to 112 as seen in Figure 18. [4.] So the big question is how to handle the increasing number of images (small objects) per web page as effective as possible.

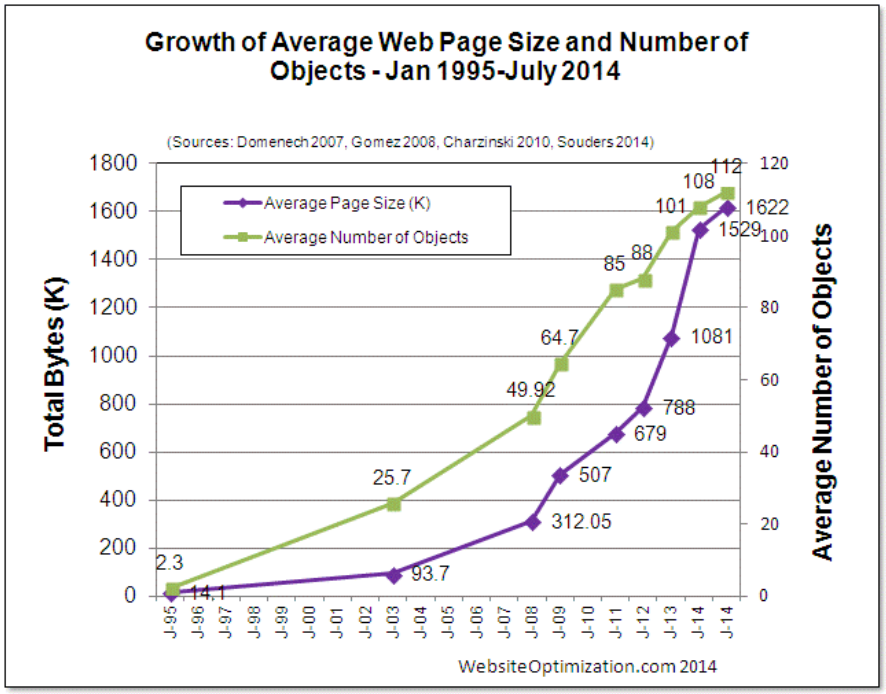

Figure 18. Average Web Page Breaks 1600K. Copied from Web Site Optimization.com (2014) [4]

The growth of the web page size was quite constant until the year 2007 but after that the growth has been exponential fast.

2.13.2 Web Page Structure

There are also some other impacts than the used HTTP protocol, network latency or network bandwidth which affect the web page load time. Application Performance Management digest (APMdigest) have collected a list consisting of 20 top factors that impact website response time. The list includes opinions asked from industry experts, for example, analysts, consultants or top vendors. It is worth stressing that this list includes only the opinions concerning the most important factors according to the informants. There are also some not so technical aspects which one might not even have thought about. The list of factors that impact website response time are shown in Table 9. [34.]

Table 9. 20 Top Factors That Impact Website Response Time. Data gathered from APMdigest (2015) [34]

20 Top Factors That Impact Website Response Time
COMPLEXITY
INTERDEPENDENCIES
CONFIGURATION AND COMMUNICATION OF COMPONENTS
LATENCY
DEMAND PEAKS
WEB PAGE SIZE
RESPONSIVE WEB DESIGN
JAVASCRIPT
WEB CONTENT AND CODE
N+1 QUERIES
THE BACK-END SERVER
THE DATABASE
THIRD PARTY SERVICES
THE NETWORK
VIRTUALIZATION
IT INFRASTRUCTURE CHANGES
ALTERED CODE
DISTRIBUTED DENIAL OF SERVICE ATTACKS (DDOS)
INFORMATION ARCHITECTURE
THE IT TEAM

Some of the above mentioned factors are not so self-evident truth. A deeper explanation is available for a few factors.

James Wylie, the director of technical product marketing from Covil has said the following concerning Demand peaks:

> "Infrastructure must be scaled to handle peak rates rather than average rates. Peaks in demand may only last for a short time, sometimes only milliseconds but they have a much longer lasting effect, impacting not only the web server and supporting systems but more importantly user experience" [34].

Trevor Parsons, a chief scientist from Logentries has said the following concerning N+1 Queries:

> "N+1 queries are a common web application anti-pattern that can cause slow website response time. It happens when a single application query

> runs other queries “automatically” - resulting in multiple queries running at once” [34].

Matt Watson, the founder and CE from Stackify has said the following concerning the database:

> “One of the top factors that we've seen in our analysis of hundreds of applications is interactions with databases; either slow database queries or too many database queries being executed per web request. As companies analyze how their code spends it's time, it is essential to see how often queries are called and how long they take to run” [34].

Some areas are more sensitive to the page load time than others. Sometimes even a few second delay is too much. One example is ecommerce. Some online shoppers even cancel the web page load after waiting three seconds.

2.14 Previous Speedy SPDY Research

There are many studies available concerning the performance difference of HTTP and SPDY. Most of the studies compare the page download time. However it is quite difficult to compare the results because of different test environment, different page structure and different SPDY features in use.
According the chromium projects pages when comparing HTTP and SPDY performance, SPDY is offering about 30 to 60% reductions in page load times [17].

As mentioned above SPDY has many different features. Some of those should improve the performance clearly such as the basic features multiplexing and header compression. Some of the advanced features are more dependent on the situation. One of them is server push. It could be problematic especially for mobile devices because it could waste the battery and the bandwidth [23].

According to the study How Speedy is Speedy from 11th USENIX Symposium on Networked Systems Design it was mentioned that sometimes SPDY can help and sometimes hurt the page load times. Most of the performance impacts rest on the use of a single TCP connection. At the same time if the network packet loss is high a set of connections seems to perform better. [33.]

3 Test Methodology and Lab Setup

This study concentrates on comparing the performance of HTTP and SPDY on the 4G (LTE) network. The main object of interest was the page down load time and its dependence on variables presented in Table 10.

Table 10. Used variables inspecting web page down load time

Used variables inspecting web page down load time
The number of small objects on the page
The size of the small objects
The delay between the proxy server and the content server
The bandwidth available between eNB and core network (S1 link)
The load in the air interface

In practise, this study began on the construction of the test environment. Partly it was possible to utilize an existing test laboratory configuration or ever, all servers and test mobiles need to be configured. Used LTE base station (eNB) was reserved only for these tests, so it was possible to control the network during the test period. Core network elements were commonly used in the test network. Because all network elements, including the content server, were in our own test laboratory network there were not big variation concerning the page download times. So it means that the load time accuracy was good enough even with a smaller amount of repetition. Figure 19 provides an overview of the used test setup.

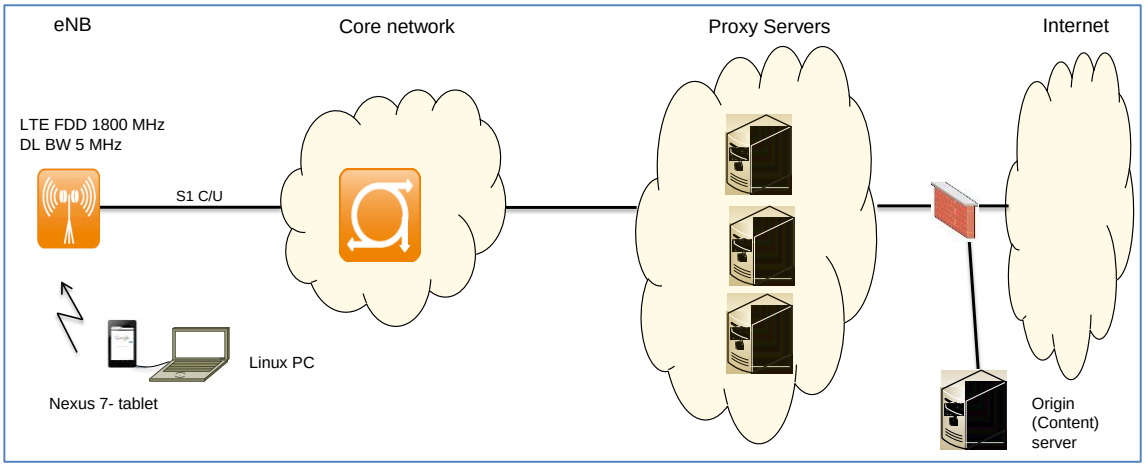

Figure 19. Used test setup

Content (origin) server was also part of the lab network. Another option would have been to use some commercial server that is connected to the Internet. Own dedicated content server ensured that it was possible to handle the delays in a controlled way.

3.1 Test Configuration

This chapter introduces the different elements of the lab network such as the client, the base station and the proxy servers. Moreover the generated web pages and tests executions are covered.

3.1.1 Client

The used client was Nexus 7 tablet with Android version 4.4.2. Chrome version was 38.0.2125.114. As mentioned in Chapter 2.8.1 a so called PAC file was used to tell the client if the proxy server was used. A PAC file is a text file including at least one JavaScript function (FindProxyForURL(url, host). Depending on the value returned by the function the browser will use a certain proxy server or direct connection to the content server. Normally Nexus 7 tablet does not support the use of PAC files. The tablet's bootloader needs to be unlocked and the root privileges need to be allowed. A Linux PC with Python3 and Android Software Development Kit (SDK) is needed to control the Nexus tablet. PC and the tablet were connected together with Universal Serial Bus (USB) cable. Android SDK enables developers to create different applications for the Android platform. Android Debug Bridge (ADB) is part of SDK and is used as a command line tool.

3.1.2 LTE Base Station eNB

The used eNB was Nokia Flexi Multiradio Base Station running in the 1800 MHz spectrum. The bandwidth was 5 MHz and antenna configuration was a 2x2 closed loop multiple input and multiple output (MIMO). LTE with MIMO includes spatial multiplexing as well as pre-coding and transmit diversity [36, 80]. As mentioned in Chapter 3 the eNB was reserved only for these tests. Thus it was difficult to load the base station to find possible differences between the HTTP and SPDY. To get even some experience the eNB parameter "Enable reduced bandwidth DL" and "Maximum number of Physical Resource Blocks (PRBs) assigned in downlink" was used. This restriction is per user equipment (UE). With 5 MHz bandwidth the maximum number of PRBs is 25. Some test cases

were done using lower number of PRBs. Figure 20 illustrates the throughput dependency on the number of PRBs in the test network (only one user). Ookla speedtest was used and the eNB bandwidth was 5 MHz as mentioned earlier.

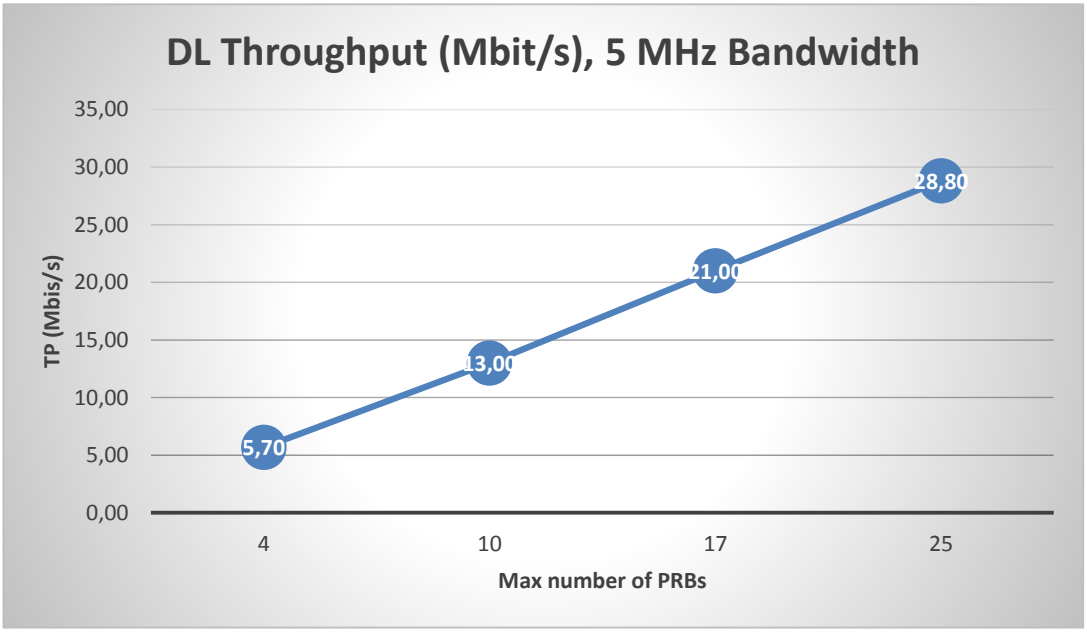

Figure 20. DL Throughput versus Max number of PRBs (S1 capacity 1000 Mbit/s)

As seen in Figure 20, the DL throughput depends on the number of PRBs.

3.1.3 Link Capacity between eNB and the Core Network

Operators have different S1 link capacity (between the eNB and the core network). During the test period it was possible to change the link capacity as presented in Table 11.

Table 11. S1 link capacity eNB and the core network

S1 link capacity
1000 Mbit/s
100 Mbit/s
10 Mbit/s

Figure 21 presents the throughput dependency on the S1 link capacity in the test network (only one user). Ookla speedtest was used and the eNB bandwidth was 5 MHz as mentioned earlier.

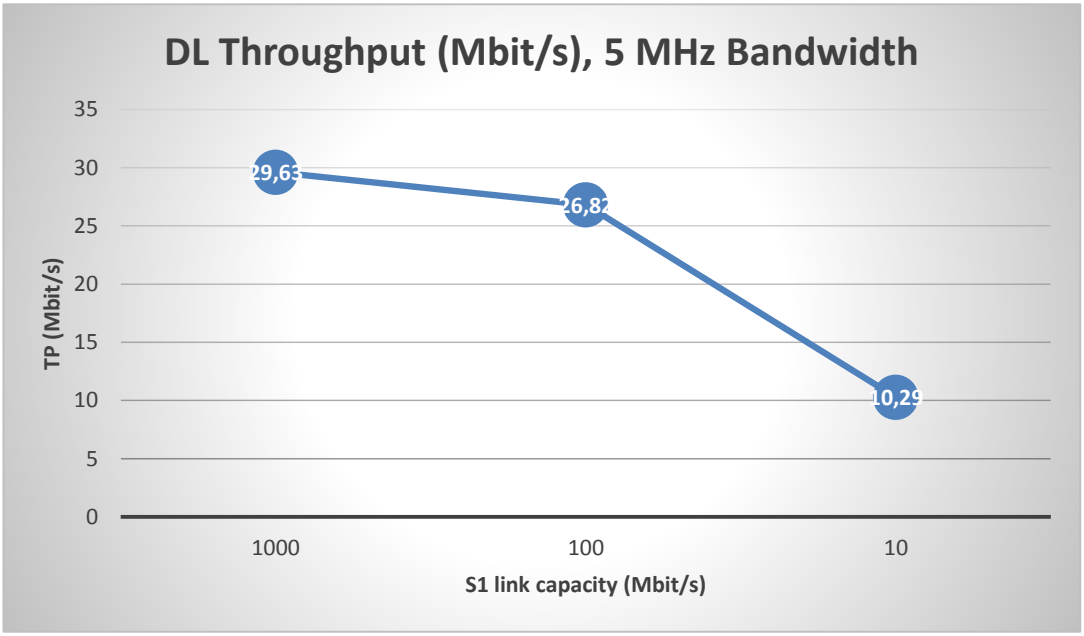

Figure 21. DL Throughput versus S1 link capacity (1000, 100 and 10 Mbit/s)

As seen in Figure 21, the DL throughput depends on the S1 link capacity.

3.1.4 Proxy Servers

It was possible to use two different kind of Proxy/SPDY-servers.
Apache proxy server with mod_spdy. Mod_spdy is an open-source Apache module which enables the activation of SPDY protocol [37] and the Nokia own experimental version based on netty.io framework [38].

The Apache proxy configuration was used to compare mainly HTTP versus SPDY performance. The own experimental proxy was used mainly to compare the caching impact. Investigating the structure of the experimental proxy was not in the scope of this study. As mentioned in Chapter 2.7.2 it is possible to activate many features when using SPDY. To keep the tests more simple only the basic features were activated.

3.1.5 Delay

All proxy servers and the content server were in the same laboratory environment. It means that there was hardly no delay. With the real environment there could be significant delay, so there was a need to simulate some network latency. In the existing Linux distributions there is a kernel component called netem, which adds Network Emulation which is used to add simulated network latency to the Linux-server. The commands to add the delay are shown in Table 12.

Table 12. Used commands to add delay between the proxy and content server

Commands used to add delay
sudo tc qdisc add dev eth1 root netem delay 10ms
sudo tc qdisc add dev eth1 root netem delay 50ms
sudo tc qdisc add dev eth1 root netem delay 100ms

In Finland the LTE network latency is between 20 to 30 ms. In lab environment, the delay is slightly smaller.

3.1.6 Web Page

Test pages were created using a script which automatically creates a page with desired size of small objects and desired number of small objects. The generated pages includes the very basic structure of web page including a variable number of objects which were represented by binary files. Table 13 presents the used web pages where the small objects (SO) number varies from 50 to 500 and the SO size varies from 1 KB to 20 KB.

Table 13. Used web pages

URL	Number of Small Object (SO)	Size of Small Object (SO) (Byte)
http://10.22.198.5/test1-050-1024.html	50	1 024
⋮	100,150, …, 450	1 024
http://10.22.198.5/test1-500-1024.html	500	1 024
http://10.22.198.5/test2-050-5120.html	50	5 120
⋮	100,150, …, 450	5 120
http://10.22.198.5/test2-500-5120.html	500	5 120
http://10.22.198.5/test3-050-10240.html	50	10 240
⋮	100,150, …, 450	10 240
http://10.22.198.5/test3-500-10240.html	500	10 240
http://10.22.198.5/test4-050-20480.html	50	20 480
⋮	100,150, …, 450	20 480
http://10.22.198.5/test4-500-20480.html	500	20 480

As illustrated in Table 13, the script creates altogether 40 web pages.

As mentioned in Figure 18 the average web page size in July 2014 was 1600 KB and the average number of objects in July 2014 was 112. As shown in Table 14 the web page size varied from 50 KB to 10 000 KB (only the small objects calculated) during this test period.

Table 14. Used web pages size variation

Name	SO size (Byte)	Number of SO	Min page size (KB)	Max page size (KB)
test1	1 024	50, 100, 150,…,500	50	500
test2	5 120	50, 100, 150,…,500	250	2 500
test3	10 240	50, 100, 150,…,500	500	5 000
test4	20 480	50, 100, 150,…,500	1 000	10 000

The reason why to choose so many different pages was the target to find a limit when the number of objects or the objects size start to be significant.

3.2 Test Execution

Figure 22 illustrates the proxy servers used. The figure is simplified, without the base station, backbone and core part.

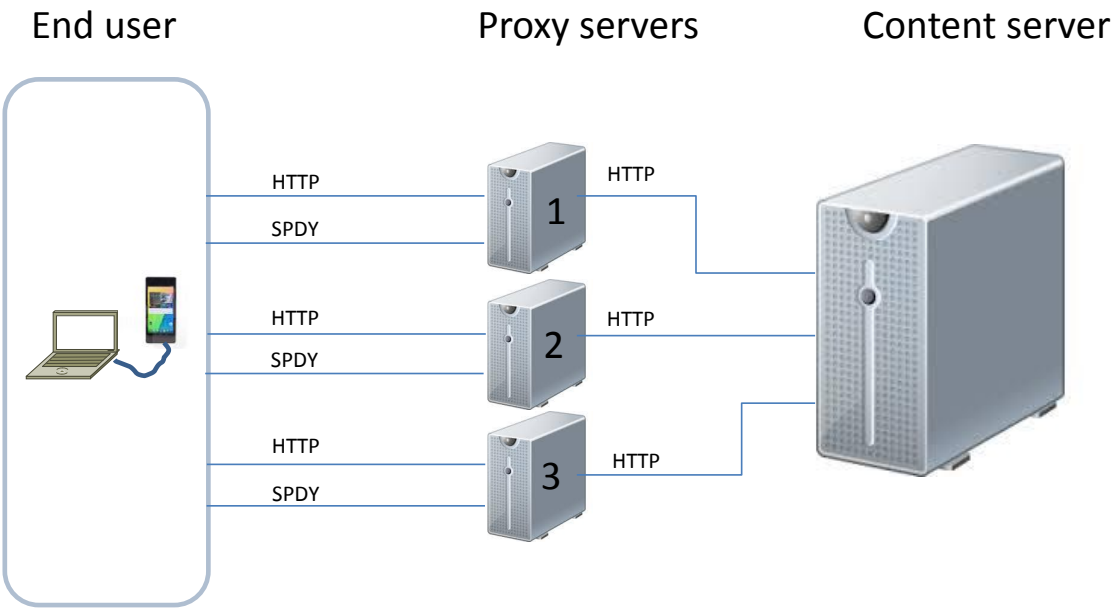

Figure 22. Different proxy servers

Each web page was loaded using six different configurations (cf. Table 15). Abbreviations mentioned in Table 15 are used later in this document.

Table 15. Used proxy server's configuration

Proxy server	Proxy server configuration	Abbreviation
1	Apache proxy server	a_http
1	Apache proxy server with spdy	a_spdy
2	Experimental proxy server	n_http
2	Experimental proxy server with spdy	n_spdy
3	Experimental proxy server with caching	n_http_c
3	Experimental proxy server with spdy and caching	n_spdy_c

All network elements, also the content server, were in the same laboratory network. However, there were some differences concerning the page load times. Therefore each page was loaded ten times and the median value was used.

The execution script also includes the cleaning of the browser's cache memory after each page download. Proxy server caching was used only with cases n_http_c and n_spdy_c. As mentioned in Table 10 the main object of interest was the page down load time and its dependence on certain variables. Table 16 illustrates the different combinations.

Table 16. Different test cases, changes of the amount of PRBs and extra RTTs

Test case	S1 link capacity (Mbit/s)	The amount of Physical Resource Blocks (PRBs)	Extra delay (rtt ms)
4.1, 4.2	1 000	25	10
4.3, 4.4	1 000	25	50
4.5, 4.6	1 000	25	100
4.7, 4.8	1 000	17	10
4.9, 4.10	1 000	4	10

The number of the small objects per web page varies from 50 to 500 (50 steps) and the size of a small object varies from 1 KB to 20 KB (1 KB, 5 KB, 10 KB and 20 KB).
Network delay was quite a small in the laboratory 4G environment, so extra 10 ms delay was added for every test case. When testing the delay effect 50 ms and 100 ms extra delay was added.

4 Results

This chapter introduces the measurement results. Main test categories includes results with the basic settings, with added network delay and with reduced number of physical blocks. Each measurement category includes the page load time comparison between HTTP and SPDY. The comparison is made with and without caching.

4.1 Basic Settings

S1 1000 Mbit/s, prb 25, extra rtt 10 ms, HTTP vs. SPDY

The first case was the basic case with a different number and size of small objects and the goal was to compare SPDY and HTTP. Figures 23 to 26 illustrates the page load time (PLT) with different parameters. Figure 27 shows the positive impact of SPDY as

compared to HTTP. PLT starts to decrease clearly when the small object number is more than 250 and the small object size is 1 KB, 5 KB or 10 KB. When the small object size is 20 KB the PLT decreasing starts after 350 small object per page.

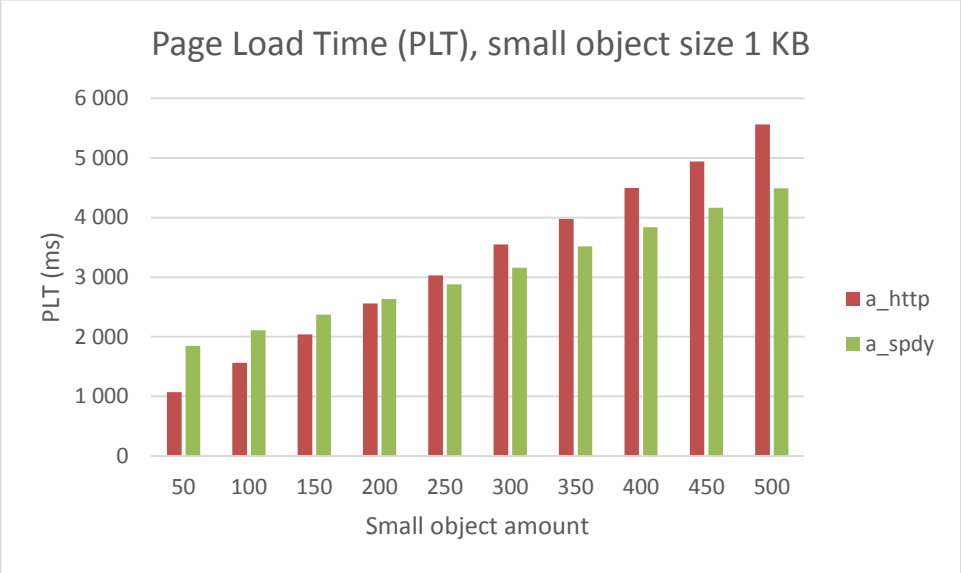

Figure 23. Test case 4.1, small object size 1 KB

SPDY starts to be more effective compared to HTTP when the number of small object is more than 250.

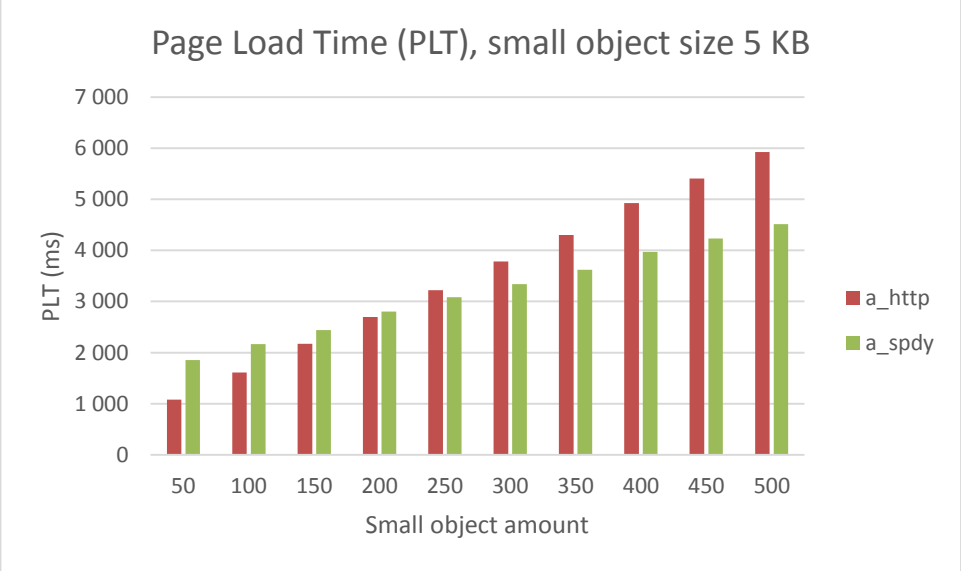

Figure 24. Test case 4.1, small object size 5 KB

SPDY starts to be more effective as compared to HTTP when the number of small objects is more than 250.

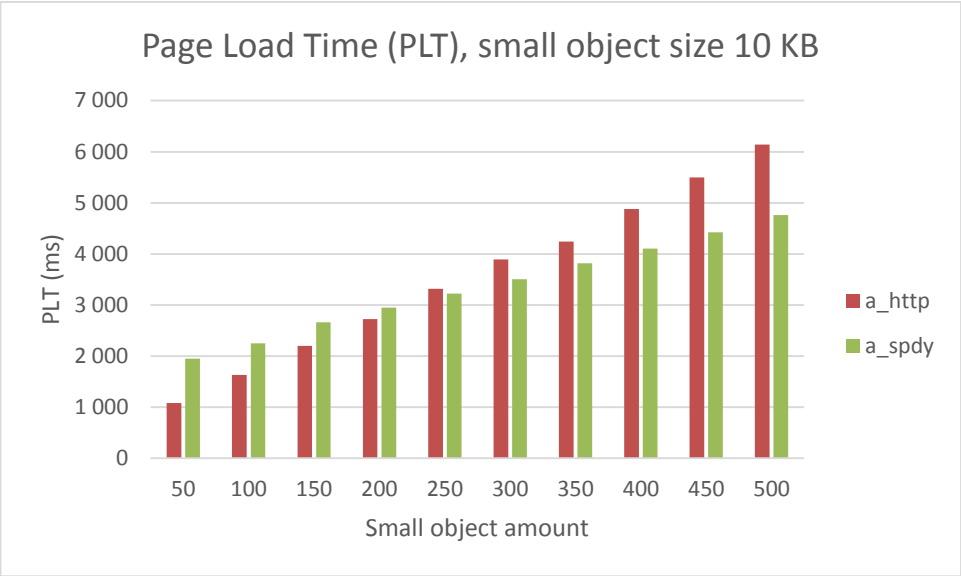

Figure 25. Test case 4.1, small object size 10 KB

SPDY starts to be more effective as compared to HTTP when the number of small objects is more than 250.

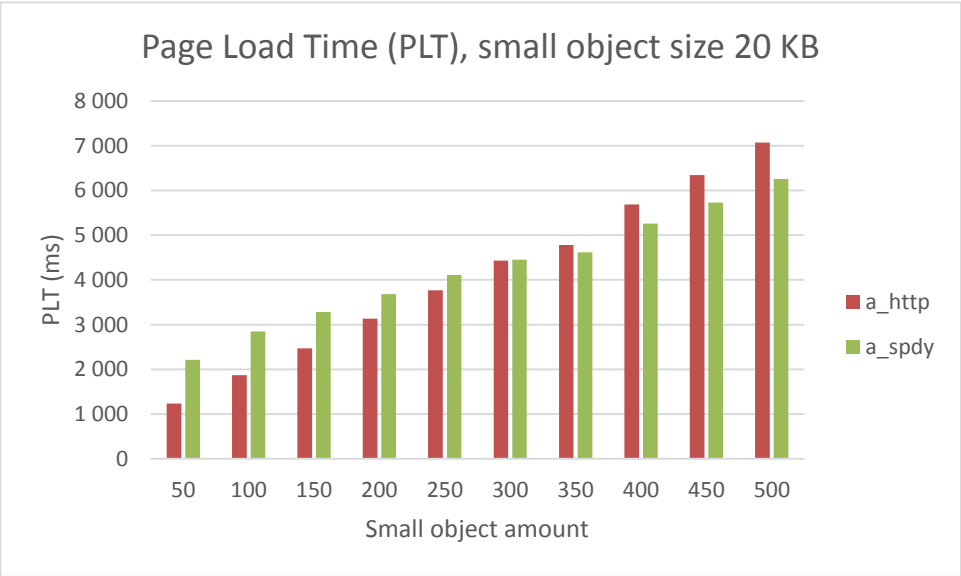

Figure 26. Test case 4.1, small object size 20 KB

SPDY starts to be more effective as compared to HTTP when the number of small object is more than 350.

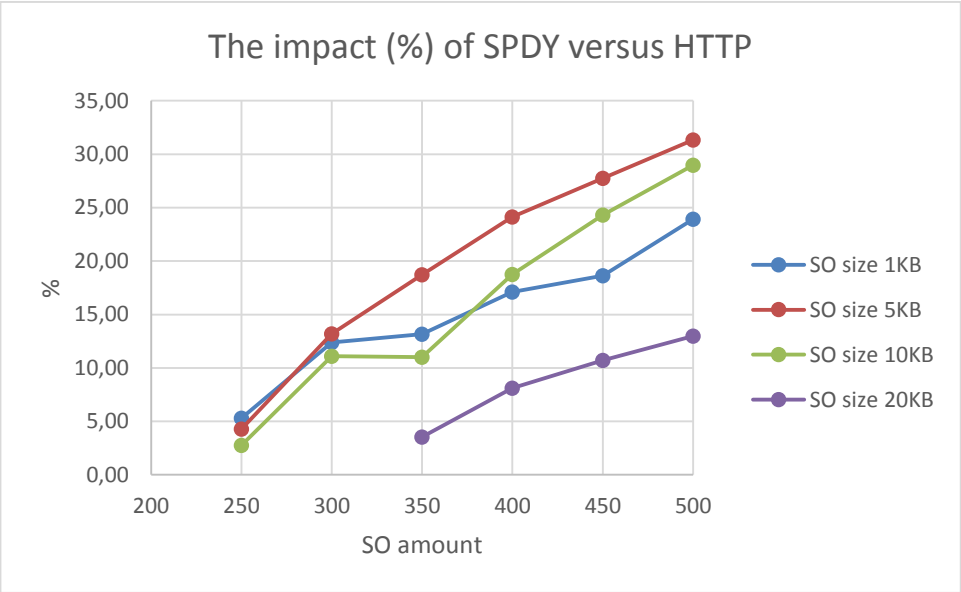

Figure 27. Test case 4.1, the positive impact of SPDY versus HTTP protocol

SPDY is more effective compared to HTTP when the small object number is more than 250. The effect is more positive, the more the small object number increases.
The case with 20 KB small object size was an exception, the positive SPDY impact starts when the small object number is more than 350. The shape of the curves with all measured object sizes looks similar, about linear.

4.2 Basic Settings and Caching

S1 1000 Mbit/s, prb 25, extra rtt 10 ms, caching allowed

The second case was the basic case with a different number and size of small objects and the goal was to investigate the proxy caching effect. Figures 28 to 31 illustrates the page load time (PLT) with different parameters. When comparing HTTP and HTTP with caching the impact was about 20 to 25% and when comparing SPDY and SPDY with caching the impact was about 5%. Figure 32 is showing the positive impact of SPDY with caching compared to HTTP. PLT started to decrease clearly when the small object number was more than 150. The most positive impact was seen when the small object size was 20KB.

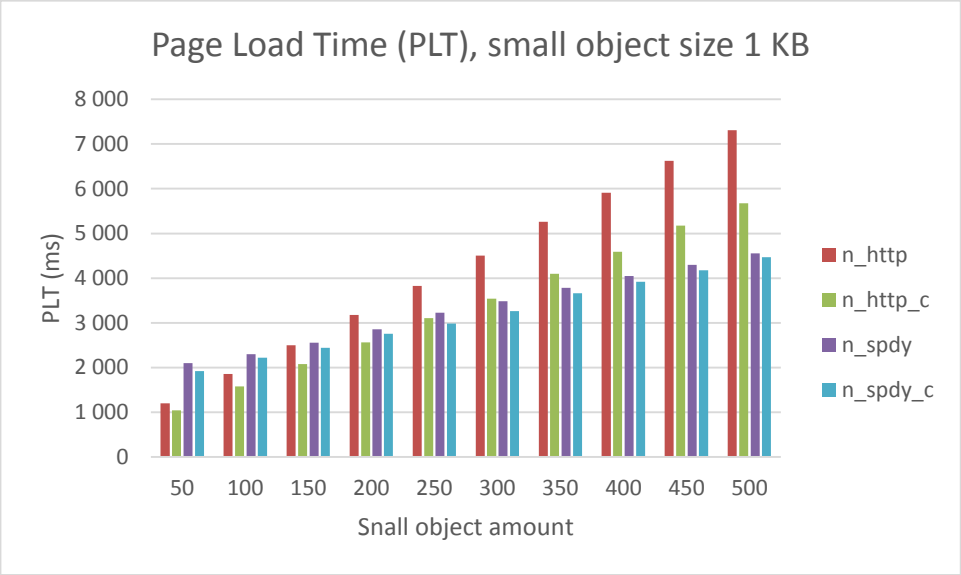

Figure 28. Test case 4.2, small object size 1 KB

PLT with caching starts to be clearly more effective when the number of small object is more than 150. Pure SPDY offers better impact than HTTP with caching when the number of small object is more than 350. The difference between SPDY and SPDY with caching is not significant.

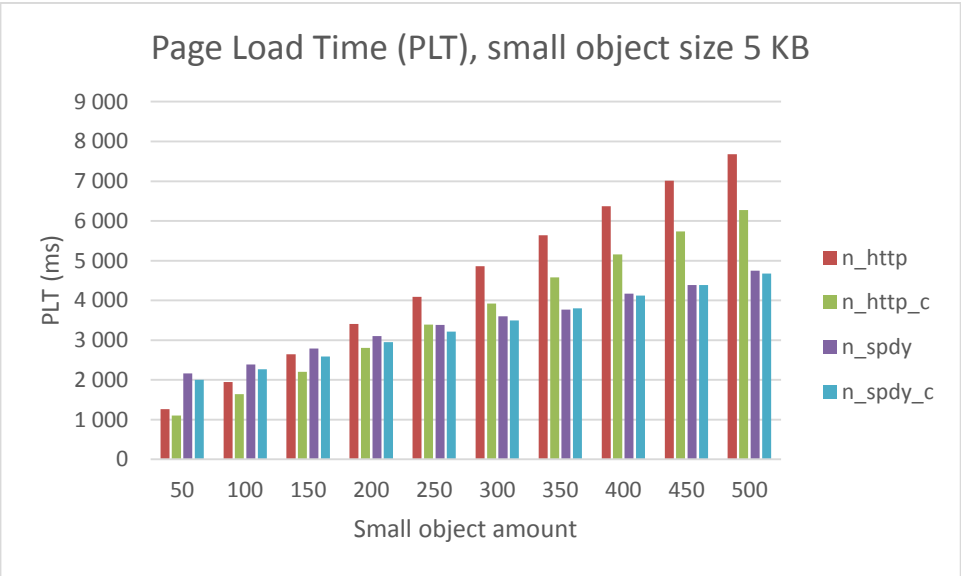

Figure 29. Test case 4.2, small object size 5 KB

PLT with caching starts to be clearly more effective when the number of small object is more than 150. Pure SPDY offers better impact than HTTP with caching when the number of small object is more than 300. The difference between SPDY and SPDY with caching is not significant.

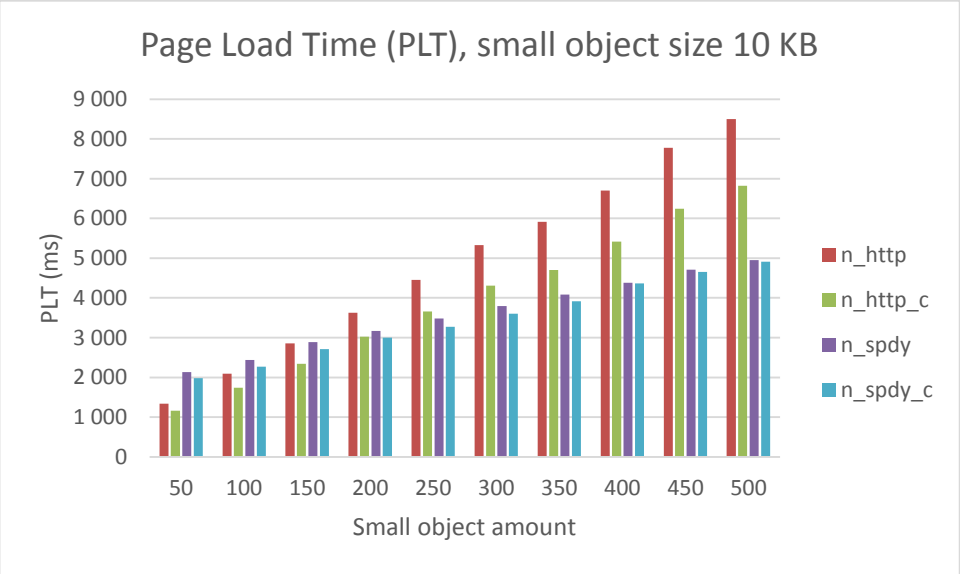

Figure 30. Test case 4.2, small object size 10 KB

PLT with caching starts to be clearly more effective when the number of small object is more than 150. Pure SPDY offers better impact than HTTP with caching when the number of small object is more than 300. The difference between SPDY and SPDY with caching is not significant.

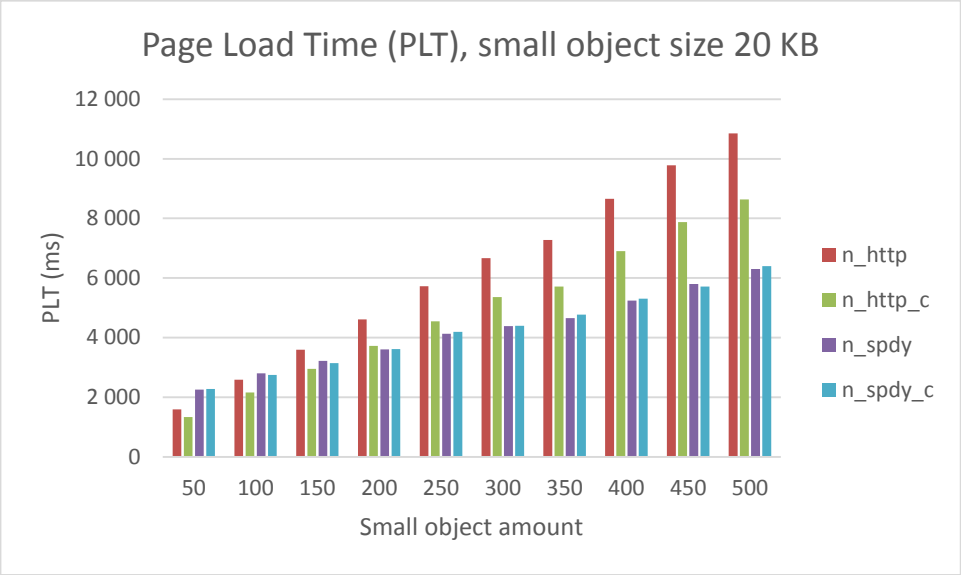

Figure 31. Test case 4.2, small object size 20 KB

PLT with caching starts to be clearly more effective when the number of small objects is more than 150. Pure SPDY offers better impact than HTTP with caching when the number of small object is more than 300. The difference between SPDY and SPDY with caching is not significant.

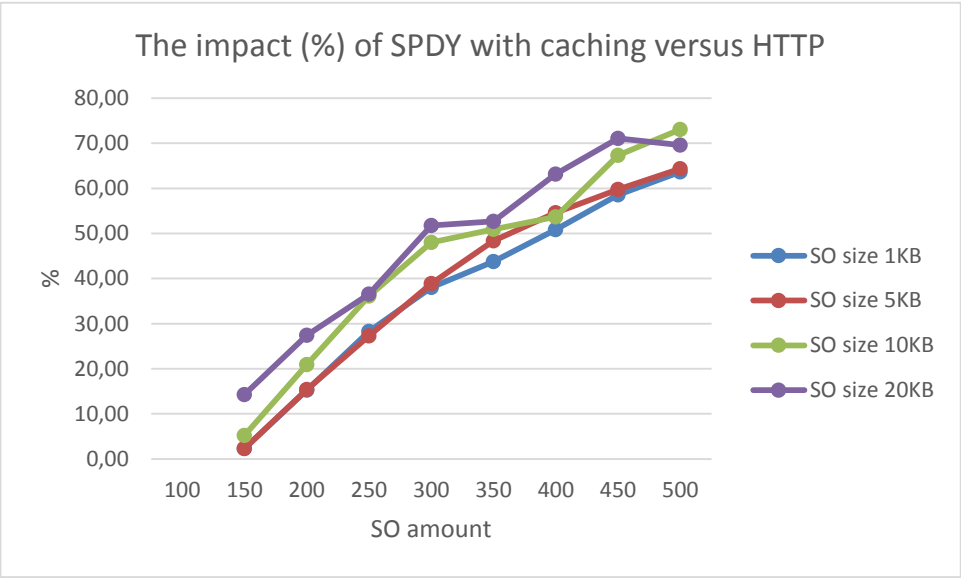

Figure 32. Test case 4.2, the positive impact of SPDY with caching versus HTTP protocol

SPDY with and without caching is more effective compared to HTTP when the small object number is more than 150. The effect is more positive, the more the small object

number increases. The shape of the curves with all measured object sizes looks similar, approximately linear.

Pure SPDY offers better impact as compared to HTTP with caching when the small object number is more than 250. The difference between SPDY and SPDY with caching is not significant.

S1 link capacity reduction did not have a significant effect on the page load time when using the above mentioned test configuration. The rest of the test cases concentrated on the effect of increased delay and reduced Physical Resource Blocks (PRB) values.

4.3 Extra Delay 50 ms

S1 1000 Mbit/s, prb 25, extra rtt 50 ms, HTTP vs. SPDY

The third case was with extra 50 ms delay and the goal was to compare SPDY and HTTP. Figures 33 to 36 illustrates the page load time (PLT) with different parameters. Figure 37 shows the positive impact of SPDY compared to HTTP. PLT start to decrease clearly when the small object number per page was more than 150. The effect was similar on all four object size.

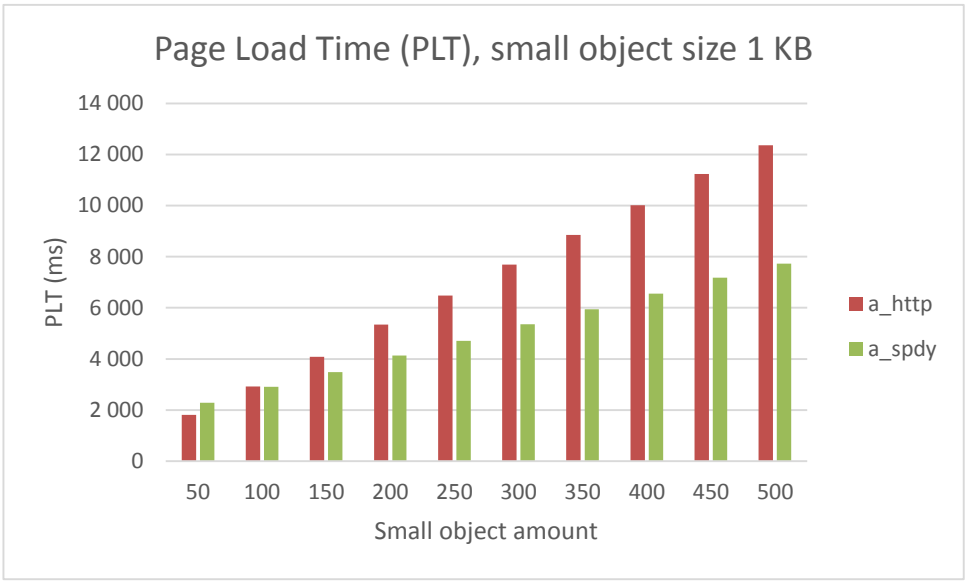

Figure 33. Test case 4.3, small object size 1 KB

SPDY starts to be more effective as compared to HTTP when the number of small object is more than 150.

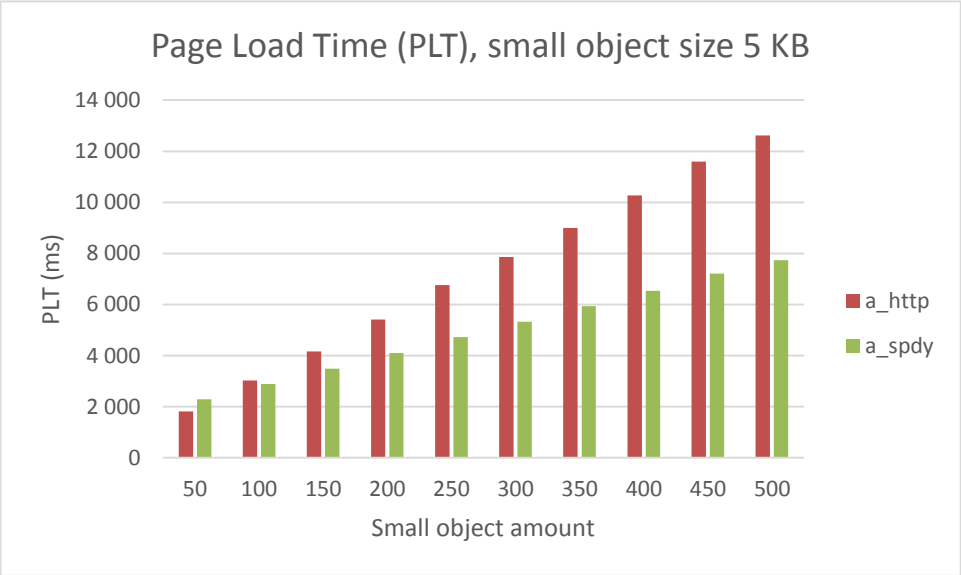

Figure 34. Test case 4.3, small object size 5 KB

SPDY starts to be more effective as compared to HTTP when the number of small object is more than 150.

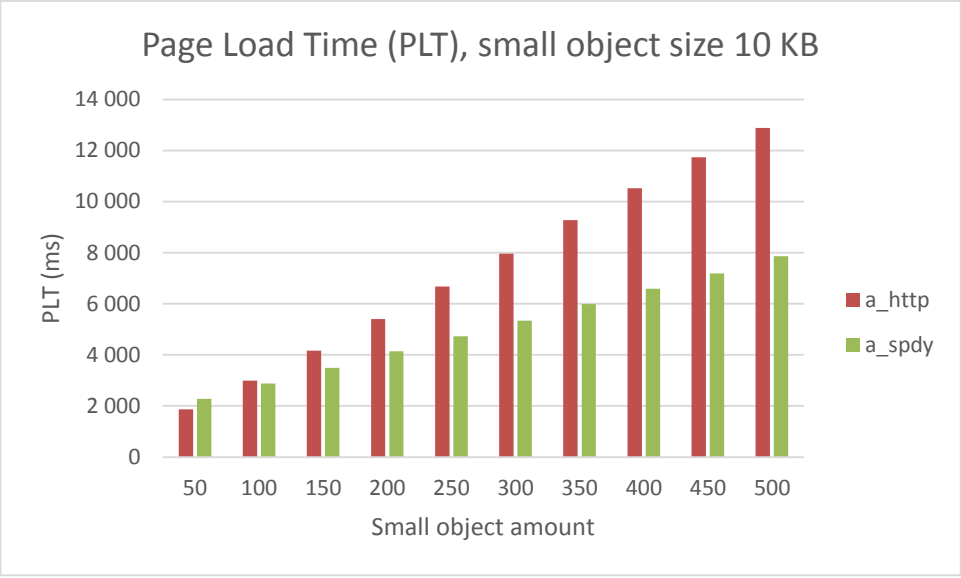

Figure 35. Test case 4.3, small object size 10 KB

SPDY starts to be more effective as compared to HTTP when the number of small object is more than 150.

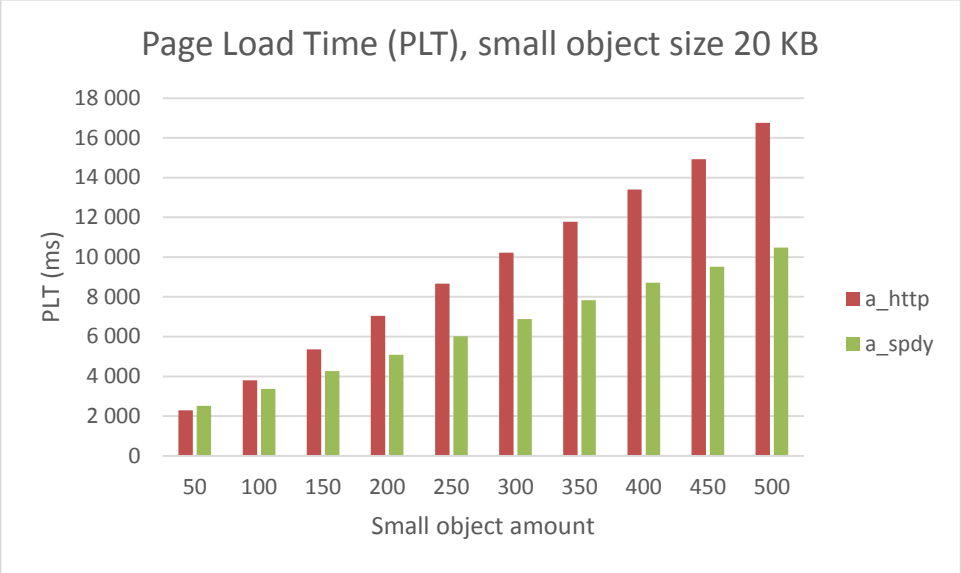

Figure 36. Test case 4.3, small object size 20 KB

SPDY starts to be more effective as compared to HTTP when the number of small object is more than 150.

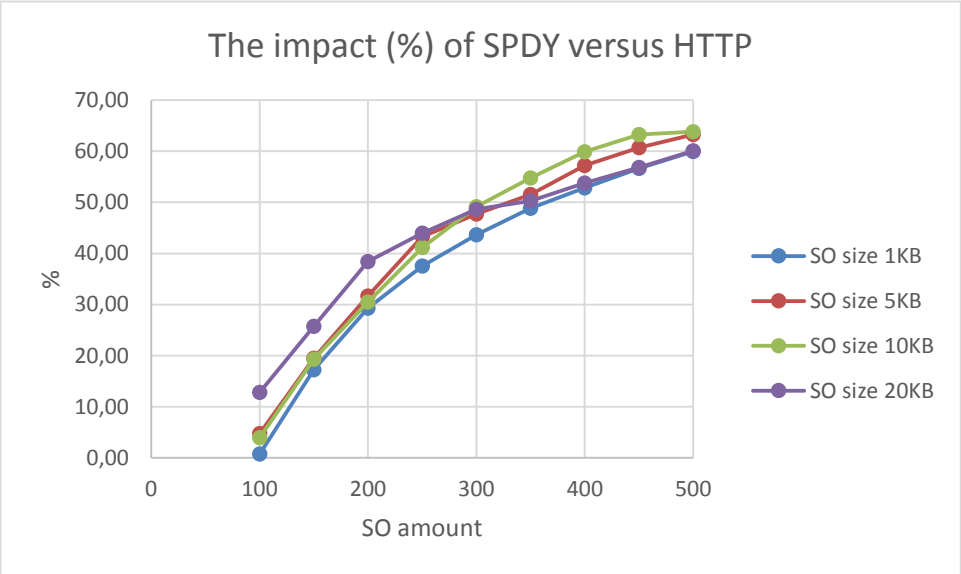

Figure 37. Test case 4.3, the positive impact of SPDY with caching versus HTTP protocol

SPDY is more effective as compared to HTTP when the small object number is more than 150. The effect is more positive, the more the small object number increases. The shape of the curves with all measured object sizes looks similar, but it is not anymore linear.

4.4 Extra Delay 50 ms and Caching

S1 1000 Mbit/s, prb 25, extra rtt 50 ms, caching allowed

The fourth case was with extra 50 ms delay and the goal was to investigate the proxy caching effect. Figures 38 to 41 illustrates the page load time (PLT) with different parameters. When comparing HTTP and HTTP with caching the impact was between 80 to 140% and the impact was increasing along the small object size. When comparing SPDY and SPDY with caching the impact was about 50% with all small object sizes. Figure 42 shows the positive impact of SPDY with caching compared to HTTP. PLT starts to decrease clearly when the small object number is more than 100. The most positive impact was when the small object size was 20 KB.

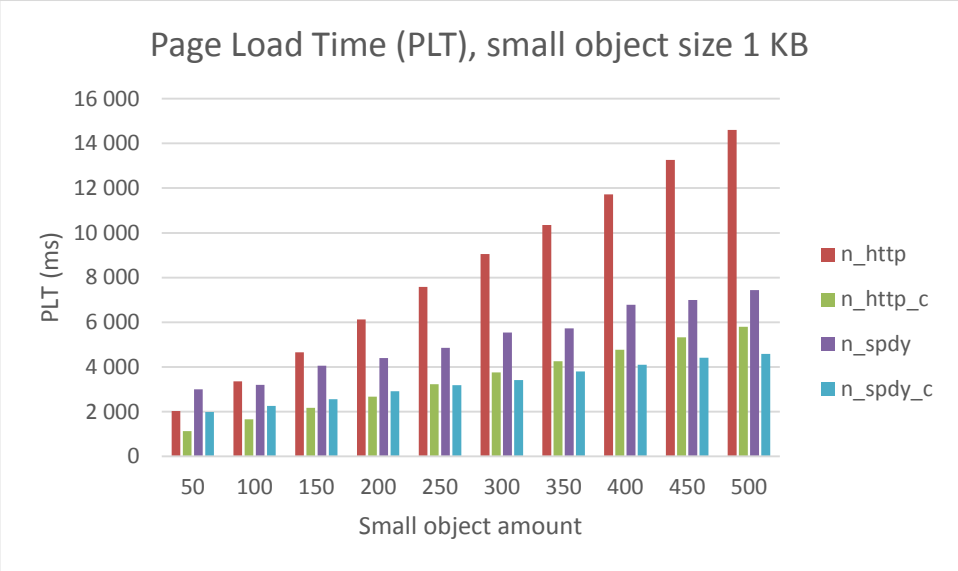

Figure 38. Test case 4.4, small object size 1 KB

PLT with caching looks all the time more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 250. SPDY with caching offers better performance compared to pure SPDY all the time.

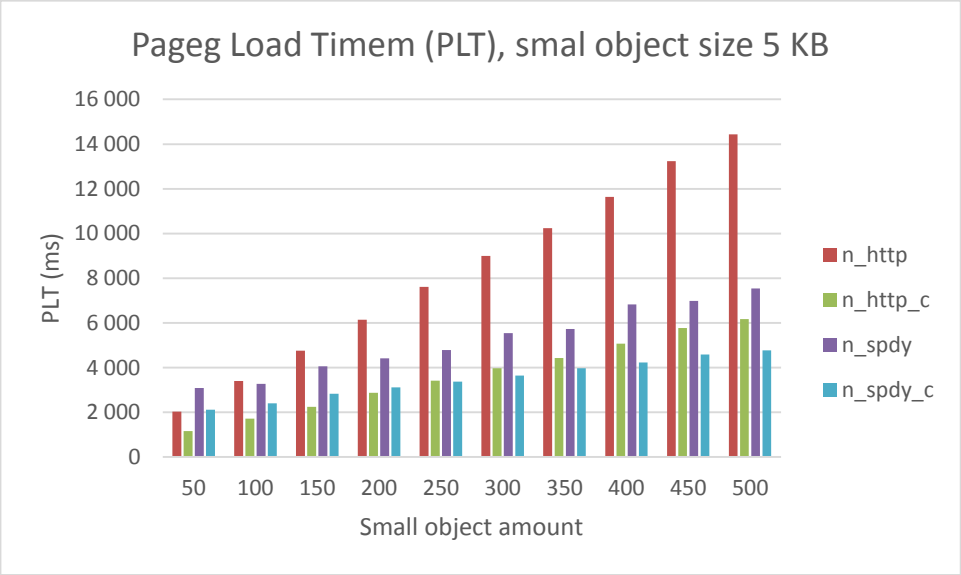

Figure 39. Test case 4.4, small object size 5 KB

PLT with caching looks all the time more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 250. SPDY with caching offers better performance compared to pure SPDY all the time.

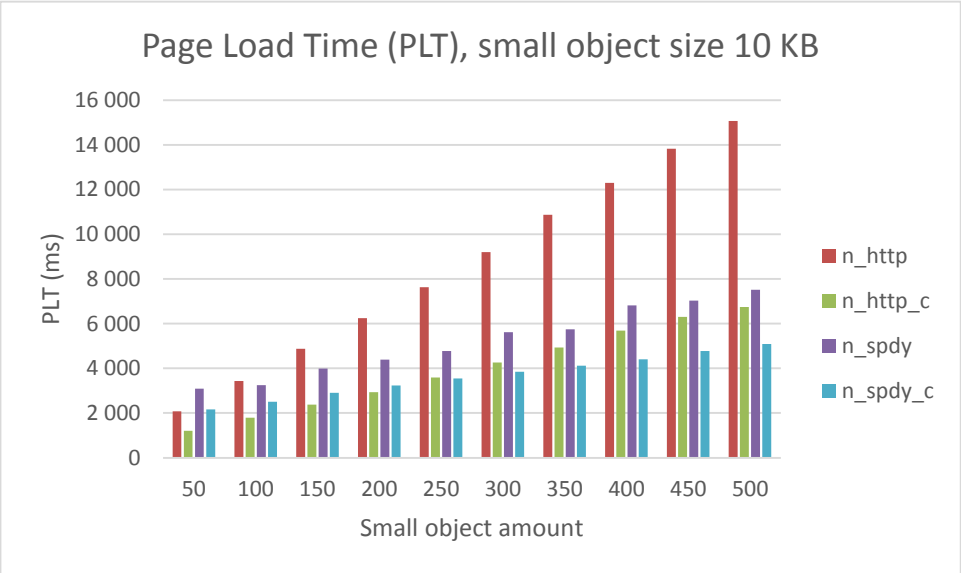

Figure 40. Test case 4.4, small object size 10 KB

PLT with caching looks all the time more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 250. SPDY with caching offers better performance compared to pure SPDY all the time.

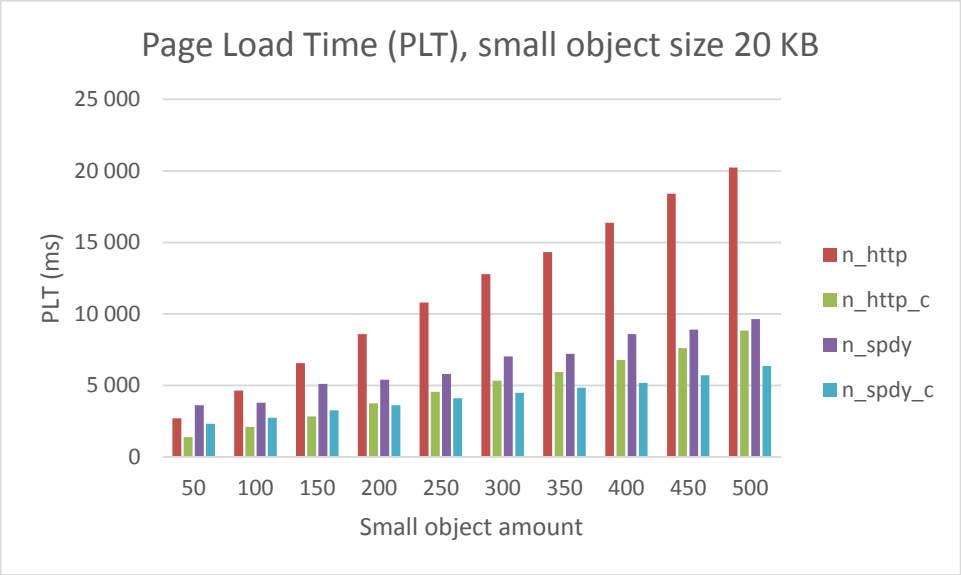

Figure 41. Test case 4.4, small object size 20 KB

PLT with caching looks all the time more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 250. SPDY with caching offers better performance compared to pure SPDY all the time.

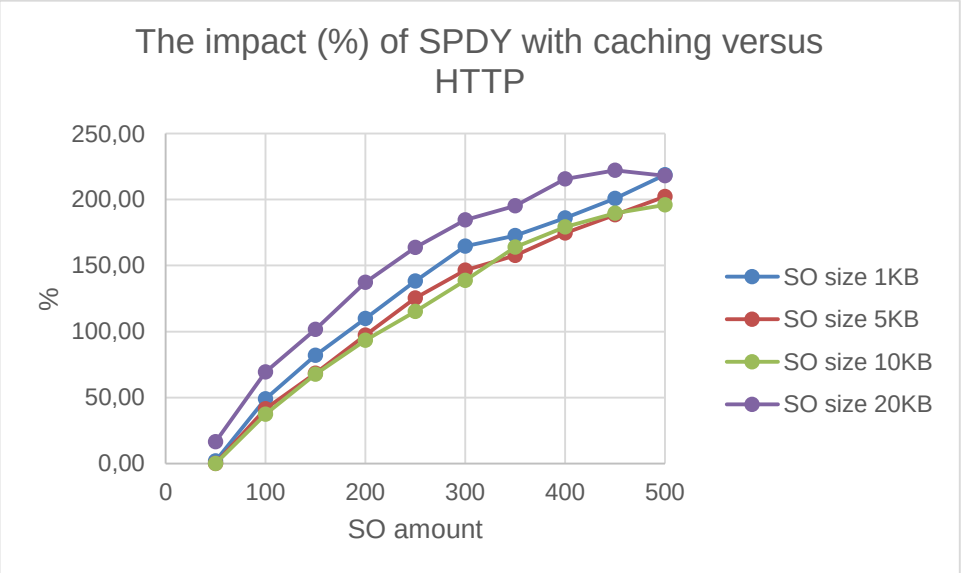

Figure 42. Test case 4.4, the positive impact of SPDY with caching versus HTTP protocol

SPDY is more effective compared to HTTP all the time. The effect is more positive, the more the small object number increases. The shape of the curves with all measured object sizes looks similar, but it is not anymore linear.

4.5 Extra Delay 100 ms

S1 1000 Mbit/s, prb 25, extra rtt 100 ms, HTTP vs. SPDY

The fifth case was with extra 100 ms delay and the goal was to compare SPDY and HTTP. Figures 43 to 46 illustrates the page load time (PLT) with different parameters. Figure 47 shows the positive impact of SPDY compared to HTTP. PLT starts to decrease clearly when the small object number per page is more than 100. The effect was similar on all four object size.

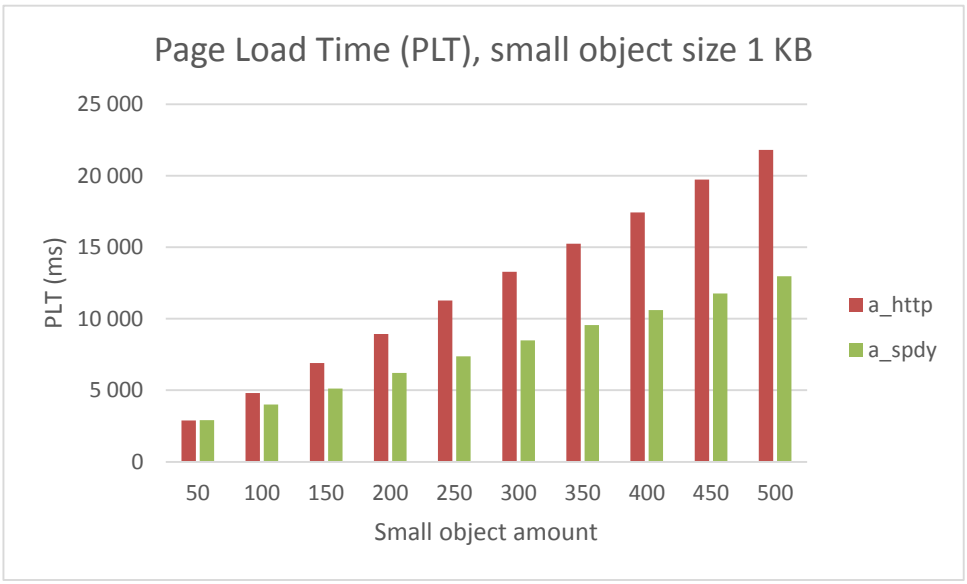

Figure 43. Test case 4.5, small object size 1 KB

SPDY starts to be more effective as compared to HTTP when the number of small object is more than 100.

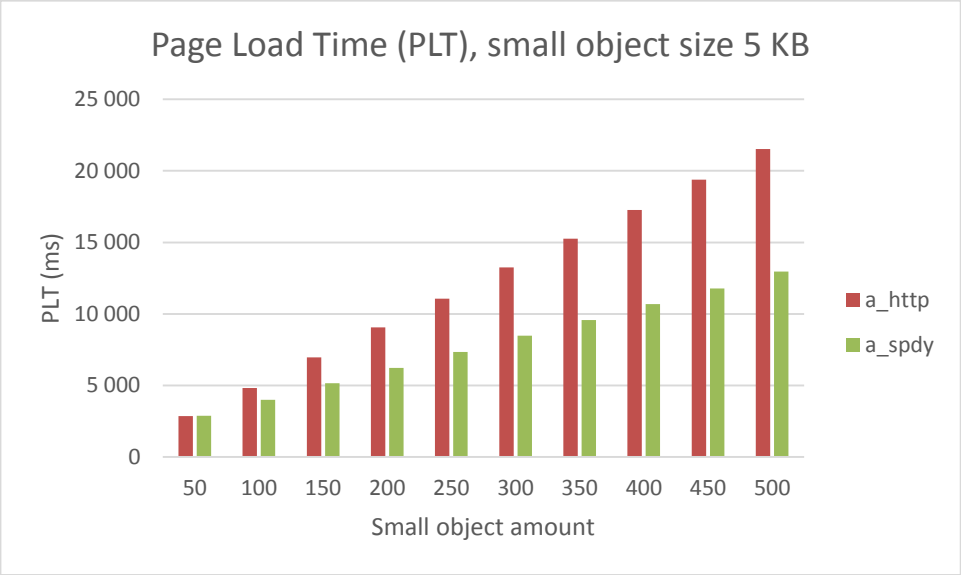

Figure 44. Test case 4.5, small object size 5 KB

SPDY starts to be more effective as compared to HTTP when the number of small object is more than 100.

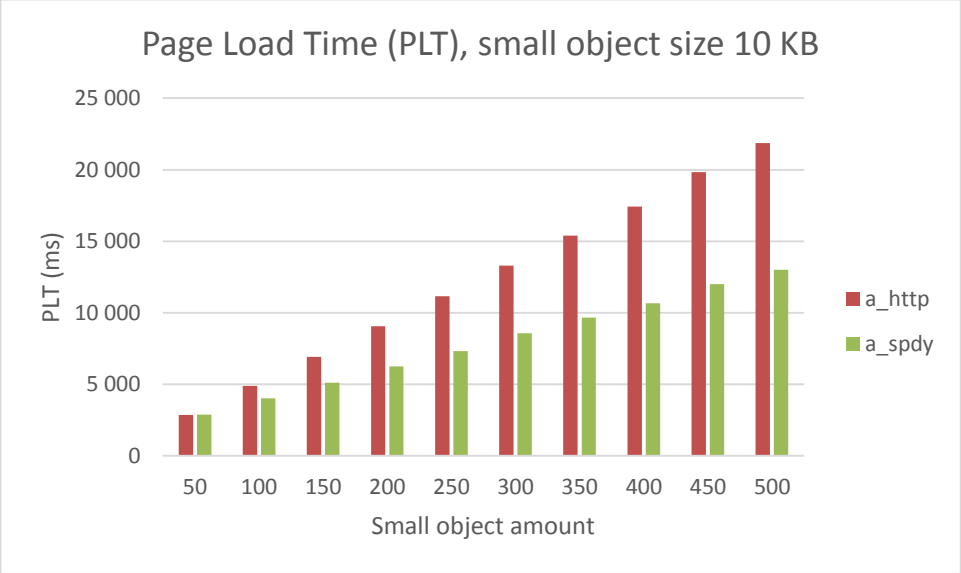

Figure 45. Test case 4.5, small object size 10 KB

SPDY starts to be more effective as compared to HTTP when the number of small object is more than 100.

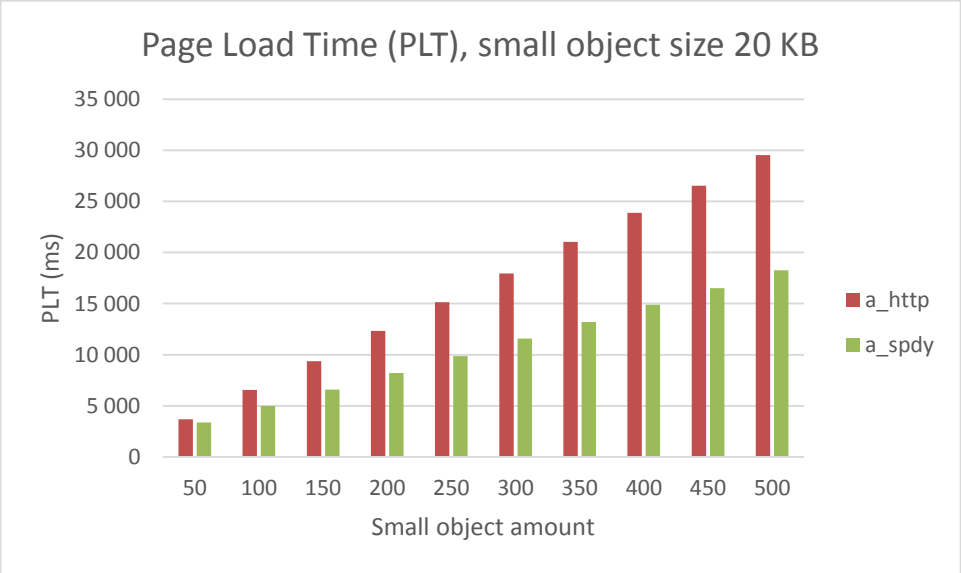

Figure 46. Test case 4.5, small object size 20 KB

SPDY starts to be more effective as compared to HTTP when the number of small object is more than 100.

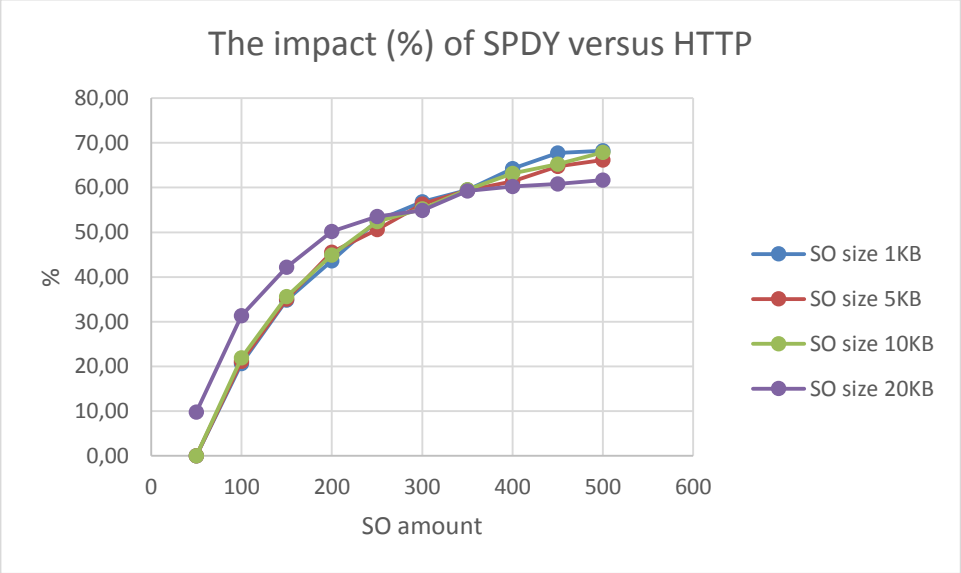

Figure 47. Test case 4.5, the positive impact of SPDY with caching versus HTTP protocol

SPDY is more effective compared to HTTP when the small object number is more than 100. The effect is more positive, the more the small object number increases. After the small object number bigger than 300 the positive impact stabilized. The shape of the curves with all measured object sizes looks similar, logarithmic.

4.6 Extra Delay 100 ms and Caching

S1 1000 Mbit/s, prb 25, extra rtt 100 ms, caching allowed

The sixth case was with extra 100 ms delay and the goal was to investigate the proxy caching effect. Figures 48 to 51 illustrates the page load time (PLT) with different parameters. When comparing HTTP and HTTP with caching the impact was between 150 to 300% and the impact was increasing until the small object number per page reached 300. When comparing SPDY and SPDY with caching the impact was about 100% with all small object sizes. Figure 52 shows the positive impact of SPDY with caching as compared to HTTP. PLT starts to decrease clearly when the small object number is more than 50.

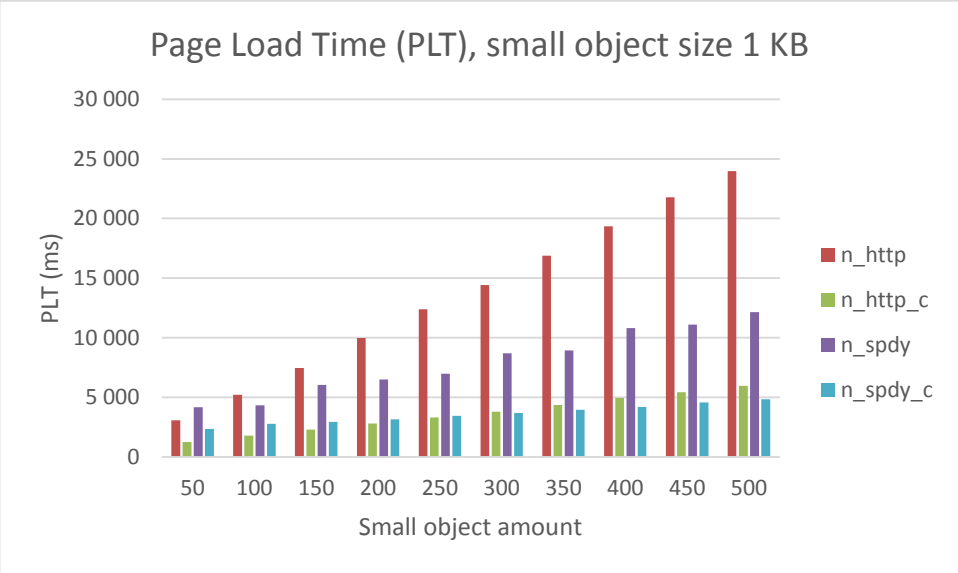

Figure 48. Test case 4.6, small object size 1 KB

PLT with caching looks all the time more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 250. SPDY with caching offers better performance compared to pure SPDY all the time.

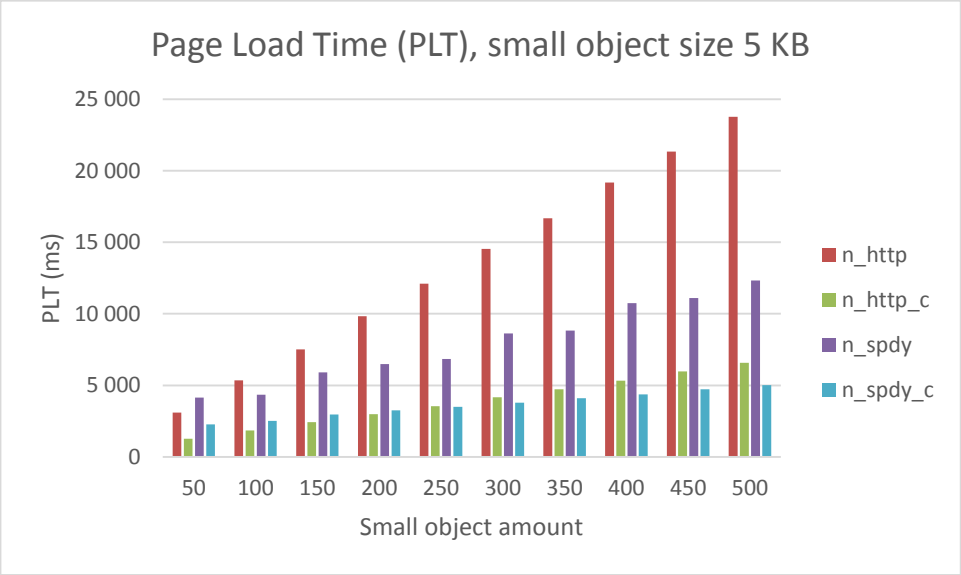

Figure 49. Test case 4.6, small object size 5 KB

PLT with caching looks all the time more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 250. SPDY with caching offers better performance compared to pure SPDY all the time.

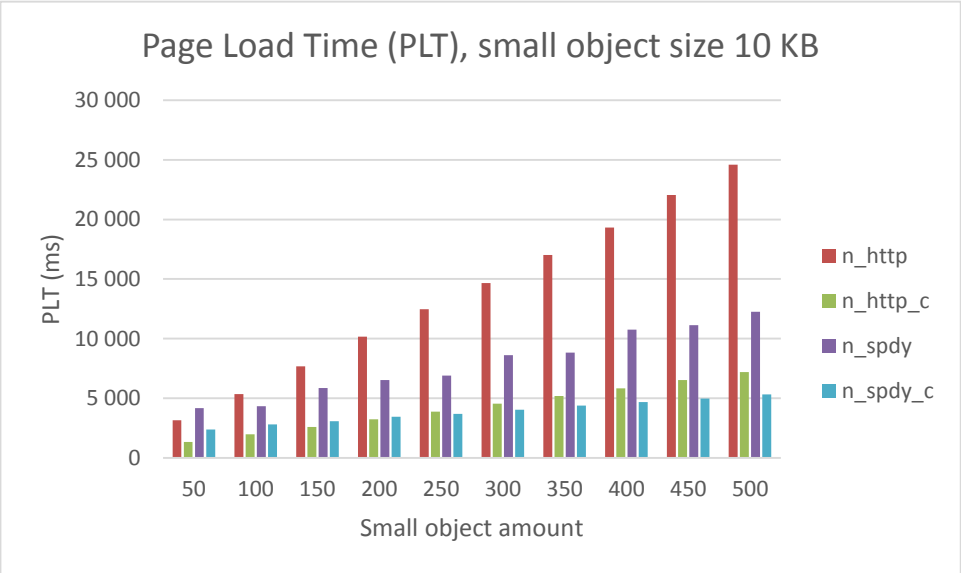

Figure 50. Test case 4.6, small object size 10 KB

PLT with caching looks all the time more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 250. SPDY with caching offers better performance as compared to pure SPDY all the time.

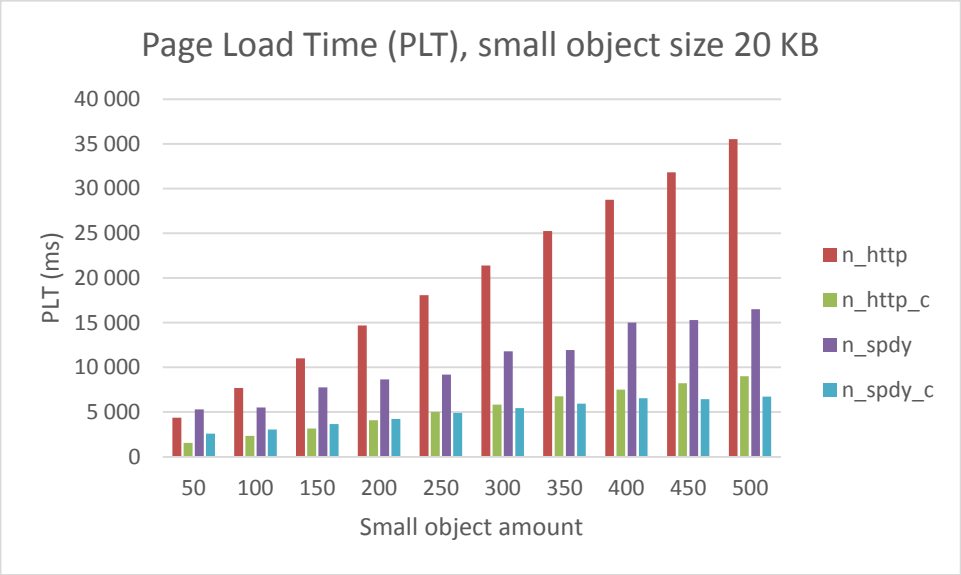

Figure 51. Test case 4.6, small object size 20 KB

PLT with caching looks all the time more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 250. SPDY with caching offers better performance as compared to pure SPDY all the time.

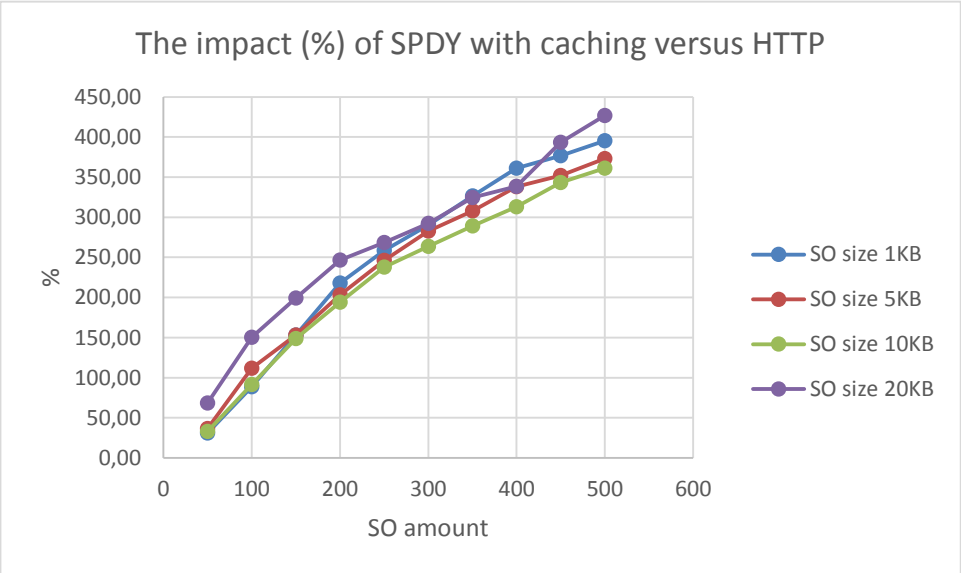

Figure 52. Test case 4.6, the positive impact of SPDY with caching versus HTTP protocol

SPDY is more effective as compared to HTTP all the time. The effect is more positive, the more the small object number increases. The shape of the curves with all measured object sizes looks similar, but it is not anymore quite linear.

4.7 Reduced Physical Resource Blocks to 17

S1 1000 Mbit/, prb 17, extra rtt 10 ms, HTTP vs. SPDY

The seventh case was with prb value 17 and the goal was to compare SPDY and HTTP. Figures 53 to 56 illustrates the page load time (PLT) with different parameters. In this case the PLT with SPDY was longer, so SPDY did not bring any positive impact. PLT difference flattened out when the number of small object per page reached 400.

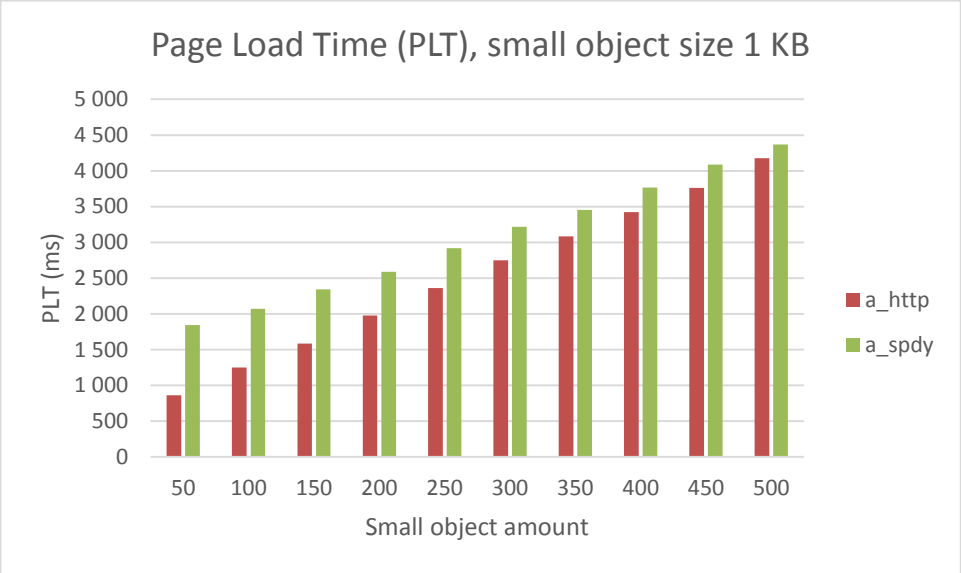

Figure 53. Test case 4.7, small object size 1 KB

SPDY does not offer any positive impact for the PLT. The effect is more negative with a small number of objects.

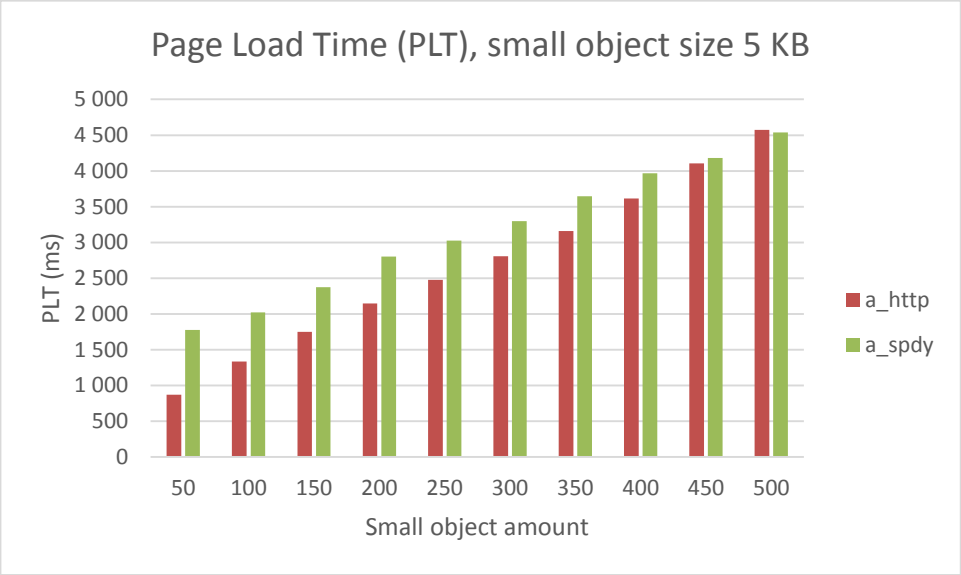

Figure 54. Test case 4.7, small object size 5 KB

SPDY does not offer any positive impact for the PLT. The effect is more negative with a small number of objects.

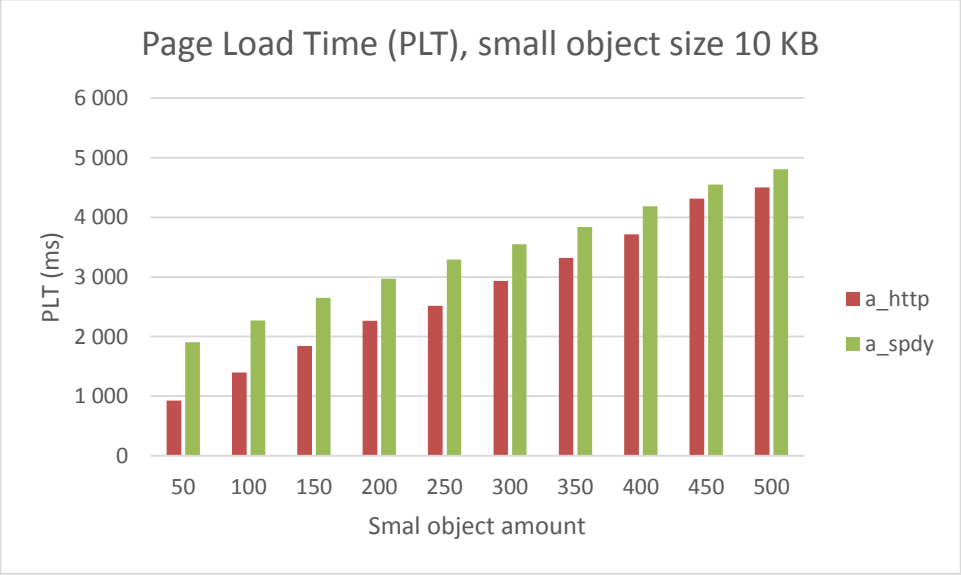

Figure 55. Test case 4.7, small object size 10 KB

SPDY does not offer any positive impact for the PLT. The effect is more negative with a small number of objects.

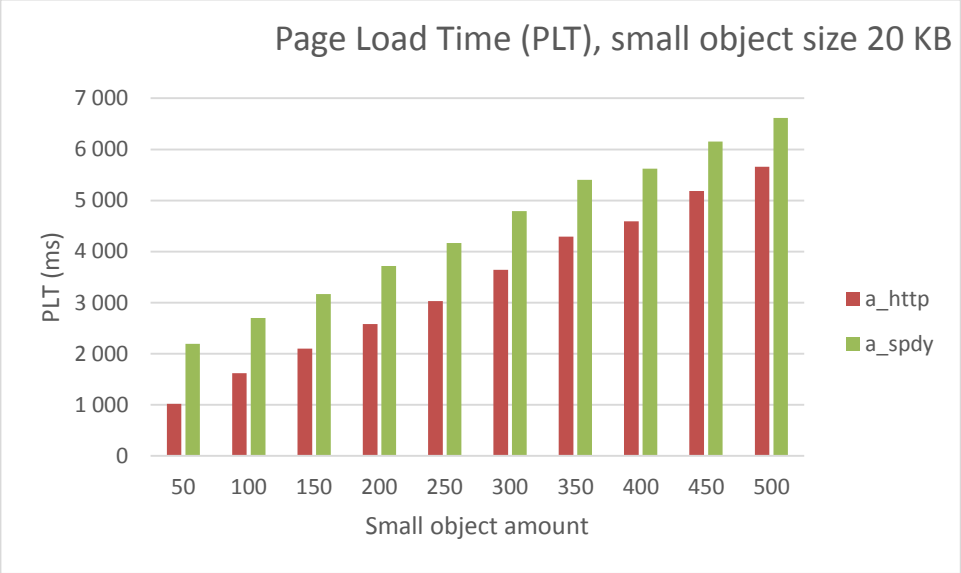

Figure 56. Test case 4.7, small object size 20 KB

SPDY does not offer any positive impact for the PLT. The effect is more negative with a small number of objects.

4.8 Reduced Physical Resource Blocks to 17 and Caching

S1 1000 Mbit/s, prb 17, extra rtt 10 ms, caching allowed

The eighth case was with prb value 17 and the goal was to investigate the proxy caching effect. Figures 57 to 60 illustrates the page load time (PLT) with different parameters. When comparing HTTP and HTTP with caching the impact was only a few per cent and when comparing SPDY and SPDY with caching the impact was also only a few per cent. Figure 61 shows the positive impact of SPDY with caching compared to HTTP. PLT starts to decrease clearly when the small object number is more than 300.

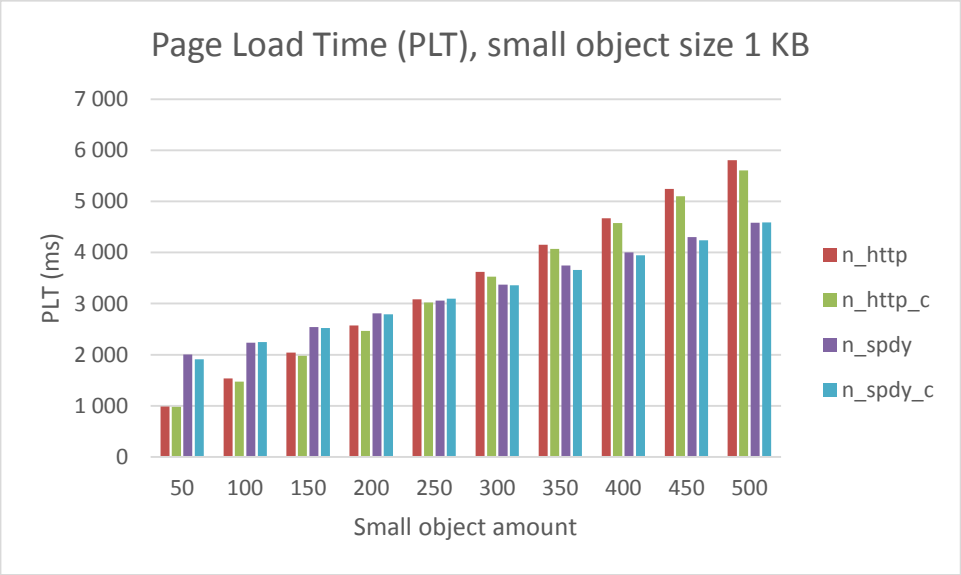

Figure 57. Test case 4.8, small object size 1 KB

PLT with caching looks all the time little bit more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 300. SPDY with caching offers little bit better performance compared to pure SPDY all the time.

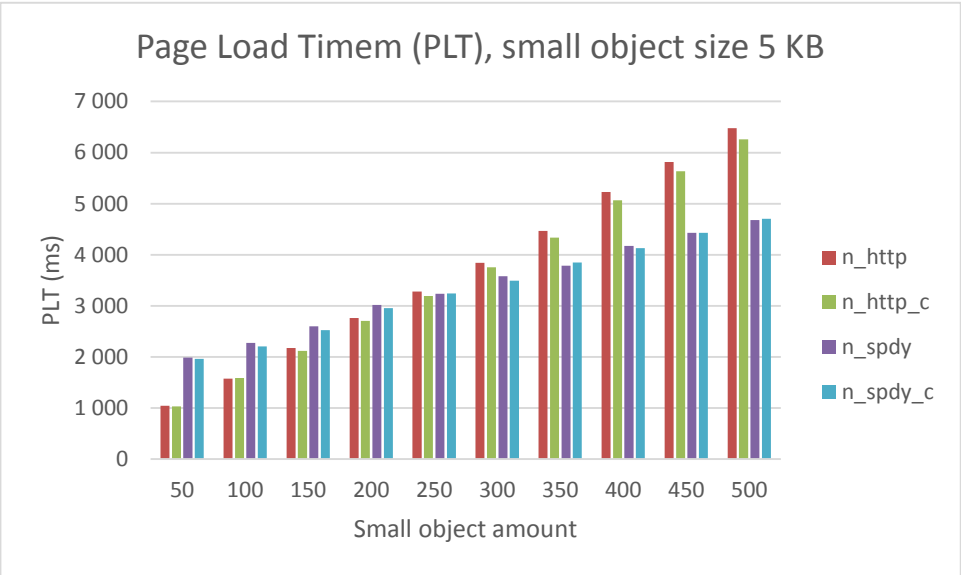

Figure 58. Test case 4.8, small object size 5 KB

PLT with caching looks all the time a little bit more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 300.

SPDY with caching offers little bit better performance compared to pure SPDY all the time.

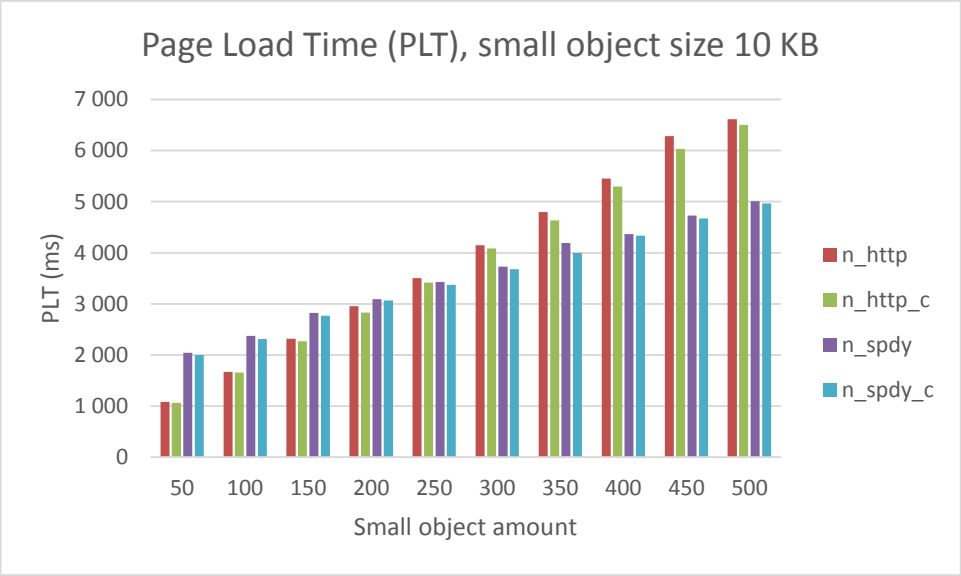

Figure 59. Test case 4.8, small object size 10 KB

PLT with caching looks all the time little bit more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 300. SPDY with caching offers little bit better performance compared to pure SPDY all the time.

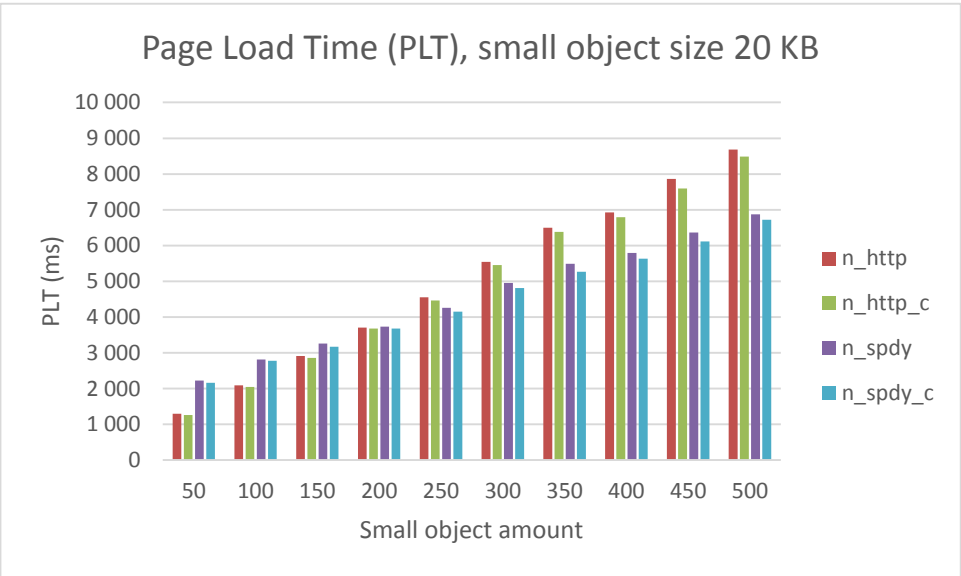

Figure 60. Test case 4.8, small object size 20 KB

PLT with caching looks all the time little bit more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 250. SPDY with caching offers little bit better performance compared to pure SPDY all the time.

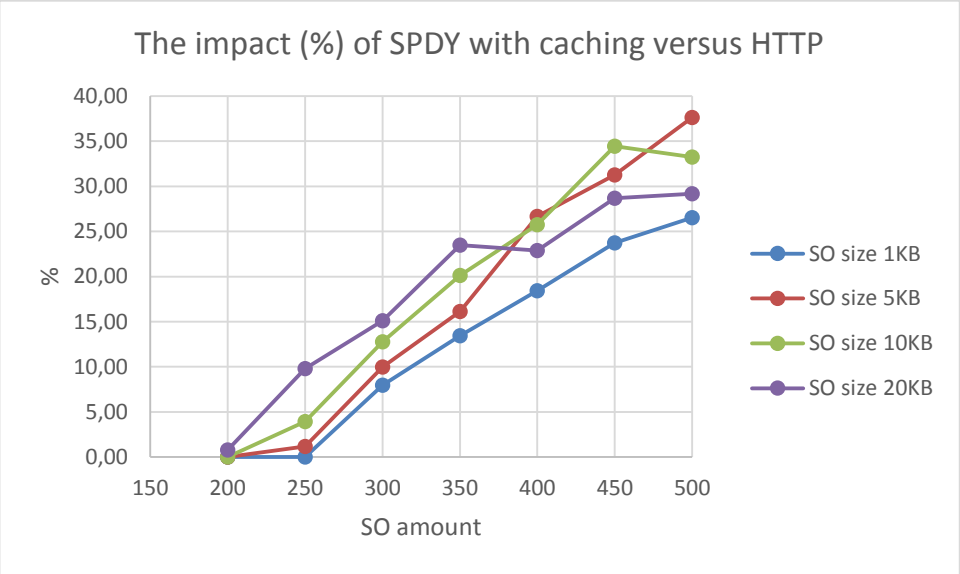

Figure 61. Test case 4.8, the positive impact of SPDY with caching versus HTTP protocol

SPDY is more effective as compared to HTTP when the small object number is more than 250. The effect is more positive, the more the small object number increases. The shape of the curves with all measured object sizes looks similar, almost linear.

4.9 Reduced Physical Resource Blocks to 4

S1 1000 Mbit/s, prb 4, extra rtt 10 ms, HTTP vs. SPDY

The ninth case was with prb value 4 and the idea was to compare SPDY and HTTP. Figures 62 to 65 illustrates the page load time (PLT) with different parameters. In this case PLT with SPDY was longer, so SPDY did not bring any positive impact. PLT difference was the shortest when the small object size was 20 KB.

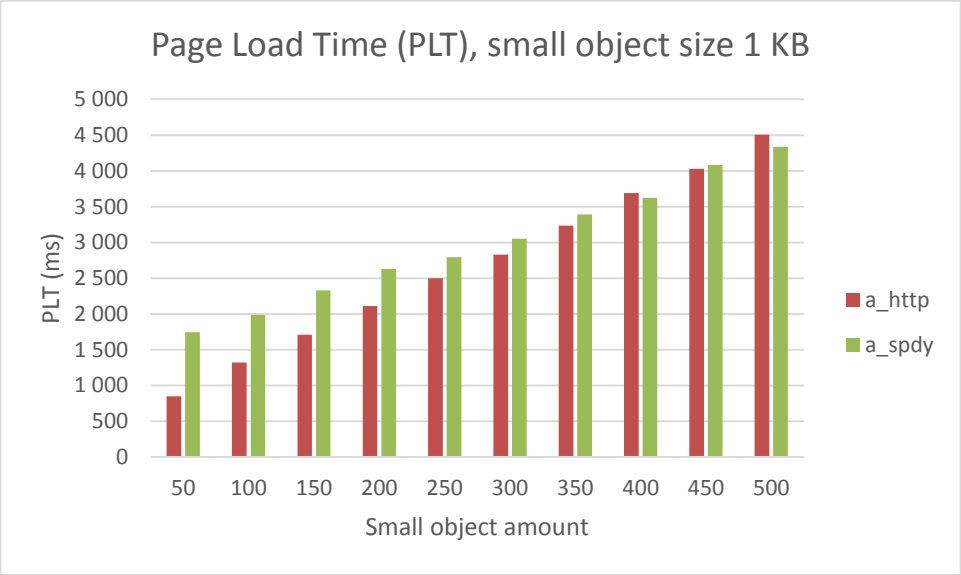

Figure 62. Test case 4.9, small object size 1 KB

SPDY does not offer any positive impact for the PLT before the small object number is 500. The effect is more negative with a small number of objects.

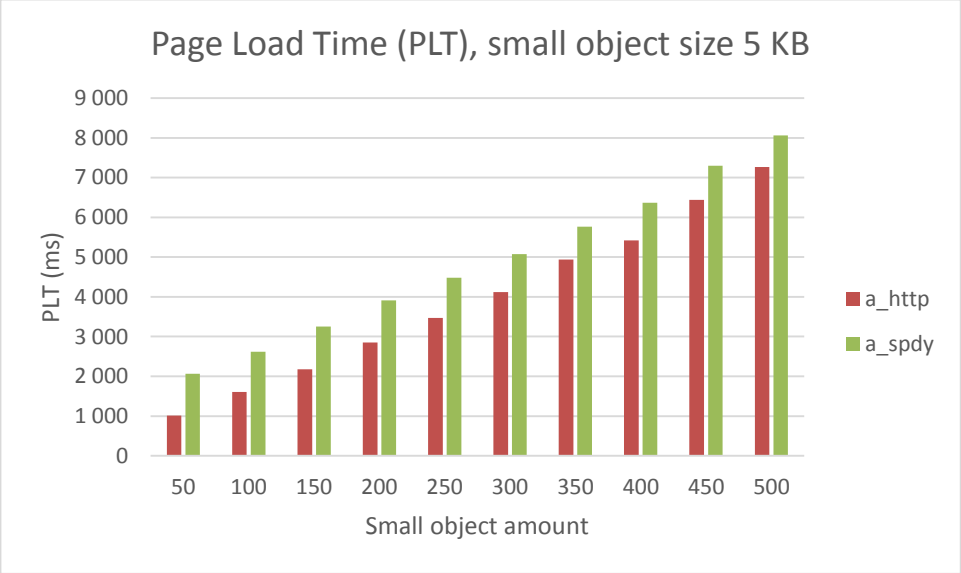

Figure 63. Test case 4.9, small object size 5 KB

SPDY does not offer any positive impact for the PLT. The effect is more negative with a small number of objects.

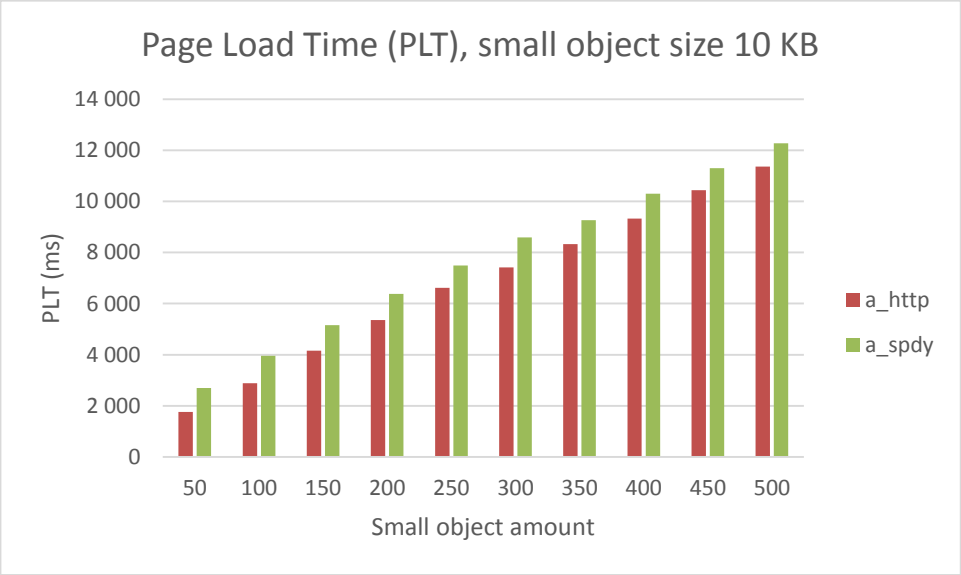

Figure 64. Test case 4.9, small object size 10 KB

SPDY does not offer any positive impact for the PLT. The effect is more negative with a small number of objects.

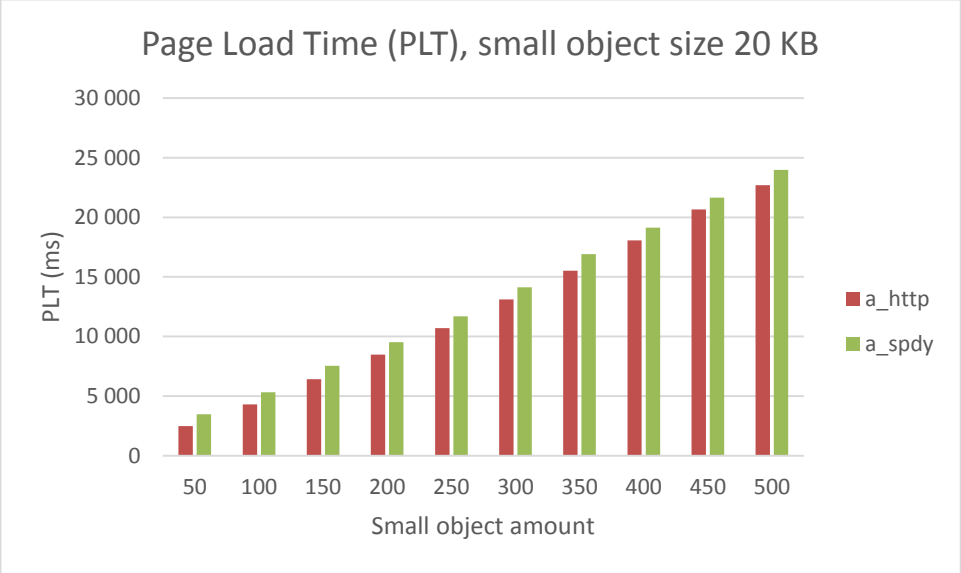

Figure 65. Test case 4.9, small object size 20 KB

SPDY does not offer any positive impact for the PLT. The PLT difference between HTTP and SPDY is very small.

4.10 Reduced Physical Resource Blocks to 4 and Caching

S1 1000 Mbit/s, prb 4, extra rtt 10 ms, caching allowed

The tenth case was with prb value 4 and the goal was to investigate the proxy caching effect. Figures 66 to 69 illustrates the page load time (PLT) with different parameters. When comparing HTTP and HTTP with caching the impact was only a few per cent and when comparing SPDY and SPDY with caching the caching impact was mostly negative. Figure 70 shows the positive impact of SPDY with caching compared to HTTP. The impact would be a little bit better when comparing SPDY without caching and HTTP. PLT starts to decrease clearly when the small object number is more than 250 and the small object size is 1 KB or when the small object number is more than 400 and the small object size is 5 KB. With other combinations the effect was insignificant.

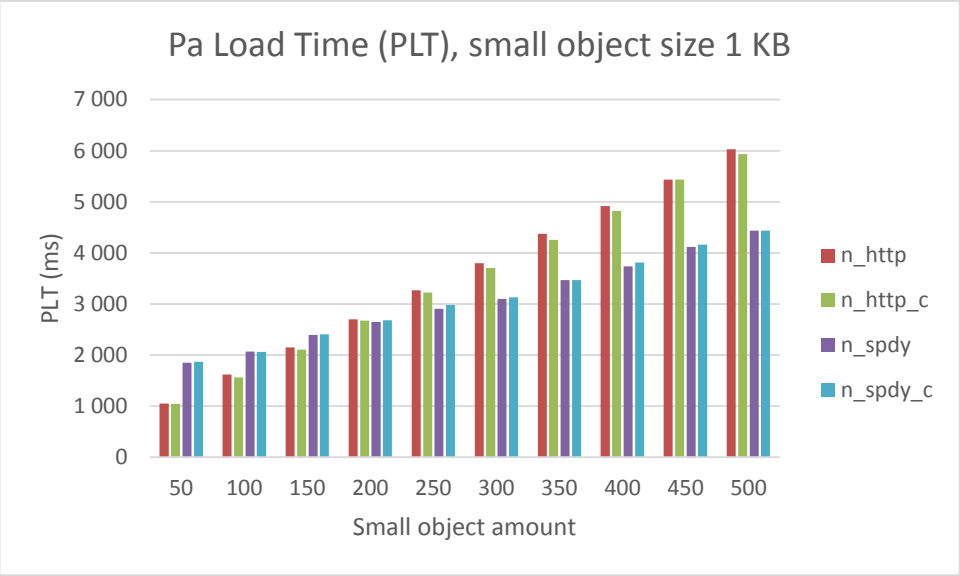

Figure 66. Test case 4.10, small object size 1 KB

PLT with caching looks all the time little bit more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 250. The difference between SPDY and SPDY with caching is not significant.

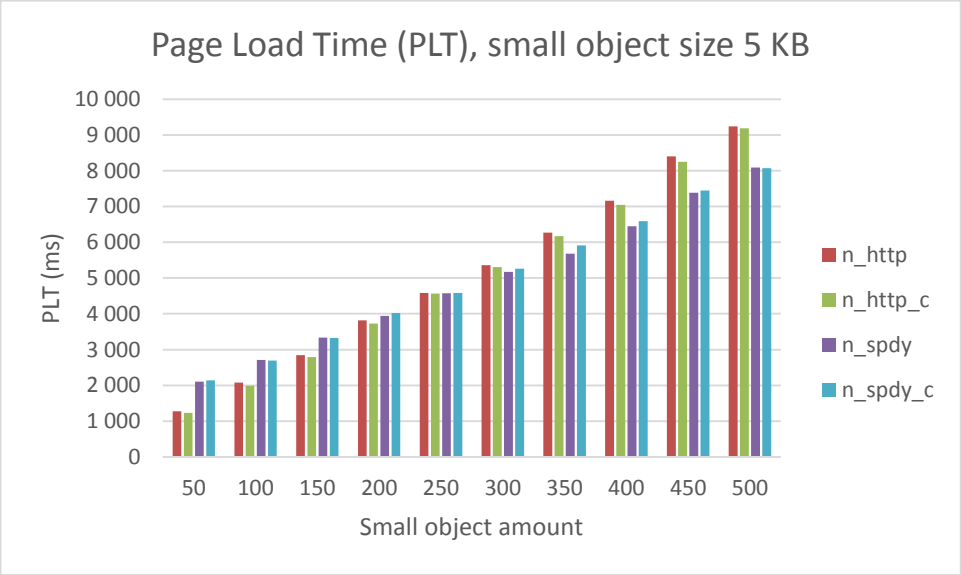

Figure 67. Test case 4.10, small object size 5 KB

PLT with caching looks all the time little bit more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 300. The difference between SPDY and SPDY with caching is not significant.

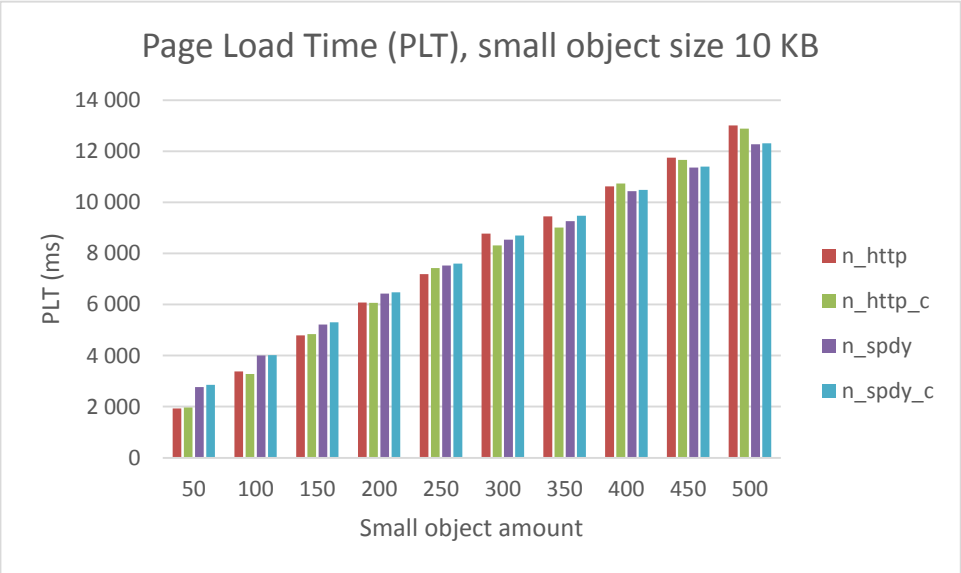

Figure 68. Test case 4.10, small object size 10 KB

PLT with caching looks all the time little bit more effective than without caching. SPDY with caching offers the best PLT when the number of small object is more than 400. The difference between SPDY and SPDY with caching is not significant.

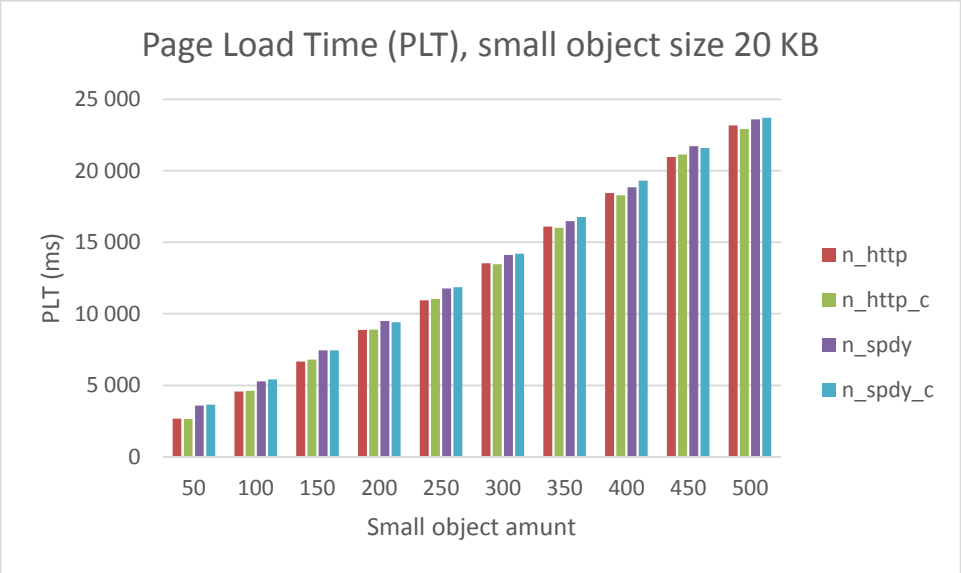

Figure 69. Test case 4.10, small object size 20 KB

PLT with caching looks all the time little bit more effective than without caching. SPDY with caching does not offer any positive impact. The difference between SPDY and SPDY with caching is not significant.

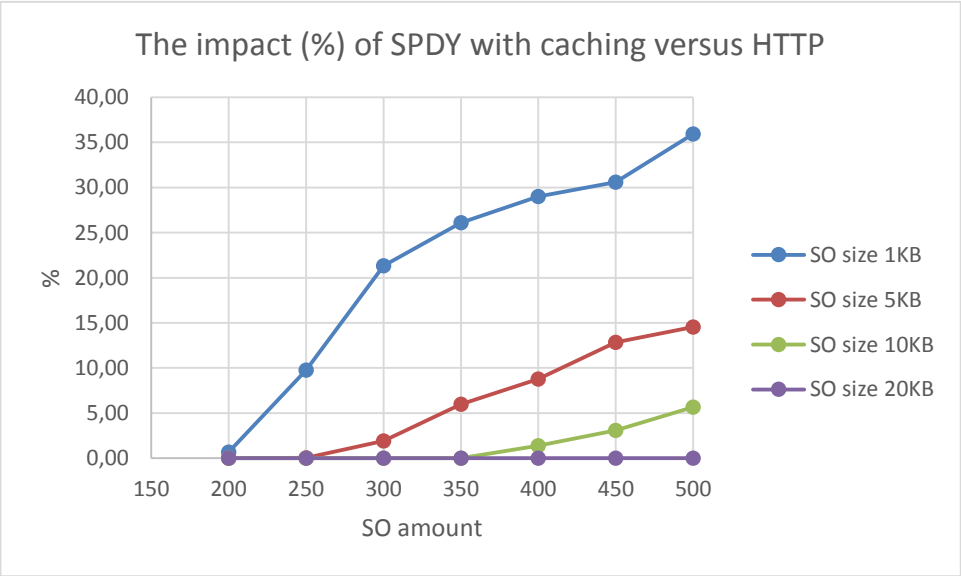

Figure 70. Test case 4.10, the positive impact of SPDY with caching versus HTTP protocol

SPDY is more effective as compared to HTTP when the small object number is more than 250 and the small object size is 1 KB. With 5 KB object size the positive impact

starts when the number of small object is more than 350. With 10 and 20 KB object size the effect is not significant.

5 Analysis and Discussion

This chapter provides the latest status of SPDY protocol and informs that the HTTP version 2 has been officially released. Further, some new HTTP version 2 features are introduced.

5.1 Goodbye SPDY, Welcome HTTP/2

The used test configuration for all test cases was the same: client, proxy servers and a content server. Most of content servers still use HTTP version 1.x protocol but the usage of SPDY-capable proxies offers the benefits of SPDY at least over the last mile. Proxies also offer a place for caching. The main purpose of this thesis was to check and compare the effectiveness of HTTP and SPDY protocol and the influence of caching.

The main object of interest was the web page load time when focusing on the effect of small object number and size and caching. Test setup was built in laboratory environment and it was reserved only for this purpose so there were no other users or any extra disturbances resulting from other users.

The reason for such a testing was that in general the structure of web pages has changed dramatically during the last few years and the old HTTP version 1.x protocol is not anymore suitable especially for fast page download which is one of the key performance indicators (KPI) for web browsing quality of experience (QoE) [39].

The interesting point is that especially mobile operators could use the web browsing QoE to determine when and where network conditions are causing reduced customer experience. When understanding adequately the relationship between web browsing QoE and network parameters factors the operator could for example decide whether to increase a cell's transmit power to improve signal strength, or decrease it to reduce handovers from possible overlapping cells. [39.]

Google's SPDY was the first real improvement for HTTP version 1.x trying to improve the web performance, especially the latency. It was shown that the existing HTTP version 1.x has become old. The SPDY protocol introduced new key features such as multiplexing, header compression, prioritization and protocol negotiation. Based on the open non-standard SPDY protocol a new standard HTTP version 2 was released. It also means that Google has removed support for SPDY in early 2016.

5.2 HTTP/2 New Features

Even though HTTP version 2 is based on SPDY, it introduces some important new changes. The most important new features are listed in Table 17. [5, Chapter 12.]

Table 17. HTTP/2 new features compared to SPDY. Data gathered from Grigorik IL [5, Chapter 12]

HTTP/2 new features compared to SPDY
SSL is not required; although IETF doesn't require SSL, many popular browsers do require it
Faster Encrypted Connections; Application Layer Protocol Negotiations (ALPN) is used as a Transport Layer Security (TSL) extension instead of Next Protocol Negotiation (NPN)
Multi-Host Multiplexing; multiplexing happens on different hosts at the same time instead of one host at a time
Faster and More Secure Compression; HPACK compression is used instead of ZLIB
Improved Prioritization; more flexible and friendlier to proxies

There are some features which could have either a positive or a negative effect on the end user's performance. Server push is at least one of them and it was already one of SPDY features. The idea of the server push is that the server sends the responses to the client's cache before separate requests. The responses in question could be images, java scripts or CSS files. The problem is that the server could send content which is already in the client's cache or which is never used. This could be problematic especially for mobile devices because it can waste battery.

The measurement of the mobile client's battery consumption when using server push could be the next useful task. It is not an easy task and it requires some special arrangements because some devices have an integrated battery.

6 Summary

The web page structure has changed dramatically during the last ten years. The average page size is roughly ten times bigger and the number of external objects per page has tripled. So it is quite clear that the old HTTP protocol is no longer suitable to serve effectively enough. Google developed its own version named Speedy (SPDY) and it has been a basis for the latest HTTP version 2.

SPDY has offered a smooth upgrade path to HTTP/2. It has introduced a set of new features which have a clear impact on reducing web latency which means reducing page load time. The main new features are full request and response multiplexing and effective header compression. From the end user point of view the web page load time is the most important indicator for the Quality of Experience (QoE).

The study concentrated mainly on two things; comparing SPDY and HTTP/1.1 and checking the effect of caching. Web caching is used to reduce the network traffic. Caching is done using a proxy server.

Test results show that the use of SPDY protocol reduces the web page load time. The effect is the more positive the longer the network delay is. If the extra delay is 100 ms the page load time is up to 60% faster when using SPDY. The positive impact starts to influence when the number of small object amount per page is more than 100 and the extra delay is more than 50 ms. If the delay is small then SPDY starts to perform better when the small object number is more than 300 per page.

The impact of caching is quite obvious. The page load time is much faster when loading it from the proxy servers' cache memory. The positive impact could be about 300%. The interesting case was to compare the difference between HTTP with caching and SPDY with caching. In this case SPDY with caching started to perform better after the number of small object was more than 250 per page.

The maximum number of Physical Resource Blocks (PRBs) were used as an indicator when trying to simulate the situation when the eNB was loaded. Value 4 means very loaded and value 25 means that the user gets all the resources. Test results showed that when reducing the PRB value to 17 and 4 the SPDY performs worse.

When comparing the difference between HTTP with caching and SPDY with caching there was some positive performance improvement when the number of small objects per page was more than 300.

References

1 StatCounter. Internet web access worldwide between May 2011 to Nov 2015. [online]
URL:
http://gs.statcounter.com/#mobile_vs_desktop-wwmonthly-201105-201205.
Accessed 15 February 2016.

2 Search Engine Watch. Time spent with the Internet in US between Feb 2013 to Jan 2014. [online]
URL:
http://searchenginewatch.com/sew/opinion/2353616/mobile-now-exceeds-pc-the-biggest-shift-since-the-internet-began#
Accessed 15 February 2016.

3 GSA. LTE user devices growth between Feb 2011 to Oct 2015. [online]
URL:
http://www.gsacom.com/downloads/pdf/LTE_user_devices_growth_021115.php4
Accessed 27 November 2015

4 WebSiteOptimization. Average Web Page Breaks 1600K. [online]
URL:
http://www.websiteoptimization.com/speed/tweak/average-web-page/
Accessed 15 February 2016

5 Grigorik I. High Performance Browser Networking. Sebastopol, CA: O'Reilly Media
URL:
http://chimera.labs.oreilly.com/books/1230000000545/ch09.html#_http_0_9_the_one_line_protocol
Accessed 15 February 2016

6 RFC. Hypertext Transfer Protocol -- HTTP/1.0. [online]
URL:
http://www.rfc-base.org/txt/rfc-1945.txt
Accessed 15 February 2016

7 RFC. Hypertext Transfer Protocol – HTTP/1.1. [online]
URL:
http://www.rfc-base.org/txt/rfc-2068.txt
Accessed 15 February 2016

8 RFC. Hypertext Transfer Protocol – HTTP/1.1. [online]
URL:
http://www.rfc-base.org/txt/rfc-2016.txt
Accessed 15 February 2016

9 Maalouf J. K2 Enterprise Security, Network Security Protocols, Part I. [online]
URL:
https://www.k2esec.com/network-security-protocols-background-part-i/
Accessed 14 February 2016

10 Maalouf J. K2 Enterprise Security, Network Security Protocols, Part II. [online]
URL:
https://www.k2esec.com/network-security-protocols-ipsec-vs-tlsssl-vs-ssh-part-ii/
Accessed 14 February 2016

11 Anttalainen T, Jääskeläinen V. Introduction to Communication Networks. Boston: Artech House; 2015.

12 RFC. HTTP over TLS. [online]
URL:
http://www.rfc-base.org/txt/rfc-2818.txt
Accessed 14 February 2016

13 Krishnamurthy B, Mogul J, Kristol D. Key Differences between HTTP/1.0 and HTTP/1.1. [online]
URL:
http://www8.org/w8-papers/5c-protocols/key/key.html
Accessed 15 February 2016

14 Stack Overflow. Difference between HTTP pipeling and HTTP multiplexing with SPDY. [online]
URL:
http://stackoverflow.com/questions/10480122/difference-between-http-pipeling-and-http-multiplexing-with-spdy
Accessed 15 February 2016

15 Akamai Technologies. Press Release. [online]
URL:
https://www.akamai.com/us/en/about/news/press/2009-press/akamai-reveals-2-seconds-as-the-new-threshold-of-acceptability-for-ecommerce-web-page-response-times.jsp
Accessed 15 February 2016

16 Sarkissian H. The Business Case for Caching in 4G LTE Networks; April 2013. [online]
URL:
http://www.wireless2020.com/docs/LSI_WP_Content_Cach_Cv3.pdf
Accessed 5 February 2016

17 The Chromium Projects. SPDY: An experimental protocol for a faster web. [online]
URL:
http://www.chromium.org/spdy/spdy-whitepaper
Accessed 15 February 2016

18 Elkhatib Y, Tyson G, Welzl M. The Effect of Network and Infrastructural Variables an SPDY's Performance;January 2014. [online]
URL:
http://arxiv.org/pdf/1401.6508v1.pdf
Accessed 15 February 2016

19 Mineki G, Uemura S, Hasegawa T. SPDY Accelerator for Improving Web Access Speed; January 2013. [online]
URL:
http://www.ieice.org/ken/paper/2012092870WV/eng/
Accessed 15 February 206

20 Chowdhury S, Sapra V, Hindle A. Is HTTP/2 More Energy Efficient Than HTTP/1.1 for Mobile Users? August 2015. [online]
URL:
https://dx.doi.org/10.7287/peerj.preprints.1280v1
Accessed 15 February 2016

21 The Chromium Projects. SPDY Protocol-Draft 3.2. [online]
URL:
https://www.chromium.org/spdy/spdy-protocol/spdy-protocol-draft3-2
Accessed 15 February 2016

22 The Chromium Projects. SPDY Proxy. [online]
URL:
https://www.chromium.org/spdy/spdy-proxy
Accessed 15 February 2016

23 Khalid J, Agarwal S, Akella A, Padhye J. Improving the performance of SPDY for mobile networks. [online]
URL:
http://research.microsoft.com/en-US/people/sagarwal/hotmobile15-poster.pdf
Accessed 15 February 2016

24 Internet Engineering Task Force (IETF). Hypertext Transfer Protocol Version 2 (HTTP/2), RFC 7540; May 2015. [online]
URL:
https://tools.ietf.org/html/rfc7540
Accessed 15 February 2016

25 Internet Engineering Task Force (IETF). HPACK: Header Compression for HTTP/2, RFC 7541; May 2015. [online]
URL:
https://tools.ietf.org/html/rfc7541
Accessed 15 February 2016

26 IETF. HPACK: Header Compression for HTTP/2. [online]
URL:
https://httpwg.github.io/specs/rfc7541.html
Accessed 15 February 2016

27 IETF. HTTP/2 Frequently Asked Questions. [online]
URL:
https://http2.github.io/faq/
Accessed 15 February 2016

28 Web Technology Surveys. Historical trends in the usage of site elements for websites. [online]
URL:
http://w3techs.com/technologies/history_overview/site_element/all
Accessed 15 February 2016

29 Web Technology Surveys. Usage of SPDY for websites. [online]
URL:
http://w3techs.com/technologies/comparison/ce-http2,ce-spdy
Accessed 15 February 2016

30 Woollaston V. Long battery life was listed as the top reason for buying a phone at 89%. [online]
URL:
http://www.dailymail.co.uk/sciencetech/article-2715860/Mobile-phone-customers-really-want-better-battery-life-waterproof-screens-poll-reveals.html
Accessed 15 February 2016

31 Weldon M.K. The Future X Network; A Bell labs perspective. Boca Ration: CRC Press; 2015.

32 HTTP Archive. Compare stats. December 2015. [online]
URL:
http://httparchive.org/compare.php
Accessed 14 February 2016

33 Wang X, Balasubramanian A, Krishnamurthy A, Wetherall S. How Speedy is SPDY. April 2014. [online]
URL:
https://www.usenix.org/conference/nsdi14/technical-sessions/wang
Accessed 15 February 2016

34 AMPdigest. 20 Top Factors That Impact Website Response Time. May 2015. [online]
URL:
http://www.apmdigest.com/website-response-time-1
Accessed 15 February 2016

35 Erman J, Gopalakrishnan V, Jana R, Ramakrishnan K. Towards a SPDY'ier Mobile Web, AT&T Labs – Research. [online]
URL:
http://conferences.sigcomm.org/co-next/2013/program/p303.pdf
Accessed 20 February 2016

36 Holma H, Toskala A. LTE for UMTS: Evolution to LTE-Advanced, Second Edition. John Wiley & Sons; 2011.

37 Google Code. mod-spdy, Apache SPDY module. [online]
URL:
https://code.google.com/p/mod-spdy/
Accessed 16 January 2016

38 Netty project. [online]
URL:
http://netty.io/
Accessed 16 January 2016

39 Sandvine. Measuring Web Browsing Quality of Experience: Requirements for Gaining Meaningful Insight. [online]
URL:
https://www.sandvine.com/downloads/general/whitepapers/measuring-web-browsing-qoe.pdf
Accessed 6 March 2016