

Master Thesis
Electrical Engineering
August 2013



Impact of HTTP Object Load Time on Web Browsing QoE

Pangkaj Chandra Paul and Golam Furkani Quraishy

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden

This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering. The thesis is equivalent to 20 weeks of full time studies.

This Master Thesis is typeset using L^AT_EX

Contact Information:

Author(1):

Pangkaj Chandra Paul

Address: Minervavägen 20 LGH 1114

371 41 Karlskrona

E-mail: pangkajpaul@gmail.com

Author(2):

Golam Furkani Quraishy

Address: Minervavägen 20 LGH 1114

371 41 Karlskrona

E-mail: utpol.quraishy@gmail.com

University advisor:

Junaid Shaikh

School of Computing

University Examiner:

Dr. Patrik Arlos

School of Computing

School of Computing

Blekinge Institute of Technology

SE-371 79 Karlskrona

Sweden

Internet : www.bth.se/com

Phone : +46 455 38 50 00

Fax : +46 455 38 50 57

Abstract

There has been enormous growth in the usage of web browsing services on the Internet. The prime reason behind this usage is the widespread availability of World Wide Web (WWW), which has enabled users to easily utilize different content types such as e-mail, e-commerce, online news, social media and video streaming services by just deploying a web browser on their platform. However, the delivery of these content types often suffers from the delays due to sudden disturbances in networks, resulting in the increased waiting time for the users. As a consequence of this, the user Quality of Experience (QoE) may drop significantly. Therefore, in order to provide successful network services, it is important to assess the impact of network disturbances on the Web browsing QoE.

In this work, user subjective tests were conducted in the context of web browsing services for an e-commerce website. The variety of network disturbances were introduced within different key parts of the web browsing transfer, such as DNS resolution, base object and the embedded objects on web pages. The main aim behind this study is to evaluate how the delay affects the QoE when applied within different parts of the transfer.

Keywords: Network Emulation, KauNet, Web-QoE, Web Browsing, Page Load Time, HTTP Object Load Time.

Acknowledgments

First of all, we would like to thank and extend our sincere gratitude to our supervisor Junaid Shaikh. His constant support, guidance and patience at every moment were the driving force in our work. It was a life changing experience. We offer him our deepest appreciation for sharing his ideas and insights, in making this work to reach its shore.

We have learnt a lot from the valuable advices and encouragements of Prof. Dr. Markus Fiedler, very special thanks to him.

We would like to extend our special gratitude to Dr. Patrik Arlos for his valuable support and small burst like talks throughout the work.

In the end, we greatly appreciate and also thank our precious friends, especially Cherry Velanginichakaravathy, Arunkumar Rajasekaran, Shafiqul Islam, Nazrul Islam, Rajib Ahmed, Abdur Razzak, Monir Hossain and Sadh Ibna Rahmat. They supported us tremendously throughout the work and we are very glad to have them as our friends.

Pangkaj Chandra Paul
Golam Furkani Quraishy
August, 2013

Preface

This Master thesis work has been conducted under the Department of Electrical Engineering with emphasis on Telecommunication System, School of Computing, at Blekinge Institute of Technology in Karlskrona, Sweden.

This Thesis includes six chapters which are briefed as follows:

Chapter-1

Chapter 1 discusses about the importance of the work that has been done and why this topic has been chosen for Master Thesis. It also gives demonstration about the research questions and the methodology to answer these questions. This chapter also discusses about the contribution of other researchers in this field till now.

Chapter-2

Chapter 2 discusses about the relevant technical background, which is a prerequisite to understand the analysis of the results.

Chapter-3

Chapter 3 covers what have been used for experimentation and how. It also describes the methodology used to set the experiment and measurement of users QoE.

Chapter-4

Chapter 4 discusses about the extraction and calculation of the data.

Chapter-5

Chapter 5 analyzes the experimental results.

Chapter-6

Chapter 6 concludes the discussion and provides ideas for future work.

Appendix

Appendix covers information about every relevant technical work that has been done.

Contents

Abstract	i
Acknowledgments	ii
Preface	iii
List of Figures	vi
List of Tables	vii
List of Abbreviations	viii
1 Introduction	1
1.1 Motivation	2
1.2 Research Question	3
1.3 Research Methodology	3
1.4 Related Works	4
2 Technical Background	7
2.1 Key Phases of HTTP Object Transfer	7
2.1.1 DNS Lookup	8
2.1.2 TCP Handshake	9
2.1.3 Request for Base HTML File	9
2.1.4 Request for Embedded Objects	9
2.2 Web Browser	10
2.3 Incremental vs All-at-Once Loading	11
2.4 User Perception of Web Application	11
3 Experimental Setup	13
3.1 End-to-End Performance Estimation	13
3.2 Network Setup Design	14
3.3 Equipment Functionalities	15
3.3.1 Measurement Point	15
3.3.2 MArC	16
3.3.3 Consumer	16
3.3.4 Network Emulator	16
3.3.5 Server	16
3.3.6 Client	16

3.4	Application Functionalities	17
3.4.1	Bind9	17
3.4.2	Apache Web Server	17
3.4.3	Codeigniter	18
3.4.4	MySQL	18
3.4.5	Fiddler	18
3.5	Assessment Scale for User Satisfaction	19
3.6	Network Emulation Parameters	19
3.7	Development of MOS Controller	20
3.8	Experiment Process	22
4	Post Processing	23
4.1	Data Processing	23
4.1.1	Network-level Data Processing	23
4.1.2	Application-level Data Processing	24
4.1.3	User Level Data Processing	24
4.2	Data Modeling	24
4.2.1	Data Selection	25
4.2.2	Calculation of Mean and Standard Deviation	25
4.2.3	Calculation of Confidence Interval	26
4.3	Acronym for Sessions	27
5	Analysis and Result	28
5.1	Impact of Delay on Page Load Time	28
5.1.1	Reference Sessions	30
5.1.2	2 Second Delay Sessions	31
5.1.3	4 Second Delay Sessions	32
5.1.4	8 Second Delay Sessions	33
5.2	Impact of Delay on Object Load Time	33
5.3	Comparison between the Delayed Objects	36
6	Conclusions	38
6.1	Answer to the Research Questions	38
6.2	Future Work	39
	Appendix	40
	Bibliography	62

List of Figures

2.1	Key Phases of HTTP Object Transfer	8
2.2	Webkit rendering engine	10
3.1	Basic Experiment System	13
3.2	Framework to obtain end-to-end performance estimate	14
3.3	Experimental setup design	15
3.4	Apache web server workflow	17
3.5	Codeigniter workflow	18
3.6	Fiddler workflow	19
3.7	Automatic Test System	21
3.8	Experiment workflow	22
5.1	Page Load Time vs MOS with 95% CI in 12 sessions	29
5.2	Page Load Time vs MOS with 95% CI in the reference sessions	31
5.3	Page Load Time vs MOS with 95% CI in 2 second delay sessions	31
5.4	Page Load Time vs MOS with 95% CI in 4 second delay sessions	32
5.5	Page Load Time vs MOS with 95% CI in 8 second delay sessions	33
5.6	Network-level Page Load Time divided based on object load time	34
5.7	Application-level Page Load Time divided based on object load time	34
5.8	Comparison between application-level object load times	36

List of Tables

3.1	ITU-T scale for quality impairment	19
3.2	Applied delay parameter for the experiment	20
4.1	Session acronym for analysis	27
D.1	Application, Network PLT and MOS with 95% CI in 12 sessions	44
E.1	Application-level HTTP Object Load Time in 12 sessions . .	45
E.2	Network-level HTTP Object Load Time in 12 sessions	45
F.1	The statistical outlier subjects in the experiment	46

List of Abbreviations

ACR	Absolute Category Rating
BIND	Berkley Internet Naming Daemon
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DNS	Domain Name System
DOM	Document Object Model
DOM	Document Object Model
DPMI	Distributed Passive Measurement Infrastructure
DAG	Central Processing Unit
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
ISPs	Internet Service Providers
ITU-T	ITU Telecommunication Standardization Sector
JSON	JavaScript Object Notation
MARc	Measurement Area Control
MOS	Mean Opinion Score
MP	Measurement Point

MVC	Model-View-Controller
PHP	Hypertext Preprocessor
CPU	Central Processing Unit
PLT	Page Load Time
QoE	Quality of Experience
QoS	Quality of Service
RDBMS	Relational Database Management System
RTT	Round Trip Time
SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URL	Uniform Resource Locator
WinHTTP	Microsoft Windows HTTP Services
WinINET	Windows Internet
XML	Extensible Markup Language

Chapter 1

Introduction

The massive growth in the number of users connected to the Internet and the emergence of bandwidth-hungry applications increased the Internet traffic enormously. There are 2.3 billion people using the Internet now and the usage of the Internet traffic increased sevenfold over the last five years [1]. As a result, efficient management of the Internet resources has become one of the key issues along with the development of new technologies. Internet Service Providers and network administrators have to deal with the increasing user base and changing network traffic for various applications.

Quality and availability are the major indicators for any service to succeed. It is a common practice to measure these issues based on conventional network performance parameters, Quality of Service (QoS) (i.e., network path quality). Measurement of QoS does not provide the full insight into the user perception for any service. User perception can be measured with a new criterion called the Quality of Experience (QoE). QoE measures the degree of end user satisfaction for a particular service. It is a subjective measurement, thus varies from user to user. Also, gathering information about QoE from user is expensive and time consuming. The challenge is to find out the correlation between objective network QoS with subjective user QoE for a particular service [2]. Service providers can adjust their resources based on these findings in order to provide more efficient services.

Usage of the service on the Internet is characterized by the requests from the user and responses from the server. The response from the server for a web page or a video might get delayed to a certain amount of time for server processing, network disturbance, client's own machine, etc. These waiting times and corresponding user perception differ from service to service. Most work in the context of QoE measurement is done for the video services. There are two major issues that the service providers take into account for a video service, these are the quality of the video and the buffering time. Hypertext Transfer Protocol (HTTP) [3] based video services emerged for managing the quality of the video. HTTP uses the Transmission Control Protocol (TCP) as a transport protocol which guarantees the packet delivery. TCP expects an acknowledgement for each packet and retransmit the packet if there is any

loss. As a result, HTTP traffic is playing a key role behind the growth of the Internet traffic [1]. The buffering still remains as an issue for video based services. There are many studies which relate the QoS to the QoE in video based services as the characteristics of video based services are well studied. But unlike video, web applications and services has many different characteristics. For instance, for different types of objects, the required network condition varies within a single web based application or service.

These HTTP/TCP based applications and web services are mainly accessed by the users through web browsing activities. A secure version of HTTP named HTTPS is mostly in use nowadays. It simply uses HTTP over Transport Layer Security (TLS) protocol, exactly as HTTP over TCP [4]. The wide ranges of applications and services over HTTP have different traffic characteristics and performance requirements. Despite the differences, every web browsing activity can be divided into four particular phases in the application-level. These phases are DNS Lookup, TCP Handshake, HTTP GET for Base HyperText Markup Language (HTML) file and HTTP GET for embedded objects. HTTPS has the same phases as HTTP with an extra phase to provide the data encryption for security. On the other hand, web browsing suffers from various network impairments. These impairments increase the loading time of HTTP objects thus increases the overall Page Load Time (PLT).

This research focuses on the phases of the HTTP object transfer and how these phases potentially affect the overall Page Load Time and as well as the user QoE if delay is occurring on the network. An e-commerce website has been designed and placed on an Apache web server. On the same machine, a DNS server has been deployed to resolve multiple DNS requests. A network emulator has been used for shaping the network which does not reorder the data packets. Taking user rating score is a bit difficult in web browsing; a tool has been developed to take the score without affecting the users browsing experience. A distributed passive measurement infrastructure has been used to capture packets in the network-level. A web debugging proxy tool was used to capture information about the HTTP objects at the application-level.

The result shows that same amount of delay on different phases of HTTP web transfer generate different Page Load Times at user end. Also, user perception of waiting times vary based on these phases.

1.1 Motivation

Essential services which are used by many people for everyday purpose are being hosted on the web for more accessibility and usability. Users are accessing these services mainly by web browsing. ISPs and web services are eager to serve their users with better quality as the quality is the deciding factor for a user to choose among the services. Moreover, using resources efficiently is always a good practice. Web browsing have some common phases during the

transfer. Management of Internet resources during a web browsing based on the phases of HTTP object transfer can improve the efficiency. This will lead to a better end user experience. In order to do that, finding out the influence of network impairments on the user QoE in different phases of HTTP object transfer can be crucial; hence, it is worthwhile to look into this factor.

1.2 Research Question

The research questions addressed throughout this thesis are listed as follows:

1. How the increased DNS lookup time affects the Page Load Time and the user QoE?
2. How the increased Base HTML Object loading time affects the Page Load Time and the user QoE?
3. How the increased Embedded Object loading time affects the Page Load Time and the user QoE?
4. Which delayed HTTP object mostly affects the user QoE?

1.3 Research Methodology

In order to answer the research questions, the existing research methods had been utilized in a controlled manner. A literature review had been performed to identify the phases of HTTP object transfer and to find out the key network-level performance parameters that affect the user QoE in web browsing activities. Then all the tools used in the experiment had been studied and tested separately. Then a preliminary experiment setup was designed which include the hardware setup and software configurations. A controller was developed to keep the whole experiment process automated. After implementing the preliminary design it had been verified with sufficient amount of dry runs. The setup had been modified and tested again until it was suitable for our hypothesis.

As a result, finally, a client-server based web browsing experiment had been setup in a controlled environment within the framework of the experiment controller and the measurement point which captures the packets from the network. An e-commerce website had been designed and deployed to give the user a real world task in the experiment. A selected amount of delay had been applied on three different phases of the HTTP object transfer. The experiment was followed by a subjective user study where user had to answer 3 specific questions (Appendix G). The data had been collected from the network and application-level in order to collect the object load time and Page Load Times. User ratings had been stored in a MySQL database.

After the completion of the experiment, all the network, application and the user level data had been gathered carefully. A combination of Perl and PHP script had been used to extract the information from network and application-level data. After the analysis, how the delayed HTTP object affects the Page Load Times had been shown for both levels. Also, the effect of delay on the HTTP objects had shown on the application-level. Finally, the extracted result had been plotted using appropriate graphs.

1.4 Related Works

QoE is still an open issue for HTTP-based web services which are accessed via a web browser. Hosfeld *et al.* [5], defined this research area as Web-QoE. They defined Web-QoE as Quality of Experience of interactive services that are based on the HTTP protocol and accessed via a browser. There are many variables that span over different disciplinary areas that can be influential in the determination of Web-QoE. Ensuring a better QoE for web services has been one of the most important research hot topics in recent years. New methods and techniques are being developed in this field for QoE assured service delivery. ITU-T Study Group 12, a standardization body is working for extending the existing recommendation ITU-T G.1030 [6], which will be a major guideline for setup and conducting experiments to find out network based models for Web-QoE measurement.

The general goal of the QoE study is to relate the perceived quality of a service to the network QoS parameters. This will allow more efficient distribution of Internet resources. Ibarrola *et al.* [7], found a strong correlation between QoS parameters and QoE scores as suggested in the ITU-T G.1030. They validate this existing recommendation and discussed a multi-agent network scenario for the web browsing experiments. In order to update existing recommendation to the present user requirements, they created a dynamically changing network environment to mimic the real world network.

In another study [2], the authors present a comprehensive analysis to evaluate the effect of different service performance parameters on the objective and subjective user behavior. They correlated a set of service performance parameters and network performance metrics for web application. Shaikh *et al.* [8] mentioned wireless network traffic is disrupted in a short-term manner quite frequently and causes longer waiting times for the web browsing. The authors also discussed the characteristics of this short-term disruption. These gaps may occur due to the user inactivity or by the network. They identified and distinguished these traffic gaps using wave-let analysis.

Collange *et al.* [9], propose a new empirical method to evaluate the traffic characteristics of the web applications sensitiveness and performance requirements. They provide correlations between various traffic features and the network performance. If application's usage forecasts are also taken into account, it may help to improve the prediction of QoE. The general relation-

ship between the waiting time and task driven user expectation is studied by Galletta *et al.* [10]. They conducted an experiment on 160 participants to find out the behavioral intentions of the user. The effect of the network delay on the user perception of an e-commerce website has been studied by Bhatti *et al.* [11]. The authors in [12, 13, 14], described the need of a quantitative relationship between network-level QoS and end user QoE. Fiedler *et al.* [12], proposed a generic formula named IQX hypothesis which establishes a logarithmic relationship between QoS and QoE parameters.

Though, the majority of the current QoE measurement models only considers the current applied stimulies, the user perception of the network performance does not depend only on the QoS parameters. According to Egger *et al.* [15], the other novel metrics need to be considered for reliable modeling of Web-QoE. Identification and measurement of these factors are crucial to the modeling of Web-QoE. Also, accuracy of the network emulation is a crucial factor in the QoE experiment. Minhas *et al.* [16] and Shaikh *et al.* [17] evaluated the Throughput and Delay performance of popular traffic shapers respectively. They also performed experiments on two hardware platforms (AMD and Intel) to study the effect of hardware on the shaping.

The research on QoE has so far been dominated by multimedia services. In [18], the conceptual model of QoE in parallel to the classical Internet architecture hourglass model is described. The QoE hourglass model is based on four layers, and each layer has a role towards the user experience (QoE). Minhas *et al.* [19] proposed the Coefficient of Throughput Variation (CoTV) as a network performance parameter. CoTV can quantify the packet delay variation and predict the changes in the network behavior. In [20], the authors described the mobile video sensitivity packet loss and packet delay variation in terms of QoE. Minhas *et al.* in [21], found that the location of disturbance of any video affects the user rating in mobile video streaming.

HTTP based video streaming is taking over the Internet traffic share. Several studies have been done to correlate the network QoS to user QoE for HTTP video streaming based on temporal structure. Huynh-Thu *et al.* In [22] investigated the temporal aspect of the video in case of jitter and jerkiness to measure the QoE. They concluded that jitter affects the QoE more than jerkiness. In [23], they proposed a no-reference temporal quality metric to measure the effect of jitter on QoE. In [24], the authors correlate the network bottlenecks with QoE for different kinds of services in terms of passive measurement on an aggregated network link. Mok *et al.* [25] divided their experiments into two parts. They build a mathematical model to determine the re-buffering duration and frequency. They mapped the network QoS to application QoS by collecting the network parameter from their own flash player. They mapped the application QoS to user QoS (i.e. QoE) by performing the subjective experiment for collecting user MOS. Finally, they showed the correlation between network QoS and QoE by plotting the findings on a radar chart.

HTTP uses TCP as a transport layer protocol. The behaviors of TCP in video streaming is well studied. As HTTP and TCP is the backbone of World Wide Web, it is important to find out the behaviors of TCP in web browsing activities. Shaikh *et al.* [26], discussed about two methods. One is for the observation of TCP connection interruption and the other is for traffic gap identification in the web transfers. TCP connection interruption behaviors and its effect on user QoE is also studied by [27, 28, 29]. The other TCP factors studied by [30, 31, 32]. Nguyen *et al.* [33], discussed that the ITU-P.800 model based on a Mean Opinion Score (MOS) is reflecting the experience of a user more directly and giving a good match to a users behavior. But, MOS may not be a complete assessment user experience. They proposed a new model to measure the user interaction with the system rather than just an opinion via MOS. Schatz *et al.* [34], studied the effect of bandwidth variation to the user ratings for different websites. According to Hosfeld *et al.* [5], the existing QoE models do not consider temporal dynamics or historical experiences of the user's which they referred to as the memory effect. They propose to consider the memory effect as a key influence factor for Web QoE modeling. They derived a new QoE model for web, considering the temporal dynamics and memory effect.

On the other hand, traditional network impairment often causes packet loss, packet delay and low bandwidth which increases the Page Load Times in an unacceptable manner [11, 35, 36]. The QoE studies [5, 15], selected 2, 4, 8, 12, and 16 seconds as the page load duration of the experiment. A variation of the delay in the factor of 2 is a common practice found in several studies. The visualization of the webpage is also a consideration to take into account. The authors of [37, 38], consider the first visual sign of progress as a relevant metric for Web-QoE study. Also, The incremental loading of website blurs the latency with rendering process [39]. Huang *et al.* [40], come up with a benchmark for evaluating performance of web applications on Smartphone and cellular data networks. They also focus on the performance bottleneck in different phases of HTTP transfer. Strohmeier *et al.* [41], explores the impact of the users task, information presentation and page load strategies on Web-QoE. They state the importance of developing task-aware and content-aware solutions for improving QoE of delay sensitive web services. Egger *et al.* [42], discussed about the relationship between waiting times and the user perception of web services.

Network impairments can cause waiting times during a web browsing session. Waiting times in different phases of a web browsing session affect the user QoE in a different manner. Contemporary QoE evaluation models for web browsing usually do not take into account the specific timing of the network outage on a web transfer. It can be a crucial investigation to find out the effects of network outage during different phases of HTTP object transfer for a more accurate measurement of Web-QoE.

Chapter 2

Technical Background

This chapter provides the technical background essential for understanding of this thesis. The general way to access the web services for the Internet user is through web browsing activities. The user uses a web browser to request some resources from a web server. Usually, a TCP connection is established between the server and the client for guaranteed transfer of all packets. The types of the content transferred during web browsing are diverse in characteristics. Different types of content need different network performance optimization for obtaining a better end user QoE. On the other hand, web browsing activities have some common phases in the transfer of the payload in any communication device [43].

2.1 Key Phases of HTTP Object Transfer

In web browsing, users transfer a number of HTTP Objects from Server to their web browser. These transfers take place within a few particular phases. These phases are DNS Lookup, TCP Handshake, request for Base HTML and request for Embedded Objects. According to ITU-T G.1030 [5], all the packets in a web transfer are divided into two phases, the TCP Handshake and the rest of the data packets. However, TCP Handshake takes place only after the DNS get resolved successfully. Also, the rest of the data packets can be divided into two parts. Few packets need to be transferred as soon as the Handshake between the Server and the browser (client) is performed. These packets are supposed to transfer the Base HTML file which enables the browser to request for the Embedded Objects in it. Figure 2.1 briefly illustrates the above mentioned phases of HTTP Object transfer.

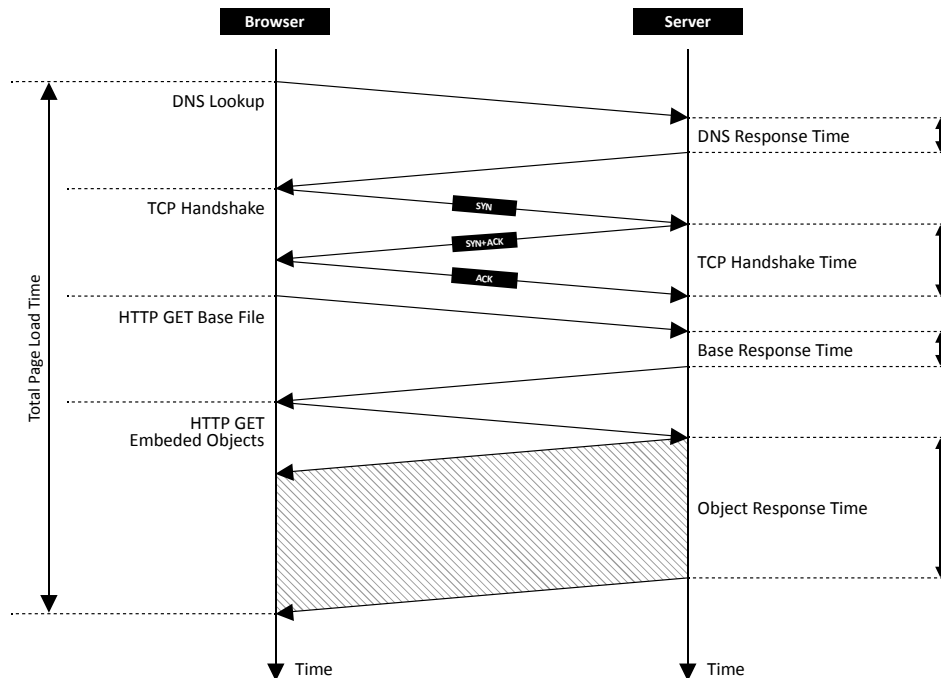


Figure 2.1: Key Phases of HTTP Object Transfer [6]

2.1.1 DNS Lookup

This section focuses on the inner workings of DNS lookup. To access a website from any browser, a user generally refer to the web address by name, not by IP address. This named web address is then queried to a DNS server to get the server IP for the particular web site. For a valid DNS query there must be three pieces of information attached to the query, these are:

- A domain name. e.g. bth.se
- Type of query. e.g. HTTP GET, HTTP POST.
- Class of query. e.g. INTERNET, CSNET, CHAOS.

DNS queries are resolved in a few different ways. A browser can resolve a query locally from the cached information in the browser or OS cache from a previous lookup. When a local query is unsuccessful then it is passed to a DNS server. The DNS server first tries to answer the query from own cache of the resource record. A DNS server can also pass query or communicate with other DNS servers to resolve the name and then send a response back. This process is called recursion. In addition, the client itself can contact multiple DNS servers to resolve a name. When a client does so, the process is called as iteration [44].

2.1.2 TCP Handshake

TCP Handshake is a crucial phase of a web transfer. This is a 3-Way Handshake which is performed by a server and a client to establish a TCP connection. This connection establish a stream between server and client which guarantees every packet to be delivered. Many connection parameters which are compulsory for the TCP connection establishment and determine the transfer quality get exchanged during this event. HTTP uses multiple TCP connections to deliver a website. If any packet gets lost during this phase, the initial connection establishment time increased significantly. Because, when the Round Trip Time (RTT) between the server and client is higher than average, the time TCP takes for Handshake become crucial. This issue has been mitigated by using HTTP pipelining which reuses existing TCP connections. Also, a new protocol named “SPEEDY” is being developed which handles all the request from a client through a single TCP connection which may reduce the waiting time for the user.

2.1.3 Request for Base HTML File

Base HTML file is the first response from any web server for the first HTTP GET request. Upon getting a request from a web browser, the server usually invokes the appropriate software for generating the response. This first response is usually a HTML file which is denoted as Base HTML in this thesis. Base HTML file is needed by the web browser render engine to create the Document Object Model (DOM) (explained in 3.2) for building the skeleton of the requested page. During the creation of the DOM, the browser passes the request to the network when it comes across any node in the DOM tree which needs to be fetched from server. The server can be the same server where the first request has been sent or any other server dependent on the URL of that object. These requests consist all the other objects essential to display and completion of the loading of that particular page. The browser takes some time to create the DOM, process all the other requests and display the content on the screen. So, this phase is very important from the user perception as this phase is responsible for the first sign of service delivery.

2.1.4 Request for Embedded Objects

Embedded Objects are the Nodes in the DOM tree which need to load from a web server. When the browser comes across with any such Node, it straight away sends a request down to the network-level to get that object from the specified address. As soon as the object gets transferred, browser shows the object on the screen. This phase has a specific significance if the users are browsing on a task-driven mode. Network disturbance can hold any Embedded Object, but if that specific object is the desired object for the user, the QoE may degrade significantly. Also, often a small delayed object at the end of the page responsible for the huge increased of overall Page

Load Time. The web browsing is an immersive flow experience which is not limited to only one single page viewed by a user. The Embedded Objects are the contents of the currently requested page. Also, these are the essential elements for the users' decision about moving to the next page. The loading of the Embedded Objects shows the fluidity of any web site. Which is an important factor for determining the user perception of that particular web site.

2.2 Web Browser

There are various web browsers available for use. The most popular web browser today is Google Chrome. Chrome is used by more than 50% of the Internet user [45]. This web browser uses Webkit as a rendering engine [46]. The WebKit is designed for web browsers to render web content. It is also used inside the Apple Safari and Opera web browser.

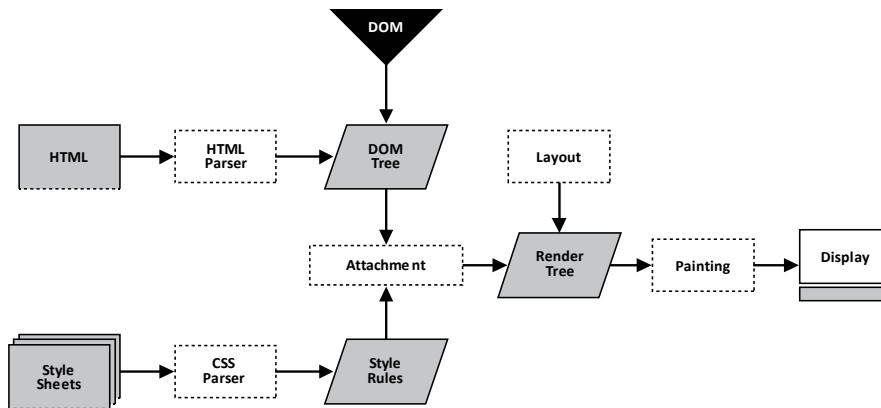


Figure 2.2: Webkit rendering engine [47]

All modern browser's implementation shares the first two components of Webkit engine. Later ones are more operating system dependent. Webkit has the following components:

- Parsing (HTML, XML, CSS, JavaScript)
- Layout
- Text and graphics rendering
- Image decoding
- GPU interaction
- Network access
- Hardware acceleration

Figure 2.2 gives a detailed workflow of Webkit. When a browser which has Webkit as a render engine receives the HTML content from a web server,

following tasks get performed by the Webkit. Firstly, the Base HTML is parsed by the parser into a DOM tree, a hierarchical model. Each element in the DOM tree is a Node. These Nodes consist of objects and texts. These are the HTML objects (ex. images) and texts are the displayable characters. If any Node contains any object or text that needs to be loaded from a URL, the rendering engine then passes those requests to the network-level. These external objects are defined as the Embedded Objects. Then the rendering component takes over and starts rendering the web page with different components. The rendering takes place by generating a tree object similar to the DOM tree. All the objects from the DOM tree and additional objects are packed into a rendering tree. Rendering starts by attaching the style information for the specific DOM objects. This attachment process is recursive with a top down approach. These processed objects then displayed on the user's screen. JavaScript is also considered at the same time as it is written for manipulation of the DOM objects. The rendered objects are destroyed when the viewing window is closed.

2.3 “Incremental” vs “All-at-Once” Loading

There are two types of loading technologies available for the web browsers, “Incremental” and “All-at-Once”. Google Chrome uses the incremental loading methodology [48]. This browser has the capability to display the partially downloaded website. In all at-once-method, all the delay is interpreted by the user as initial delay. The browser displays the web page into the screen after downloading the page completely. So, any delay during the transfer of that page is added up to the initial delay. On the other hand, partially loaded websites blur the user waiting time with the rendering process [39]. It is important to show the first visual sign of transfer to the user. Moreover, in the experimental process, the incremental loading is an essential feature to visualize the effect of various amounts of delay in different phases.

2.4 User Perception of Web Application

Usually any service should be served as fast as possible. Interruption during the service consumption may occur due to network impairments or unavailability of the particular service. There is always a very small time scale disturbance which goes unnoticed by the naked human eye. Services can be very fast also where human cannot cope up with the speed. The timing of any service should keep the human perception factor in consideration. There are three main time limits to keep in mind for web users [49].

Instantaneous experience: 0.1 second is about the time limit experienced by the user which gives the impression that the system is reacting instantaneously. This time limit is very important for instant messaging services.

Uninterrupted experience: 1.0 second is about the limit that the user flow of thoughts stays the loading is uninterrupted. Generally, the user does not notice any delay less than 1 second. In case of more than 1 second delay user feels that the service is not being served instantaneously.

Loss of attention: 10 seconds is about the time limit when user usually thinking to leave the service. In some cases, user moves to another tab and let the service to be loaded. But in the most cases, this is now the time limit for the user to quit. This time limit has been updated to 8 second [42] with respect to the present day scenario.

Chapter 3

Experimental Setup

In order to investigate the user perception of web browsing for different network variations, an experiment setup was designed. Three main computers were used to deliver an e-commerce websites to the user. A server where the website was hosted, a network emulator for emulating network conditions and a client where the site was viewed by the user using a web browser.

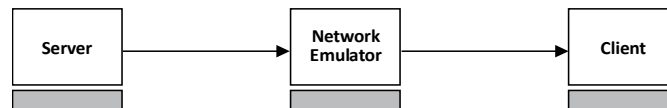


Figure 3.1: Basic Experiment System

Figure 3.1 illustrates the three main components of experiment system and their connections. This chapter will provide detailed description about the experimental process, including the tools and techniques, which were deployed to perform the user subjective tests.

3.1 End-to-End Performance Estimation

According to ITU-T Recommendation G.1030 framework, estimation of end-to-end performance in packet networks depends on various process which is illustrated in Figure 3.2 [6]. This framework estimates end-to-end performance using below steps:

- Network Performance was assessed from two principal information sources: measurement and modeling. In this experiment, DPMI [50] was used to assess the packet transfer performance.
- The goal of the Application Performance Assessment was to produce a key metric of application performance based on estimated network performance and information. This metric would be used to develop a model for measuring user QoE considering the governing protocol in use with specified options.

- Perceptual models were used to obtain desired end-to-end performance. These models interpreted users interest of an application performance to an estimate of quality experienced by the users.

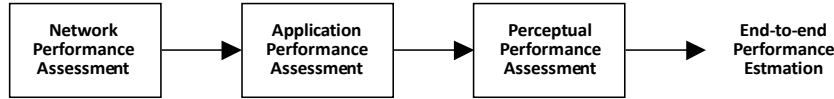


Figure 3.2: Framework to obtain end-to-end performance estimate [6]

3.2 Network Setup Design

A client-server model was implemented to conduct the experiment. The server and client were prepared with Ubuntu 10.10 32 bit and Window7, respectively. The packet transfer between server and client was going through a network emulator and a measurement point consist of two Endace DAG 3.5E cards [50]. All devices were connected using a 10 Mbps link as the DAG cards were configured to capture packet from 10Base-T link. The server was configured with PHP [51] (version 5.3) and Apache [52] (version 2.1) to act as a web server. A DNS server was also implemented on the same machine for translating users requested URL to IP address. The widely used free open-source software for Linux systems, Bind9 was installed to setup the DNS server. The server machine worked as a web server and as well as a DNS server. The client was on Windows operating system. Windows was the most popular operating system among the Internet users [53]. This is the reason behind choosing Windows. The client was also equipped with Google Chrome browser to provide the users a familiar environment. Google Chrome was used as a web browser by more than 50% Internet users around the world [45].

The network was controlled by a network emulator. The emulation was done by using KauNet as it provides packet driven deterministic network emulation and did not reorder packets [54]. This client-server model was connected to a Distributed Passive Measurement Infrastructure (DPMI) [50] via a machine named Measurement Point (MP). This MP was equipped with two DAG cards as mentioned earlier in this chapter. Each DAG card had two transceivers. The server is connected to the DAG transceivers d10 and client is connected to d20 through a router. The other two transceivers d11 and d21 were connected to network emulation as illustrated in Figure 3.3.

The Measurement Point captured the packets between server and client which were going through the network emulator at two points, before and after the network emulator. This was done for the validation of the network emulator that the emulator was applying the delay accurately. This distributed architecture had a controller named “MARC” [50] which provides basic instruction to the MP (details in 4.4.1) for capturing packets. Another

machine named Consumer collected the captured packets from MP and stored into an appropriate file format (details in 4.4.3). An application-level object information capture tool named Fiddler [50] (details in 4.5.5) was installed on the client machine to capture the information about the HTTP object transfer for further analysis. Moreover, the client was also setup as an experiment controller to automate the experiment process (details in 4.5.8). The red line illustrated in Figure 3.3 is indicating the control connections for the automated experiment controller.

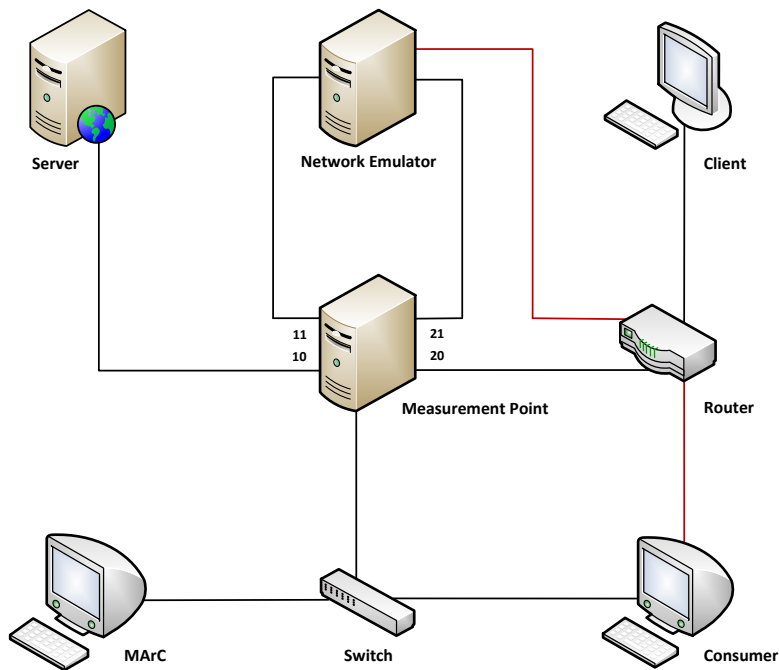


Figure 3.3: Experimental setup design

3.3 Equipment Functionalities

3.3.1 Measurement Point

The function of the Measurement Point (MP) was to capture packets based on the filter set by MArc. The MP collects packet before and after network emulator in both directions. MP was equipped with two Endace DAG 3.5E cards. These cards were synchronized using a TDS-2 connected to a Acutime Global Positioning System (GPS) antenna to obtain an accurate timestamp up to a resolution of 60 ns [55]. Finally, MP sent the captured packets to consumers.

3.3.2 MArC

Measurement Area Control (MArC) was the core part in DPML. MArC provides the set of rules for MP. The rules decided the what, when and where to capture packets. MArC used the user supplied rules for specific projects and translates into the MPs understandable format. If any rule became a performance issue for MP, it had the ability to modify that rule [50].

3.3.3 Consumer

A Linux based system namely Consumer was used to receive the incoming packets from MP. The Consumer was connected with MP using a switch. It was used for collecting the packets and saving into the CAP file formats.

3.3.4 Network Emulator

In this experiment, client and server communicated with each other via a network emulator. This emulator works as a gateway between server and client. The client and server were placed in two different subnets to route traffic through the network emulator. A Linux version emulation tool named KauNet was placed on that machine to emulate the network. KauNet is an open-source emulator tool implemented based on FreeBSD 7.3. It was developed by Johan Garcia, Per Hurtig and Anna Brunstrm of Karlstad University. KauNet improves Dummynets approach by adding deterministic restrictions on network parameter such as: bandwidth, packet loss, bit errors, packet reordering and delay change. By providing pattern generated file, it can control network behavior based on per-packet or per-milliseconds [54].

3.3.5 Server

The server was setup both as a web server and a DNS server. An e-commerce website had been deployed in the server. This website had 3 pages for the users to buy a product. The DNS server was prepared to resolve 12 domain names for 12 different sessions. The users were browsing the same website in each session under 12 different domain names. As this thesis is investigating the effect of the delayed DNS lookup object along with other two HTTP objects, DNS resolving was required for each of the 12 sessions.

3.3.6 Client

When a user request a web address through a web browser, the browser first checks its own cache and then the local OS cache for DNS resolution as well as other recourses respectively. This experiment needed to send the DNS and other consecutive requests every time to the server. In order to ensure that, the DNS cache of the local OS was disabled to restrict any caching of previous DNS lookup results. Also, the web browser in use, the Google

Chrome was set to the “Incognito” [56] mode. This mode does not cache any object as it deletes everything fetched in the current session when the browser window is closed. The client had requested everything to the server in each session.

3.4 Application Functionalities

3.4.1 Bind9

Domain Name System (DNS) translates domain names to numerical IP address which helps to locate computer services and devices worldwide. This system works as a phone book in the Internet by translating human-friendly hostnames to IP addresses. To enable this service in this experiment, a Linux based free open-source software named Berkley Internet Naming Daemon (Bind - Version 9) [57] was installed in the server machine. In the present day, Bind9 is widely used to provide DNS service. It is very easy to install and configure. A sample configuration snippet is added in Appendix A.

3.4.2 Apache Web Server

In a client-server model, the web server plays a vital role in communication. Apache is the most popular web server [58] which is an open-source software and available for every major OS. The main role of a web server is taking HTTP (S) requests from a client and reply with HTTP (S) response. Usually a web server serves all these responses in the form of web page consisting of static and dynamic contents [52]. The workflow of Apache web server is shown in Figure 3.4.

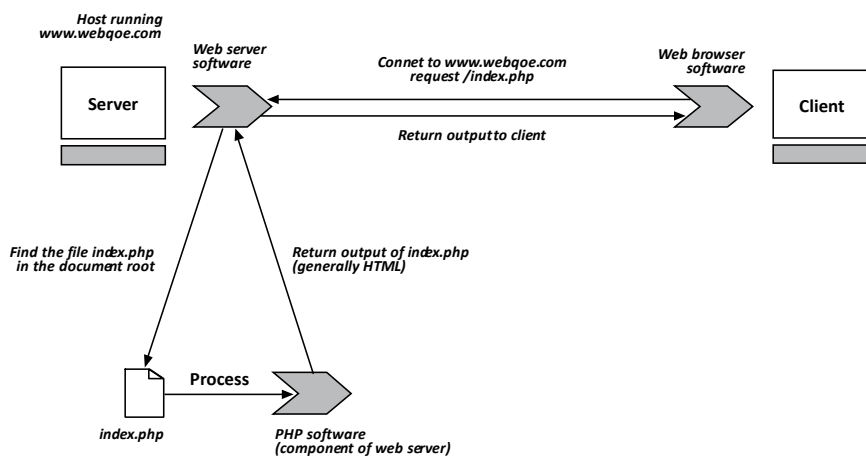


Figure 3.4: Apache web server workflow

3.4.3 Codeigniter

Codeigniter is an open-source, lightweight, powerful web application framework to build a full-featured dynamic websites with PHP [59]. The initial goal of Codeigniter was to provide a rich set of libraries and helpers to developers for developing a website from scratch code. Codeigniter develops based on the popular Model-View-Controller (MVC) framework. From the name of MVC it clearly implies that this design pattern (Figure 3.5) will allow developers to separate code into three parts:

- The model will maintain database interaction
- Viewers will be used for show data in the user interface
- Controllers will handle user interaction that affects model and views

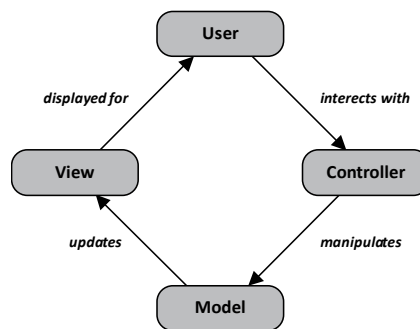


Figure 3.5: Codeigniter workflow

3.4.4 MySQL

Database software is a computer program which is designed to store, find and modify data in an efficient way. MySQL is a free and open-source relational database management system (RDBMS) tool [60]. Web developers can develop web applications which can serve thousands of users per second by accessing terabytes of data by using MySQL. MySQL is popular for easy installation, small size and faster speed, easy-to-use interface with other programming languages and many more.

3.4.5 Fiddler

Fiddler is an open-source web debugging proxy tool for Windows platform which will capture all HTTP (S) traffic between the client computer and the Internet [61]. It is a very handy tool for web developer for debugging web transfers. It also provides various features like inspection of traffic, analysis of incoming and outgoing data, display a time line for web transfers. Fiddler can capture traffic using WinHTTP, WinINET and WebSocket. Figure 3.6 illustrates the workflow of Fiddler for Google Chrome as it uses WinINET. It can

be used to debug traffic from any application like Internet Explorer, Google Chrome, Mozilla Firefox, Apple Safari and more which supports proxy.

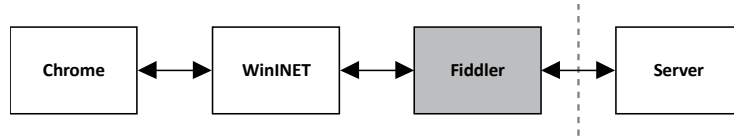


Figure 3.6: Fiddler workflow

It is also used to capture traffic from Windows Phone, iPhone/iPad/iPod as a proxy server. The session transfer info can be exported into both .saz(zip compatible) and HTTP archive (.har) file format.

3.5 Assessment Scale for User Satisfaction

ACR rating scale recommended by ITU-T P.800 [62], was used to collect the user satisfaction about how they perceive any service. In the experiment, a user gave MOS using this five point scale. The scale was defined as, Excellent (5), Good (4), Fair (3), Poor (2) and Bad (1) as detailed in Table 3.1.

Table 3.1: ITU-T scale for quality impairment

Scale	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Good Perceptible, but not annoying
3	Fair	Fair Slightly, annoying
2	Poor	Poor Annoying
1	Bad	Very Annoying

3.6 Network Emulation Parameters

There were total 12 sessions in the experiment. Delay had been applied in 9 sessions while the other 3 were the reference session without any delay. The reference sessions were Session 1, 5 and 9. After each reference session, there were three sessions, where network disturbances were introduced. The delay was introduced respectively to the DNS Lookup, Base HTML and Embedded Object phase on the first page of the 3 page e-commerce web site as shown in the Table 3.2. Each time after three disturbed sessions, reference session was repeated once again.

The web page was fetched several times without any delay to count the number of the packets in the first page of the 3 page e-commerce web site. The packets were captured by the MP in these dry runs of the experiment. Also, packets were captured by Wireshark in server and client machine. The transfer of the packets was analyzed and compared after capturing from these three points. The goal of these dry runs was to identify the considered phases of the HTTP object transfer by the serial number of the packets.

KauNet can apply the delay on any specific packet. There was a need to identify the serial number of the packet for that reason. The number of packets was varying in every transfer but the variance was very small. Specific packets were identified for each phase of the transfer with careful observation. This process became a bit easier at the end because the serial number of the packet for DNS Lookup, TCP Handshake and Base HTML file were static in all of the transfer. The rest of the packets were belonging to the Embedded Objects phase. The experiment was tested by applying the following delay on the chosen packets. The experiment was modified until getting the desired effect.

Table 3.2: Applied delay parameter for the experiment

Session Name	Applied Network Delay
Session 1	No delay
Session 2	2 sec delay in DNS Lookup
Session 3	2 sec delay in Base HTML
Session 4	2 sec delay in Embedded object
Session 5	No delay
Session 6	4 sec delay in DNS Lookup
Session 7	4 sec delay in Base HTML
Session 8	4 sec delay in Embedded object
Session 9	No delay
Session 10	8 sec delay in DNS Lookup
Session 11	8 sec delay in Base HTML
Session 12	8 sec delay in Embedded object

3.7 Development of MOS Controller

An experiment controller named MOS Controller was developed and setup to automate the experiment process. This controller was primarily managed

by a PHP framework named Codeigniter. The design and workflow of the MOS Controller is illustrated in Figure 3.7.

A set of operations are needed to be executed for each user session in the experiment. Changing the network emulator settings, starting the Consumer to collect the experimental data and saving the user given scores in the database. Web browsing is an intense flow of experience where the user goes through a number of pages to complete the desired task. The essential operations needed to be performed during the experiment might disturb the flow of user experience during the experiment. The experiment controller overcome this issue.

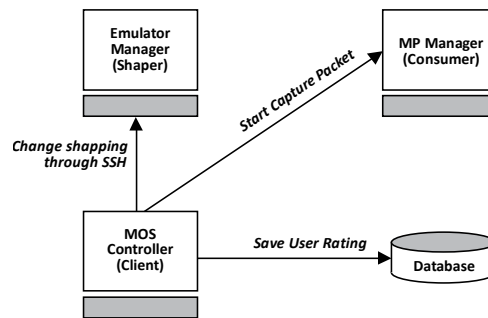


Figure 3.7: Automatic Test System

The Emulator Manager and MP Manager were also developed to assist the MOS Controller. The Emulator Manager was developed by using bash script which takes a delay pattern as an input parameter from the experiment controller through SSH. The MOS Controller used a text file to manage the session ID and based on that session ID, it loaded the delay pattern model as described in Table 3.2. Two bash scripts were developed in Emulator Manager. One used for flushing all the network shaping rules while the other used for applying the new rules. A code snippet for both shell script is available in Appendix B.

The MP Manager was developed to capture packets in the Consumer for the current user session. The MOS Controller called a PHP script by using Output-Buffering technique to execute a shell script. Output-Buffering technique is used to write anything in output buffer. A command can be given to a remote machine using this technique without waiting for the response of that particular command. This technique helped MOS Controller to collect all the packets for the current user session without placing its call in waiting. A code snippet for both PHP and the shell script is available in Appendix B.

In conclusion, the MOS Controller was developed to handle both of the Managers. It also controlled the whole experiment flow as well as saved user MOS into the MySQL database.

3.8 Experiment Process

A task-driven experimental procedure was setup to investigate the user perception in web browsing. Users were browsing an e-commerce website to buy a product in the experiment (described in Table 3.2). The website and network scenario will be same for all users. Initially, a training session was conducted for 5 minutes to give an introduction about the experiment process. During this session all the essential steps were explained to the users.

A sign up page was used at the beginning of the experiment to take user information and save into database. The user had to give ratings for the first page of each session. These ratings were stored in a MySQL database in the client machine. A schematic workflow of a complete web browsing session for one user is illustrated in Figure 3.8.

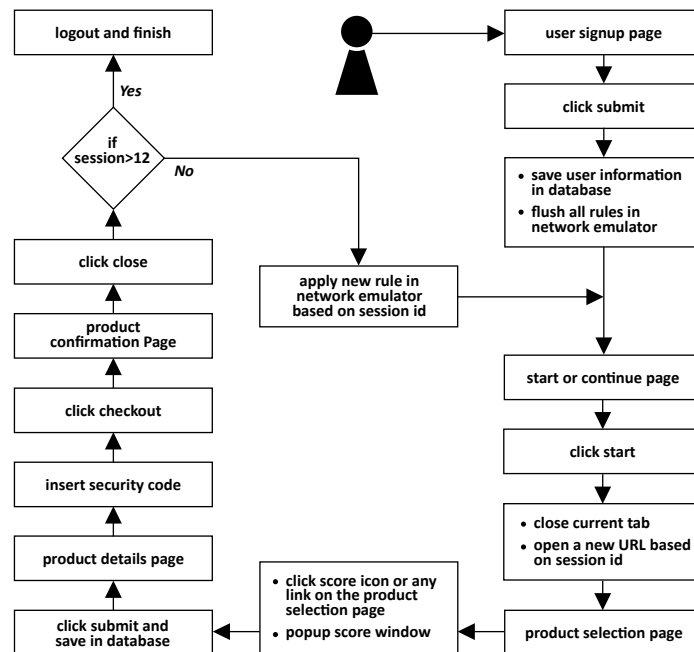


Figure 3.8: Experiment workflow

The e-commerce website was designed and developed to mimic the current e-commerce trend for giving the user a familiar experience. This website consists of 3 pages where the first page was a product selection page where the user had to select a product. On the second page, detailed description about the selected product had been shown. This page also contained a checkout process. To complete the checkout process, the user needed to insert a security code into a text field from a CAPTCHA image. The CAPTCHA image was used to mimic the payment procedure. Successful insertion of the CAPTCHA image code allowed the user to redirect into a payment confirmation page.

Chapter 4

Post Processing

This chapter explains the post processing of the data obtained from the experiment defined in the previous chapter. Some important points should be kept in the consideration here. The users had to browse an e-commerce website consisting of three web pages in every session of the experiment. They were browsing this e-commerce website to buy any one of the products. The e-commerce website was browsed under different network settings specified for that particular session. Also, the users were asked three questions as mentioned in Appendix G on every page.

4.1 Data Processing

Data had been collected on three different levels. Firstly, the network-level packets were collected using the Measurement Point (MP). Secondly, the application-level object data were collected using Fiddler, a web debugging proxy tool. Thirdly, the user-level subjective opinions were collected in the form of the MOS [62] using 5-points ACR scale. All data were stored using appropriate file formats for further analysis. The details of these processes have been explained in the following sections.

4.1.1 Network-level Data Processing

The Measurement Point (MP) collected all the packets from the network-level during the client-server communication. The Client Requests were captured in d20 and d11 whereas, the Server Responses were captured in d10 and d21 as illustrated in Figure 3.3. Each of the packets was collected with timestamp by the MP and stored locally in the binary file format (.cap) by the Consumer (see Section 3.3.3). Initially, these cap files were only machine-readable. A tool was developed to convert these machine-readable files into human-readable text file format by using Libcap library utilities. A sample output snippet of the text file has been added in Appendix H. A Perl script was developed to analyze and obtain desired information from those text files. For instance, the Perl script extracted the timestamp from the client request

captured in d20 and the server response for that particular request captured in d21. The d20 is the first capture point of any packet after leaving the client machine and the d21 is the last capture point of any packet before entering the client machine. The time difference between these two timestamps is considered to be the loading time for a particular object in the network-level. A code snippet for this process is available in Appendix I.

4.1.2 Application-level Data Processing

In addition to network-level, application-level data was also collected during the experiment. As mentioned previously, application-level data were collected by using a web debugging proxy tool called Fiddler [61]. Fiddler collected information about each of the client request and corresponding response object from the server in JSON format. A sample output snippet JSON data has been added in Appendix J (a). Usually, each web transfer has two basic parts, Request and Response Header. As we can see from Appendix J (a), the Fiddler stores information about the objects like; timings, cache, etc. Under timings sections, it stores the total time taken by different phases like: DNS resolve, create TCP connection, waiting time, etc. The load time for all the objects in a web page is listed with the timestamp. This data was stored in .har file format. A PHP script was used to extract the desired data from .har files. This script collected timestamp for each client request, server response, time taken for DNS resolve and creating a TCP connection. A code snippet for this process is available in Appendix J (b).

4.1.3 User Level Data Processing

As discussed previously, during experiment on every page users were asked three questions as mentioned in Appendix G. The user given MOS and the answers for other two questions were stored in the local MySQL database of the client machine. Each of the entry in the database was consist of user id, page name, page id, user MOS (in Integer format) and answer of the other two questions. The user given MOS was extracted from the database by using MySQL queries with a PHP script.

4.2 Data Modeling

A total of forty six subjects participated in the experiments. The mean age of the users were 24.5, where the minimum age was 21 years and the maximum age was 32 years. There were 9 female and 37 male subjects. All of them were daily users of web browsing services.

As mentioned in Section 3.6, the delay was applied only on the first page of the three pages in each session. Therefore, the Page Load Time was calculated only for the first page. It was done by taking the timestamp from first DNS query to last object transfer which completed the fetching of the

page. Also, the Page Load Time was calculated from application-level and as well as from network-level data for each of the page.

As discussed previously, users browsed the site for twelve different sessions. After extracting the data, twelve data files was created having Page Load Time, object load time and MOS from application-level, network-level and user-level data. The Mean, Standard Deviation and Confidence Interval of the Page Load Time, object load time and MOS was calculated from each of the data files.

4.2.1 Data Selection

Among the forty six subjects, six subjects (2 female and 4 male) were statistical outliers, who gave a static score in all the 12 sessions (Available in Appendix F). The calculated Page Load Time and other data of these six subjects were removed. Finally, forty subjects were taken into account for further analysis.

4.2.2 Calculation of Mean and Standard Deviation

The Mean, Standard Deviation and Confidence Interval for the Page Load Time and MOS of each sessions is calculated from the selected 40 users.

Let,

X be the Page Load Time for the first page of any session

Y be the user given MOS for the first page of any session

The Mean Page Load Time is defined as,

$$\bar{X} = \frac{1}{N} \sum_{i=1}^N X_i$$

Where $N = 40$, $X_i =$ Page Load Time for the i^{th} user of any session.

The Mean MOS is defined as,

$$\bar{Y} = \frac{1}{N} \sum_{i=1}^N Y_i$$

Where $N = 40$, $Y_i =$ MOS given by the i^{th} user for any session.

The Standard Deviation of the Page Load Time for the first page of any session is defined as,

$$\delta_p = \sqrt{\sum_{i=1}^N \frac{(X_i - \bar{X}_i)^2}{N - 1}}$$

The Standard Deviation of the user given MOS for the first page of any session is defined as,

$$\delta_m = \sqrt{\sum_{i=1}^N \frac{(Y_i - \bar{Y}_i)^2}{N - 1}}$$

4.2.3 Calculation of Confidence Interval

Once all the results of Mean Page Load Time and MOS score was calculated. The Confidence Intervals (CI) for all the Mean Page Load Time and MOS score were calculated as the Mean is always associated with CI, 95 %. The fitness of the calculated Mean of the Page Load Time and MOS score can be obtained using 95% CI.

The 95% of CI for Page Load Time of any session is defined as,

$$[\bar{X} - \gamma_p, \bar{X} + \gamma_p]$$

Where marginal error γ_p is defined as,

$$\gamma_p = 1.96 * \frac{\delta_p}{\sqrt{N}}$$

δ_p = Standard Deviation of Page Load Time for the first page
 N = Number of User

Similarly, The 95% of CI for the MOS of any session is defined as,

$$[\bar{Y} - \gamma_m, \bar{Y} + \gamma_m]$$

Where marginal error γ_m is defined as,

$$\gamma_m = 1.96 * \frac{\delta_m}{\sqrt{N}}$$

δ_m = Standard Deviation of the user given MOS for the first page
 N = Number of User

4.3 Acronym for Sessions

All the data had been divided based on the 12 sessions in the experiment. The acronym mentioned in Table 4.1 is used to represent a particular session with its applied delay.

Table 4.1: Session acronym for analysis

Session Name	Session Acronym	Applied Network Delay
Session 1	Ref 01	No delay applied
Session 2	DNS 2s	2 sec delay in DNS Lookup
Session 3	Base 2s	2 sec delay in Base HTML
Session 4	Obj 2s	2 sec delay in Embedded object
Session 5	Ref 02	No delay applied
Session 6	DNS 4s	4 sec delay in DNS Lookup
Session 7	Base 4s	4 sec delay in Base HTML
Session 8	Obj 4s	4 sec delay in Embedded object
Session 9	Ref 03	No delay applied
Session 10	DNS 8s	8 sec delay in DNS Lookup
Session 11	Base 8s	8 sec delay in Base HTML
Session 12	Obj 8s	8 sec delay in Embedded object

Chapter 5

Analysis and Result

This chapter explains the detailed description about the obtained results. These results are based on the data collected from the web browsing experiment explained in Chapter 3. The extraction and calculation of these data has also been explained in the previous chapter. The considered three phases of HTTP object transfer, namely the DNS lookup phase, the request for Base HTML file phase and the request for Embedded Objects phase would be compared within the same network delay sessions.

The effect of the delayed HTTP object on the Page Load Time is shown for all the experimental sessions. Also, the object load time is shown based on the four phases of HTTP object transfer explained in Chapter 2. These four phases of HTTP object transfer for any web page accumulate the total load time of that particular page. This chapter explains the impact of the object load time on the overall web browsing QoE. How the object load time which are divided into four phases determines the overall Page Load Time and also shape the end user perception.

5.1 Impact of Delay on Page Load Time

This section gives a detailed description about the obtained Page Load Time on the basis of the different delayed HTTP objects. This Page Load Times were calculated from two different points between the communication of the server and the client. There were 12 sessions for each user and every session had a specific network condition. The data was captured from the network and application-level. Also, the users were given MOS for each session according to their perception. At the end, a network-level Page Load Time, an application-level Page Load Time and a user given MOS is representing each experimental session.

The Figure 5.1 exhibits the network and application-level Page Load Time and the user given MOS for all the sessions. In this figure, the X-axis represents all 12 sessions in the experiment. Average Page Load Time calculated from the network and application-level data is on the primary Y-axis. The corresponding user given MOS is on the secondary Y-axis.

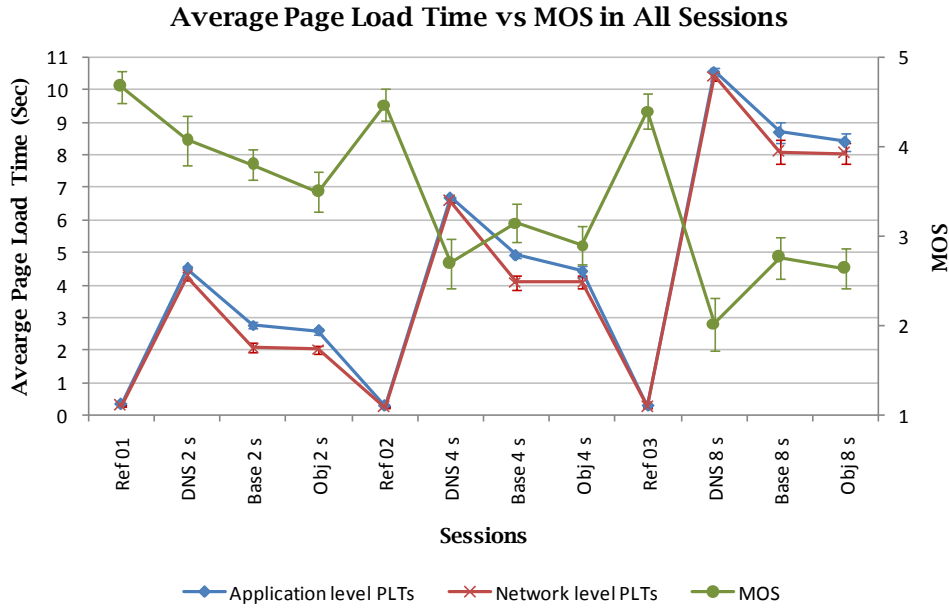


Figure 5.1: Page Load Time vs MOS with 95% CI in 12 sessions

In general, the longer the user has to wait for any service, the less satisfied the user becomes [42]. People do not want to wait unnecessarily for anything, as the time is a non-recurring, non-reproducible entity. The Figure 5.1 shows an expected overall degradation of user given MOS while the Page Load Time is increasing. The overall user satisfaction is degrading from 2 seconds to 4 seconds and from 4 seconds to 8 second delay sessions. But this degradation is not linear. The user given MOS is not following the typical inverse relationship of waiting times and user satisfaction in some sessions. For instance, in the 2 seconds delay session of the request for Base HTML file and request for Embedded object phase, though the Page Load Time is decreasing but the user given MOS is also decreasing. The user given MOS should increased according to the relationship. The users are being less satisfied though the Page Load Time is getting smaller than the previous session. Moreover, the user given MOS has the same characteristics in 2, 4 and 8 second delay sessions. There is an exception in 2 second DNS lookup delay session. The Page Load Time is considerably higher in this session but the user given MOS is comparatively good. It will be explained in the later section.

While the same amount of delay was applied in the different phases of HTTP object transfer, the resulting Page Load Time are not same. For instance, while 2 second delay was applied in the DNS lookup, request for Base HTML file and request for Embedded object phase, the resulting Page Load Time is 4.5 s, 2.8 s and 2.6 s respectively. Which is exhibiting that the network delay produce different Page Load Time based on the phases of HTTP object transfer.

The application-level Page Load Time is always higher than the network-level Page Load Time [42]. This time difference is caused by the client machine as it processes all the packets obtained from the network through the TCP/IP layers to reach the appropriate application. Figure 5.1 exhibits the time difference between the application-level and network-level Page Load Times. Moreover, these Page Load Time are showing the same characteristics in the same amount of delay sessions. On the other hand, though the network-level Page Load Time is almost similar for all the requests for Base HTML file and request for Embedded Object sessions, the obtained Page Load Time from the application-level is not same. In this case, the browser was taking more time for processing the packets of the Base HTML file. The browser uses the Base HTML file for creating the entire skeleton of that particular page. Delay in the different phase of the HTTP object transfer is generating a different amount of Page Load Times and being rated differently by the end user. This result shows that, measuring the timing of the delay during a web transfer, more particularly, measuring the timing of the delay on the basis of the phases of HTTP object transfer is a crucial factor for web browsing QoE. The later sections will explain the results of particular delay sessions.

5.1.1 Reference Sessions

The reference session is the session where no delay was applied and average Page Load Time is almost same. Every user was browsing the web site for three times without any delay during the experiment. These sessions were placed at the 1st, 5th and 9th position. The reference sessions were coming when the amount of delay was getting changed. For instance, 3 sessions with 2 seconds delay and then a reference session and then 3 sessions with 4 seconds delay and so on. These sessions were placed there to ease the memory of the user from previously experienced delays.

Figure 5.2 exhibits that the users' overall perception were degrading towards the e-commerce site as they were experiencing increasing delays in each session between the 2nd and 3rd reference session. One of the reasons behind this degradation can be the memory effect [5]. The patience level of the users was decreasing as they were experiencing delays in several sessions. No delay was applied in these reference sessions but it seems from the MOS that the users were perceiving some delay. That is why, the overall Page Load Time is same for each reference session but the MOS is degrading. Though the load time of any particular web page is tightly coupled with the QoE, but there are some influences of the context where that page is being fetched. In other words, it is very important to identify the point of time when any disturbance is occurring in the network during a single page transfer or an entire browsing session. Because the user perception for web browsing activities depend on this time dynamics. This figure also shows this time dynamics of the human perception.

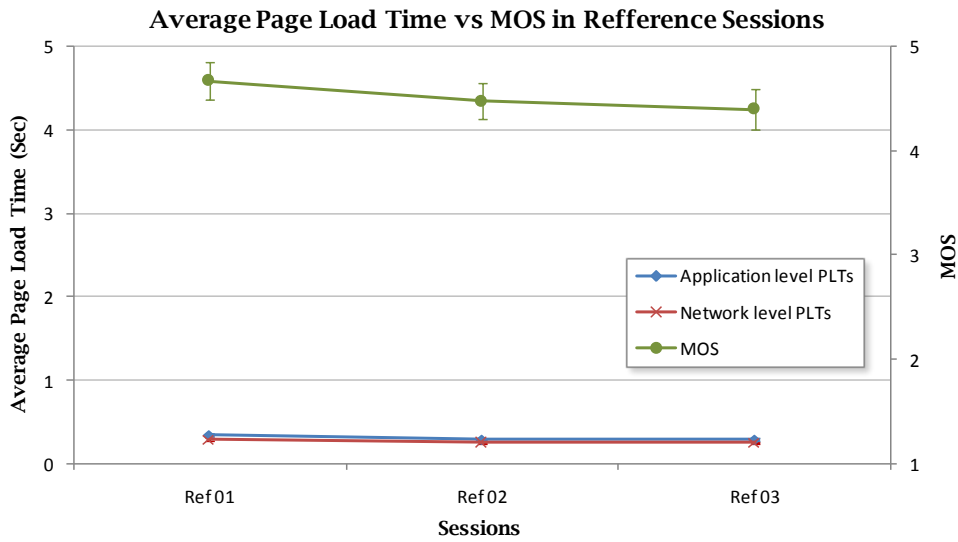


Figure 5.2: Page Load Time vs MOS with 95% CI in the reference sessions

5.1.2 2 Second Delay Sessions

In Figure 5.3, the 3 sessions where 2 second delay was applied on the considered 3 phases are on the X-axis. Average Page Load Time calculated from the network and application-level data is on the primary Y-axis. The corresponding user given MOS is on the secondary Y-axis.

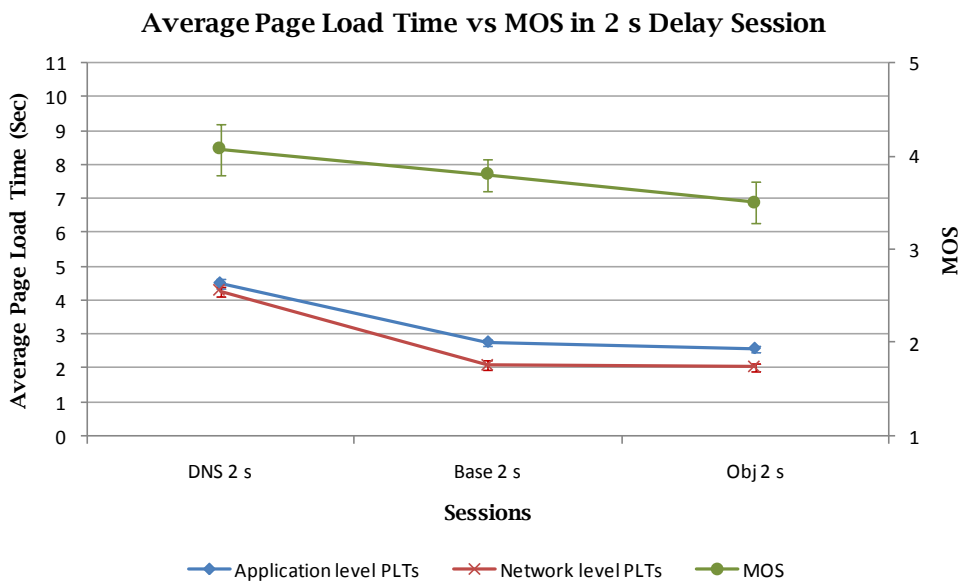


Figure 5.3: Page Load Time vs MOS with 95% CI in 2 second delay sessions

The 2 second delay in the DNS lookup phase results in around 4.5 seconds in Page Load Time, for Base HTML and Embedded Object it were around 2.8 and 2.6 seconds respectively. Though the Page Load Time in the DNS lookup sessions was significantly higher than the other two sessions, user ratings for this session is “Good”, the other two sessions were rated “Fair”. It can be seen from this figure that the users were tolerant to the delay of the DNS lookup session. This was the first session in the experiment where users were experiencing any delay because the very first session was a reference session. This can be one reason for this tolerance. Page Load Time in Base HTML session was slightly higher than Embedded Object session even though the Embedded Object session was getting more ratings. Moreover, it seemed users were less tolerant if the delay occurs in the middle of the page loading. In this case, the delay is more perceivable to the user.

5.1.3 4 Second Delay Sessions

In Figure 5.4, the X-axis represents all the 4 second delay sessions, Page Load Time calculated from network and application-level data is on the primary Y-axis and the corresponding user given MOS is on the secondary Y-axis.

The 4 second delay sessions had the same trend as the previous figure 5.3 with respect to the Page Load Time. The amount of Page Load Time generated by the DNS lookup session was still significantly higher than the other two sessions. In this case it is noticeable to the user as this was rated “Poor”. The ratings of the Base HTML and Embedded Object sessions were “Fair” and “Poor” respectively. Among the 3 sessions, the DNS lookup session has the most effect on the MOS.

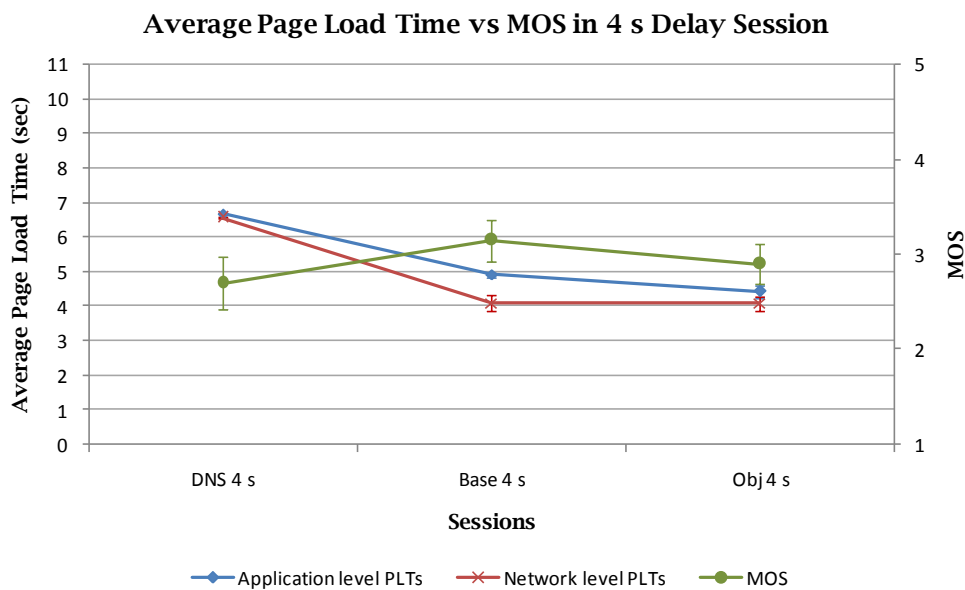


Figure 5.4: Page Load Time vs MOS with 95% CI in 4 second delay sessions

5.1.4 8 Second Delay Sessions

In Figure 5.5, the X-axis represents all the 8 second delay sessions, Page Load Time calculated from network and application-level data is on the primary Y-axis and the corresponding user given MOS is on the secondary Y-axis.

The patience level of the user breaks down at 10 seconds [49]. The DNS lookup phase generated more than 10 seconds Page Load Time in 8 seconds delay session. On the other hand, the 8 seconds delay sessions also had the same trend as the previous figures with respect to the Page Load Time. All 3 sessions were rated “Poor” for 8 second delay as the Page Load Time had increased. The Page Load Time in the DNS lookup phase is the highest Page Load Time between the 12 sessions and predictably the less tolerable by the user as it had the worst rating.

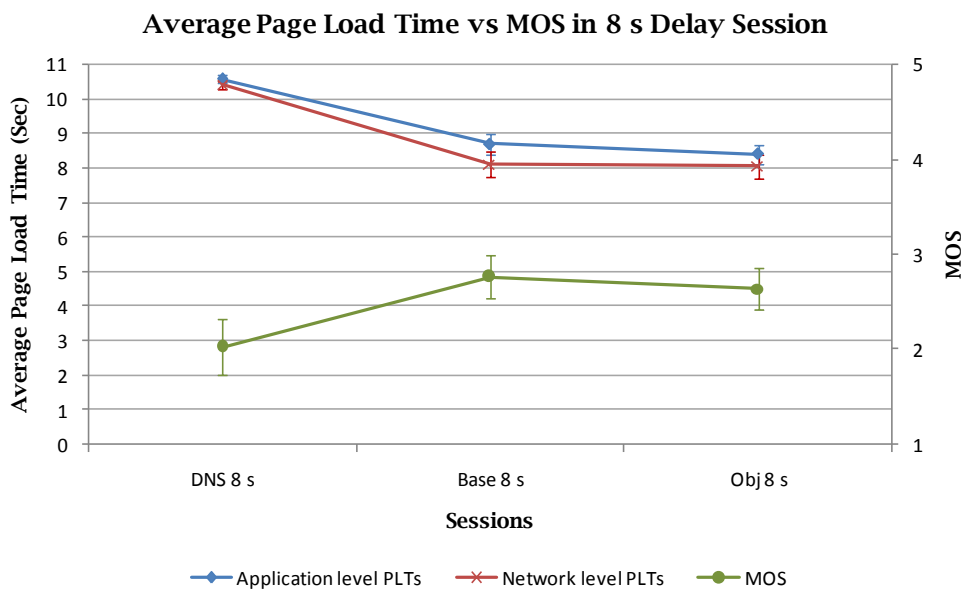


Figure 5.5: Page Load Time vs MOS with 95% CI in 8 second delay sessions

5.2 Impact of Delay on Object Load Time

In this section, the Page Load Time is divided into the four phases of HTTP object transfer. The specific object that transferred within a specific phase had been identified and the load time had been calculated. The accumulated object load times in a phase represent the transfer time of that phase. The Page Load Time of a web page is the sum of all the object load times in it. It is crucial to identify how the disturbance in the network affecting the object load time thus affecting the Page Load Time.

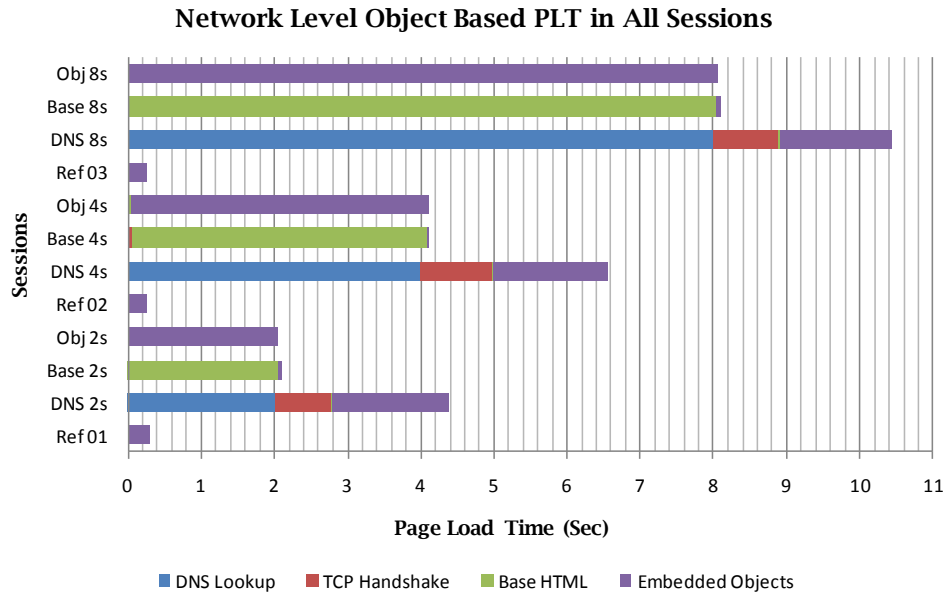


Figure 5.6: Network-level Page Load Time divided based on object load time

In Figure 5.6 and 5.7, the X-axis represents the Page Load Time divided on the basis of object load time. These object load times were calculated from the network and application-level data consecutively. The Y-axis represents all 12 sessions starting from the bottom to the up.

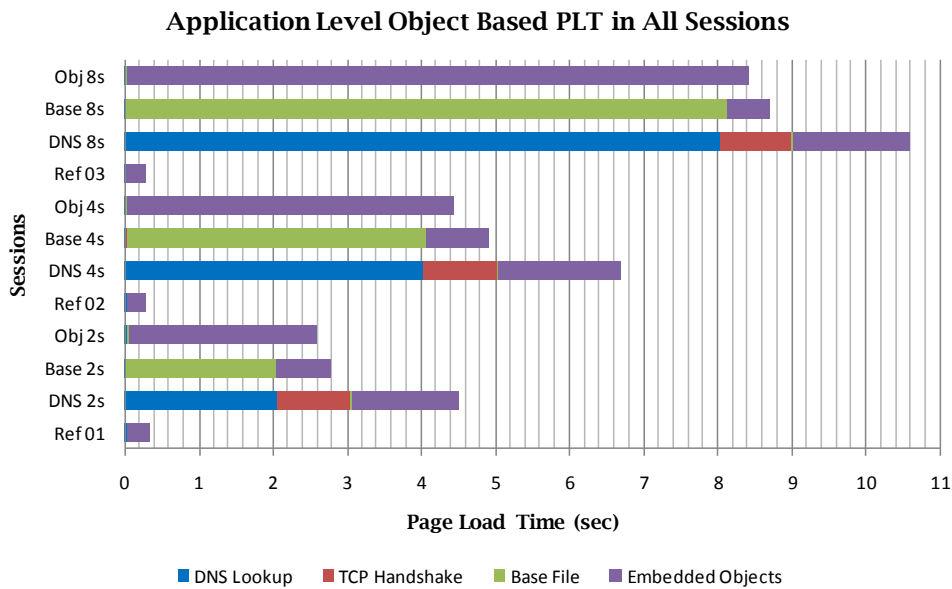


Figure 5.7: Application-level Page Load Time divided based on object load time

In Figure 5.6, it can be seen from the reference sessions that if there is no delay, the Page Load Time mostly consists of the Embedded Object load time. The other 3 phases are very small in the proportion to the Embedded Object load time. The applied delay remains same for the DNS lookup phase but there are additional delays in the rest of the 3 phases. In the Base HTML sessions and the Embedded Object sessions, applied delay consist most of the Page Load Time. The applied delay in these two phases is not generating any additional delay on the network-level.

The Figure 5.7 shows that the application-level object load time have the similar pattern as the network-level in all sessions except the Base HTML sessions. In the Base HTML sessions, the applied delay causes some additional delay on the remaining phases. This additional delay is caused by the client machine for processing the network-level packets. Details on the impact of the object load time will be discussed in the next section.

It can be seen from these figures that the same delay applied in different phases yields in different Page Load Times. Every phase of the HTTP object transfer has a different impact on the entire page transfer. This is the sole reason behind these different Page Load Times.

A browsing session starts by resolving the DNS of any domain name to an IP address. The browser first looks in its own cache for resolving the DNS and then go for the OS cache and then to the specified DNS server. If the server cannot return a successful response, the request passes to other DNS servers. As a result it goes to a recursive state to resolve the DNS query. A browser sends request DNS lookup request in every one second if the first request is unsuccessful. During this waiting period the browser stays idle as it cannot progress to the next phase. The delay in this phase affects the rest of the transfer as this is the first phase of the entire transfer.

The Base HTML file transfer phase is responsible behind the very basic construction of any web page. In this phase, the browser deals with the creation of the basic building blocks of the web page as well as sending a request for the Embedded Objects. The browser does a lot of processing for a very short period of time. It has been seen from the network-level figure that there is no extra delay occurring for the delayed Base HTML file. But on the application-level figure the applied delay gets increased. This extra time is clearly taken by the host machine to process all the packets.

The Embedded Object phase is the last phase of the whole transfer. Typically, there are several objects in a web page. When the browser creates the DOM of a web page, it allocates the space for each and every object on that page. Any delay in this phase does not have any consequence on the rest of the transfer. That is why the Embedded object phase yields the lowest Page Load Time compare to the other two phases.

5.3 Comparison between the Delayed Objects

This section will give a comparison between the effect of the delayed objects on the overall Page Load Time. In Figure 5.8 (a-d), the X-axis represents the Page Load Time divided on the basis of object load time. The Y-axis represents the sessions where delay was applied.

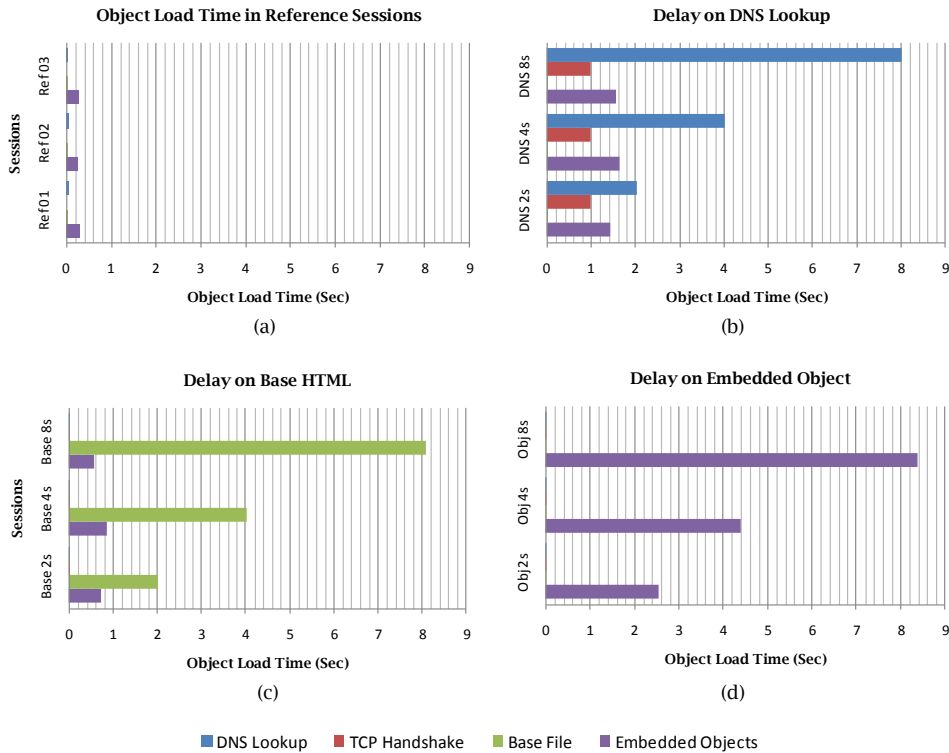


Figure 5.8: Comparison between application-level object load times

Figure 5.8 (a) exhibits the 4 phases when no delay was applied. This figure will be the base for describing the other delayed sessions. The delay on the DNS lookup phase results in a Page Load Time which is approximately 2.5 seconds higher than the actual applied delay which can be seen in Figure 5.8 (b). The delay in the DNS lookup phase is perceived by the user as an indicator of service unavailability. It is because, the browser cannot start the TCP Handshake with the server if there is an unresolved DNS. Moreover, the delayed DNS response causes additional delay to the Embedded Object phase. The Embedded Objects are taking significantly higher amount of time to load. The DNS lookup is the initial task for a web browser, if there is a delay in this task, it affects the rest of the transfer phases. In the delayed DNS lookup sessions, the entire communication in each phase is loading slower than reference sessions, resulting a higher amount of Page Load Time.

Typically, the delays in the Base HTML file are also perceived by users as initial delay. But in this case, the users can see the resources are being transferred from the server, something is loading on the screen. Figure 5.8 (c) exhibits the amount of additional delay caused by the delayed Base HTML file, the Embedded Object load time gets almost doubled. But this increased Embedded Object load time is relatively small compared to the whole Page Load Time. The host processing time is higher in this phase as this is the phase where the DOM for the requested page is created. Though the Page Load Time for this session is always higher than the Embedded Object session, user given score for this session is relatively higher. Because the user can see the loading of the page. This loading blurs the waiting time of the users.

The delay applied to the Embedded Object affect less of the total Page Load Time. Figure 5.8 (d) shows that the applied delay remains almost transparent for the Embedded Object. But any delay in the Embedded Object is more visible to the users because the users had to wait for the object to be loaded for finishing the task and move on to the next page. As the experiment conducted was a task driven one, the Embedded Object sessions are rated less than the Base HTML sessions in all amounts of delay sessions.

The disturbance on the network affects the different phases of HTTP object transfer during web browsing differently. The most amount of additional delay is caused by the DNS Lookup object. It increases the loading time of all the phases significantly. The delay in the DNS is perceived as an indication of service unavailability. Also, after a disturbed DNS lookup, the whole website loads slower than the usual. Disturbed DNS lookup may degrade the user QoE significantly. On the other hand, Embedded Object delays are also crucial for task driven web browsing experience.

This research comes to a conclusion that the different phases of HTTP object transfer affect the overall Page Load times in a different manner. Also, the perception of the user depends on these phases hence, the QoE. Web browsing is not a single request-response transaction. It is considered as a continuous flow of experience where the user come across a series of waiting times. Also, the end user QoE does not linearly depend on the Page Load Times [42]. It is crucial to find out this time dynamics for the HTTP object transfer in web browsing. The waiting times caused by the network outage in the different phases of HTTP object transfer has different implications. Firstly, the increase in the Page Load Times deviate depends on the particular phase. Secondly, the perception of the end user is different for different phases. Monitoring of network performance parameters based on these phases during web browsing can lead to a more efficient modeling of the user QoE.

6.1 Answer to the Research Questions

The answer to the research questions are listed below.

Question 1. How the increased DNS lookup time affects the Page Load Time and the user QoE?

Answer 1. The DNS lookup phase is the very first phase for any web transfer. Any delay in this phase affects the rest of the web transfer. For instance, the Embedded Objects take significantly longer time than usual. The Embedded Objects loading time in the no delay sessions is approximately 0.25 second but in the delayed DNS lookup phase it becomes approximately 1.5 second. The users seem to accept less than 5 second delay at the beginning of the service consumption but react to more than 5 seconds delay. Interruption during this phase of the transfer perceived by the users as service unavailability.

Question 2. How the increased Base HTML Object loading time affects the Page Load Time and the user QoE?

Answer 2. The delay in the Base HTML file causes additional delay to the rest of the transfer. The Embedded Objects loading time get almost double from its original loading time. Interruption during this phase perceived by the users as initial delay. The user tolerance to this phase is higher than other phases as the perception of the delay is blurred by the loading of the page.

Question 3. How the increased Embedded Object loading time affects the Page Load Time and the user QoE?

Answer 3. The delay in the Embedded Object remains almost same as it does not affect any other objects in the remaining web transfer. But the delay in this phase is more visible to the users. As a result, increased waiting time in this phase perceived as a disturbance in the service. Interruption in the loading of the website during this phase effect the user more than the delayed Base HTML phase, even though the Page Load Time for the delayed Base HTML phases are higher than the delayed Embedded Object phases.

Question 4. Which delayed HTTP object mostly affects the user QoE?

Answer 4. Interruption during DNS lookup phase causes most affect to the end user QoE. Between all delay sessions (2, 4 and 8 seconds), the unresolved DNS lookup causes the highest amount of Page Load Time. Also, the user ratings are the lowest for this phase among the considered 3 phases. Moreover, disturbance during this phase is an indication of service unavailability which is crucial for the QoE of any web service.

6.2 Future Work

Future research can be done on a multi agent network where the network condition changes dynamically. This research only used delay as the network performance parameter. The experiment can be done using other network performance parameters. Furthermore, the user perception for any web service usually builds upon by browsing several web pages in that service, to be more specific by a browsing session with that service. It will be interesting to investigate how the user loyalty of a web service changes from session to session while experiencing different amount of random delay in different web pages in the service.

APPENDIX A

Bind9 installation and configuration on Ubuntu Machine (Ubuntu 10.10)

1. apt-get install bind9
2. cd */etc/bind/*
3. open *named.conf.local*
4. Add below line and save
*# This is the zone definition. replace example.com with your
domain name zone "webqoe.com" {
 type master;
 file "/etc/bind/zones/www.webqoe.com.db";
};*
5. cd */etc/bind/zones*
6. create a new file named *www.webqoe.com.db*
7. Add below line and save
*\$TTL 604800
@ IN SOA ns1.webqoe.com. admin.webqoe.com. (
 2010052001
 28800
 3600
 604800
 38400
);
webqoe.com. IN NS ns1.webqoe.com.
webqoe.com. IN MX 10 mail.webqoe.com.
www IN A 10.0.1.1
mail IN A 10.0.1.1
ns1 IN A 10.0.1.1*
8. Restart bind9 service by */etc/init.d/bind9 restart*

APPENDIX B

B1. Code snippet for flushing shaping rule

```
#!/bin/bash
ipfw -f flush
ipfw -f pipe flush
```

B2. Code snippet for apply new shaping rule

```
#!/bin/bash
cd /home/shaper/KauNet/KauNet2.0-Linux/patt_gen
rm -rf $1
./patt_gen -del -pos $1 data 1000 $2
ipfw -f flush
ipfw -f pipe flush
ipfw add allow all from any to any
ipfw add 1 pipe 100 $3 from 10.0.1.1 to 192.168.0.101 in
ipfw pipe 100 config bw 1Mbit/s pattern $1
```

B3: Code snippet for Output-Buffering technique using PHP

```
<?php
function connection_close()
{
    while (ob_get_level()) ob_end_clean();
    header('Connection: close');
    ignore_user_abort(1);
    ob_start();
}
/**
 * Flush a closed HTTP connection from client and
 * clean server context
 */
function connection_flush()
{
    $size = ob_get_length();
    header("Content-Length: $size");
    ob_end_flush();
    flush();
    ob_clean();
}
echo "start time".time(). " <br>";
$file_name=$_POST['file_name'];
connection_close();
connection_flush();
$exe = shell_exec('sh /home/ats/pskill.sh ' . $file_name );
?>
```

B4: Code snippet for start packet capture using shell script

```
#!/bin/bash
file=$1

out=$(ps aux | grep capdump)
set -- $out
pid=$2
echo $pid."PID"

if [ -n "$pid" ];then
sudo -u root kill -SIGINT $pid
fi

cd /home/test/captured_packet/
sudo -u root capdump -o $file -i eth1 01::01
```

APPENDIX C

The `/etc/network/interfaces` file in server and network emulator machine in the experiment setup has been edited by adding below parameters to set the IP address as permanent.

Server

```
auto eth0 iface eth0 inet static
address 10.0.1.1
netmask 255.255.255.0
broadcast 10.0.1.255
gateway 10.0.1.2
```

Emulator

```
auto eth2 iface eth2 inet static
address 10.0.1.2
netmask 255.255.255.0
broadcast 10.0.1.255
gateway 10.0.1.0
```

auto eth1

```
iface eth1 inet static address 192.168.0.100
address 10.0.1.1
netmask 255.255.255.0
broadcast 192.168.0.255
gateway 192.168.0.1
```

The client was a windows based system, the permanent IP has been set by the following:

Client

```
IP: 192.168.0.101 Subnet mask: 255.255.255.0
Default gateway: 192.168.0.100
Preferred DNS: 10.0.1.1
```

APPENDIX D

Table D.1: Application, Network PLT and MOS with 95% CI in 12 sessions

Sessions	App. PLT	95%	Net. PLT	95%	MOS	95%
Ref 01	0.342409159	4.43%	0.292182	3.90%	4.675	3.79%
DNS 2s	4.502886715	2.67%	4.274608	3.22%	4.075	6.76%
Base 2s	2.768166081	3.51%	2.087137	6.67%	3.8	4.47%
Obj 2s	2.589356838	3.29%	2.044322	6.05%	3.5	6.34 %
Ref 02	0.288937851	4.05%	0.251089	3.46%	4.475	3.84%
DNS 4s	6.687728525	0.26%	6.569023	0.22%	2.7	10.13%
Base 4s	4.92109056	1.12%	4.101746	5.49%	3.15	6.76%
Obj 4s	4.432940434	3.92%	4.094404	4.94%	2.9	7.36%
Ref 03	0.293040417	4.00%	0.257019	3.75%	4.4	4.45%
DNS 8s	10.57524588	1.12%	10.42729	1.12%	2.025	14.49%
Base 8s	8.701062049	3.57%	8.10618	4.64%	2.765	8.34%
Obj 8s	8.410090949	3.31%	8.060771	4.19%	2.645	8.27%

APPENDIX E

Table E.1: Application-level HTTP Object Load Time in 12 sessions

Sessions	DNS Lookup	TCP Handshake	Base HTML	Embedded Object
Ref 01	0.036	0	0.004225	0.302184159
DNS 2s	2.04655	0.9879	0.0266	1.441836715
Base 2s	0.02075	0.00285	2.0065	0.738066081
Obj 2s	0.028575	0.002825	0.02	2.537956838
Ref 02	0.02795	0	0.006125	0.254862851
DNS 4s	4.02945	0.986925	0.023975	1.647378525
Base 4s	0.023775	0.003225	4.041225	0.85286556
Obj 4s	0.017675	0.002875	0.01895	4.393440434
Ref 03	0.019875	0	0.005075	0.268090417
DNS 8s	8.013925	0.979575	0.02365	1.558095885
Base 8s	0.021725	0.003125	8.0939	0.582312049
Obj 8s	0.021275	0.00285	0.019225	8.366740949

Table E.2: Network-level HTTP Object Load Time in 12 sessions

Sessions	DNS Lookup	TCP Handshake	Base HTML	Embedded Object
Ref 01	0.000389	0.001673325	0.003067	0.28705363
DNS 2s	2.016485	0.755590516	0.019549	1.58298377
Base 2s	0.000384	0.004277615	2.054158	0.02831795
Obj 2s	0.000376	0.003618966	0.013422	2.02690485
Ref 02	0.000386	0.001272163	0.004147	0.2452837
DNS 4s	4.002876	0.964393146	0.019168	1.58258575
Base 4s	0.000388	0.054941851	4.033155	0.01326068
Obj 4s	0.000386	0.018443015	0.012738	4.06283689
Ref 03	0.000386	0.001367164	0.003623	0.251643
DNS 8s	8.01071	0.873579502	0.019373	1.52362668
Base 8s	0.000387	0.016133142	8.022449	0.06721071
Obj 8s	0.000387	0.003980407	0.012889	8.04351505

APPENDIX F

This user had been filtered out for being statistical outliers. They gave static score for all the 12 sessions.

Table F.1: The statistical outlier subjects in the experiment

User No	10	16	25	32	41	48
Session 1	4	4	5	4	4	5
Session 2	4	5	5	4	4	5
Session 3	4	5	5	4	4	5
Session 4	4	5	5	4	4	5
Session 5	4	4	5	4	4	4
Session 6	4	5	5	4	4	5
Session 7	4	5	5	4	4	5
Session 8	4	5	5	4	4	5
Session 9	4	5	5	4	4	4
Session 10	4	5	5	4	4	4
Session 11	5	5	4	4	4	4
Session 12	4	5	5	5	4	5

APPENDIX G

The questions that were asked to the user in the experiment are listed below.

Q1. Which page is this?

Options:

- Product Selection
- Product Details and Payment
- Thank You

Q2. How do you feel about its loading time?

Options:

- Excellent (5)
- Good (4)
- Fair (3)
- Poor (2)
- Bad (1)

Q3. Would you be willing to use this Internet Service again?

Options:

- Yes
- No

APPENDIX H

A. Snippet of Captured Client request:

```
[ 33]:d20:mp0458:5.295785486500:LINK( 460):CAPLEN( 460):
IPv4(HDR[20])[Len=446:ID=32072:TTL=128:Chk=45027:DF Tos:0]:
TCP(HDR[20]DATA[196]): [AP]: 192.168.0.101:50879 --> 10.0.1.1:80
[0000] 00 02 1E F1 66 64 00 16 17 5C 69 91 08 00 45 00 |....fd...\i...E.|
[0010] 01 BE 7D 48 40 00 80 06 AF E3 C0 A8 00 65 0A 00 |..}H@.....e..|
[0020] 01 01 C6 BF 00 50 78 14 E9 E5 77 4F BE 5E 50 18 |.....Px...wO.^P.|
[0030] 01 00 5D 90 00 00 47 45 54 20 2F 20 48 54 54 50 |..]...GET / HTTP|
[0040] 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 77 77 2E |/1.1..Host: www.|
[0050] 77 65 62 71 6F 65 32 2E 63 6F 6D 0D 0A 43 6F 6E |webqoe2.com..Con|
[0060] 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C |nection: keep-all|
[0070] 69 76 65 0D 0A 41 63 63 65 70 74 3A 20 74 65 78 |live..Accept: tex|
[0080] 74 2F 68 74 6D 6C 2C 61 70 70 6C 69 63 61 74 69 |t/html,applicati|
[0090] 6F 6E 2F 78 68 74 6D 6C 2B 78 6D 6C 2C 61 70 70 |on/xhtml+xml,app|
[00A0] 6C 69 63 61 74 69 6F 6E 2F 78 6D 6C 3B 71 3D 30 |lication/xml;q=0|
[00B0] 2E 39 2C 2A 2F 2A 3B 71 3D 30 2E 38 0D 0A 55 73 |.9,*/*;q=0.8..Us|
[00C0] 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C |er-Agent: Mozill|
[00D0] 61 2F 35 2E 30 20 28 57 69 6E 64 6F 77 73 20 4E |a/5.0 (Windows N|
[00E0] 54 20 36 2E 31 29 20 41 70 70 6C 65 57 65 62 4B |T 6.1) AppleWebKit|
[00F0] 69 74 2F 35 33 37 2E 32 32 20 28 4B 48 54 4D 4C |it/537.22 (KHTML|
[0100] 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 |, like Gecko) Ch|
[0110] 72 6F 6D 65 2F 32 35 2E 30 2E 31 33 36 34 2E 32 |rome/25.0.1364.2|
[0120] 39 20 53 61 66 61 72 69 2F 35 33 37 2E 32 32 0D |9 Safari/537.22.|
[0130] 0A 52 65 66 65 72 65 72 3A 20 68 74 74 70 3A 2F |.Referer: http:/|
[0140] 2F 6C 6F 63 61 6C 68 6F 73 74 2F 75 73 65 72 6D |/localhost/userm|
[0150] 6F 73 2F 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F |os/..Accept-Enco|
[0160] 64 69 6E 67 3A 20 67 7A 69 70 2C 64 65 66 6C 61 |ding: gzip,defla|
[0170] 74 65 2C 73 64 63 68 0D 0A 41 63 63 65 70 74 2D |te,sdch..Accept-|
[0180] 4C 61 6E 67 75 61 67 65 3A 20 65 6E 2D 55 53 2C |Language: en-US,|
[0190] 65 6E 3B 71 3D 30 2E 38 0D 0A 41 63 63 65 70 74 |en;q=0.8..Accept|
[01A0] 2D 43 68 61 72 73 65 74 3A 20 49 53 4F 2D 38 38 |-Charset: ISO-88|
[01B0] 35 39 2D 31 2C 75 74 66 2D 38 3B 71 3D 30 2E 37 |59-1,utf-8;q=0.7|
[01C0] 2C 2A 3B 71 3D 30 2E 33 0D 0A 0D 0A |,*;q=0.3.... |
[01D0]
```

B. Snippet of Captured Client request:

```

[ 36]:d21:mp0458:5.315309941750:LINK(1514):CAPLEN(1434):
IPv4(HDR[20])(Len=1500:ID=20374:TTL=63:Chk=6776:DF Tos:0):
TCP(HDR[20]DATA[5b4]): [A]: 10.0.1.1:80 --> 192.168.0.101:50879
[0000] 00 16 17 5C 69 91 00 02 1E F1 66 64 08 00 45 00 |...\i....fd..E|
[0010] 05 DC 4F 96 40 00 3F 06 1A 78 0A 00 01 01 C0 A8 |..O.@.?..x.....|
[0020] 00 65 00 50 C6 BF 77 4F BE 5E 78 14 EB 7B 50 10 |.e.P..wO.^x..{P|
[0030] 00 6C 21 AC 00 00 48 54 54 50 2F 31 2E 31 20 32 |.!....HTTP/1.1 2|
[0040] 30 30 20 4F 4B 0D 0A 44 61 74 65 3A 20 54 68 75 |00 OK..Date: Thu|
[0050] 2C 20 31 34 20 46 65 62 20 32 30 31 33 20 30 38 |, 14 Feb 2013 08|
[0060] 3A 31 30 3A 34 32 20 47 4D 54 0D 0A 53 65 72 76 | |:10:42 GMT..Serv|
[0070] 65 72 3A 20 41 70 61 63 68 65 2F 32 2E 32 2E 31 |er: Apache/2.2.1|
[0080] 36 20 28 55 62 75 6E 74 75 29 0D 0A 58 2D 50 6F |/6 (Ubuntu)..X-Po|
[0090] 77 65 72 65 64 2D 42 79 3A 20 50 48 50 2F 35 2E |/wered-By: PHP/5.|
[00A0] 33 2E 33 2D 31 75 62 75 6E 74 75 39 2E 31 30 0D |/3.3-1ubuntu9.10.|
[00B0] 0A 56 61 72 79 3A 20 41 63 63 65 70 74 2D 45 6E |/.Vary: Accept-En|
[00C0] 63 6F 64 69 6E 67 0D 0A 43 6F 6E 74 65 6E 74 2D |/coding..Content-|
[00D0] 45 6E 63 6F 64 69 6E 67 3A 20 67 7A 69 70 0D 0A |/Encoding: gzip..|
[00E0] 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 |/Content-Length: |
[00F0] 31 38 33 36 0D 0A 4B 65 65 70 2D 41 6C 69 76 65 |/1836..Keep-Alive|
[0100] 3A 20 74 69 6D 65 6F 75 74 3D 31 35 2C 20 6D 61 |/: timeout=15, ma|
[0110] 78 3D 31 30 30 0D 0A 43 6F 6E 6E 65 63 74 69 6F |/x=100..Connectio|
[0120] 6E 3A 20 4B 65 65 70 2D 41 6C 69 76 65 0D 0A 43 |/n: Keep-Alive..C|
[0130] 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 74 65 78 |/ontent-Type: tex|
[0140] 74 2F 68 74 6D 6C 0D 0A 0D 0A 1F 8B 08 00 00 00 |/t/html.....|

```

APPENDIX I

A. Perl snippet for getting timestamp for request object

```
#!/usr/local/bin/perl-wT
$input=$ARGV[0];
$con_name=$ARGV[1];

open (sa1,"<$input") or die "cannot open the file";
$i=0;
while (<sa1>){
    @arra[$i]=$_;
    $i++;
}
close (sa1);

$lenarra=@arra;    @vech = @arra; @gettimestamp; $aa = "1";
$bb = "1";         $time = "0";  $name = "0";  $re = "1";
$sta="0";         $limiter = "0";  $count = 0;   $o_count = 0;

for($h1=0;$h1<$lenarra;$h1++)
{
    if(($vech[$h1] =~ m/^\:d20:mp0458:/) && (($vech[$h1] =~ m/^\: TCP/) ||
($vech[$h1] =~ m/^\: UDP/)))
    {
        $limiter1 = 1;
        @da1=split(':', $vech[$h1]);
    }
    elsif(($limiter1 == 1) && ($vech[$h1] =~ /GET/) && ($da1[1] == 'd20'))
    {
        @b1=split(/\ /, $vech[$h1]);
        @c1=split(' ', $b1[0]);
        $string3 = $c1[1].".".$b1[1].".".$b1[2].".".$b1[3].".".$b1[4]."\t";
        $string3 =~ s/([a-fA-F0-9]{2})/chr(hex $1)/eg;
        $string3 =~ s/[\n\r\s]+//g;
        @e1=split(/\ /, $vech[$h1+1]);
        @f1=split(' ', $e1[0]);
        $string4 = $f1[1].".".$e1[1].".".$e1[2].".".$e1[3].".".$e1[4]."\t";
        $string4 =~ s/([a-fA-F0-9]{2})/chr(hex $1)/eg;
        $string4 =~ s/[\n\r\s]+//g;
        $ad1 = $ad1 + 1;
        $outstr1 = $string3." ".$string4 ;
        if($outstr1 =~ /$con_name/)
        {
            push(@gettimestamp, $da1[3]);
        }
    }
}
print $gettimestamp[0];
```

B. Perl snippet for getting timestamp for response object

```
#!/usr/local/bin/perl-wT

$input=$ARGV[0];
$con_size=$ARGV[1];
$con_type=$ARGV[2];
open (sa1,"<$input") or die "cannot open the file";
$i=0;
while (<sa1>){
    @arra[$i]=$_;
    $i++;
}
close (sa1);

$lenarra=@arra;
@vech = @arra;
@posttimestamp;
$aa = "1";
$bb = "1";
$time = "0";
$name = "0";
$re = "1";
$sta="0";
$limiter = "0";
$count = 0;
$o_count = 0;

for($h1=0;$h1<$lenarra;$h1++){

    if(($arra[$h1] =~ m/^\:d21:mp0458:/) &&
    (($arra[$h1] =~ m/^\:TCP/) || ($arra[$h1] =~ m/^\:UDP/)))
    {

        $limiter = "1";
        @da=split(':', $arra[$h1]);
    }
    elsif(($limiter == 1) && ($arra[$h1] =~ m/^\-Leng/))
    {
        @w=split(/\ /, $arra[$h1]);
        @x=split(' ', $w[0]);
        $string7 = $x[1].".".$w[1].".".$w[2].".".$w[3].".".$w[4]."\t";
        @v=split(/\ /, $arra[$h1+1]);
        @u=split(' ', $v[0]);

        $string8 = $u[1].".".$v[1].".".$v[2].".".$v[3].".".$v[4]."\t";
        @g=split(/\ /, $arra[$h1+2]);
        @h=split(' ', $g[0]);

        $string9 = $h[1].".".$g[1].".".$g[2].".".$g[3].".".$g[4]."\t";

        $hexout5 = $string7." ".$string8;
        @hexout5 = split(" ", $hexout5);
        $hexout5 = join(" ", @hexout5);
        @hed = split("3A", $hexout5);
        @out = split("0D", $hed[1]);
        $contlen = $out[0];
    }
}

```

```

$contlen =~ s/([a-fA-F0-9]{2})/chr(hex $1)/eg;
$contlen =~ s/[\n\r\s]+//g;

$ad = $ad + 1;
$limiter = "0";

$link = (/+/ , $da[4]);

@b=split(/\s/, $arra[$h1+5]);
@c=split(',', $b[0]);

$string = $c[1]."". $b[1]."". $b[2]."". $b[3]."". $b[4]."\t";
@e=split(/\s/, $arra[$h1+5+1]);
@f=split(',', $e[0]);

$string1 = $f[1]."". $e[1]."". $e[2]."". $e[3]."". $e[4]."\t";
@q=split(/\s/, $arra[$h1+5+2]);
@p=split(',', $q[0]);

$string2 = $p[1]."". $q[1]."". $q[2]."". $q[3]."". $q[4]."\t";

$hexout = $string. "". $string1. "". $string2 ;

my ($domn, $program) = $hexout =~ m{(. *3A)(. *)};
my ($hexstr, $hexspl) = $program =~ m{(. *0D)([^\?]*)};

$string =~ s/([a-fA-F0-9]{2})/chr(hex $1)/eg;
$string =~ s/[\n\r\s]+//g;
$string1 =~ s/([a-fA-F0-9]{2})/chr(hex $1)/eg;
$string1 =~ s/[\n\r\s]+//g;

$hexstr =~ s/([a-fA-F0-9]{2})/chr(hex $1)/eg;
$hexstr =~ s/[\n\r\s]+//g;

$ad = $ad + 1;
$outstr = $string. "". $string1 ;

if($contlen == $con_size && $hexstr =~ /$con_type/){
    push(@posttimestamp, $da[3]);
}
}
}
print $posttimestamp[0];

```

APPENDIX J

A. Snippet of a object in HAR file

The image shows a snippet of a HAR file object with the following structure and values:

- timings**
 - blocked : -1
 - ssl : 0
 - dns : 2005
 - wait : 24
 - receive : 0
 - send : 0
 - connect : 991
- cache**
- response**
 - httpVersion : "HTTP/1.1"
 - status : 200
 - bodySize : 1836
 - redirectURL : ""
 - statusText : "OK"
 - headersSize : 276
- headers**
- content**
- cookies**
- request**
 - queryString
 - bodySize : 0
 - method : "GET"
 - httpVersion : "HTTP/1.1"
 - url : "http://www.w ebqoe2.com"
 - headersSize : 406
- headers**
- cookies**
- startedDateTime : "2013-02-14T09:11:43.0087890+01:00"
- time : 3020
- connection : "50878"

B. PHP snippet for saving each object timestamp & calculate total page load time for each object from Application & Network level

```

<?php
error_reporting(E_ERROR);
set_time_limit(6000);

$key_name = array();
$con_arr = array();
$object_arr = array();
$object_key_list = array();

// remove id page_load_time folder exists
if(is_dir("page_load_time")){
    rmdir("page_load_time");
    mkdir("page_load_time");
}

$file_lists = getDirectoryList("F_Data/");

foreach($file_lists as $file_list)
{
    $folder_temp = explode(".har", $file_list);
    if (is_dir($folder_temp[0]) ) {
        mkdir($folder_temp[0], 0777);
    }
    $data_temp = file_get_contents("F_data/" . $file_list);

    $data = str_replace("'", "", $data_temp);
    $user_sess = 1;
    $json = (array)json_decode($data);
    $entries = $json["log"]->entries;
    for($user_sess<=12;$user_sess++){

        $ret_data =
ex_sess($entries,$user_sess,$key_name,$con_arr,$folder_temp[0]);
        $key_name = $ret_data["key_name"];
        $con_arr[] = $ret_data["content"];
        $page_array = slice_array_page($ret_data["content"]);

        // Write each session data into file
        $page_array_obj =
write_to_file($page_array["page1"],$page_array["page2"],$page_array["page3"],$u
ser_sess,$folder_temp[0]);
    }
}

```

```

function ex_sess($entries,$sess_id,$key_name_temp,$con_arr,$user_id){

    $con_arr_temp = array();
    for ($i = 0 ; $i<count($entries); $i++){
        if (strpos($entries[$i]->request->url, 'webqoe'.'$sess_id' . '.com') !== false) {
            // Creating Session array
            $parts = explode('/', $entries[$i]->request->url);
            if(end($parts) != null){
                $con_key = end($parts);
                $key_name_temp[$con_key] = $con_key;
                $req_time =
getMpdata($user_id,$sess_id,$con_key,$entries[$i]->response->content-
>mimeType,$entries[$i]->response->bodySize,'get');
                $res_time =
getMpdata($user_id,$sess_id,$con_key,$entries[$i]->response->content-
>mimeType,$entries[$i]->response->bodySize,'post');

                $con_arr_temp[$con_key] = array("total_time"
=>$entries[$i]->time,

                "start" =>$entries[$i]->startedDateTime,"dns" => $entries[$i]->timings->dns,

                "mp_start" => $req_time, "mp_end" => $res_time);
            } else{
                $con_key = $entries[$i]->request->url;

                $key_name_temp[$con_key] = $con_key;

                $req_time =
getMpdata($user_id,$sess_id,"Host:www",$entries[$i]->response->content-
>mimeType,$entries[$i]->response->bodySize,'get');
                $res_time =
getMpdata($user_id,$sess_id,$con_key,$entries[$i]->response->content-
>mimeType,$entries[$i]->response->bodySize,'post');
                $con_arr_temp[$con_key] = array("total_time" =>
$entries[$i]->time,

                "start" =>$entries[$i]->startedDateTime,"dns" => $entries[$i]->timings->dns,

                "mp_start" => $req_time, "mp_end" => $res_time);

                // Get dns time stamp
                $dns_req_time =
getMpdata($user_id,$sess_id,$con_key,$entries[$i]->response->content-
>mimeType,$entries[$i]->response->bodySize,'dns');
                $dns_time = explode(" ", $dns_req_time);
                $con_arr_temp["dns"] = array("mp_start" => $dns_time[0],
"mp_end" =>$dns_time[1]);
            }
        }
    }
    return array("content"=>$con_arr_temp,"key_name"=>$key_name_temp);
}

```

```
function getMpdata($user_id,$sess_id,$con_name,$con_type,$con_size,$type)
{
    $path = "M_data/" . $user_id . "/" . $user_id . "_sess_" . $sess_id . ".txt";
    $output = "";
    if($type == 'post'){
        $output = shell_exec("perl cappostread.pl $path $con_size $con_type");
    } else if ($type == 'get'){
        $filename = pathinfo($con_name, PATHINFO_FILENAME);
        $output = shell_exec("perl capgetread.pl $path $filename");
    } else {
        $filename = pathinfo($str, PATHINFO_FILENAME);
        $output = shell_exec("perl capreaddns.pl $path ");
    }
    return $output;
}

function slice_array_page($arr_tmp){
    $page1_array = array();
    $page2_array = array();
    $page2_array_temp = $arr_tmp;
    foreach($arr_tmp as $key=>$value){
        if(preg_match("/checkout_pid/", $key)){
            break;
        } else {
            $page1_array[$key] = $value;
            unset($page2_array_temp[$key]);
        }
    }

    $page3_array = $page2_array_temp;
    foreach($page2_array_temp as $key=>$value){
        if(preg_match("/thank_you/", $key)) {
            break;
        } else {
            $page2_array[$key] = $value;
            unset($page3_array[$key]);
        }
    }

    return array("page1" => $page1_array, "page2" => $page2_array, "page3" =>
    $page3_array);
}
```

```

function getTimeDiff($t1,$t2){
    $sstr = "+";
    $time_temp = substr($t1, ($pos = strpos($t1, 'T')) !== false ? $pos + 1 : 0);

    $t1 = array_shift(explode($sstr, $time_temp));

    $time_temp = substr($t2, ($pos = strpos($t2, 'T')) !== false ? $pos + 1 : 0);

    $t2 = array_shift(explode($sstr, $time_temp));

    $t1_tmp = getMsecond($t1);
    $t2_tmp = getMsecond($t2);

    $t_diff = $t2_tmp - $t1_tmp ;

    return $t_diff;
}

function getMsecond($time1){

    $t1 = explode(":", $time1);
    $totaltime = ($t1[0]*60*60) + ($t1[1]*60) + $t1[2];
    return $totaltime;

}

function write_to_file($sarrs_page1,$sarrs_page2,$sarrs_page3,$sess_id, $user_id_fol)
{

    $log_data = "Page1 \n \n";
    $log_data .= "Object name \t total_time \t elapsed_time \t start \t mp_total_time \t
mp_start \t mp_end \n";
    $sstr = "+";
    // Save page1

    $init_time= $sarrs_page1["http://www.webqoe". $sess_id . ".com/"]["start"];
    // Remove favicon.ico entry
    $dns_page1_temp = $sarrs_page1["dns"];
    unset($sarrs_page1["favicon.ico"]);
    unset($sarrs_page1["dns"]);
    foreach($sarrs_page1 as $key=>$value){
        // Get dns timestamp
        if(preg_match("/http:\\\\www.webqoe/", $key)) {
            //echo $key;
            $elapsed_time = getTimeDiff($init_time,$value["start"]);
            $sarrs_page1_dns["dns"]["total_time"] = $value["dns"];

            $sarrs_page1_dns["dns"]["start"] = $value["start"];
            $sarrs_page1_dns["dns"]["elapsed_time"] = $elapsed_time;
            $sarrs_page1_dns["dns"]["mp_start"] =
            $dns_page1_temp["mp_start"];
            $sarrs_page1_dns["dns"]["mp_end"] = $dns_page1_temp["mp_end"];

```

```

        $log_data .= "DNS" . " \t " . $value["dns"] . " \t " .
        $sarrs_page1_dns["dns"]["elapsed_time"] . " \t " . $value["start"] .
        " \t " . ($dns_page1_temp["mp_end"] -
        $dns_page1_temp["mp_start"]) . " \t " . $dns_page1_temp["mp_start"] . " \t
        " . $dns_page1_temp["mp_end"] . " \n";

        //unset($sarrs_page1["dns"]);

        $diff = $value["total_time"] - $value["dns"];

        $time_next = getNextTimestamp($value["start"], $value["dns"]);
        $elapsed_time = getTimediff($init_time, $time_next);
        $sarrs_page1[$key]["elapsed_time"] = $elapsed_time;
        $sarrs_page1[$key]["total_time"] = $diff;
        $sarrs_page1_temp["base_file"] = $sarrs_page1[$key];

        $log_data .= $key . " \t " . $diff . " \t " .
        $sarrs_page1[$key]["elapsed_time"] . " \t " . $time_next .
        " \t " . ($value["mp_end"] -
        $value["mp_start"]) . " \t " . $value["mp_start"] . " \t " . $value["mp_end"] . " \n";

        unset($sarrs_page1[$key]);
        $sarrs_page1 = $sarrs_page1_dns + $sarrs_page1_temp +
        $sarrs_page1;
    } else{
        $elapsed_time = getTimediff($init_time, $value["start"]);

        $sarrs_page1[$key]["elapsed_time"] = $elapsed_time;

        $log_data .= $key . " \t " . $value["total_time"] . " \t " .
        $sarrs_page1[$key]["elapsed_time"] . " \t " . $value["start"] .
        " \t " . ($value["mp_end"] - $value["mp_start"]) . "
        \t " . $value["mp_start"] . " \t " . $value["mp_end"] . " \n";
    }
}

// Save page2
$log_data .= "\n Page2 \n \n";
$init_time = 0;
foreach($sarrs_page2 as $key=>$value){
    if(preg_match("/checkout_pid/", $key)) {
        $init_time = $sarrs_page2[$key]["start"];
        $elapsed_time = getTimediff($init_time, $value["start"]);

        $sarrs_page2[$key]["elapsed_time"] = $elapsed_time;
        $log_data .= $key . " \t " . $value["total_time"] . " \t " .
        $sarrs_page2[$key]["elapsed_time"] . " \t " . $value["start"] .
        " \t " . ($value["mp_end"] - $value["mp_start"]) . " \t
        " . $value["mp_start"] . " \t " . $value["mp_end"] . " \n";
        $sarrs_page2_temp["base_file_2"] = $sarrs_page2[$key];

        unset($sarrs_page2[$key]);
        $sarrs_page2 = $sarrs_page2_temp + $sarrs_page2;
    }
}

```

```

else {
    $elapsed_time = getTimediff($init_time,$value["start"]);

    $sarrs_page2[$key]["elapsed_time"] = $elapsed_time;
    $log_data .= $key . "\t " . $value["total_time"] . "\t " .
    $sarrs_page2[$key]["elapsed_time"] . "\t " . $value["start"] .
    "\t " . ($value["mp_end"] - $value["mp_start"]) . "\t " .
    $value["mp_start"] . "\t " . $value["mp_end"] . "\n";
    }

// Save page3
$log_data .= "\n Page3 \n \n";
$init_time= $sarrs_page3["thank_you.php"]["start"];
foreach($sarrs_page3 as $key=>$value){
    $elapsed_time = getTimediff($init_time,$value["start"]);
    $sarrs_page3[$key]["elapsed_time"] = $elapsed_time;
    $log_data .= $key . "\t " . $value["total_time"] . "\t " .
    $sarrs_page3[$key]["elapsed_time"] . "\t " . $value["start"] .
    "\t " . ($value["mp_end"] - $value["mp_start"]) . "\t " . $value["mp_start"] . "\t " .
    $value["mp_end"] . "\n";
}

$fh = fopen($user_id_fol . "/" . $user_id_fol . "_sess_". $sess_id . ".xls" , 'w+');
fwrite($fh, $log_data);
fclose();

//Need to turn on to create page load time
write_page_load_time($sarrs_page1,$sarrs_page2,$sarrs_page3,$user_id_fol,$sess_id)
;

return
array("page1"=>$sarrs_page1,"page2"=>$sarrs_page2,"page3"=>$sarrs_page3) ;
}

function
write_page_load_time($sarrs_page1,$sarrs_page2,$sarrs_page3,$user_id,$sess_id)
{
    $pageload_time_title = "\t \t page1 \t page2 \t page3 \t mp_page1 \t mp_page2 \t
mp_page3 \n";
    $load_time_page1_mp = current($sarrs_page1);
    $load_time_page2_mp = current($sarrs_page2);
    $load_time_page3_mp = current($sarrs_page3);

    $load_time_page1 = end($sarrs_page1);
    $load_time_page2 = end($sarrs_page2);
    $load_time_page3 = end($sarrs_page3);
}

```

```

$pageload_time = ($load_time_page1["elapsed_time"] +
($load_time_page1["total_time"]/1000)) . " \t " .
($load_time_page2["elapsed_time"] +
($load_time_page2["total_time"]/1000)) . " \t " .
($load_time_page3["elapsed_time"] +
($load_time_page3["total_time"]/1000)) . " \t " .
($load_time_page1["mp_end"] -
$load_time_page1_mp["mp_start"]) . " \t " .
($load_time_page2["mp_end"] -
$load_time_page2_mp["mp_start"]) . " \t " .
($load_time_page3["mp_end"] -
$load_time_page3_mp["mp_start"]) . " \t " .
" \n";
if(file_exists("page_load_time/session_". $sess_id . ".txt"))
{
    $fh = fopen("page_load_time/session_". $sess_id . ".txt" , 'a+');
    fwrite($fh, $user_id . " \t \t " . $pageload_time);
    fclose();
} else {
    $fh = fopen("page_load_time/session_". $sess_id . ".txt" , 'a+');
    fwrite($fh, $pageload_time_title);
    fwrite($fh, $user_id . " \t \t " . $pageload_time);
    fclose();
}
return null;
}

function getDirectoryList ($directory)
{
    // create an array to hold directory list
    $results = array();
    $handler = opendir($directory);

    // open directory and walk through the filenames
    while ($file = readdir($handler)) {

        // if file isn't this directory or its parent, add it to the results
        if ($file != "." && $file != "..") {
            $results[] = $file;
        }
    }
    closedir($handler);
    return $results;
}

```

```

// Get next time format with time difference
function getNextTimestamp($time1, $diff){
    $sstr = "+";
    // Store time stamp part
    $time_final = "";
    $time_temp1 = explode("T",$time1);
    $time_final .= $time_temp1[0]. "T";
    $temp2 = explode($sstr, $time_temp1[1]);

    // Get timestamp after +diff seconds

    $time_temp = substr($time1, ($pos = strpos($time1, 'T')) !== false ? $pos + 1 : 0);
    $t1 = array_shift(explode($sstr, $time_temp));
    //echo "time: " . $t1 . "\n";
    $diff = $diff/1000;
    $t1 = explode(":", $t1 );
    $totaltime = ($t1[0]*60*60) + ($t1[1]*60) + $t1[2] + $diff;

    $time_temp3 = formatSeconds($totaltime);
    $time_final .= $time_temp3 . "+" . $temp2[1];
    return $time_final;
}

/* Convert milisecond into date time format */

function formatSeconds( $seconds )
{
    $hours = 0;
    $milliseconds = str_replace( "0.", "", $seconds - floor( $seconds ) );

    if ( $seconds > 3600 )
    {
        $hours = floor( $seconds / 3600 );
    }
    $seconds = $seconds % 3600;

    return str_pad( $hours, 2, '0', STR_PAD_LEFT )
        . gmdate( ':i:s', $seconds )
        . ( $milliseconds ? ".$milliseconds" : "" )
    ;
}
?>

```

Bibliography

- [1] “World Telecommunication/ICT Indicators Database.” <http://www.itu.int/ITU-D/ict/statistics/>, 2013. [Online; Accessed: 21-Jan-2013].
- [2] J. Shaikh, M. Fiedler, and D. Collange, “Quality of experience from user and network perspectives,” *annals of telecommunications-Annales des télécommunications*, vol. 65, no. 1-2, pp. 47–57, 2010.
- [3] “RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1.” <http://www.faqs.org/rfcs/rfc2616.html>. [Online; Accessed: 07-Feb-2013].
- [4] “RFC 2818 - HTTP Over TLS.” <http://www.faqs.org/rfcs/rfc2818.html>. [Online; Accessed: 07-Feb-2013].
- [5] T. Hoßfeld, S. Biedermann, R. Schatz, A. Platzer, S. Egger, and M. Fiedler, “The memory effect and its implications on web qoe modeling,” in *Proceedings of the 23rd International Teletraffic Congress*, pp. 103–110, ITCP, 2011.
- [6] “Estimating end-to-end performance in IP networks for data applications.” http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.1030-200511-I!!PDF-E&type=items, 2005. [Online; Accessed: 18-Jan-2013].
- [7] E. Ibarrola, F. Liberal, I. Taboada, and R. Ortega, “Web qoe evaluation in multi-agent networks: Validation of itu-t g. 1030,” in *Autonomic and Autonomous Systems, 2009. ICAS’09. Fifth International Conference on*, pp. 289–294, IEEE, 2009.
- [8] J. Shaikh, M. Fiedler, P. Arlos, and D. Collange, “Modeling and analysis of web usage and experience based on link-level measurements,” in *Teletraffic Congress (ITC 24), 2012 24th International*, pp. 1–8, IEEE, 2012.
- [9] D. Collange and J.-L. Costeux, “Passive estimation of quality of experience,” *Journal of Universal Computer Science*, vol. 14, no. 5, pp. 625–641, 2008.
- [10] D. F. Galletta, R. M. Henry, S. McCoy, and P. Polak, “When the wait isnt so bad: The interacting effects of website delay, familiarity, and breadth,” *Information Systems Research*, vol. 17, no. 1, pp. 20–37, 2006.

- [11] N. Bhatti, A. Bouch, and A. Kuchinsky, "Integrating user-perceived quality into web server design," *Computer Networks*, vol. 33, no. 1, pp. 1–16, 2000.
- [12] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *Network, IEEE*, vol. 24, no. 2, pp. 36–41, 2010.
- [13] H. Cui and E. Biersack, "On the relationship between qos and qoe for web sessions," tech. rep., Technical Report RR-12-263, EURECOM, Sophia Antipolis, France, 2012.
- [14] T. Hosfeld, M. Fiedler, and T. Zinner, "The qoe provisioning-delivery-hysteresis and its importance for service provisioning in the future internet," in *Next Generation Internet (NGI), 2011 7th EURO-NGI Conference on*, pp. 1–8, IEEE, 2011.
- [15] S. Egger, P. Reichl, T. Hosfeld, and R. Schatz, "time is bandwidth? narrowing the gap between subjective time perception and quality of experience," in *Communications (ICC), 2012 IEEE International Conference on*, pp. 1325–1330, IEEE, 2012.
- [16] T. N. Minhas, M. Fiedler, J. Shaikh, and P. Arlos, "Evaluation of throughput performance of traffic shapers," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pp. 1596–1600, IEEE, 2011.
- [17] J. Shaikh, T. N. Minhas, P. Arlos, and M. Fiedler, "Evaluation of delay performance of traffic shapers," in *Security and Communication Networks (IWSCN), 2010 2nd International Workshop on*, pp. 1–8, IEEE, 2010.
- [18] T. N. Minhas and M. Fiedler, "Quality of experience hourglass model," in *Computing, Management and Telecommunications (ComManTel), 2013 International Conference on*, pp. 87–92, IEEE, 2013.
- [19] T. N. Minhas, M. Fiedler, and P. Arlos, "Quantification of packet delay variation through the coefficient of throughput variation," in *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, pp. 336–340, ACM, 2010.
- [20] T. N. Minhas, O. Gonzalez Lagunas, P. Arlos, and M. Fiedler, "Mobile video sensitivity to packet loss and packet delay variation in terms of qoe," in *Packet Video Workshop (PV), 2012 19th International*, pp. 83–88, IEEE, 2012.
- [21] T. N. Minhas and M. Fiedler, "Impact of disturbance locations on video quality of experience," *Quality of Experience for Multimedia Content Sharing, EuroITV*, 2011.

- [22] Q. Huynh-Thu and M. Ghanbari, “Temporal aspect of perceived quality in mobile video broadcasting,” *Broadcasting, IEEE Transactions on*, vol. 54, no. 3, pp. 641–651, 2008.
- [23] Q. Huynh-Thu and M. Ghanbari, “No-reference temporal quality metric for video impaired by frame freezing artefacts,” in *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pp. 2221–2224, IEEE, 2009.
- [24] P. Varga, G. Kún, G. Sey, I. Moldován, and P. Gelencsér, “Correlating user perception and measurable network properties: Experimenting with qoe,” in *Autonomic Principles of IP Operations and Management*, pp. 218–221, Springer, 2006.
- [25] R. K. Mok, E. W. Chan, and R. K. Chang, “Measuring the quality of experience of http video streaming,” in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pp. 485–492, IEEE, 2011.
- [26] J. Shaikh, M. Fiedler, T. Minhas, P. Arlos, and D. Collange, “Passive methods for the assessment of user-perceived quality of delivery,” *SNCNW 2011*, p. 73, 2011.
- [27] I. Din, N. A. Saqib, and A. Baig, “Passive analysis of web traffic characteristics for estimating quality of experience,” in *GLOBECOM Workshops, 2008 IEEE*, pp. 1–5, IEEE, 2008.
- [28] J. Shaikh, M. Fiedler, D. Collange, P. Arlos, and T. N. Minhas, “Inferring user-perceived performance of network by monitoring tcp interruptions,” *Network Protocols and Algorithms*, vol. 4, no. 2, pp. 49–67, 2012.
- [29] J. Shaikh, M. Fiedler, P. Arlos, T. Minhas, and D. Collange, “Classification of tcp connection termination behaviors for mobile web,” in *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*, pp. 1111–1115, IEEE, 2011.
- [30] Y. Ito and T. Tahara, “Evaluation of user-satisfaction in online shopping web services with tcp variables,” in *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*, pp. 469–473, IEEE, 2011.
- [31] D. Collange, M. Hajji, J. Shaikh, M. Fiedler, and P. Arlos, “User impatience and network performance,” in *Next Generation Internet (NGI), 2012 8th EURO-NGI Conference on*, pp. 141–148, IEEE, 2012.
- [32] R. Kooij, R. van der Mei, and R. Yang, “Tcp and web browsing performance in case of bi-directional packet loss,” *Computer Communications*, vol. 33, pp. S50–S57, 2010.

- [33] L.-T. Nguyen, R. Harris, J. Jusak, and A. PUNCHIHewa, "Modelling of quality of experience for web traffic," in *Network Applications Protocols and Services (NETAPPS), 2010 Second International Conference on*, pp. 84–89, IEEE, 2010.
- [34] R. Schatz, S. Egger, and A. Platzner, "Poor, good enough or even better? bridging the gap between acceptability and qoe of mobile broadband data services," in *Communications (ICC), 2011 IEEE International Conference on*, pp. 1–6, IEEE, 2011.
- [35] M. Andrews, J. Cao, and J. McGowan, "Measuring human satisfaction in data networks," in *Proceedings of IEEE INFOCOM06*, 2006.
- [36] A. Bouch, A. Kuchinsky, and N. Bhatti, "Quality is in the eye of the beholder: meeting users' requirements for internet quality of service," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 297–304, ACM, 2000.
- [37] H. Cui and E. Biersack, "Trouble shooting interactive web sessions in a home environment," in *Proceedings of the 2nd ACM SIGCOMM workshop on Home networks*, pp. 25–30, ACM, 2011.
- [38] D. Olshefski and J. Nieh, "Understanding the management of client perceived response time," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, pp. 240–251, ACM, 2006.
- [39] Y. X. Skadberg and J. R. Kimmel, "Visitors flow experience while browsing a web site: its measurement, contributing factors and consequences," *Computers in human behavior*, vol. 20, no. 3, pp. 403–422, 2004.
- [40] J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and P. Bahl, "Anatomizing application performance differences on smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 165–178, ACM, 2010.
- [41] D. Strohmeier, S. Jumisko-Pyykko, and A. Raake, "Toward task-dependent evaluation of web-qoe: Free exploration vs. who ate what?," in *Globecom Workshops (GC Wkshps), 2012 IEEE*, pp. 1309–1313, IEEE, 2012.
- [42] S. Egger, T. Hossfeld, R. Schatz, and M. Fiedler, "Waiting times in quality of experience for web based services," in *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*, pp. 86–96, IEEE, 2012.
- [43] "Understanding Mobile Web Browser Performance: Velocity 2011 - O'Reilly Conferences, June 14 - 16, 2011, Santa Clara." <http://velocityconf.com/velocity2011/public/schedule/detail/18241>, 2011. [Online; Accessed: 03-Dec-2012].

- [44] “RFC 1035 - Domain names - implementation and specification.” <http://www.faqs.org/rfcs/rfc1035.html>. [Online; Accessed: 07-Feb-2013].
- [45] http://www.w3schools.com/browsers/browsers_stats.asp, 2013. [Online; Accessed: 01-Mar-2013].
- [46] “The WebKit Open Source Project.” <http://www.webkit.org/>. [Online; Accessed: 01-Mar-2013].
- [47] <http://taligarsiel.com/Projects/howbrowserswork1.htm>. [Online; Accessed: 12-Jan-2013].
- [48] “How Chromium Displays Web Pages - The Chromium Projects.” <http://www.chromium.org/developers/design-documents/displaying-a-web-page-in-chrome>. [Online; Accessed: 11-Jan-2013].
- [49] “Response Times: The 3 Important Limits.” <http://www.nngroup.com/articles/response-times-3-important-limits/>. [Online; Accessed: 14-Jan-2013].
- [50] P. Arlos, M. Fiedler, and A. A. Nilsson, “A distributed passive measurement infrastructure,” in *Passive and Active Network Measurement*, pp. 215–227, Springer, 2005.
- [51] “PHP: Hypertext Preprocessor.” <http://php.net/>. [Online; Accessed: 19-Feb-2013].
- [52] “Apache: HTTP Server Project.” <http://httpd.apache.org/>. [Online; Accessed: 26-Feb-2013].
- [53] “OS Platform Statistics.” http://www.w3schools.com/browsers/browsers_os.asp. [Online; Accessed: 08-Mar-2013].
- [54] “KauNet: Deterministic Network Emulation.” <http://www.kau.se/en/kaunet>. [Online; Accessed: 22-Feb-2013].
- [55] P. Arlos, *On the Quality of Computer Network Measurements*. Blekinge Institute of Technology Doctoral Dissertation Series, Blekinge Institute of Technology, 2005.
- [56] “Incognito mode (browse in private).” <http://support.google.com/chrome/bin/answer.py?hl=en&answer=95464>. [Online; Accessed: 07-Jan-2013].
- [57] “Bind9.” <http://wiki.debian.org/Bind9>. [Online; Accessed: 17-Feb-2013].

- [58] “October 2012 Web Server Survey.” <http://news.netcraft.com/archives/2012/10/02/october-2012-web-server-survey.html>. [Online; Accessed: 02-Jan-2013].
- [59] “Codeigniter: A fully baked PHP framework.” <http://ellislab.com/codeigniter>. [Online; Accessed: 24-Feb-2013].
- [60] “MySQL: The worlds most popular open source database.” <http://www.mysql.com/>. [Online; Accessed: 04-Mar-2013].
- [61] “Fiddler: Web Debugging Proxy.” <http://www.fiddler2.com/fiddler2/>. [Online; Accessed: 26-Jan-2013].
- [62] “Methods for subjective determination of transmission quality.” http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-P.800-199608-I!!PDF-E&type=items, 1996. [Online; Accessed: 18-Jan-2013].