ENERGY EFFICIENT MOBILE WEB VIA SCRIPTS&STYLESHEETS BASED TRANSCODING

A THESIS SUBMITTED TO
THE BOARD OF CAMPUS GRADUATE PROGRAMS
OF MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

BY

HÜSEYİN ÜNLÜ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
SUSTAINABLE ENVIRONMENT AND ENERGY SYSTEMS PROGRAM

JANUARY 2019

Approval of the Board of Graduate Programs

Prof. Dr. Gürkan Karakaş
Chairperson

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Ceren İnce Derogar
Program Coordinator

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Yeliz Yeşilada Yılmaz
Supervisor

Examining Committee Members

Assoc. Prof. Dr. Yeliz Yeşilada Yılmaz	Computer Engineering Prog. METU NCC	
Assoc. Prof. Dr. Enver Ever	Computer Engineering Prog. METU NCC	
Prof. Dr. Simon Harper	School of Computer Science The University of Manchester	

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Hüseyin Ünlü

Signature :

ABSTRACT

ENERGY EFFICIENT MOBILE WEB VIA SCRIPTS&STYLESHEETS BASED TRANSCODING

Ünlü, Hüseyin
M.Sc., Department of Sustainable Environment and Energy Systems Program
Supervisor : Assoc. Prof. Dr. Yeliz Yeşilada Yılmaz

January 2019, 119 pages

Mobile devices have become essential in our daily lives and the requirements of the mobile platform are increasing everyday. Although mobile devices nowadays are technically much more stronger than the desktops in the past and their battery capacity is increasing at around 3% per year, they still have some limitations in terms of battery size, processing power and device memory. These limitations have effects on browsing web pages since they are not totally designed for mobile use and it takes more power than necessary on the client side. In order to save energy and extend the battery life, there are some guidelines for web site programmers. However, most programmers are not aware of these guidelines and therefore most web sites do not adhere to these guidelines. Important components of modern web sites are the scripts that make them dynamic and stylesheets that are used for visual rendering. These two are external components of web sites that are shown to have effect on the downloading time of web pages. This MSc thesis first investigates the effect of scripts and stylesheets on the energy consumption of web pages on mobile devices, and then propose two techniques, which are (1) concatenating

external script and stylesheet files and (2) minifying external script and stylesheets, that can be used to transcode web pages to improve energy consumption and therefore improve battery life on the client side, without changing the look&feel of the web pages and without adding extra load on the client side or the server. The evaluation results show that the proposed techniques achieved statistically significant energy saving.

Keywords: Transcoding, Energy Saving, Browsing, Mobile Web, Web Engineering

ÖZ

WEB SAYFALARININ ENERJİ TASARRUFU İÇİN SCRİPT VE STİL ŞABLONLARINA DAYALI DÖNÜŞTÜRÜLMESİ

Ünlü, Hüseyin
Yüksek Lisans, Sürdürülebilir Çevre ve Enerji Sistemleri Programı
Tez Yöneticisi : Doç. Dr. Yeliz Yeşilada Yılmaz

Ocak 2019, 119 sayfa

Mobil cihazların günlük hayatımızdaki yeri çok önemli yerlere ulaştı. Bunun sonucunda mobil platform gereksinimleri hergün artmaktadır. Günümüzdeki mobil cihazlar teknik olarak geçmişteki masaüstü bilgisayarlara göre çok daha güçlü olmasına ve batarya kapasitesi her yıl %3 artmasına rağmen hala batarya kapasitesi, cihaz hafızası ve işlemci kapasitesi konularında bazı kısıtlamalar ile karşı karşıyadır. Bu kısıtlamaların sonucunda web sayfalarına erişim kullanıcı tarafında bataryanın hızlı ve gerektiğinden fazla olarak tükenmesine neden olabilmektedir. Enerji dostu web sayfası için önerilen teknikler olmasına rağmen çoğu programcı bu tekniklerin farkında olmadan web sayfası yaratabiliyor. Günümüz modern web sayfalarının önemli dış bileşenlerinden stil şablonları sayfayı görsel olarak geliştirmeye, script ise dinamik bir web sayfası yaratmak için gereklidir. Ancak, bu iki bileşenin sayfanın yüklenme süresine olumsuz yönde etkileri olduğu saptanmıştır. Bu yüksek lisans tezi, ilk olarak script ve stil şablonlarının web sayfalarının enerji tüketimine olan etkilerini ortaya koymayı ve sonrasında ise script ve

stil şablonları ile ilgili önerilen iki tekniği kullanarak web sayfalarını daha az enerji tüketeceği bir şekile dönüştürmeyi ve bunların sonucunda batarya ömrünü geliştirmeyi amaçlamaktadır. Bu dönüştürme tekniklerinin amacı web sayfasının görünüşünü değiştirmeden ve kullanıcı tarafında veya sunucu tarafında ekstra yük oluşturmadan, kullanıcı tarafında enerji verimliliği sağlamaktır. Değerlendirme sonuçları kullanılan tekniklerin istatiksel olarak kayda değer bir enerji tasarrufu sağladığını göstermektedir.

Anahtar Kelimeler: Dönüştürme, Enerji Tasarrufu, Tarama, Mobil Web, Web Mühendisliği

To my family, friends and Merve,
Thanks for always being there for me.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor Assoc. Prof. Dr. Yeliz Yeşilada for the continuous support of my master study, for his patience, motivation, understanding and immense knowledge. Her guidance helped me in all the time of my bachelor studies and throughout my master. research and writing of this thesis. I could not have imagined having a better supervisor and mentor for my study.

Beside my advisor, my special thanks to the examining committee members: Assoc. Prof. Dr. Enver Ever and Prof. Dr Simon Harper for their precious time and effort reading my thesis and for their valuable comments.

Thanks also go to my friend Altınok Darıcı for his valuable feedback to improve this study. I would also like to thank Assoc. Prof. Dr. Enver Ever for his guidance on the network and modeling parts of the study. I am also very grateful to Dr. Şükrü Eraslan for his help during my bachelor degree and throughout my master.

I would thank all my friends, who believed in me and who laughed with me. Especially the beautiful people whom I met at METU NCC for being in my life: Anıl, Güzin, Yiğit, Ege, Erdem, Burak, Melike and Berk.

Finally, I would like to thank my family Ata Ünlü, Emine Betül Aksoy and Ahmet Ünlü for supporting, motivating, understanding and encouraging me during my life. The last thanks go to Merve, the special person in my life. I am extremely grateful for motivating and supporting me and not letting me give up.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2G	Second-Generation Mobile Network
3G	Third-Generation Mobile Network
4G	Forth-Generation Mobile Network
CDN	Content Delivery Network
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DNS	Domain Name System
DOM	The Document Object Model
FTP	File Transfer Protocol
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IBM	International Business Machines Corporation
ICAP	Internet Content Adaptation Protocol
ICT	Information and Communication Technologies
JS	JavaScript
LTE	Long-Term Evolution
MEMS	Microelectromechanical systems
SMS	Short Message Service
SPDY	SPeeDY Protocol
SPSS	Statistical Package for the Social Sciences
SW	Shapiro Wilk Test
TCP/IP	Transmission Control Protocol/Internet Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WWW	World Wide Web

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Statement

Global warming is one of the most crucial threats for our world. Indeed, our world is now facing with its effects such as the rise in water level and the temperature. One of the most important cause of global warming is carbon emission. Since the beginning of the industrial revolution, the amount of carbon emission has been increasing [138]. There are various sectors that cause carbon emission with different rates. Information and Communication Technologies (ICT) is one of the contributing sectors with the rate of 2% - 2.5% in overall carbon emission, which is almost higher than the contribution of Global Aviation Industry which is around 2% [15, 46]. Thus, the carbon emission of ICT cannot be negligible. When the contribution of main sectors in ICT is analyzed, the breakdown is as follow: energy requirements of personal computers and monitors contribute 40%, data centers contribute 23% and fixed and mobile telecommunications contribute 24% [46].

In the past, the only way to access World Wide Web (the web), which is a leading source of information, was via desktop computers. In 1994, the first mobile device, Simon, was produced by IBM [47]. This mobile device, which is the ancestor of smart phones, started a new category of technology. Since then, the popularity of mobile devices has been increasing and users access the web via mobile devices. Now, mobile devices has an important role in our daily lives. Today, the number of mobile devices is almost the same with the people who has access to drinkable

(%)

	Year												
	2004	2005	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Desktop computer	10,0	11,6	24,0	28,1	30,7	33,8	34,3	31,8	30,5	27,6	25,2	22,9	20,3
Portable computer (Laptop, Tablet PC)	0,9	1,1	5,6	9,1	11,2	16,8	22,6	27,1	-	-	-	-	-
Portable computer (Laptop, netbook, tablet)	-	-	-	-	-	-	-	-	-	40,1	43,2	-	-
Portable computer (Laptop, netbook)	-	-	-	-	-	-	-	-	31,4	-	-	36,4	36,7
Tablet computer	-	-	-	-	-	-	-	-	6,2	-	-	29,6	29,7
Mobile phone (incl. smart phone)	53,7	72,6	87,4	88,1	87,6	90,5	91,9	93,2	93,7	96,1	96,8	96,9	97,8
Fixed line telephone	81,6	81,3	72,7	68,4	61,9	56,1	51,4	45,5	37,9	34,6	29,6	25,6	20,2

TurkStat, Survey on Information and Communication Technology (ICT) Usage Survey in Households and by Individuals, 2004-2017

Figure 1.1: Gadget Ownership Statistics in Turkey between 2004-2017 [23]

water [110]. Figure 1.1 shows the gadget ownership in Turkey, between 2004 and 2017 [23]. From this figure, it can be seen that the number of mobile devices is increasing rapidly while the number of desktop computers is decreasing.

Another survey shows the number of Internet subscribers and the number of mobile telephone subscribers in Turkey, between 1994 and 2015 (see Figure 1.2). This figure shows that the number of mobile device subscribers has been increased rapidly in Turkey, since 1994 [32]. According to another survey, smartphone dependency is also increasing between Americans. Today, just over one in ten American adults are using Internet on smartphones only. In other words, they own smartphone while they do not have traditional home broadband service [50].

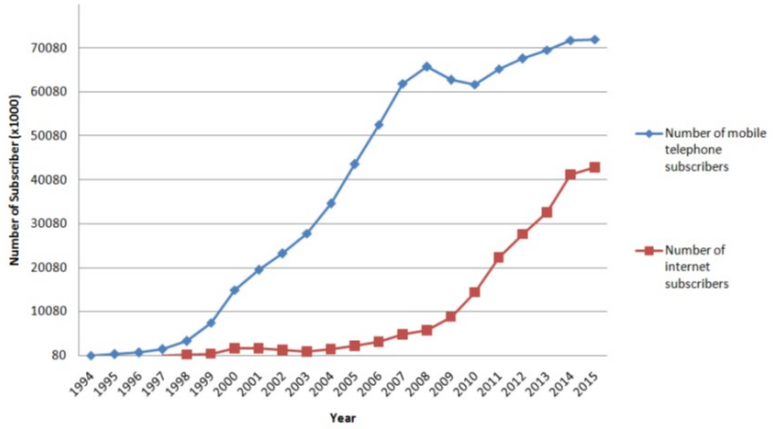

Figure 1.2: The Number of Internet Subscribers and the Mobile Devices in Turkey between 1994-2015 [32]

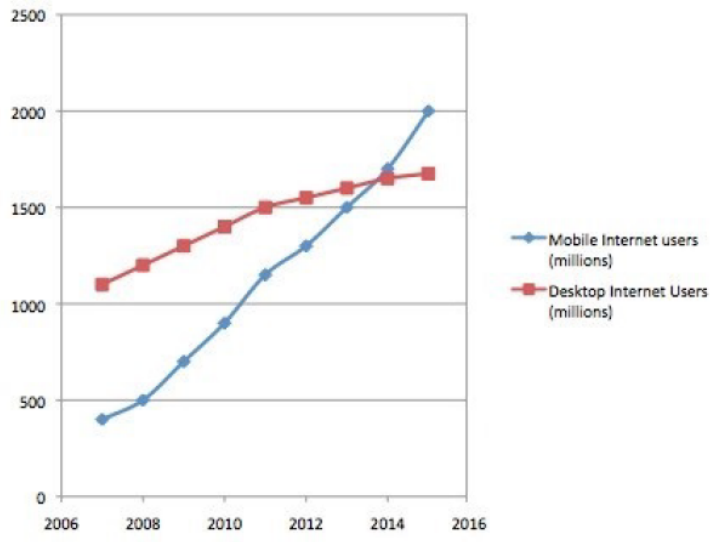

Figure 1.3: Web Access from Desktop to Mobile, Forecast in 2011 [110]

According to a forecast in 2011, it was expected that mobile web would overtake desktop web by 2014 (see Figure 1.3). In 2011, the number of people who access the web through their mobile phones was 900 million and this number was 1.4 billion for desktop Internet users. Their forecast estimates that there will be 1.7 billion mobile web users and 1.65 billion desktop web users. Today, when we compare the number of web access from desktop and mobile, it can be seen that mobile web overtook desktop web (see Figure 1.4). Another forecast estimates the number of connected devices per person (see Figure 1.5). According to this forecast, the number of connected devices per person is 3.47, in 2015 and this ratio will reach to 6.58 by 2020.

Alongside the increasing web access from mobile, web pages have also been improving for better user experience. For that purpose, they include more images and videos in high resolution and the size of the web pages is also increasing. Since 2011, the average web page size increased more than 2 MB [72]. By 2017 July, the average web page size increased to 3034 KB while it was 929 KB in 2011. The expectation is that the average size will be more than 4 MB by 2019 (see Figure 1.6). Indeed, 16% of the web pages are greater than 4 MB today [72]. The number of requests are also increasing by the time. The number of requests increased from 86 to 109 between December, 2011 and December, 2017 [40].

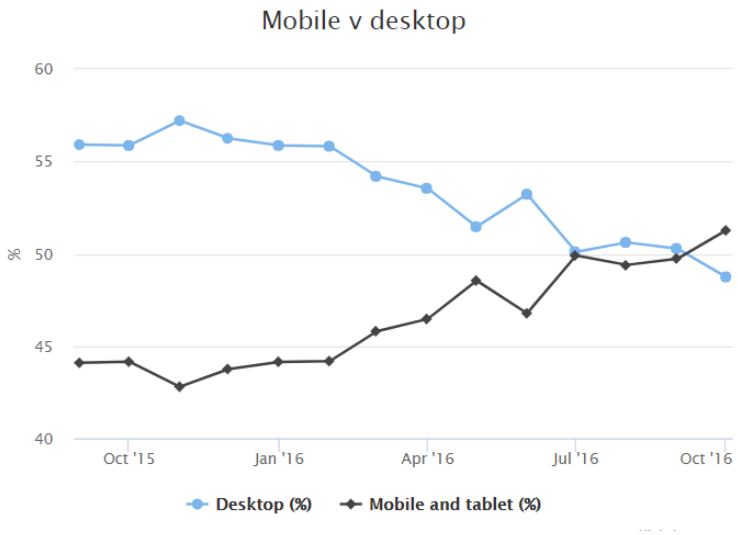

Figure 1.4: Web Access from Desktop to Mobile [136]

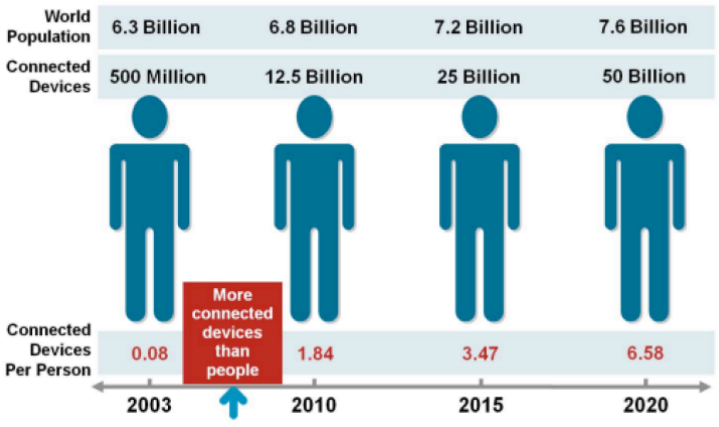

Figure 1.5: The Number of Connected Devices versus the Population [70]

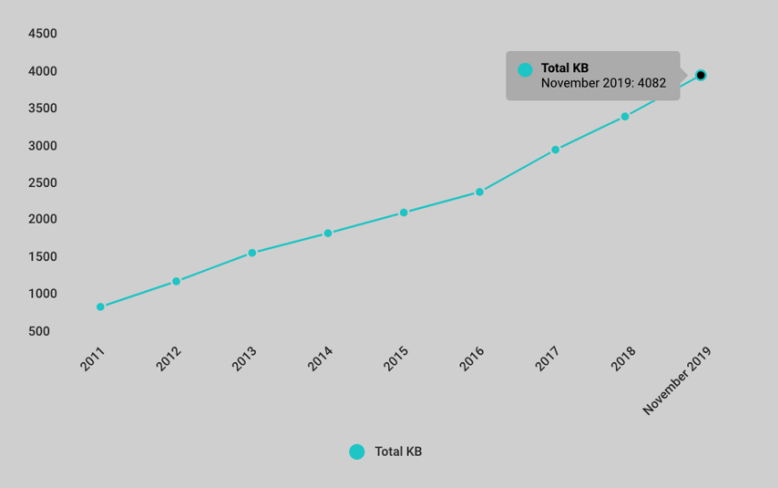

Figure 1.6: The Average Web Page Size between 2011 and 2019 [72]

Web pages consist of different elements and the main web page elements can be listed as HTML, scripts, stylesheets, images, videos and fonts. The reason behind the increasing size of web pages is that web pages include more elements with higher quality and this increases the size. Moreover, they are more dynamic, hence they include more scripts and stylesheets. Figure 1.7 shows the page growth from 2011 to 2017, broken out by content type. According to the figure, it can be seen that the the biggest pie of the size includes images and videos. Furthermore, the size of images and videos has been increasing over the years. Apart from images and videos, the size of scripts and stylesheets is also high. Page size is a factor that negatively affects the performance and increases the energy consumption of a web site but it is not the only factor [72].

Over the years, the web has been evolved to give better user experience and the mobile devices become the primary way to connect Internet. Alongside the improvements in the web, the computers and network have also been improved. Now, computers have big screens, fast network connections, power access, powerful processing power and almost endless memory. Current mobile devices are also very powerful, especially if they are compared the computers in the past. Indeed, they are as powerful as an average desktop now [150]. However, mobile device users may not have sufficient user experience as they expect since most of the web pages are not used in ideal conditions and they are not in average case. The reason is that mobile devices

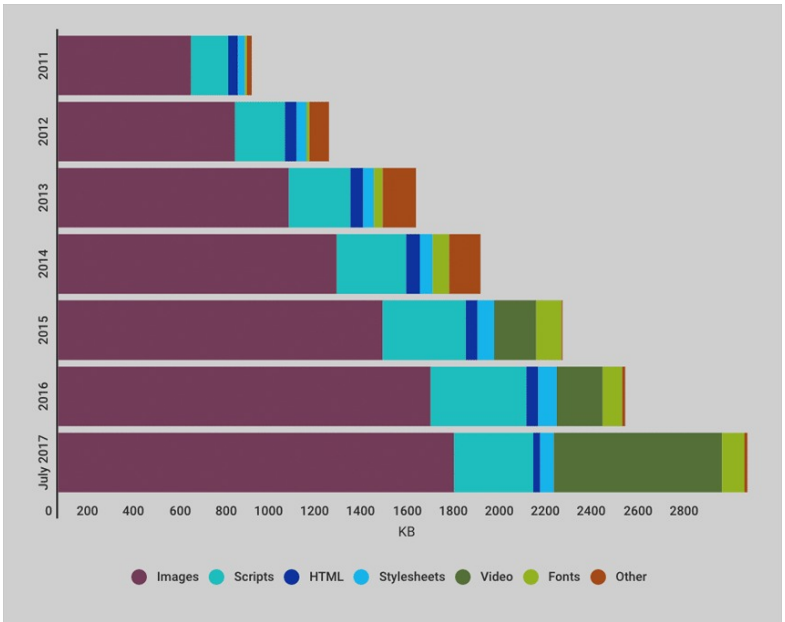

Figure 1.7: Page Growth Broken by Content Type between 2011 and 2017 [72]

have some limitations and these limitations can be listed as screen size, limited bandwidth of the connection, the number of connections, device memory, processing power and the battery [71, 100, 98].

Mobile devices are much smaller than desktop computers in size and this is the main reason of the mentioned limitations. Today, web pages are designed to be shown in large screens to provide better user experience, with a rich content. However, when these web pages are requested by mobile devices, all of the content requested can be rendered but the user is not able to see all of the content due to lack of space. Some of the web pages have also specific mobile versions to overcome this problem but the client may still request the desktop version.

Another result of the smaller size is the processing power. Due to the lack of space, it is not possible to integrate similar hardware of desktop computers to mobile devices and this results with lack of processing power. Thus, mobile devices are slower and have lower performance compared to desktop computers [121].

As the size is small, mobile devices have less device memory. Although their memory is much more than the desktop computers in the past, now less memory capacity is a constraint in mobile devices. Images and scripts are one of the problem related with the memory. Since they are loaded in to the DOM tree, if the number of scripts or images is high, it may cause slow performance or a crash, as a result of the limited memory [150].

Limited bandwidth of the connection is another constraint for mobile devices. Wi-Fi and cellular connection are slower than the wired connection if we do not consider 5G as there are some claims that 5G will be faster than the wired connection [143]. Thus, it consumes more power when a web page is loaded slowly and it requires more request and response [100].

To sum up, mobile devices have some limitations compared to the desktop computers. Mobile devices are more likely to be used for browsing. However, the mentioned limitations cause more energy consumption while browsing and this leads battery to drain fast [77]. To illustrate, less number of connections cause latency and under a latency, devices consume more energy. Furthermore, limited bandwidth increases the loading time and cause request-response round trips and this also drains the battery. For example, the battery life of an iPhone 7 is different under different connection types. It is up to 384 hours on standby, 13 hours on the cellular connection and 15 hours on Wi-Fi connection [24].

Web pages consist of different elements and energy consumption of these elements differs in each web page. In a study, energy consumption of web page elements is analyzed in top web sites (see Figure 1.8). Their measurements were performed in 16 popular web sites are measured the energy consumption of images, JavaScript, CSS and the other elements, on a mobile phone. The results show that energy consumption of JavaScript and CSS is high alongside images. To illustrate, 10 Joules of energy is needed to download and render JavaScript in the Wikipedia page, which is about 30% of total energy to download and render the page. Another example, web site of Apple consumes 12 Joules to load and render CSS, which is about 40% of the total energy consumption.

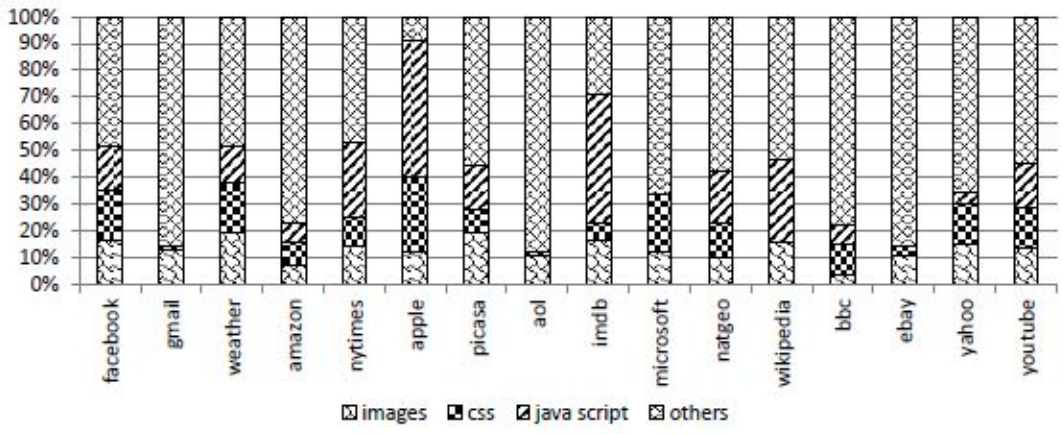

Figure 1.8: Energy Consumption of Web Page Elements in Top Web Sites [134]

The energy hungry elements in a web page are images, scripts and stylesheets, as mentioned before. However, they are very important elements for a web site especially for presentation and interaction. CSS and JavaScript are the core technologies alongside HTML to create a web site. Cascading Style Sheets (CSS) is one of the most used technology in stylesheets and describes how HTML elements are to be displayed on screen, paper, or in other media. It saves a lot of work. It can control the layout of multiple web pages all at once. JavaScript is the leader programming language to make web pages interactive. JavaScript code mainly runs on the client side so it puts extra load on the client side. JavaScript engine of the browsers is responsible from the work on the client side and their performance may differ with the browsers and platforms (see Figure 1.9). Indeed, JavaScript engine performance is slower in mobile devices compared to the desktop versions [150]. Although current mobile devices have higher performance, they parse and execute web pages ten times slower than desktop computers [109]. When a JavaScript code is downloaded, code execution is done in the Central Processor Unit (CPU) of the device and power is needed to execute. In this case, processing power of the device is proportional with the performance of the CPU. In other words, high processing power means less time in code execution. However, the processing power of the mobile devices is limited. Thus, JavaScript execution causes faster drained batteries [150].

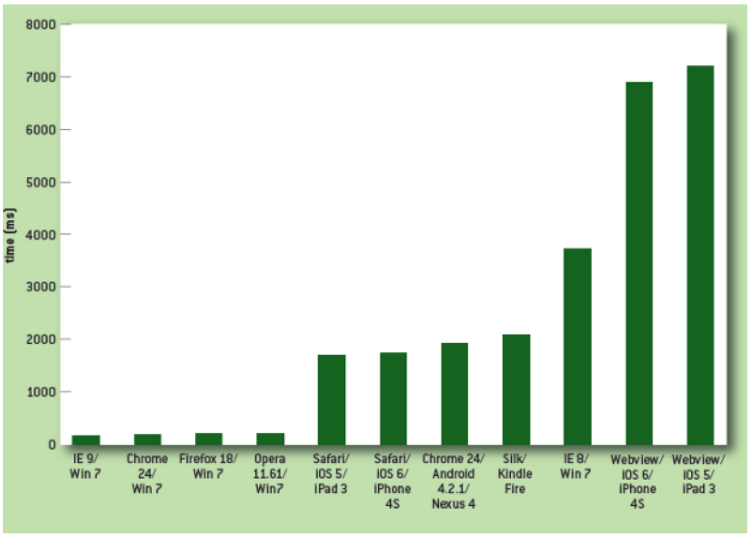

Figure 1.9: JavaScript Engine Performance Comparison among the Browsers [150]

The mentioned limitations of mobile devices cause more energy consumption while browsing and this leads battery to drain fast. Thus, the lifespan of the battery is shortened and they become waste. The battery waste contribute many potentially hazardous compounds as it includes zinc, lead, nickel, alkalines, manganese, cadmium, silver, and mercury. Recycling batteries may be a caution but it keeps heavy metals out of landfills and the air [126]. Thus, it is important to increase the lifespan of a battery of the mobile device to contribute sustainability.

1.2 Objective

In this thesis, the overall objective is to save energy on the client side with transcoding. The number of mobile devices is increasing and people browse from mobile more than desktop computers. However, mobile devices have some constraints and these constraints cause battery to drain faster. In the literature, there are different guidelines for high performance web sites. However, there are almost no study that analyzes the impact of these guidelines on energy saving. There are two main ideas behind energy saving in web sites: (1) reducing the payload and (2) reducing the number of requests.

Our goal is to transcode web pages without modifying their look&feel and without adding extra load to the client or the server side with the goal of energy saving via scripts and stylesheets optimization. Two techniques will be implemented in a proxy server: (1)concatenating external JavaScript and CSS files to reduce the HTTP request and (2)minification of CSS and JavaScript to reduce the size.

At the final stage, we aim to have energy efficient mobile devices while browsing. However, the energy consumption of proxy server cannot be negligible. For that purpose, a analytical model will be included to show the relation between energy saving on the client side and energy consumption of the proxy server.

1.3 Contributions

The contributions of this thesis are listed as follow:

Novelty: In the literature, there are various studies about transcoding in the proxy. Also, there are different guidelines for high performance web sites. However, almost none of these guidelines were evaluated with respect to energy. Our system implements two of these techniques in the proxy server and evaluate their impact in terms of energy saving, which is novel.

Improving battery life of mobile devices: The aim of this thesis is to achieve energy saving on the client side to improve the battery life of the mobile devices. When a client requests a web page, most of the duration is passed on the client side while downloading the resource. As this duration becomes longer, mobile device will be connected to the Internet longer and there will be more request and response. In this thesis, our goal is to decrease this duration with concatenation of external files and minification. Thus, we aim to improve battery life of the mobile device by saving energy on the client side.

No extra load: Our system does not put extra load on the client side or on the server side. All of the transcoding is done on the proxy server. Client does not need to install extra program and developer does not need to modify the web page.

Automated transcoding: In the literature, there are different examples of studies that attempts to save energy on the client side. Moreover, there are different guidelines for developers to modify their web page in order to achieve higher performance in their web sites and there are many number of web sites designed without considering these guidelines. In our system, web pages are automatically transcoded in the proxy server.

Contribution to Sustainability: The aim of this thesis is to achieve energy saving while browsing on a mobile device. Saving energy on a mobile device will increase the lifespan of the battery and it will contribute to sustainability when we consider the harmful effects of the battery disposal. The techniques are implemented on a proxy server to provide energy saving on the client side. However, in order to provide overall energy saving, the energy consumption of the proxy cannot be negligible. In this thesis, we developed a model between the energy consumption of the proxy server and the energy saving on the client side. Thus, we contribute to the sustainability.

1.4 Thesis Outline

The remaining part of this thesis is organized as follow:

Chapter 2 Related Work: This chapter gives detailed literature review about mobile web. Firstly, different ways to access Internet and the concept of HTTP are given. Secondly, energy related studies in hardware, network and software are given. Following the energy related studies, transcoding is presented with its different types and some studies about transcoding are given. Lastly, guidelines for high performance web sites are given and CSS and JavaScript related guidelines are discussed in energy aspect.

Chapter 3 Methodology: In this chapter, the techniques used in this study are given in detail. Then, architecture of the study is mentioned. Lastly, it gives the implementation details of our study.

Chapter 4 Evaluation: This chapter gives the results of our system with discussion. We have two research questions to evaluate our system: *"Does the system allow saving on the client side energy by concatenating external files to reduce the number of HTTP connections?"* and *"Does the system allow saving energy on the client side by minifying scripts and stylesheets to reduce the size of the file?"*. The measurements in desktop and mobile are done to answer these questions. Apart from the results and its discussion, equipment, material and tools are also mentioned in this chapter.

Chapter 5 Modeling Sustainability: This chapter gives details of our sustainability model. As mentioned before, the goal of this thesis is to provide energy saving on the client side. However, the energy consumption of the proxy cannot be negligible when sustainability is considered. In this chapter, our model is discussed.

Chapter 6 Conclusion and Future Work: This chapter gives the conclusion of our thesis mentioning the possible future work.

CHAPTER 2

RELATED WORK

The main goal of this research is to decrease the energy consumption of web browsing based on script and stylesheets optimization and transcoding, without changing the look&feel and without putting extra load on server or client side. Therefore, this chapter first reviews the foundations on web technologies, discusses the previous work and highlights the gaps in the literature. The first section includes some background information about HTTP and ways to access web from mobile devices. In the second section, energy related studies from hardware, network and software levels are given. Following the energy related studies, transcoding is explained with different transcoding techniques and implementations of these techniques. In the next section, recommendations for web site developers from seven different guidelines are discussed. Then, techniques on JavaScript and CSS are explained and implementations about these studies are given. The last section discusses the gaps in the literature.

2.1 Background

Mobile web allows users to access browser based Internet services from mobile devices such as smartphones or tablets, instead of desktop computers. The difference of mobile web from traditional web is that communication is done through a mobile or wireless network instead of a fixed-line service. However, both of them uses the same protocols at application layer such as Hyper Text Transfer Protocol (HTTP) and Transmission Control Protocol/Internet Protocol

(TCP/IP). Mobile web users can access both desktop web pages and mobile web pages. Although there are some constraints of mobile devices, their important characteristics can be listed as being portable, personal, easy to use and having access to network connection [77]. This section first introduces the concept of HTTP request and response and then explains different ways that can be used to access the web from mobile devices.

2.1.1 Fundamentals of HTTP Protocol

Users can access web via mobile devices or desktop computers. HTTP introduces the protocol of requests and responses to access web from both mobile and traditional devices and it is used by the web since 1990. The main idea behind this protocol consists of requests and responses. The client should establish a connection and create requests to download the content of a web page. In other words, client send requests to the server and server sends the response to the client. There may be different level of elements as an intermediary form between client and server such as proxy and gateway. Briefly, requests and responses are generated by the client and the server in order to provide a communication between them and take the content of a web site or data [74].

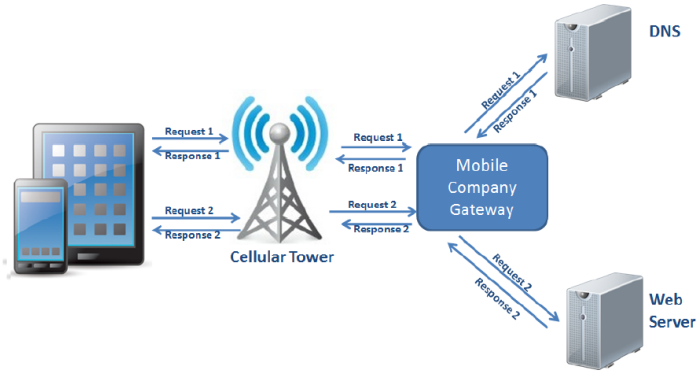

Figure 2.1: HTTP Request from a Mobile Device [91]

In every HTTP specification, there are different rules [74]. To illustrate, in HTTP 1.0, different connections are established for each request and response. On the other hand, in HTTP 1.1,

requests and responses are sent in order and then the communication stops after all the traffic is finished. In HTTP 1.1, the client does not need to wait the response for the request before sending a new request on the same connection. This feature is called as pipelining and it improves the performance [44]. Although there are different rules for each specification, the path of the HTTP requests consist of the same steps. Firstly, when a client visits a web page first time, it is needed to take the physical address (IP) of a web page from Domain Network Service (DNS). Client sends the first request to take the IP address of a web page from DNS and this request passes through cellular tower and then mobile company gateway, before delivering to DNS. Domain Network Service sends the request, IP address, to the client through the mobile company gateway and cellular tower. After client takes the physical address of the web site, second request is generated in order to download the content of the web site. Similar to first request, it passes through nearest cellular tower and then mobile company gateway. When it is delivered to web server, it will generate a response and sends the content of the web site to the client, from the same path (see Figure 2.1).

When a user requests a web page, HTTP request is generated. Actually, more than one HTTP request is needed to show the content of a typical HTML page as web pages composed of different part such as CSS, scripts, images and videos and they are included as external files in general. For each external element, there should be an HTTP request. However, increasing number of HTTP requests means a waiting time for the client and this duration may be longer under low bandwidth and high latency. Thus, battery of the mobile device drains with the increased loading time. Around 80% of this loading time is used for loading the source and client side processing [71]. The remaining 20% of this period is used for displaying the content. This is same for both desktop and mobile devices. In a study [134], according to the measurement, the energy consumption of establishing the connection for downloading and uploading is around 12 Joules. In the same study, they found another interesting result related with the traffic. They measured the energy consumption of uploading 8 KB at 1 KB per iteration and at once. Their result shows that iteration consumes 5% more energy than at once. This shows the impact of HTTP traffic over mobile devices. In another study [150], they suggest that decreasing the number of HTTP requests and the size of the resources will improve the web page performance. In our study, the

Table 2.1: Comparison of Ways to Access Web

	Pros	**Cons**
Native Applications	*High performance *Hardware access *Push notifications *Standalone application	*Compatible with only one platform *High development cost *One needs to maintain multiple applications
Mobile Web Applications	*Low development cost *Cross-platform *Straightforward development *Do not take up any memory or storage on the user's device	*Extra browser layer *Low performance *No hardware access *Internet connection is required to work
Hybrid Applications	*Cross-platform *Moderate development cost *Moderate performance *Hardware access	*Extra Webview layer
Web Sites	*Can be displayed in all platforms *Straightforward development *Low development cost	*No hardware access
Widgets	*Standalone applications	*Compatible with only one platform

main goal is to decrease the size of the resources and the number of HTTP requests in order to reduce the energy consumption on the client side which is suggested by the mentioned studies.

2.1.2 Ways to Access Web from Mobile Devices

Mobile device users may access web via five different ways: native applications, web applications, hybrid applications, web sites and widgets. The summary of advantages and disadvantages of each way is given in Table 2.1 which are introduced as follow:

1.Native Applications are specific for a mobile platform, firmware and operating system. They are directly installed onto a mobile device and the client needs to download the application from an application store such as Apple Store or Google Play [84]. They are installed and run as a standalone application which means that no web browser is needed. Native applications can access to hardware of the mobile device which means that they can interface with the device's native information features and hardware such as camera and accelerometer. This advantage of native applications gives better user experience [88]. The main drawback of native applications is that they are compatible with only one platform and they should be reimplemented to be used in different platforms and this makes them expensive [84]. To illustrate, one needs to have one application for Android and one application for IOS. To illustrate, Twitter and Pokemon Go are examples of native applications.

2.Mobile Web Applications are Internet-enabled applications that have specific functionalities for mobile devices. They have to be accessed via mobile web browsers. Since they are accessed through web browsers of mobile devices, there is no need to download and install them on to device, so they do not take up any memory or storage on the user's device. Indeed, they are combination of HTML5, JavaScript and Cascading Style Sheets (CSS) [88]. Opposite to native applications, mobile web applications can be used cross-platforms which means if a mobile browser supports HTML5, JavaScript and CSS, they can be used on the device. Its development process is more straightforward comparing with native applications and this takes attention of non-programmer developers [125, 84]. Moreover, mobile web applications can create an icon over o mobile device which makes them similar to native application and this means that user does not need to use bookmarks [125]. However, there is an extra layer, browser, for mobile web applications and because of this layer their performance are lower than native applications [125, 84]. Another disadvantage of mobile web applications is that they are dependent on the Internet connection so, they cannot be used without Internet. For example, when you enter Facebook from the browser, it is an example of mobile web application.

3.Hybrid Applications can be introduced as a combination of native applications and web applications [122]. They can be installed from an application store like native applications, however,

they are web applications. They are combination of HTML, JavaScript and CSS and run in a simplified browser in the application. This simplified browser is called Webview [97]. Hybrid applications are cross-platform which means that they can be used on different platforms with one codebase. Since they are coded once to be performed in different platforms, they are cheaper than native applications [122]. Moreover, they have ability to access device features thanks to solutions like PhoneGap [33]. However, their performance are poor since there is an extra layer which is responsible for displaying the user interface and running the JavaScript code. Yelp and Instagram are examples of hybrid applications.

4.Web Sites are typically available in two versions: mobile web sites and desktop web sites. Desktop computers have big screens, fast connection and high speed processors. Desktop web pages are designed based on these computer specifications to provide better user experience. However, mobile devices have some limitations unlike desktop computers. Although desktop web pages can be seen from mobile browsers, some web pages have also mobile versions which are designed to be rendered under these limitations. Responsive web design shows the same web page in different ways. If the web page is requested from mobile device, mobile version will be loaded, otherwise desktop version will be loaded from the server. The idea behind loading mobile web pages is same with responsive concept. When user requests a web page from a mobile browser, server checks the user-agent identifier of the request and sends the specific version of the web page or the one designed based on responsive concept. However, there may be some differences in the layout and content of the web page, between mobile and desktop versions [88].

Web sites are displayed on a web browser. A web browser is a program with a graphical user interface for displaying HTML files, used to navigate the web. Nowadays, there are different browsers in the market such as Google Chrome, Microsoft Edge, Mozilla Firefox, Apple Safari and Opera [30]. According the statistics, Google Chrome is the most popular browser with 76.9% in August, 2017 [30]. In mobile platforms, there are different browsers to access web from mobile. The most popular ones can be listed as Apple Safari, Google Chrome, Opera Mini and Mobile, Mozilla Firefox Mobile, Amazon Silk, Blackberry Browser, Nokia Browser and

Microsoft Edge Mobile.

5.Widgets are small and task specific applications. Similar to native applications, they are standalone and they communicate with a specific web server. They cannot be used cross-platforms and they need to be modified to be used over different platforms [84]. Samsung TouchWiz is an example for widgets.

2.1.3 Summary

In summary, this section first introduces the HTTP. It is based on requests and responses in order to download the content of a web page, from server to the client. It is also mentioned that under low bandwidth and high latency, the loading time of the resources is increased. Thus, energy is consumed more. Another important factor about energy consumption is number of requests and the energy consumption will be higher if the number of requests increases. After introducing HTTP, five different ways to access web from mobile devices are mentioned. The summary of these ways can be seen in Table 2.1 introducing the advantages and disadvantages of mobile web applications, native applications, hybrid applications, web sites and widgets. Widgets and native applications uses Internet in their background but they are task specific. Moreover, they are not suitable for cross-platforms. Mobile web applications and hybrid applications are cross platforms but again they are designed to perform specific task. The last category, web sites, can be accessed from all of the browsers without any restriction. Responsive web sites manages the version of the web site. If it is requested from a mobile device, mobile version of the site is downloaded. Otherwise, desktop version is downloaded. Web sites are displayed with browsers. Browsers are only way to access the Internet which is not task specific. Another feature about browsers that makes them a good candidate for this study is that they are not blocked by applications security. Moreover, as they are not task specific, there are lots of web sites that can be included in our evaluation set. On the other hand, mobile applications may be blocked by applications security and they are task specific and this is why this study focuses on to web sites.

2.2 Review of the Energy Related Work

There are different studies in reducing the carbon footprint of mobile communication, saving energy from web pages in different levels and improving the battery life of mobile devices. If we consider mobile devices, we can see that this can be done at the hardware, software and network level.

2.2.1 Hardware Level

There are different kind of studies to reduce the energy consumption of mobile devices in the hardware level focusing different areas such as reducing the energy consumption of the processor [81, 108], embedding renewable energy resources into mobile devices [105, 137], analyzing the role of the processor in energy consumption of mobile web browser [151] and reducing the energy consumption of the screen [129].

Energy saving is studied by Intel in their different processor architectures. In Intel Core Duo they introduced high performance with low power consumption [81]. Their energy saving has evolved with Intel Core 2 Duo [108]. In [151], they state the role of the CPU in energy consumption of mobile web browsers. Their results show that the impact of CPU on mobile web browser energy consumption depends on the network latency. If there is a low network latency, high CPU performance achieves energy efficiency in browser and also faster web page load. However, under high latency, low CPU performance makes the system energy efficient without change in browsing performance.

Some studies focus on using renewable energy resources to provide battery life improvements in mobile devices. In [105], they integrated thermoelectric generator into CPU heat pipe to use the waste heat of the processor and generate electricity. Another approach in this study is integrating photo voltaic unit into a mobile device and generating power from environmental illumination. In another study [137], they introduce a tool that estimates the output energy power level of Micro Electromechanical Systems (MEMS) which converts low frequency vibration to electrical

energy. This energy scavenger is used for mobile computing and wireless sensor applications. There are also studies that focus on saving energy from screens. [129] is an example for the studies that attempts to optimize the power consumption, focusing on the display.

In summary, energy saving is studied at hardware level in different aspects. It is important to reduce the energy consumption in hardware level. However, hardware is expensive and it evolves slowly [91]. Also, software and network have an impact on the energy consumption of the hardware. Thus, it is not enough only to reduce the energy consumption on only hardware level, so other aspects need to be considered.

2.2.2 Network Level

There are different studies about energy consumption on network level such as analyzing the energy consumption of the mobile network [69], the impact of different communication technologies on mobile devices [116, 147, 119, 118, 54], using localization [57] and caching [124] to improve energy saving in mobile networks. Network has an impact on energy consumption of mobile devices, however the energy consumption of the network is very high compared to consumption of mobile devices. In [69], they show the energy consumption of mobile of a customer for a terminal and for mobile network. According to the their results, energy consumption of a customer for a terminal is 0.83 Wh/day while it is 120 Wh/day for the mobile network. Moreover, their results also show the net electricity consumption of data center per day for one mobile phone user is 70 Wh. When the energy consumption of the terminal is compared with mobile network, it may be negligible. However, they state that mobile devices are energy starving because of their battery limitations.

It is argued in [116] that as the operation duration of the mobile networked device increases, the problem about energy consumption is hard to be solved if mobile networking will be the same. According to the results of the study, Iphone 3G lasts 300 hour in standby, 5 hours with 3G connection and 6 hours with Wi-Fi. Their solution is adopting the network paradigm to information-centric approach and using multi access mobile networking. In this approach,

surrounding networks are used together. Their results are promising but infrastructural change is needed.

Mobile devices have different energy consumptions under different network technologies. In [119], they analyze the energy consumption of mobile device entities such as data communication (bluetooth, Wi-Fi, 2G, 3G), cellular link services (hand-off, sms, voice call, video call), display, screen update, mobile tv and CPU. [147] gives the energy consumption of mobile Youtube under 3G and WLAN. [118] is another study about analyzing the impact of different network technologies and it gives the energy consumption comparison for 2G and 3G using different tasks such as sms, voice service and data connection. Their results show that 3G has more energy consumption especially for data connection and voice service. In [54], they present an energy consumption characteristics of three different networking technologies: 3G, GSM and Wi-Fi and based on the obtained measurements, they put energy consumption model for each networking technology.

The technology in mobile network is evolving and 5G is the latest technology that may overtake wired connection in performance [143]. Energy efficiency is one of the priorities of 5G and the industry has begun to design for energy efficiency. In [62],the authors argue the energy efficient techniques for 5G network and states the challenges ahead. Although there are some studies on energy consumption of 5G network, their survey shows that there are many technical, regulatory, policy, and business challenges still remain to be addressed before 5G is released.

In a study, they developed a system, Senseless, which uses localization techniques but not only limited with Global Positioning System (GPS). They use combination of accelerometer, GPS and 802.11 access point for localization services and they increased the battery life from 9 hours up to 22 hours [57]. Caching is also used for improving the performance of mobile environment. In [124], they propose a cooperative web caching system for ad-hoc networks, which enables mobile terminals to share web pages to decrease the energy consumption.

To sum up, mobile devices consume more energy when they are connected to a network. Furthermore, they consume different amount of energy under different networking topologies. Although

the energy consumption of the network is much higher than energy consumption of mobile device itself, the energy consumption of mobile devices cannot be negligible when the battery of the mobile devices is considered. Mobile devices uses network to connect Internet and download the web pages. Thus, the energy consumption of the network is important to reduce the energy consumption of web pages. However, it is not enough to focus on only network level to reduce the energy consumption of web sites.

2.2.3 Software Level

Software is another field which is related with the energy consumption and there are different studies that handle the software in order to reduce the energy consumption. These studies handle the effect of the operating system and the programming language [115, 112], refactoring [120, 117], code obfuscation [60, 123] and the effect of the networking protocol [37]. There are also some studies related with the energy consumption of web sites [134] and the effect of different JavaScript libraries on energy consumption [103]. Energy efficiency is also worked by the browsers [63, 90, 140, 141, 59].

Operating systems have an impact on the energy consumption [115]. In [115], they compared the energy consumption of two different operating system: Android and Angstrom Linux. The main difference between two operating systems is that Android only uses Java and Angstrom Linux uses both Java and native C. They compared these operating systems by two different sorting algorithms. The result shows that Android provides better virtual machine designs but consumes more energy. In another study [112], they compared Java, JavaScript and C++ in terms of energy consumption and performance in Android applications. Their results show that there is no winner, each one has its advantages in different scenarios. Apart from operating systems and the programming languages, energy consumption of a mobile application may differ when it is from different domains while providing similar services [142].

Code refactoring is the process of restructuring existing computer code without changing its external behavior. It can be used for improving the performance. In [120], they applied good

programming practices and code refactoring to reduce battery consumption of scientific mobile applications on Android devices. The benchmarks in the study includes array copying, matrix traversal, string handling, arithmetic operations, exception handling, object creation and primitive data types and applying code refactoring to these benchmarks, they reduced the energy usage between 2% and 99%. Their results are same on different devices that runs Android operating system. On the other hand, refactoring may increase the energy consumption. [117] argues that god class refactoring has harmful effect on power consumption. Refactoring god classes derives excessive message traffic and this increases a system's power consumption.

Code obfuscation is the most common approach to prevent piracy by deliberation act of creating source or machine code to make it difficult for humans to understand. It is the most common approach to prevent piracy and suggested by both Microsoft and Google. However, it has an impact on energy consumption of the applications. In [60], ten Android applications were analyzed with the impact of code level obfuscation executing number of scenarios. According to the results of this study, code obfuscation may increase the energy consumption by %15 and performance by %20. A similar study was also conducted by Sahin et al. and in their study, they conducted effects of 18 code obfuscations executing 21 use scenarios on 11 different Android applications on four different mobile phone platforms [123]. Their study admits that obfuscation can both increase and decrease the energy consumption. However, it is more likely to increase energy consumption.

The networking protocol has an impact on energy consumption and loading time of a web site. SPDY is an experimental protocol developed by Google as a part of the "Let's make the web faster" initiative. Although the main aim of this protocol is not energy efficiency, it is a step for energy saving with reducing the latency. The comparison of HTTP and SPDY shows that SPDY achieves 64% reduction in page load duration. SPDY is an application layer protocol and it is using Transmission Control Protocol (TCP) as transport layer. The only change is in the user agent and web server applications and they do not modify the network infrastructure. This protocol compresses the request header and this makes the size of the content less than regular. Moreover, it allows multiple requests on a TCP session. Another feature of the protocol is that it

allows bi-directional streams which means that server can send the data without a request from the client. If there is a bottleneck, the solution is that client can block the request or prioritize it [37].

Analyzing the energy consumption and web site optimization is another aspect under software level. There are some studies about energy consumption of web sites and [134] is one of them. In this study, they measured the energy consumption of top web pages. They show the consumption of the elements such as JavaScript, CSS and images. According to their measurements, energy consumption for 3G setup connection is 12 Joules. JavaScript and CSS consume very high amount of energy in some of the web pages. To illustrate, in Apple web page, the energy consumption of downloading and rendering CSS is 12 Joules while the total amount is 46 Joules. Mobile version of Wikipedia consumes 36 Joules of energy to download and render the page and the energy consumption of JavaScript is around 10 Joules. They analyzed these pages and they replaced the 5 CSS files with one file and they achieved 5 Joules drop in energy consumption, which is about 40% of the total energy consumption of web page rendering for Apple web page. Moreover, they obtained 9.5 Joules of energy reduction in mobile version of Wikipedia by replacing the JavaScript file. Thus, the energy consumption of JavaScript and CSS is high and they can be reduced applying some techniques.

JavaScript is very crucial for any web site. However, the energy consumption of JavaScript is high [134]. There are different JavaScript libraries and they can be used to do same task. In [103], they analyzed the energy consumption of JavaScript based mobile web applications. In their study, they developed the same application using different JavaScript libraries and according to their result there are no significant differences between the eight implementations which use different JavaScript libraries.

Web sites consume more energy as they get richer in content and become more dynamic using more JavaScript [134, 91]. Although there are some studies and guidelines about energy consumption of web pages, it is not enough to work on only content of the web sites since there is an extra layer to render the web pages, which is browser. In the market, there are various browsers and they are in competition with each other. Energy saving is one of the issues in

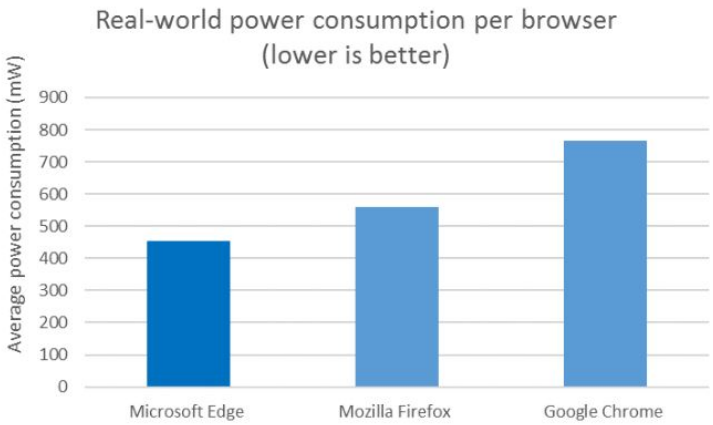

Figure 2.2: Real World Average Power Consumption in Windows 10 (version 1607) [141]

this competition. Most of the browsers provide energy saving with plug-ins, add-ons and different modes (see Table 2.2). The main techniques behind these energy saving tools are blocking Flash contents, reducing the payload by not transferring images, reducing the performance or compressing over the cloud [52, 76].

Browsers provide users to choose energy saving mode or plug-ins. This feature is offered by most of the browsers with different techniques. Browser companies are doing tests to measure the energy efficiency of each other in every update. These tests are done by Chrome[63], Opera[90], Firefox and Microsoft (on Microsoft Edge)[140, 140]. According the latest tests performed by Microsoft, Edge is the most power efficient browser (see Figure 2.2). The results of these tests change rapidly for the newer test done by different companies.

Apart from the sector, there are some scientific studies that aim to improve the energy efficiency of the browsers. [59] worked on to reduce the energy consumption to load web pages on smartphones . They derived general principles for energy efficient web page loading and they presented three effective techniques. Their two techniques, network-aware resource processing and adaptive content paint aims to address energy inefficiency issues of the current mobile web browsers in its content processing and graphic processing pipelines. Application assisted scheduling is the third technique that balances the trade-off between the energy saving and the

QoS. They implemented the proposed techniques on Firefox and Chromium browsers evaluating on real world web sites and using smartphones. Their results show that average 24.4% system energy saving on Chromium is achieved.

2.2.4 Summary

In summary, energy efficiency is an important issue in hardware, network and software fields. In hardware field, using renewable energy is trending. Processors are always designed to be more energy efficient. There are some studies that analyze the impact of the CPU in energy consumption of web pages. There are also some studies that focus on energy consumption of different parts such as display. For the network, there are some studies that analyze the energy consumption of mobile devices under different network technologies. It is mentioned that, data centers and mobile networks consume high amount of energy. There are different approaches to reduce the energy in mobile networks. In software field, there are studies that compare different operating systems and different programming languages in terms of energy efficiency. Some studies attempt to reduce the energy consumption with modifying code execution. However, the studies about the web is very limited. Google developed an experimental protocol that aims to reduce the latency. Moreover, it is an competition among browser companies to be the most energy efficient browser. Most of the browsers offer different plug-ins or modes for energy saving. Furthermore, they improve their energy efficiency in order to compete with each other. However, there are not enough improvements in software field for web.

The aim of this thesis is to reduce the energy consumption of web sites via script and stylesheet based transcoding. The web stands between software, hardware and network. The mobile device has to be connected a mobile network in order to send HTTP request and download the content of the web site. While rendering the web page, processor and other components of the device, such as screen, is used. JavaScript is the key technology of dynamic web pages and it is executed in the CPU and it has high energy consumption on the processor. Thus, these three levels (software, hardware and network) are important for the energy consumption of web

pages and this is why some important studies from these areas are mentioned. The next section explains what is transcoding and adaptation, including three different types of transcoding and gives some transcoding examples from the literature.

Table 2.2: Energy Saving in Browsers

Device Category	Browser Name	Energy	Plug-in	Mode	Ref.
Desktop	Internet Explorer	Power Plan of Windows OS		✓	[102]
Desktop	Opera	Battery Saver		✓	[113]
Desktop	Google Chrome	Power Saver	✓		[66]
Desktop	Apple Safari	Safari Power Saver	✓		[99]
Desktop	Mozilla Firefox	Power Saver		✓	[75]
Mobile	Google Chrome	Power Saver	✓		[66]
Mobile	Mozilla Firefox Mobile	Power Saver		✓	[75]
Mobile	Opera Mini	Data Saving	✓		[114]

2.3 Transcoding and Adaptation

Transcoding is the process of transforming web pages into alternative forms in order to have improvements for different purposes. To illustrate, it can be applied to provide web accessibility for disabled people, increase the performance or improve the usability of a mobile device. Transcoding can be done to improve the energy efficiency or loading time of the web page. In the process of transcoding, when client requests a web page from the web server, first it passes through the transcoding server. In the transcoding server, transcoding techniques are applied to the content of the web page and then the modified content is sent to the client from the transcoding server. This modification can be changed according to type of the device or network bandwidth [135]. There are three types of transcoding which are client-side, server-side and proxy-side [98]. The difference between the types is the location of the transcoding server.

In client-side transcoding, modification is done in the device. After client receives the content, transcoding is applied on the device [98]. In other words, browser is responsible for the transcoding process [53]. The main purpose of client-side transcoding is to provide more suitable content considering the limitations of the device. In this type, since transcoding is done according to user preferences, modified version of the web page fits better. However, since the transcoding is done on by device itself, the device should be powerful. If the limitations of mobile devices considered, it can be said that client-side transcoding is not feasible in terms of saving energy [148].

Server-side transcoding is the other type of transcoding. The transformation is done on the server side and client receives the modified content from the server. This process is invisible to the client since all the process is done at the server-side [53]. The first advantage of the server-side transcoding is that there is no need for any change on the client-side. Another advantage is that it sends minimum amount of data so there is no need for high bandwidth. However, the web servers are private servers and there might be servers that do not support transcoding [148].

The third type of transcoding is proxy-side transcoding. Proxy is a specialized server that sits

Figure 2.3: The Architecture of Proxy-Side Transcoding

between the client and the web server (see Figure 2.3). The transcoding process is performed in the proxy, not in the client-side or server-side [98]. It is transparent to the user after the address of transcoding server is set by the client. When client requests a web page, it sends the request to the proxy. Then, proxy sends a request to the server and the content is sent to the proxy. After transcoding, the web page content is sent to the client [148]. Proxy-side transcoding does not put extra work on client or server side but the connection speed between the client, proxy and server is the drawback, as the proxy is added as an extra component to the network architecture. [148].

There are also examples of transcoding which is a combination of server-side and client-transcoding. Video messaging service is an example for this combination [95]. With the client-side transcoding upload traffic can be reduced and on the server-side low quality video can be delivered to the recipient. Thus, network traffic generated from upstream and downstream transmission can be reduced. This allows client to have different choices to record and transmit the video. To illustrate, client may record high quality video and send the low quality version. The advantage of this combination is reducing the transmitted data in the network and it provides less network cost, battery usage and latency.

There are different purposes of transcoding. In Table 2.3, 11 different transcoding techniques are given with different implementation examples [91]. The location of the transcoding (server, proxy or client) is also given in the table alongside some features of the implementation such as modifying the look& feel of the page, adaptation for people with special needs, screen size adaptation and battery life improvement.

Table 2.3: Transcoding Methods [91] [P/wSN: People with Special Needs, *: Reverse-Proxy , x: No, ?: No Information]

Technique	Implementation	Reference	Location			Modify the Look & Feel	Adaptation for P/wSN	Screen Size Adaptation	Battery Life Improvement
			Server	Proxy	Client				
Text Magnification	IGoogle, Browser	[53, 17]	✓	x	✓	✓	✓	✓	x
Color Scheme Changes	Color Enhancer, High Contrast, PAN	[53, 85, 8]	x	✓	✓	✓	✓	x	x
Alternative Text Insertion	Image Alt Text Viewer, PAN	[53, 85, 8]	x	✓	✓	✓	✓	✓	✓
Page Reaarrangement	BETSIE	[98, 53, 132, 128]	✓	✓	x	✓	✓	✓	x
Simplification	FOM, [149]	[53, 64, 149]	✓	✓	x	✓	x	✓	✓
Summarization	SSD Browser, [93]	[53, 98, 51, 93, 94]	x	✓	✓	✓	x	✓	?
Image Consolidation	Google PageSpeed	[68, 71, 38, 73]	✓	✓*	x	x	x	x	?
Responsive Image	Apple Retina, Mobify, Google PageSpeed, HTML5 & CSS	[79, 71, 48, 100, 38, 26]	✓	x	✓	x	x	✓	x
Data Compression	Google Data Saver,Research by using Squid Proxy	[55, 71, 13, 65, 8]	✓	✓	✓	x	x	x	x
Reducing the Number of Redirects	Apache Reverse Proxy and mapping rules	[150, 6]	✓	✓*	x	x	x	x	?
Expiration Header	Content Delivery Network, Proxy-Cache of Corporations	[71, 111]	✓	✓	✓	x	x	x	✓
Dynamic Adaptation	Energy-aware adaptation for mobile applications	[78]	x	x	✓	?	?	?	✓

When we analyze the related work on transcoding methods from the Table 2.3, the number of studies that achieve batter life improvement is very low. Alternative text insertion is used on the client and proxy side but it modifies the look and feel of the page. Simplification is another transcoding method that achieves battery life improvement but again it modifies the look and feel. Expiration header is also validated to achieve battery life improvement and implemented on server, proxy and the client side. Dynamic adaptation is another method to achieve energy saving on the client side that changes the behavior of the application according to behavior of the mobile device [78].

In our approach, the transcoding will be done on the proxy side. The reason is that server side transcoding needs the knowledge of the developer and the techniques should be applied by all the web developers. If the location of the transcoding would be on the client side, it needs the knowledge of the client. Furthermore, it puts extra work on the client side, which is the mobile equipment. As discussed, mobile devices have some limitations and client side transcoding would be harmful for the energy and the performance. Another important criteria is to modify look&feel of the web page. Our approach does not change the look&feel of the page and user can see the original web page.

There are different guidelines for web site developers to design high performance web sites. These guidelines may also reduce the energy consumption of the web sites. However, most of them were not evaluated in terms of the impact on energy saving. The next section covers recommendations from seven different guidelines for performance improvement and energy saving.

2.4 Guidelines for Performance Improvement and Energy Saving

It is a fact that high performance is very crucial for web sites. The performance of a web site affects the speed and energy consumption. As mentioned before, speed is one of the most competitive factor between the websites. Page speed is also used as a ranking factor for Google Mobile First Index [127]. Besides, it is not important that there is an effective artificial intelligence model behind the scenes or a brilliant mathematical model, if the website cannot be

displayed in a given time, they loose visitors [107]. According to a study based on Google and Microsoft engineers' experience New York Times, visitors of the web site visit less often if it is slower than a competitor by more than 250 milliseconds [96].

Another study shows the online consumer behavior when there is a delay in displaying the web page [41]. According to the results, 57% of online customers abandon a site after waiting for a page to load and 80% of these people does not return to the web site and almost half of these consumers go on to tell about their negative experience. Besides, 7% loss occurs in conversions with 1 second delay [144]. It also causes 11% less page view and 16% less customer satisfaction. Slow loading also causes around 1.73 billion pounds loss in global sales, according to online customer data platform [104]. Thus, it is admitted that web pages should reduce the page load time under 3 seconds [42].

There are various guidelines for high performance and faster web sites. Besides, Google and Yahoo also include some suggestions about faster web sites for the developers. Although these guidelines does not include any suggestion about energy efficiency, some of the suggestions may be also applicable for energy saving. There are two main ideas behind reducing the energy consumption of a website: (1) to minimize HTTP requests and (2) to reduce the size of the components. If the suggestions for faster web sites are analyzed, these two ideas can be seen behind most of the suggestions. The table that summarizes these guidelines can be found in Appendix A. These guidelines are listed as follow:

1. Yahoo Developer Network [6]
2. Google Developer [28]
3. High Performance Web Sites [130]
4. Even Faster Web Sites [131]
5. Rules for Mobile Performance Optimization [71]
6. Making the Mobile Web Faster [100]

7. The Evolution of Web Development for Mobile Devices [150]

From the table (see Table A.1) summarizes these guidelines, it can be seen that there are various recommendations about JavaScript and CSS to increase the performance of a web site (*they are specified as italic*). As mentioned before, the energy consumption of JavaScript and CSS is high [134]. Moreover, they have impact on speed of a web site. Although these recommendations are not directly suggested to reduce the energy consumption of the web site, some of them may provide energy saving. However, there are almost no studies that evaluate their impact on energy consumption. The next section explains the CSS and JavaScript related guidelines and gives the implemented tools about these techniques.

2.4.1 CSS&JavaScript Related Guidelines

It is showed that there are lots of recommended techniques on CSS and JavaScript which are core elements of dynamic web sites and their energy consumption is high. In this section, the recommended techniques on CSS and JavaScript are explained in detail. Energy saving in some of these guidelines are shown scientifically and some of them are implemented as tools that work on different sides such as client, proxy or server. However, might be relevant but not shown scientifically. Table 2.4 summarizes the CSS and JavaScript related guidelines and its related work and implementations. However, most of these techniques have not been evaluated in terms of energy saving or performance. Some techniques have some implementations so that they can be used while developing a web site. The recommended techniques on CSS and JavaScript are as follows:

Table 2.4: Techniques on CSS & JavaScript [Yes: shown scientifically , No: irrelevant, ?: might be relevant but not shown scientifically]

Technique	Implementation	Ref.	Location			Modify the Look & Feel	Battery Life Improvement
			Server	Proxy	Client		
Avoid CSS @imports	Google PageSpeed Module	[16]	✓			?	?
Avoid document.write	Google Lighthouse Tool	[4]	✓			?	?
Combine images with CSS sprites	Google PageSpeed Module	[38]	✓			?	?
	Twes+	[91]		✓		No	Yes
Combine external CSS and Javascript	Google Apache Module	[150]	✓			?	?
	Google Closure Compiler	[5]	✓			?	?
	Google PageSpeed Module	[9]	✓			?	?
	Mobify Jazzcat	[26]	✓		✓	?	?
	IBM Worklight Studio	[11]	✓			?	?
	Grunt contrib-concat	[82]	✓			?	?
Minify JavaScript and CSS	YUICompressor	[43]	✓			?	?
	JSCompress	[27]	✓			?	?
	JSMin	[67]	✓			?	?
	Minify	[106]	✓			?	?
	CSSNano	[29]	✓			?	?
	csso	[49]	✓			?	?
	Microsoft Ajax Minifier	[1]	✓			?	?
Minification and concatenation	JS&CSS Script Optimizer	[92]	✓			?	?
	Merge+Minify+Refresh	[87]	✓			?	?
	Dependency Minification	[145]	✓			?	?
	Minqueue	[83]	✓			?	?
	Combine and Minify	[10]	✓			?	?
	Granule	[139]	✓			?	?
	Jawr	[25]	✓			?	?
	The Asset Pipeline	[3]	✓			?	?
	Codekit	[7]	✓			?	?
	Prepros	[34]	✓			?	?
Remove unused CSS	Firefox Dust-me selectors extension	[14]	✓			?	?
	Chrome Developer Tools	[56]	✓			?	?
	Gtmetrix	[31]	✓			?	?
	unused-css.com	[35]	✓			?	?
	mincss	[58]	✓			?	?
	uncss	[80]	✓			?	?
	CSS remove and combine	[101]	✓			?	?
	Who killed my battery?	[134]	✓			?	Yes
Put scripts at bottom	JS&CSS Script Optimizer	[92]	✓			?	?
Remove duplicate scripts	uniq.js	[133]	✓			?	?
Write efficient JavaScript	Who killed my battery?	[134]	✓			?	Yes

Avoid CSS @import

@import is used for importing the rules of a CSS file into another CSS file. However, since it leads to extra requests, Google and Yahoo suggest to eliminate @import [6, 28]. Moreover, @import behaves like using <link> at the bottom of the page in Internet Explorer. Since it is suggested to use CSS at the top for progressive rendering, the best way is not to use @import [6].

Google has PageSpeed Module that supports flattening CSS imports. This module works on the server side and it first finds linked and inlined CSS. Then, all @import rules are flattened and replaced with the contents of the imported file. It repeats the process for each imported file recursively. However, this module only rewrites the CSS referenced by <style> and <link> and rewriting style attributes are not supported. Another limitation of the module is that not all proprietary constructs or CSS3 are found. If there is an unhandled syntax, the module lefts the CSS file unflattened. The size of the flattened CSS is also important for the module since if the size of the file is more than 2048 bytes, the flattening process will be canceled because the performance can be worse than when not flattened[16].

Avoid CSS expressions

CSS properties can be set dynamically by CSS expressions. However, they are dangerous as they can negatively affect the performance. To illustrate, CSS expressions is used to set the background color to alternate in different time periods. The following code is an example for CSS expression:

background-color:expression((newDate().getHours()%2 ? "#B8D4FF":"#F08A00");

In the given code, JavaScript is used for evaluating the expression. However, this expression is evaluated more than expected. According to Yahoo Developer Network [6], more than 10000 evaluations are generated as a result of moving the mouse around the page. Thus, developers should be aware using CSS expressions. Moreover, if the JavaScript code is not efficient enough, page will be loaded more slowly [130]. Yahoo suggests using one-time expressions to decrease

the number of evaluations and increase the performance. Yahoo also suggests to use event handlers to replace CSS expressions [6].

Avoid document.write

document.write() is a way to inject external scripts. However, the display of main page content can be delayed tens of seconds under slow connections. Google [28] suggests not to use document.write(). Google Ligthouse tool, which reports every instance of document.write(), can be used to eliminate these functions [4]. However, this tool only finds all of the document.write() functions and they should be replaced by the developer.

Combine images with CSS sprites

CSS sprites can be used for reducing the number of images need to be downloaded by consolidating the images. Multiple images are combined in one large image using CSS sprites and it reduces the number of requests to only one and improves the performance significantly [71]. When multiple images are combined into one large image, the web page fetches it at once and then individual images are displayed using CSS background positioning, as needed on the page. Yahoo [6] have some suggestions to optimize the CSS sprites. Firstly, the images should be arranged horizontally since it results in a smaller file size compared to vertically. Secondly, similar colors should be combined in one sprite in order to have lower color count. Lastly, there shouldn't be big gaps between the images. Although it does not affect the size of the file, big gaps cause more memory in order to decompress the image into a pixel map. To illustrate, 100*100 image includes 10 thousand pixels whereas 1000*1000 image consists of 1 million pixels.

Google PageSpeed Module Sprite Images Module is one way to combine images with CSS sprites. This filter detects PNG and GIF images used as backgrounds in CSS. Then, it combines these images into single large image with background positioning declarations in order to make the page appears as it was before. However, it does not support JPG images and works with only CSS backgrounds, not <img> tags [38].

Twes+ is another implementation of consolidating images with CSS sprites. It works on proxy

side and reduces the energy consumption of web pages. It does not put extra load on the client side and server side. Furthermore, it does not modify the look&feel of the page. One of the limitation of Twes+ is that it does not support the images inside the CSS and the background images inside the HTML. Another limitation is that it does not support GIF animations [91].

Defer parsing of JavaScript

The idea behind defer parsing of JavaScript is same with the idea of putting scripts at bottom. Most of the scripts are not needed before the page is rendered [71]. To illustrate, the user does not use the scripts that supports interactive user behavior, such as drag and drop, until the page is loaded. This is also same with script execution, they are not needed to execute before the page loaded. Especially, poorly optimized scripts from third parties, such as advertisements, social media wizards and analytics supports, should be deferred [71]. Apart from these third party scripts, the scripts of the web pages should be deferred to parse [71, 6, 28, 131]. It is suggested to defer as much as scripts until ”onload”. According to Yahoo [6], if a script can be deferred, it should be moved to the end of the page so that page can be downloaded faster.

Develop Smart Event Handlers

An event handler is a callback routine that operates asynchronously and handles inputs received into a program (events). However, too many event handler cause less responsive web pages. Event delegation, suggested by Yahoo [6], is an approach to overcome this problem. This approach suggests to attach only one event handler to the div wrapper instead of one handler for each button and it can be figured out which button it is originated from at the code level.

Keep components under 25K

This is a recommendation from Yahoo which is related with Iphone caching features. Iphone cache components cannot be more than 25K which is the uncompressed size. Thus, minification is suggested alongside gzip the components [6].

Put scripts at the bottom

According to HTTP/1.1 Specification [21], browsers should not download more than two components in parallel per host name. When a script is being downloaded, it blocks parallel downloads. To illustrate, more than two images can be downloaded at the same time if they are served from multiple host names. However, is a script is downloading, it is not possible to download any other element even if it is served from different host name. Thus, scripts should be moved to the bottom of the page in order not to block downloading other elements of the page [130, 6]. Wordpress has a plug-in JS&CSS Script Optimizer which put scripts at bottom alongside minification and concatenation of external files [92].

Put stylesheets at the top

Stylesheets are responsible from formatting the elements in the page. If they are included at the bottom, browser will not have an information about how to locate the elements. There are different approaches from different browsers. to illustrate, Internet Explorer waits until all stylesheets are downloaded to render the page. Firefox first renders the page elements and then change it if the stylesheet changes the formatting. However, this may be disruptive for the user [130]. On the other side, in Internet Explorer, user sees the white page until the stylesheets are downloaded [6]. Moreover, putting stylesheets at the top loads the page faster, according to the research of Yahoo [6]. When the stylesheets are in the HEAD of the page, page is rendered progressively. The HTML specification also states that stylesheets should be included in the HEAD of the page [45].

Remove duplicate scripts

In a web page, duplicate scripts tends to be seen more likely as the number of developers in the development team increases[130, 71, 6]. According to a review of top 10 U.S. web sites, two of them contain a duplicate script [6]. Apart from the team size, number of scripts is an important factor for duplicate scripts. Duplicate scripts cause wasted JavaScript execution and also unnecessary HTTP requests and this causes to slow loading. Some specific JavaScript libraries (such as unique).js can be used to eliminate duplicate scripts on the server side but you

need to be owner of the web site [133].

Remove unused CSS

Stylesheets are required to be downloaded and parsed before rendering the page. Rendering the page is blocked until the browser loads the stylesheets before the page even if the stylesheet is an external file and cached. Moreover, CSS engine evaluates every rule in the file in every stylesheet is loaded. Many web sites often use same external CSS file for all of their pages although all of the rules inside the file does not be needed on the current page. Thus, many web sites have unused CSS. As a solution, these unused stylesheets recommended to be removed [28]. In the study conducted by Thiagarajan et al., they reduced the energy consumption of Apple web page by removing unused CSS. They have obtained 5 Joules reduction in energy consumption which is 40% of the energy consumed by CSS [134].

There are different tools to eliminate unused CSS. Dust-me selectors extension is a Firefox extension and CSS remove and combine works as Google Chrome extension [14, 101]. Google Chrome Devtool also have unused CSS feature [56]. There are also online tools such as GTmetrix [31] and unused-css.com [35]. mincss and uncss are tools that removes unused CSS from the stylesheet files [58, 80]. They can potentially be relevant for energy saving, but these tools works on the server side and you need to be owner of the web site to use these tools.

Replace click events with touch events

Everts recommends to replace click events with touch events on touchscreen devices [71]. When user taps the screen, onclick event waits around 300 milliseconds to allow user start another movement apart from a click. However, it reduces the responsive performance that that a user expect from the website. Instead of using onclick event, touchend event can be used with using touchstart end touchmove events to ensure unexpected behavior experienced by the user.

Simplify pages with HTML5 and CSS 3.0

A simpler page means smaller size and faster load and simple DOM (Document Object Model) means faster JavaScript execution. HTML5 specification contains new structural elements yields

a simpler and more efficiently parsed page. CSS 3.0 has also new features which support to speed up page rendering. These features can create smaller size web pages with rounded borders, animation, shadows, transitions and other graphical effects to replace images [71].

Simplify and use efficient CSS selectors

Inefficient CSS selectors have cost on a web site and there are different rules about the selectors. It is important that developers should avoid a few common, yet costly, CSS selector patterns [131]. The key to efficient CSS selectors is to have some control over how long it takes the browser to match the selectors against the elements in the document.

Use CSS animations and CSS transitions instead of JavaScript

Zakas [150] recommends to use CSS animations or CSS transitions to avoid using JavaScript. CSS animations and CSS transitions are much more efficient way for the devices since JavaScript based animations creates the appearance of animation by running more code at frequent periods. Besides, CSS animations and CSS transitions create these effects in more optimal way as they bypass the CPU.

Use CSS instead of images

It is well known that the energy consumption of images in a web page is high [150]. According to the study, the average size of images per a web page is 793 KB. Thus, eliminating the number of images is recommended [150, 100]. CSS3 provides number of properties to create images such as buttons with less size. To illustrate, a button can be created by 4 lines of CSS code and it may replace multiple images. Thus, fewer bytes is needed to create the button with CSS properties instead of including an image [150].

Write efficient JavaScript

The speed of JavaScript execution is dependent on different variables about how the code is written. Since the efficiency of the code is related with the CPU it has also effects on energy consumption since the execution consumes energy on the client side. In [131] there is a chapter

about writing efficient JavaScript and they recommend different techniques for writing efficient JavaScript. The first way to speed up JavaScript execution is about managing the scope. They suggest to avoid including constructs that artificially augment the scope chain as out of scope variables take longer access than local variables. If there are more than one out of scope value, the recommendation is to store it in a local variable in order to minimize the longer access. The performance of a script is also related with the way of the data stored and accessed. Local variables and literal values are the fastest ones and they can be used to improve the performance of accessing array items and object properties so it is recommended that if any array item or object property are used more than one time, they should be stored in a local variable. Flow control is another important determinant. According to Souders, ”if” statements should be used with range of values or small number of discrete variables. ”Switch” statements should be used for between 3 and 10 discrete values and array lookup should be used for a larger number of discrete values. Loops also have an important impact on execution of JavaScript and they suggest reversing the order where the items are processed so that the control condition compares the iterator to zero since this way is faster than comparing a value to a nonzero number. Thus, array processing will be faster. ”HTMLCollection” objects are another important factors of the performance of JavaScript execution since query of the DOM is needed for matching nodes when a property accessed on one of the objects. Thus, ”HTMLCollection” properties should be used only when necessary and the frequently used values should be used in local variables. Performance is also related with operations on strings such as string concatenation. Furthermore, it is more slower in Internet Explorer compare to the other browsers and this can be avoided by calling join() in order to merge the strings. Trimming is also important for strings, especially the algorithm used depending on the size of the string. In their study, Thiagarajan et al. reduced the energy consumption of by 5.5 joules by redesigning the page with a more efficient JavaScript [134].

Make JavaScript and CSS external

JavaScript and CSS can be added to a page as external or inline. However, they have some differences in terms of performance issues. If they are included as inline in an HTML document, the

first disadvantage is that it makes the project difficult to read and develop [28]. Separating CSS and JavaScript into different files provides better readability. Apart from this, making JavaScript and CSS external will provide better performance based on the suggestions in the guidelines [6, 130, 131, 28]. Most of the times HTML documents are not cached since their content changes constantly. However, CSS and JavaScript are less dynamic compared to HTML, their content often do not change for weeks or month [130]. If these are inlined in an HTML document, there will be unnecessary bytes downloaded from the server in every change in HTML. External files will increase the HTTP requests, however, external JavaScript and CSS files can be cached in order not to download the same rules every time. Thus, this will reduce the size of the HTML document without increasing the number of HTTP requests [6]. Yahoo suggests to cache external JavaScript and CSS files if the users on a site have multiple page views per a session and many of the pages reuse same scripts and stylesheets. According to the suggestion, this will improve the performance of the response time and also the bandwidth of the data center [130].

On the other hand, there are many web sites that fall into middle of these metrics. The best solution for these web sites is to make JavaScript and CSS external. The only suggestion to include them inlined is on home pages that have few page view per session. Another suggestion for front pages, which are the first of many page views, is to inline JavaScript and CSS in the front page. However, the external files should be downloaded dynamically after the page has finished loading and the other pages reference the external files that are already in the cache [6].

Combine external CSS and JavaScript

Each external resource means an extra HTTP request. It is suggested to make JavaScript and CSS external unless they are used in home pages that have few page view per session. However, if there are numbers of external files, it will result with numbers of extra HTTP requests. As a solution, external JavaScript and CSS files should be concatenated [71, 150, 28]. In the development stage, multiple CSS and JavaScript files makes the process easier since it is better to follow [71]. However, an ideal web page should have one JavaScript file and one CSS file as external [150].

Traditionally, concatenation is done at build time. However, there are some services to make this process at runtime using content delivery network (CDN). Google Apache Module is a concatenation service which requires server support so, you need to be owner. This module concatenates files dynamically at runtime. It uses special URL format to download multiple files using a single request.

http://www.example.com/assets/js/main.js

http://www.example.com/assets/js/utils.js

http://www.example.com/assets/js/lang.js

mod_concat in Apache Module combines these requests into one URL as follow:

http://www.example.com/assets/js??main.js,utils.js,lang.js

This URL concatenates main.js, utils.js, and lang.js into a single response in the order specified. In the combined URL, double question marks indicates to the server that this URL should use the concatenation behavior. [150]

There are various tools to concatenate external CSS and JavaScript. However, most of them works on the server side (see Table 2.4). Thus, only the owners of the web sites can use these tools to optimize with concatenating external files. Also, the primary objective of these tools is to reduce the loading time of the web sites and reduction in energy consumption is not an aspect for these tools.

Google has different tools and modules to concatenate external files such as CSS and JavaScript. Apart from Apache,which concatenates multiple JavaScript files, Google Closure Compiler[5] also combines external JavaScript files into one and PageSpeed Module can combine external CSS files [9].

Mobify Jazzcat is a commercial software that concatenates CSS and JavaScript files, one external file for CSS and one for JavaScript. They state that this technique speeds up the web sites. However, the drawback of this technique becomes when there is an error in one of the scripts.

JavaScript files are not concatenated, the website only will block the script with error and skips to next script. On the other hand, the rest of the file will be failed to execute, if they are concatenated [26].

IBM Worklight Studio [11] (works with Eclipse) is an environment to develop mobile applications. It has also support to concatenate external script and stylesheets files for desktop browser and mobile web environments. Concatenation is controlled by a number of different parameters, such as the structure of the HTML, the type of the resources to be concatenated, and the attributes of these resources. The order of the resources in the HTML is preserved. As a result, the concatenation process does not have any negative effects in terms of code dependencies or functionality.

Minify CSS and JavaScript

Minification refers to removing unnecessary characters, such as white spaces and comments, from the code in order to reduce the size of a file. Therefore, it will improve the load times. According to a survey, in ten top U.S. web sites, 21%size reduction was achieved by minification [6]. According to Souders [130], minification achieves 20% size reduction for JavaScript, typically. Apart from reducing the bandwidth consumption and latency, minification makes difference for cacheable object by reducing the size of the object to be cached [71]. Gzip compression does not help in this regard since objects are cached after decompressed.

There are different tools to minify JavaScript and CSS. The YUI Compressor [43] is a JavaScript compressor which, in addition to removing comments and white-spaces, obfuscates local variables using the smallest possible variable name. It is also able to safely compress CSS files. Apart from using in server side, it has also online tool. JSCompress [27] is an online JavaScript compressor that allows you to compress and minify all of your JS files by up to 80% of their original size. JSMin [67] is another JavaScript minifier which works in server side. Minify is also an online JavaScript and CSS minifier [106].

Most of the minification tools also concatenate external files alongside minification. JS&CSS Script Optimizer [92] is an open source WordPress plug-in that groups several scripts into single

file and combine several CSS files into single files and minify CSS. Merge+Minify+Refresh [87] is another WordPress plug-in that merges CSS and JavaScript files into groups and then minifies the generated files using Minify for CSS and Google Closure for JavaScript. Dependency Minification [145] and Minqueue [83] are also Wordpress plug-ins to concatenate and minify external scripts and stylesheets.

The tools for minification and combining external files are not limited with the mentioned ones. CSSNano [29] and csso [49] minify CSS. Moreover, they are recommended by Google Pagespeed. CombineAndMinify [10] is a package that combines CSS and JavaScript files and minifies them. Moreover, it uses cookieless domains and implements far future caching and inserts version numbers in urls to ensure browser always loads the latest versions. Microsoft AJAX Minifier [1] is CSS and JavaScript minification library to use in .NET applications. Granule [139] is an optimization solution for Java-based web applications that combines and compresses JavaScript and CSS files. Jawr [25] is another library that combines, compresses and minifies JavaScript and CSS files.

When we look at different frameworks, they also have libraries to concatenate and minify JavaScript and CSS files. Furthermore, Ruby on Rails has the Asset Pipeline [3] that concatenates and minifies or compress a JavaScript and CSS assets. Moreover, Django has a Python script [146] for automated merge and minify of JS and CSS files. There are also different task runners and they have plug-ins to optimize JavaScript and CSS. For example, Grunt [19] is a JavaScript task runner and it has a plug-in [82] to concatenate files. Apart from the frameworks, same features can be found in different standalone desktop applications. To illustrate, Codekit [7] and Prepros [34] can minify and concatenate JavaScript and CSS files.

2.4.2 Lessons Learnt

In this section, we analyzed guidelines for performance improvement and energy saving in detail. Recommendations from 7 different guidelines, both from the sector and scientific papers, are included. When we analyze the Table A.1, it can be seen that there are various recommendations

on JavaScript and CSS. Then, JavaScript and CSS related guidelines are explained. There are some implemented tools on these guidelines and there are some scientific studies that analyzes the impact of these recommendations. The related tools may work on different sides (server, client or proxy). Most of them are online tools but again it means that you need to be the owner as you have to work on server side. Besides, almost none of them are evaluated in terms of energy efficiency.

In [89], the authors analyzed the impact of performance enhancement techniques on web applications. They applied six different techniques which are caching objects using HTTP expires, reducing number of HTTP requests, caching of interpreted PHP scripts, caching query results, web server configuration tuning and caching pages using a proxy server. In the experiments, they used three open source web applications in different categories such as Joomla (portal), PhpBB(community) and OsCommerce(e-commerce). According to their effectiveness calculation, the tested guidelines have different ranking on the applications, positively in general. However, they did not evaluate the guidelines in terms of energy saving.

In a study conducted by Bunse and Rohde [61], the impact of six common guidelines suggested by Google are analyzed which avoid unnecessary objects, static methods should be preferred, use static final for constants, avoid internal properties, use enhanced for-loop syntax and avoid float. They tested these guidelines with different applications in iOS and Android. The results show that the impact of the guidelines changes with operating system. In overall, systematically applying the guidelines can reduce the energy consumption by 9% and performance by %1 in average.

Twes+ is an example for evaluating the impact of the guidelines in terms of energy saving but did not evaluate JavaScript and CSS related techniques [91]. It is a transcoding system based on two main services, redirect transcoding and image transcoding. In redirect transcoding, they attempted to save energy by reducing the number of redirects and in image transcoding, their technique was image consolidation. Their results show that redirect service can success 4.6% cumulative processor energy reduction and image transcoding can success 7% reduction in cumulative processor energy, which is equal to between a 40 to 60 minutes saving in a battery

of a mobile device. The transcoding occurs in te proxy so that it does not put extra load on the client or server side.

The recommendations on CSS and JavaScript can be categorized into different groups: avoid certain constructs, positions of the constructs, removal, minification and the number of external files. However, the aim of these guidelines are not energy saving directly but some of them may achieve energy saving. The main idea of reducing the energy consumption of web pages is to reduce the size of the files and reduce the number of HTTP requests. As these recommendations are not directly related with energy saving and our aim is to reveal that some of these guidelines may achieve energy saving we need to find the techniques that reduces the size of the components and number of HTTP requests. For that purpose, two recommendations were chosen: (1) minification to reduce the size of the components and (2) concatenation of external files to reduce the number of HTTP requests.

2.5 Summary

To sum up, in this chapter we have started with a background section in which we gave the concept of HTTP request and different ways to access web from mobile devices. Although the HTTP request is same for mobile and desktop, there are different ways to access Internet from mobile, such as native applications, mobile web applications, hybrid applications, responsive sites and widgets. In the next section, different studies from literature that are related with energy are mentioned in three fields: hardware, network and software. Following the energy section, transcoding and its types are explained. Transcoding can be done on server side, client side or proxy side. Transcoding can be used for different purposes such as improving the performance or improving the usability based on different devices. A table that summarizes transcoding techniques and their implementations is also included in the section. In the next section, recommendations for web site developers from seven different guidelines are summarized. This section shows that there are lots of techniques on JavaScript and CSS. These techniques are explained and their implementations are given in the last section.

Energy efficiency of web sites is important when we consider the limitations of mobile devices, especially the battery size. Furthermore, web access from mobile devices is increasing. There are different studies that handle energy consumption in hardware, software and network level. Web sits between these three levels. There are different guidelines for web developers to design efficient web sites. Although these guidelines are not designed to improve energy efficiency, the main idea behind increasing the performance and energy efficiency may be the same: decreasing the number of HTTP requests and reducing the size of components. However, most of the developers are not aware of these guidelines. Indeed, some of the best web sites include some performance pitfalls. Thus, the transcoding should not be in the server side so that it can be applicable for all of the web sites. When the limitations of mobile devices are considered, it can be seen that client side transcoding is not feasible. The number of studies that attempt to reduce the energy consumption of web sites is very low. Twes+ is an example that attempts to reduce energy consumption of web sites based on proxy side transcoding but it did not evaluate JavaScript and CSS related techniques.

The next section describes our proposed approach to reduce the energy consumption over the client. There are several contributions of our proposed solution and these can be listed as follow. In our approach, we focus on the energy saving of web pages as they can be reached from any device, any operating system and any browser. In short, we transcode the web pages to save energy. Transcoding can be done for different purposes to improve the constructed software and our approach is to improve the energy saving of the web sites with different techniques which is novel. We transcode the web pages on a proxy so that there will be no extra load on the server or the client. There are some tools but they work on the server side so you need to be the developer of the web site to use these tools. In our approach, the transcoding is performed on a proxy which is automated. Another important feature of our approach is that it does not modify the look&feel of the page and improves the battery life. These features of the proposed suggestions have not been evaluated before. Thus, our proposed solution is to transcode web sites on proxy server based on two techniques on JavaScript and CSS: (1) concatenating external JavaScript and CSS files and (2) minification to reduce the energy consumption without modifying the look&feel on the web page and without putting extra load on the client or server side.

CHAPTER 3

RESEARCH METHODOLOGY

In this chapter, we describe our novel approach to transcode web pages without modifying their look&feel and without adding extra load to the client or the server side with the goal of energy saving via scripts and stylesheets optimization. Our approach interferes the network while browsing and transcodes the web page on the proxy server, between the client and the server. Two services are implemented on the proxy: (1) concatenating external script and stylesheet files to reduce the number of HTTP requests and (2) minifying script and stylesheet files to reduce the size of the components.

This chapter consists of two sections. First section explains the network architecture of our system and the second section gives the details of the implementation including details about the proxy and the services of our system.

3.1 Architecture

The architecture of our system consists of the following three components:

Server is the component where the source of the web page is located and served from here.

Proxy is responsible for the transcoding and is located between the client and the server.

Client is the component that requests the web page via using the browser.

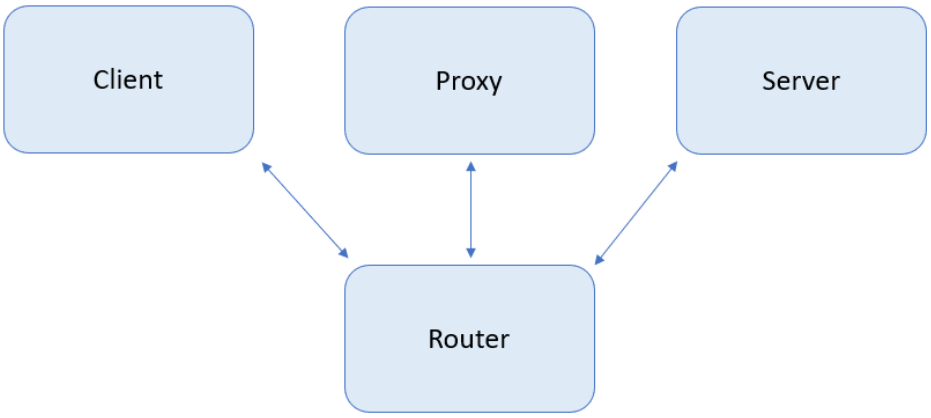

Figure 3.1: The Architecture of Experimental Local Network Approach

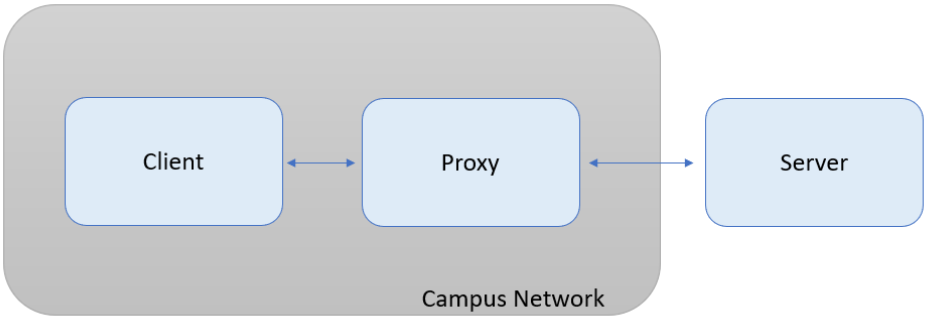

Figure 3.2: The Architecture of Real Network Approach

In the architecture of our system, when a client sends a request, it goes through proxy to the server. The response comes from the server and proxy interferes the network, transcodes the web page and send the response to the client. The example architecture in our campus network is illustrated in Figure 3.2. Web pages are requested from the real IP address of the web page. For this purpose, we are using Middle East Technical University Northern Cyprus Campus network to connect proxy and the client. When a client requests a web page, the request goes through the proxy to the real server and then the content is sent from the server, proxy interferes the network and sends the transcoded content to the client using the campus network. However, local network is used during the evaluation of our services to eliminate the factors that may

Table 3.1: Strength and Weaknesses of the Content Adaptation Mechanisms [12]

Evaluation Criteria	**Mechanisms in rough order from "best" to "worst"**
Squid independence	ICAP, eCAP, ACLs, Client Streams, code hacks
Processing speed	eCAP or Client Streams or ACLs or code hacks, ICAP
Development effort (header adaptation)	ACLs, code hacks, Client Streams, eCAP, ICAP
Development effort (content adaptation)	eCAP, ICAP, Client Streams, code hacks
Versatility	code hacks, eCAP, ICAP, Client Streams, ACLs
Maintenance overheads	ACLs, eCAP, ICAP, Client Streams, code hacks

affect the results of the tests. In this experimental local network architecture, there is a router to connect server, proxy and the client locally. All the web pages are downloaded to the local server and served to the client via our local network (see Figure 3.1).

3.2 Software Architecture and Implementation

This section describes the details of the implementation (see Figure 3.3). We have two different service (consolidation and minification) located on the proxy. Squid Caching Proxy [39] is used as the proxy infrastructure. Squid is an open source caching proxy that is used for the Web and supports HTTP, HTTPS, FTP and some other protocols. Although it's main purpose is caching to reduce the bandwidth and response time, it can be used for different purposes. To illustrate, it can be used for access control. It can run under most of the operating systems. In our implementation, it runs on Linux OS.

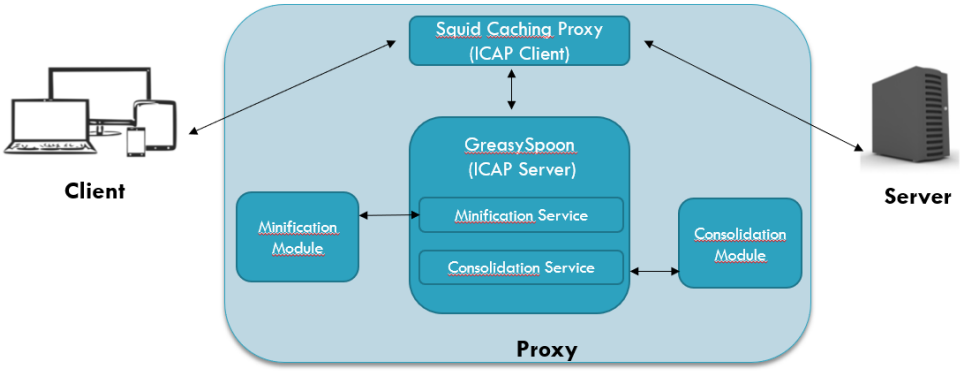

Figure 3.3: The Software Architecture

Table 3.2: Summary of Content Adaptation Mechanisms [12]

Mechanism	**Request**		**Response**	
	Header	**Body**	**Header**	**Body**
ICAP	yes	yes	yes	yes
Client Streams	no	no	yes	yes
eCap	yes	yes	yes	yes
ACLs	yes	no	changes with versions	no
Code Hacks	yes	yes	yes	yes

Squid Caching Proxy also supports different content adaptation mechanisms that may analyze, block, capture or modify the messages. This feature is very important for our implementation as we need to modify the content of the web site. There are different adaptation mechanisms for Squid and they can be listed as Client Streams, eCap, Squid.conf ACLs, Code Hacks and Internet Content Adaptation Protocol (ICAP). Each mechanism has some advantages and disadvantages. Table 3.1 summarizes the strength and weaknesses of each mechanism based on frequently used evaluation criteria. Another important point for these mechanisms is that some of them are limited in their scope. Table 3.2 summarizes what messages and what message parts the mechanisms can adapt.

It is essential that the content adaptation mechanism have to modify both header and body part of the response and this is why we cannot use ACLs for our implementation. Another important criteria is proxy independence as we want to run our project in different systems. When we analyze Table 3.1, it can be seen that ICAP is the most suitable choice for the proxy independence criteria and we chose ICAP as a content adaptation mechanism.

ICAP specifies how an HTTP proxy can outsource content adaptation to an external ICAP server [22]. For this purpose, our proxy works as an ICAP client. ICAP is a protocol that receives HTTP messages from the ICAP client and transforms it on the ICAP server. Moreover, non-HTTP contents can be modified by the ICAP server. When an HTTP message reaches to the ICAP client, it sends the message to the ICAP server and the transformation is done here. An ICAP client may have many ICAP servers and an ICAP server may have many ICAP clients.

As we need to do the transformation over ICAP, we need an ICAP server. There are different

Table 3.3: ICAP Servers and Supported Languages [22]

ICAP Server	**Supported Language**
C-ICAP	C
Traffic Spicer	C++
ICAP-Server	Python
POESIA	Java
GreasySpoon	Java and Javascript

ICAP servers that work with Squid Proxy (see Table 3.3). Each of these ICAP servers support different languages and some of them are not open source (Traffic Spicer and POESIA). Our choice is GreasySpoon because of the language preference. GreasySpoon also supports Ruby and Python with necessary extensions. It is inspired from the GreasyMonkey which is an extension of Firefox and makes content adaptation on the client side. GreasySpoon is transparent for the client which means that it is client platform independent. When a content transferred from client to the server, GreasySpoon modifies it dynamically [18]. To set the connection between the ICAP client and ICAP Server, Squid should be configured [22].

In our system, we use only response modification over GreasySpoon. When a response is received by the Squid Proxy, it is sent to the GreasySpoon and transcoding is done on the fly and redirected to the client. We created two services that work over the ICAP server: (1) consolidation and (2) minification (see Figure 3.3), which are explained in detail below:

3.2.1 Services

In this section, services of our system are explained. As mentioned earlier, CSS and JavaScript are essential components for a web page and their energy consumption is high. In Chapter 2, recommendations for energy efficient and high performance web sites are discussed in detail. Our system is based on two of these recommendations: (1) concatenating external JavaScript and CSS files to reduce the number of requests and (2) minifying JavaScript and CSS to reduce the size of the components. We explain below these services implemented on our ICAP Server:

3.2.1.1 Consolidation Service

A web page may include external JavaScript and CSS files. In Chapter 2, it is mentioned that each external file needs an HTTP request and this cause more energy consumption so guidelines recommend to include one external CSS file and one external JavaScript file for a web page. Our consolidation service follows this recommendation. In summary, it concatenates external JavaScript and CSS files to one JavaScript and one CSS file.

Once the client requests a web page and the HTTP message comes to the proxy, it is redirected to our ICAP server. Here, our consolidation service, implemented using Java, takes the HTTP message and runs consolidation module sending the URL of the requested web page and the HTTP message as parameters. Consolidation module is written using C# and manages the concatenation of the external JavaScript and CSS files. First, it finds all JavaScript and CSS files from the HTML. Then, these files are downloaded to our proxy server using C# HttpClient Class [20]. Each external JavaScript is downloaded and appended to last.js file and each external CSS is downloaded and appended to last.css file on our proxy server. The client needs to request the last.js and last.css files from the proxy instead of the server. For this purpose, Apache Web Server is installed on our proxy as there should be a web server on our proxy so that user may request these files [2].

The next and last step of consolidation module is to parse HTML. This module finds all .js and .css file calls from the HTML and deletes them. Then, it puts an external script or CSS call to request last.js or last.css from the IP address of our proxy. To illustrate, our HTML includes these external script calls:

<script type="text/javascript" src="script1.js" language="javascript">

<script type="text/javascript" src="script2.js" language="javascript">

<script type="text/javascript" src="script3.js" language="javascript">

Consolidation module removes all of the above external script calls and puts one external script

call to request the last.js file from the IP address of the proxy as follow:

<script src="http://194.170.10.2/last.js" >

Finally, user requests one concatenated .js and .css file from the IP address of the proxy instead of the server. If the number of external CSS or script files is less than or equal to one, no transcoding is done by the module.

3.2.1.2 Minification Service

Minification refers to removing white spaces and comments from the code in order to reduce the size of a file. In Chapter 2, it is mentioned that minification may achieve around 20% size reduction and it is recommended by different guidelines. Our minification service follows this recommendation. In summary, minified external JavaScript and CSS files are sent to the client by this module.

This service works very similar to consolidation module. HTTP message reaches to our ICAP server and our minification service runs minification module and sends the URL and the HTTP message as parameters. Similar to consolidation, minification service is written in Java and minification module is written in C#.

Minification module is a script that manages the minification process. It first finds all .js and .css file calls from the HTML and download these external files into our proxy using C# HttpClient. Once any .js or .css file call is found in HTML code, this line is replaced. Instead of the web site server, minified .js or .css file is reached from our proxy server. To illustrate:

<link rel="stylesheet" href="css1.css" type="text/css" />

is replaced with

<link rel="stylesheet" href="http://194.170.10.2/css1.css" type="text/css" />

Minification of the JavaScript and CSS is done by open source tools. YUI Compressor[43] is

used for CSS minification and JSMin[67] is used for JavaScript minification.

YUI Compressor is a command line tool that works on Linux OS. It is written in Java and requires Java to run. It both minifies JavaScript and CSS. It has different options. To illustrate, JavaScript can be minified with obfuscation or without obfuscation. The CSS compression algorithm uses finely tuned regular expressions.

JSMin is one of the most popular JavaScript minifiers. It is a filter that removes unnecessary white spaces and comments from JavaScript files. It works as a command line tool. It requires ASCII or UTF-8 as the character set. It does not obfuscate the code for instance it does not modify quoted strings and regular expression literals.

3.3 Summary

In this chapter, we described the methodology behind our study. We implemented two services on a proxy server: (1) concatenating external JavaScript and CSS files to reduce the number of requests and (2) minifying JavaScript and CSS files to reduce the size of the components with the goal of energy saving. We explained the architecture of our proposed approach and also explained how it is implemented for experimental purposes. Our experimental architecture is based on local network to evaluate our services without the factors that may affect the tests. The real architecture is to be sure that our system works in real world scenario. Squid Caching Proxy is used as the proxy and it acts as an ICAP client. When HTTP message comes to the proxy, it is redirected to ICAP server and modification is done over this ICAP server. Finally, client receives the message from the proxy. We have implemented two different services on ICAP server: consolidation and minification. The important feature of our system is that there is no need for any setup or modification on the client side or on the server side. The only setup is to set the proxy settings of the browser on the client side. The next section investigates the effects of our implementation on energy consumption

CHAPTER 4

EVALUATION

In this chapter, we describe the evaluation of our services in terms of energy saving while browsing. In the previous chapter, the services of our system are described and this section answers the question *"Does our system provides energy saving on the client side?"*. This question is answered comparing the energy consumption while browsing with and without using our system. The tests are performed over a desktop computer and a mobile device.

This chapter is organized as follow: Section 4.1 presents our research questions. In Section 4.2, the test materials for the evaluation are described. Section 4.3 explains the equipments used to implement the architecture and the tools used for the measurements during the experimental tests. Then, test methodology is given in Section 4.4. Section 4.5 gives the results of the evaluation process and in Section 4.6 these results are discussed. In Section 4.7, chapter is summarized.

4.1 Research Questions

In this study, we aim to reduce the energy consumption of the web sites while browsing by concatenating the external JavaScript and CSS files to reduce the number of requests and minifying external JavaScript and CSS to reduce the size of the components. We have asked the following research questions to validate our system:

1. *Does our system allows energy saving on the client side by concatenating external JavaScript and CSS files to reduce the number of HTTP connections?*

2. *Does our system allows energy saving on the client side by minifying external JavaScript and CSS to reduce the size of the components?*

4.2 Test Materials

The test materials to validate energy saving of our system are selected from Alexa Top 100 [1] list on October 8, 2018. However, in order to eliminate the external factors (such as the latency in the network) that may affect our measurements, these web pages are downloaded to the local server to be used in our measurements. Some of the web pages are eliminated in this process. There are several reasons to eliminate these web pages. Firstly, different language versions of the same web pages such as *google.com* and *google.com.tr* are eliminated and the original version of these web pages are kept. Secondly, web pages that include inappropriate content are eliminated. After these two steps, 70 web pages remained from the top 100 list. The next step was to eliminate the web sites that does not work locally and downloaded with errors. As a result of these steps, 35 web pages remained for our evaluation.

The remained 35 web pages were tested using Google Chrome DevTools and checked to be sure there is no modification on local. To do that, missing files that are served from other servers are downloaded and the code is adjusted respectively. All of the errors were fixed. Finally, these 35 web pages were ready for our evaluation.

The web page should include at least two external CSS or JavaScript file to be used in the evaluation of our concatenation service. Eight of these web pages include one or less external JavaScript or CSS files and eight of them were modified after transcoding. Thus, 19 web pages remained for the evaluation of concatenation or concatenation+minification service (see Table 4.1).

[1] https://www.alexa.com/topsites

Table 4.1: Evaluation Set for Our Services (X: Irrelevant)

Rank	Web Site	Number of JavaScript Files	Number of CSS Files	Concatenation	Minification	Concatenation+Minification
3	facebook.com	1	10	✓	✓	✓
4	baidu.com	1	0	X	✓	X
5	wikipedia.org	2	0	✓	✓	✓
6	qq.com	8	2	✓	✓	✓
9	yahoo.com	14	13	✓	✓	✓
10	sohu.com	8	1	✓	✓	✓
14	twitter.com	1	3	✓	✓	✓
15	amazon.com	0	5	✓	✓	✓
19	360.cn	14	3	✓	✓	✓
25	blogspot.com	2	2	X	✓	X
32	alipay.com	1	0	X	✓	X
43	imdb.com	7	3	✓	✓	✓
45	microsoft.com	2	1	✓	✓	✓
52	tumblr.com	13	9	X	✓	X
55	naver.com	7	6	✓	✓	✓
56	office.com	5	5	✓	✓	✓
59	wordpress.com	1	5	✓	✓	✓
66	paypal.com	5	2	✓	✓	✓
69	microsoftonline.com	2	1	✓	✓	✓
70	soso.com	1	0	X	✓	X
72	stackoverflow.com	2	2	✓	✓	✓
74	github.com	3	3	✓	✓	✓
89	roblox.com	10	2	✓	✓	✓
95	bbc.com	6	6	X	✓	X
100	quora.com	2	1	✓	✓	✓

To evaluate minification service, a web page should include at least 1 external CSS or JavaScript file. Three of these web pages do not include any external JavaScript or CSS files. Thus, these web pages were eliminated. Seven of the remained web pages were modified after minification. Finally, 25 web pages remained to evaluate the minification service (see Table 4.1).

4.3 Equipments and Tools

The architecture of our system consists of the server, the proxy and the client. In order to eliminate external factors and differences in these components that could affect the measurements, three identical desktop computers are used as these three components. The processor of these computers is Intel Core i7 model 4770 with the base frequency 3.4 GHz and the turbo frequency 3.9 GHz. The processor consists of 4 cores and the size of the installed memory is 16 GB. The operating system of the server and the proxy is Ubuntu 16.04 and the desktop client is Windows 10. We have set the brightness of the screens to 50% during the tests. Cisco 2901 Integrated Services Router is used to provide the network between these computers.

For the tests done over mobile device, the router and the client computer is replaced. The router is replaced with a switch which is Cisco Catalyst 2960-X and Netgear Wireless-N 300 Access Point (WN802T) is connected to the switch to provide wireless network. The client desktop is replaced with Samsung Galaxy S3 Mini (GT-I8200Q). Its processor is 1.2 GHz dual core Cortex-A9 and it has 1 GB RAM. The installed operating system is Android 4.2.2. The screen resolution is 480 x 800 pixels and the brightness is set to 50% during the tests. No external microSD card is installed and only PowerTutor application is installed to be used in our tests. The mobile device connects to the local network via Wi-Fi over the wireless access point and the switch.

For the evaluation, different tools used on the desktop and mobile to measure the power or energy consumption, the size of the components and the number of requests. These tools are described as follow:

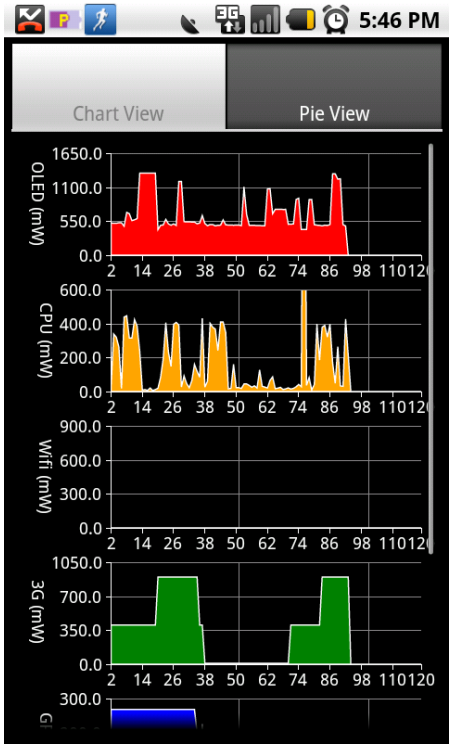

Figure 4.1: The GUI of PowerTutor

PowerTutor: It is an Android application to display the power consumption of major system components such as display, CPU, network interface and different applications and we used this tool for the measurements done on the mobile client (see Figure 4.1). It was developed by Ph.D. students from University of Michigan, in 2011. Software developers may see the impact of design changes on power efficiency with the help of this application. It also let users to determine how the battery life is impacted by various actions. It produces text file output as well to see detailed results. Its power model is built by direct measurements during control of device power management states including the CPU, OLED/LCD, Wi-Fi, 3G, GPS and audio. Their model estimates the energy consumption within 5% of actual values, according to their measurements. In our experiments, the measurement process is set to start after the total power and the CPU power become stable.

Intel Power Gadget: We used this tool for the measurements done on the desktop client and cumulative processor energy (Joules), cumulative processor energy (mWh) and average proces-

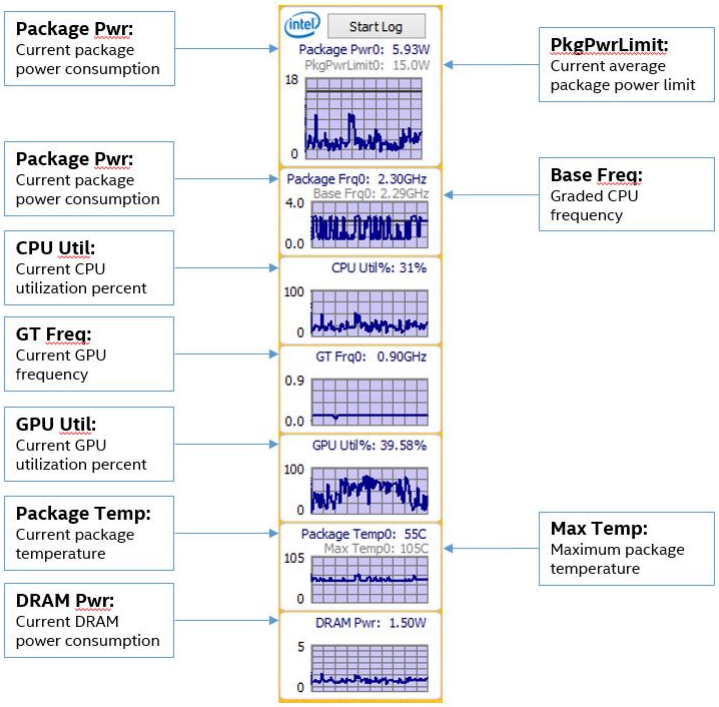

Figure 4.2: The GUI of Intel Power Gadget

sor power (Watt) data is used from the log file generated by the tool. It is a software based tool to monitor power usage. It works on Windows, Linux and Mac operating systems and it supports Intel Core processors from second generation to seventh generation but not Intel Atom. Thus, this tool is suitable for our equipment. It estimates the real time processor package power information by using the energy counters of the processor. As it does not need any hardware instrumentation, many developers are interested in this tool. For the evaluation, version 3.5 is used on Windows 10 operating system. This version consists of several components such as temperature, frequency and power usage (see Figure 4.2). It also creates log files to see the elapsed timed, package power limit, GT frequency, processor temperature, average and cumulative power of the processor.

Google Chrome DevTools: This tool is used to record number of requests, loading time, JavaScript and CSS files and the size of the components on the desktop client. It is a web developer tool that helps developers to edit web pages on the fly. It is built directly into Google

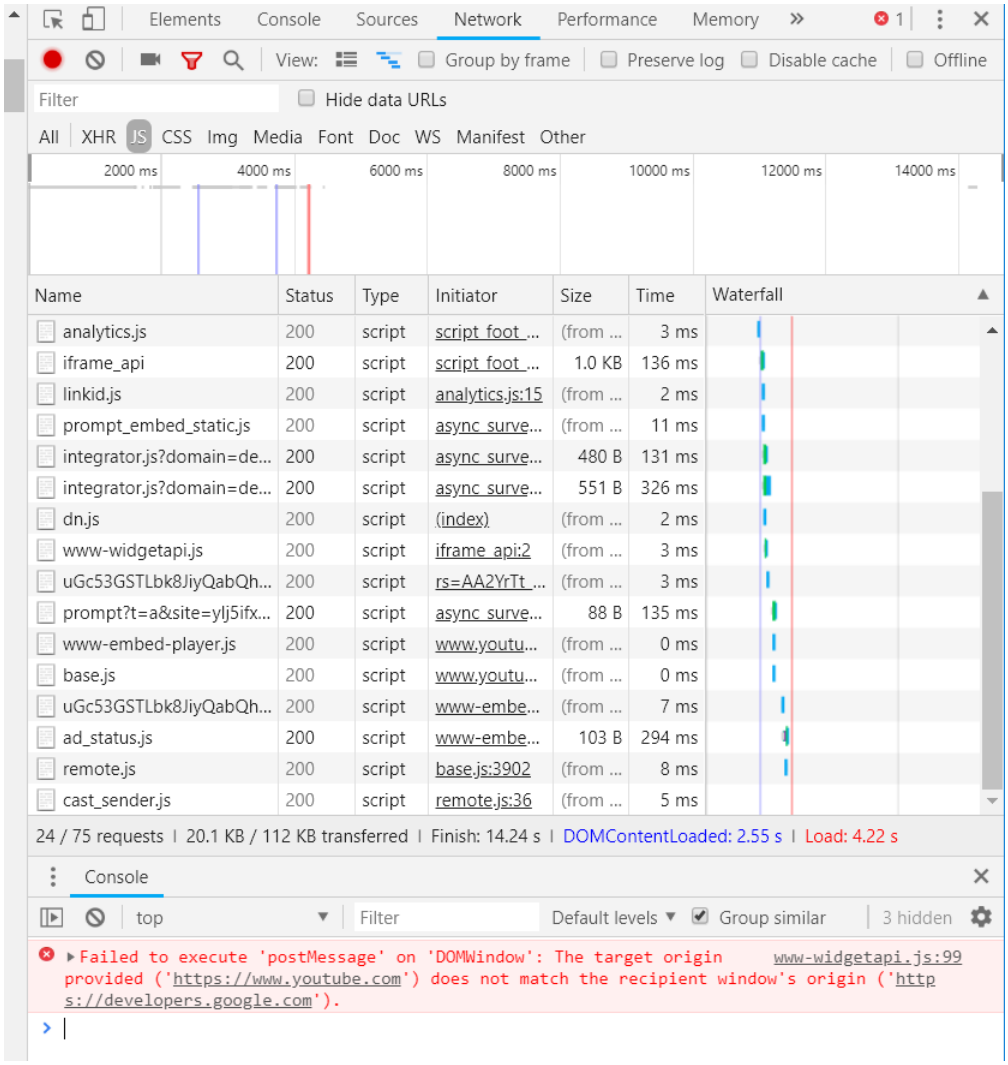

Figure 4.3: Google Chrome DevTools

Chrome and developers can record the process of loading of the web pages via this tool. Thus, it helps to diagnose problems quickly and build better and faster web pages. Developers may view and change the contents of web pages via DevTools. There are 9 categories under DevTools which are elements, console, sources, network, performance, memory, application, security and audits (see Figure 4.3). In this figure, JavaScript files are showed from the network module with their status, size, initiator and load time. Moreover, this tool can simulate different networks such as GPRS, 2G, 3G, 4G, DSL or WiFi. In our tests, Regular 4G (4 Mb/s download speed,

250 kb/s upload speed and 20ms RTT) is used as it is the closest technology that can be simulated. Another important feature for our tests is caching. The cache is disabled during our tests using this tool.

4.4 Test Methodology

The evaluation methodology is constructed over our research questions. In order to see the impact of the two services implemented, they are evaluated both independently and dependently. Firstly, the web pages are evaluated with our minification service. Secondly, they are evaluated with concatenation service. Finally, the impact of both concatenation and minification is evaluated. Our metrics are number of the requests, page load duration and power/energy consumption for concatenation service. For minification service metrics, number of the requests is replaced with the size of the components.

The evaluation is done over two different architectures to see the impact of our services (see Figure 4.4 and Figure 4.5). Figure 4.4 shows the architecture without transcoding. Web pages are requested directly from the server without any transcoding. In the other architecture, proxy sits between the server and the client and web pages are transcoded before served to the client (see Figure 4.5). These two cases are done over a local network in order to eliminate the external factors such as the latency.

Figure 4.4: The Evaluation Architecture without Transcoding

The evaluation is performed over two types of client: desktop and mobile. The tests are also performed over the desktop computer as the improvements can be seen better on a desktop computer because of its powerful processor and limitless sources.

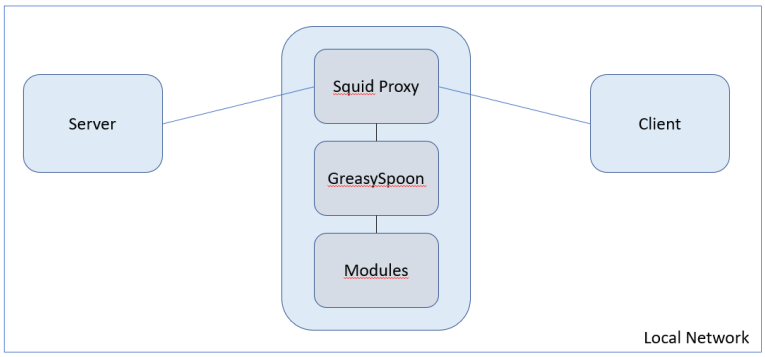

Figure 4.5: The Evaluation Architecture with Transcoding

During the evaluation over the desktop client, Regular 4G (4 Mb/s download speed, 250 kb/s upload speed and 20ms RTT) is simulated by Google Chrome DevTool. The cache is disabled and the history is deleted over the browser. Also, the cache and the data on the proxy and the cache on the server are cleared to see the real impact of the services. Google Chrome DevTool is used to record the page load duration, number of requests and the size of the components. Intel Power Gadget is used to record energy consumption. All of the other applications apart from the browser and Intel Power Gadget are stopped during the evaluation over the desktop client. The duration for the energy measurement is set to total 30 seconds. The first 10 seconds is idle and browser is launched in this period. Then web page is requested and waited for 10 seconds to load the page. Just after, we switch to Intel Power Gadget and after waiting 10 seconds the measurement is done. Indeed, requesting web pages are less than 10 seconds for most of the web pages but this duration is constant. This duration is set after the tests done to see the load time of the web pages. Network Panel of DevTool is used to see the duration of the page load time.

During the evaluation over the mobile client, the cache and the history of the browser is cleared same as the evaluation over the desktop client. The mobile device is connected to the network via Wi-Fi as we do not want Internet affect our tests. PowerTutor is used to measure the energy consumption over the mobile client and no other application is installed on the mobile device. The duration of the test lasts 90 seconds. In the first 30 seconds, browser is launched. Then,

web page is requested and waited for 30 seconds. After this duration, we switch to PowerTutor and the test duration finishes after 30 seconds. The duration over the mobile client is longer than the duration over the desktop client as load time of the web page is increased and switching between the applications (browser and PowerTutor) takes more time compared to desktop client. The tests are repeated if any unexpected behavior (such as fluctuation) is observed.

As explained above, the evaluation of our services is done in the ”wild” which means that real web pages are requested by real users to see the real impact of the services. As the cache is disabled and the history is cleared to eliminate the factors may affect our results, the results may differ (page load time may be shortened) in real life usage with enabling cache.

4.5 Results

In this section, we present the results of three different services (concatenation, minification and concatenation+minification) of our research over two clients: desktop and mobile.

4.5.1 Results on Desktop Client

Firstly, the mean, the median and the standard deviation of the number of requests, total size in KB and the load time in seconds are calculated for before and after transcoding (concatenation, minification and concatenation+minification) (see Table 4.2, 4.3 and 4.4). The results show that the number of requests is decreased after the transcoding over concatenation and concatenation+minification and the total size of the components is decreased after the transcoding over minification and concatenation+minification services. For example, the mean value of the number of the requests decreased from 76.80 to 69.35 after transcoded by the concatenation service (see Table 4.2). As given in Section 4.2, we have 19 web pages to test concatenation and concatenation+minification and 25 web pages to test minification service. Thus, the before values in these tables are different as the number of samples is different.

The next step is to see the impact of the decrease in the number of the requests and the total

Table 4.2: Mean, Median and Standard Deviation of Each Sample of the Number of Requests, Total Size (KB) and the Load Time (s) for Concatenation Service

	Number of Requests		Total Size (KB)		Load Time (s)	
	Before	After	Before	After	Before	After
Mean	76.80	69.35	1035.32	1035.37	2.19	2.41
Median	47.50	39.00	811.00	811.00	1.70	2.10
Standard Deviation	83.15	75.93	965.99	965.34	1.88	1.82

Table 4.3: Mean, Median and Standard Deviation of Each Sample of the Number of Requests, Total Size (KB) and the Load Time (s) for Minification Service

	Number of Requests		Total Size (KB)		Load Time (s)	
	Before	After	Before	After	Before	After
Mean	55.44	55.44	899.50	663.53	1.90	3.77
Median	46.00	46.00	598.00	330.00	1.44	3.24
Standard Deviation	45.84	45.84	891.70	660.85	1.71	2.47

Table 4.4: Mean, Median and Standard Deviation of Each Sample of the Number of Requests, Total Size (KB) and the Load Time (s) for Concatenation+Minification Service

	Number of Requests		Total Size (KB)		Load Time (s)	
	Before	After	Before	After	Before	After
Mean	76.80	69.35	1035.32	913.65	2.19	3.49
Median	47.50	39.00	811.00	729.00	1.70	3.14
Standard Deviation	83.15	75.93	965.99	919.24	1.88	2.02

size on energy consumption of the device. To see that, Intel Power Gadget is used to record Cumulative Processor Energy (Joules), Cumulative Processor (Total processor energy) Energy (mWh), Average Processor Power (Watt), Cumulative IA (Energy of the CPU/processor cores) Energy (Joules), Cumulative IA Energy (mWh), Average IA Power (Watt). The mean, median and the standard deviation calculations of these measurements (before and after transcoding)

are given in Table 4.5, 4.6, 4.7, 4.8, 4.9, 4.10. The results show that our services decreased the cumulative processor and ia energy and the average processor and ia power. To illustrate, mean cumulative processor energy is decreased from 119.48 Joules to 106.53 Joules after transcoding over our concatenation service (see Table 4.5). The rest of the results for each web page can be see in Appendix B.

Table 4.5: Mean, Median and Standard Deviation of Each Sample of the Cumulative Energy and the Average Power of the Processor for Concatenation Service

	Cumulative Processor Energy (Joules)		Cumulative Processor Energy (mWh)		Average Processor Power (Watt)	
	Before	After	Before	After	Before	After
Mean	119.48	106.53	33.19	29.12	3.94	3.49
Median	121.24	101.83	33.68	28.28	3.98	3.38
Standard Deviation	21.20	16.90	5.89	4.02	0.72	0.57

Table 4.6: Mean, Median and Standard Deviation of Each Sample of the Cumulative Energy and the Average Power of IA for Concatenation Service

	Cumulative IA Energy (Joules)		Cumulative IA Energy (mWh)		Average IA Power (Watt)	
	Before	After	Before	After	Before	After
Mean	39.82	32.53	11.07	8.45	1.30	1.06
Median	39.25	30.77	10.90	8.54	1.29	1.02
Standard Deviation	13.68	10.01	3.78	3.33	0.43	0.33

These results are also evaluated statistically to determine whether or not the difference between before and after transcoding over our services is significant. First of all, Shapiro-Wilk (SW) Test is applied to see whether or not the difference between the results is normal distribution or not. If the p value is greater than 0.05 it means that the data come from a normally distributed population. If this p value is less than 0.05, the data do not come from normally distributed population. Here, the value of 0.05 comes from 95% confidence level. Based on the p value, Paired-Dependent T-Test or Wilcoxon Signed Rank Test is applied to check whether the dif-

Table 4.7: Mean, Median and Standard Deviation of Each Sample of the Cumulative Energy and the Average Power of the Processor for Minification Service

	Cumulative Processor Energy (Joules)		Cumulative Processor Energy (mWh)		Average Processor Power (Watt)	
	Before	After	Before	After	Before	After
Mean	115.94	108.80	32.21	29.77	3.83	3.58
Median	115.68	112.00	32.13	28.10	3.82	3.33
Standard Deviation	20.84	18.69	5.79	4.90	0.70	0.62

Table 4.8: Mean, Median and Standard Deviation of Each Sample of the Cumulative Energy and the Average Power of IA for Minification Service

	Cumulative IA Energy (Joules)		Cumulative IA Energy (mWh)		Average IA Power (Watt)	
	Before	After	Before	After	Before	After
Mean	37.71	33.08	10.49	9.09	1.24	1.09
Median	37.03	34.86	10.29	8.19	1.22	1.15
Standard Deviation	13.37	11.79	3.71	3.28	0.42	0.39

Table 4.9: Mean, Median and Standard Deviation of Each Sample of the Cumulative Energy and the Average Power of the Processor for Concatenation+Minification Service

	Cumulative Processor Energy (Joules)		Cumulative Processor Energy (mWh)		Average Processor Power (Watt)	
	Before	After	Before	After	Before	After
Mean	119.48	109.05	33.19	29.91	3.94	3.59
Median	121.24	108.00	33.68	29.90	3.98	3.56
Standard Deviation	21.20	19.45	5.89	5.26	0.72	0.65

ference is significant or not. Paired-Dependent T-Test is for normal distribution ($p > 0.05$) and Wilxocon Signed Rank Test is for not-normal distribution. Then, using the p value of Paired-Dependent T-Test or Wilcoxon Signed Rank Test, the significance of the data can be interpreted. If this p value is less than 0.05 (comes from 95% confidence interval), it can be interpreted that

Table 4.10: Mean, Median and Standard Deviation of Each Sample of the Cumulative Energy and the Average Power of IA for Concatenation+Minification Service

	Cumulative IA Energy (Joules)		Cumulative IA Energy (mWh)		Average IA Power (Watt)	
	Before	After	Before	After	Before	After
Mean	39.82	34.09	11.07	9.09	1.30	1.12
Median	39.25	35.32	10.90	9.40	1.29	1.17
Standard Deviation	13.68	10.71	3.78	3.11	0.43	0.37

there is a significant difference before and after the transcoding over our services. However, the effect size cannot be interpreted with p value. To interpret the effect size, Eta Squared Statistics is used for the Paired-Dependent T-Test and the r value is used for the Wilcoxon Signed Rank. For Eta Squared Test 0.01 is small, 0.06 is moderate and 0.14 is large effect size and for the r value of Wilcoxon Signed Rank Test, 0.1 is small, 0.3 is medium and 0.5 is large effect size.

The result of the mentioned statistical tests to see the significance of the difference is presented in Table 4.11, 4.12, 4.13. The results show that our services achieved significance difference over the load time, number of the requests, total size of the components, cumulative processor energy (Joules), cumulative processor energy (mWh), average processor power (Watt), cumulative IA energy (Joules), cumulative IA energy (mWh), average IA power (Watt). All statistical calculations are done by using IBM SPSS Statistics tool. Paired-Dependent T-Test is applied for some of the data and Wilcoxon Signed Ranked Test is applied for some of the data, based on the result of Shapiro Wilk Test. To illustrate, for the result of the concatenation service, Wilcoxon Signed Rank Test is applied for the number of requests, cumulative processor energy (Joules), cumulative processor energy (mWh) and average IA power (Watt). Paired Dependent T-Test is applied for the load time, average processor power (Watt), cumulative IA energy (Joules) and cumulative IA energy (mWh) (see Table 4.11). t, degrees of freedom (df), significance level (Sig.) and eta squared values are calculated with Paired-Dependent T-Test and r values are calculated with Wilcoxon Signed Rank Test. For example, Paired-Dependent T-Test shows that there is a significant difference in concatenation service for the number of requests before and after transcoding,

with t(18) = -2.560, p = 0.019 and the value of eta squared is 0.267 which indicates large effect size. Wilcoxon Signed Ranked Test shows that there is a significance difference for the cumulative processor energy (Joules) before and after transcoding with concatenation service, with Z = -3.823 and p = 0.000 with large effect size (r=0.620).

Table 4.11: The Results of the Paired-Dependent T-Test and Wilcoxon Signed Rank Test for Concatenation Service

	Paired-Dependent T-Test				Wilcoxon Signed Rank Test		
	t	df	Sig.	Eta Squared	Rank	Test	r
Number of Requests					-3.927	0.000	0.637
Total Size (KB)							
Load Time (s)	-2.560	18	0.019	0.267			
Cumulative Processor Energy (Joules)					-3.823	0.000	0.620
Cumulative Processor Energy (mWh)					-3.823	0.000	0.620
Average Processor Power (Watt)	6.150	18	0.000	0.678			
Cumulative IA Energy (Joules)	5.275	18	0.000	0.607			
Cumulative IA Energy (mWh)	4.483	18	0.000	0.528			
Average IA Power (Watt)					-3.823	0.000	0.620

Table 4.12: The Results of the Paired-Dependent T-Test and Wilcoxon Signed Rank Test for Minification Service

	Paired-Dependent T-Test				Wilcoxon Signed Rank Test		
	t	df	Sig.	Eta Squared	Rank	Test	r
Number of Requests							
Total Size (KB)					-3.623	0.000	0.512
Load Time (s)					-4.373	0.000	0.618
Cumulative Processor Energy (Joules)					-3.969	0.000	0.561
Cumulative Processor Energy (mWh)					-3.619	0.000	0.512
Average Processor Power (Watt)					-3.529	0.000	0.499
Cumulative IA Energy (Joules)					-3.754	0.000	0.531
Cumulative IA Energy (mWh)	4.754	24	0.000	0.485			
Average IA Power (Watt)	4.885	24	0.000	0.499			

Table 4.13: The Results of the Paired-Dependent T-Test and Wilcoxon Signed Rank Test for Concatenation+Minification Service

	Paired-Dependent T-Test				Wilcoxon Signed Rank Test		
	t	df	Sig.	Eta Squared	Rank	Test	r
Number of Requests					-3.927	0.000	0.637
Total Size (KB)					-3.409	0.001	0.553
Load Time (s)					-3.920	0.000	0.636
Cumulative Processor Energy (Joules)					-3.823	0.000	0.620
Cumulative Processor Energy (mWh)					-3.783	0.000	0.614
Average Processor Power (Watt)					-3.825	0.000	0.620
Cumulative IA Energy (Joules)	3.679	18	0.002	0.429			
Cumulative IA Energy (mWh)					-3.823	0.000	0.620
Average IA Power (Watt)					-3.785	0.000	0.614

4.5.2 Results on Mobile Client

Our services are also evaluated over a mobile phone. PowerTutor is used to trace average power on the mobile client in total 90 seconds interval (30 seconds for the request). Results show that our concatenation, minification and concatenation+minification services achieved 3.94%, 2.50% and 3.34% average power saving over the mobile client, respectively (see Table 4.14 and Table 4.15). The rest of the results for each web page can be seen in Appendix C.

Table 4.14: Mean Value of the of Average Power Comparison for Concatenation and Concatenation+Minification Services over Mobile Client

	No Transcoding	Concatenation	Concatenation+Minification
Mean Value of the Average Power (mW)	654	628	632

Table 4.15: Mean Value of the of Average Power Comparison for Minification Service over Mobile Client

	No Transcoding	Minification
Mean Value of the Average Power (mW)	652	636

Table 4.16: The Results of the Paired-Dependent T-Test and Wilcoxon Signed Rank Test for the Mobile Client

	Paired-Dependent T-Test				Wilcoxon Signed Rank Test		
	t	df	Sig.	Eta Squared	Rank	Test	r
Concatenation	5.502	18	0.000	0.627			
Minification					-3.560	0.000	0.549
Concatenation+Minification					-3.825	0.000	0.620

These mobile results are also evaluated statistically to determine whether or not the difference between before and after transcoding over our services is significant (see Table 4.16). For example, Paired-Dependent T-Test shows that there is a significant difference in concatenation service over the mobile client for the average power (mW) before and after transcoding, with t(18) = -5.502, p = 0.000 and the value of eta squared is 0.627 which indicates large effect size. Wilcoxon Signed Ranked Test shows that there is a significance difference for the average power (mW) before and after transcoding with minification service over mobile client, with Z = -3.560 and p = 0.000 with large effect size (r=0.549).

In this section, we presented the results of our transcoding services and the statistical test results. The result of statistical tests shows that the difference before and after transcoding is significant for all of the services in our research with large effect size.

4.6 Discussion

In this section, we discuss our major findings with respect to the research questions mentioned in Section 4.1. The aim of this thesis is to save energy by concatenating and/or minifying external JavaScript and CSS files to reduce the number of HTTP connections and/or reduce the size of the components. Thus, two services are implemented on the proxy server: (1) concatenation and (2) minification. The evaluation of these services are done in three levels: (1) concatenation, (2) minification and (3) concatenation+minification.

Our results show that the services implemented achieved statistically significant energy saving. Our three services (concatenation, minification and concatenation+minification) reduced the cumulative processor energy (mWh) by 12.26%, 7.57% and 9.87% and the average processor power (Watt) 11.35%, 6.39% and 9.01% respectively in average of our evaluation set. The statistical tests show that there is a significant improvement and the effect size is large.

The results show that the highest improvement is achieved with concatenation service, without minification. Although concatenation+minification reduces the number of requests and the size

of the components, by only concatenating the external files, more energy saving is achieved. There may be different factors that affect this but our major finding is the load time. The average load time of concatenation service increased by 10.2% while this rate is 59.49% for concatenation+minification and 98.67% for minification. As the concatenation service does not increase the load time so much, it achieved the highest energy saving.

When we analyze the load time results, it can be seen that concatenation is the fastest service and minification is the slowest service. Concatenation+minification has the moderate speed although it manage both concatenation and minification and its speed is better than minification only. The reason is that minification service starts different process for each external JavaScript and CSS file. On the other hand, concatenation+minification first concatenates the external files and then it minifies the concatenated files. Thus, the number of minification process is decreased over the concatenation+minification service.

The impact of the services are also evaluated over the mobile client. The average power measurement results over the mobile client show that our concatenation, minification and concatenation+minification services achieved 3.94%, 2.50% and 3.34% average power saving, respectively. The percentage of the energy saving over the mobile client is less than the energy saving over the desktop client. One reason of this may be that different evaluation intervals are used for the mobile and the desktop client. The interval for the measurements over the mobile client is 90 seconds (30 seconds idle, 30 seconds for the request and 30 seconds idle again) and over the desktop client is 30 seconds (10 seconds idle, 10 seconds for the request and 10 seconds idle again). This interval is changed for the desktop and mobile client as switching between PowerTutor and the browser takes more time than launching the browser on the desktop client. Another reason for different intervals is that loading time of the web page is increased over the mobile client, therefore to better observe the time difference of the services is obtained less over the mobile client as the time interval is increased.

The guidelines mentioned in Chapter 2 states that minification can achieve around 20% size reduction. In our study, minification service achieved up to 26% reduction in the total size of the web site.

One aim of this thesis is to increase the battery life of the client device while browsing. Thus, we analyzed the impact of this improvement on the battery and we assumed that the client is a laptop works on a battery. For the specifications, the battery of the closest laptop (ASUS N580VD) is assumed with 47Wh capacity. As the average power of the result does not show the average total system power, two different assumption is used: the average processor power is 100% of the total system power and 70% of the total system power. The mean values of each service are used to calculate the battery duration to analyze the improvement in the battery duration in an overview.

The result of the calculation shows that our concatenation, minification and concatenation +minification services can achieve 13%, 7% and 9% improvement in the battery life on the client side with the assumption of processor power equals to total system power (see Table 4.17) and 9%, 5% and 7% improvement in the battery life on the client side with the assumption of processor power equals to 70% of the total system power (see Table 4.18).

In summary, our results show that the implemented services may achieve energy saving on the client side and this energy saving is statistically significant. Our battery life analysis show that these services improve the battery life on the client side.

In the evaluation of our services, caching is disabled to see the impact of the services without any effect. However, this system can be used for large scale systems to answer lot of clients and caching on the proxy may be applied to reduce the number of requests between the proxy and the main server. Once the content is transcoded on the proxy, it can be served to many clients and this may decrease the energy consumption of the proxy and the load time of the web page.

4.7 Summary

In this chapter, the evaluation process of our system is described. We have asked two research questions to evaluate the impact of our services. The equipments and the tools used during the evaluation are presented. The methodology behind our evaluation process is explained. Finally,

the numerical results given and their impact on mobile devices is discussed. The next section gives the detail of the model between the energy consumption of the proxy and the energy saving from the clients with the aim of total energy saving.

Table 4.17: Battery Life Analysis with the Assumption of Total System Power

100%	% Reduction in Average Processor Power	Battery Duration (hour)		Improvement (hour)	Improvement (%)
		Before	After		
Concatentation	11.35	11.93	13.47	1.54	13
Minification	6.39	12.27	13.13	0.86	7
Concatentation+Minification	9.01	11.93	13.09	1.16	10

Table 4.18: Battery Life Analysis with the Assumption of 70% of Total System Power

100%	% Reduction in Average Processor Power	Battery Duration (hour)		Improvement (hour)	Improvement (%)
		Before	After		
Concatentation	11.35	11.93	12.97	1.04	9
Minification	6.39	12.27	12.86	0.59	5
Concatentation+Minification	9.01	11.93	12.72	0.79	7

CHAPTER 5

MODELING SUSTAINABILITY

The evaluation results show that our system achieves statistically significant energy saving on the client side. However, proxy is an extra component in our system and its energy consumption cannot be negligible if we aim overall system energy efficiency to contribute sustainability. Thus, this section gives the detail of our basic model and its result over our concatenation service to see whether our system achieves overall energy efficiency or not.

In evaluation of our services, caching is disabled to see the real impact. However, this system can be used for large scale systems to answer lot of clients and caching on the proxy may be applied to reduce the number of requests between the proxy and the main server. Once the content is transcoded on the proxy, it can be served to many clients. Thus, our model is based on caching the web page sources on the proxy. The data for this model is based on our concatenation service.

Our approach for modeling the sustainability is as follow: In the first request of the web page, all of the web page sources are downloaded to our proxy server and the transcoding is done. The source with the transcoded files are cached over the proxy. After the first request of the web page, if any client requests the same web page, the sources are served from the proxy instead of the server. Here, our assumption is that there is no change in the web page during this process. Our model eliminates the server after the first request of the web page. Thus, the energy consumption of the server is ignored after the first request of the web page.

In our model, we attempt to find how many clients should request the same web page to achieve

energy efficiency in the overall system. Thus, the comparison is based on the total system energy of n clients with and without transcoding. The total energy consumption is calculated with two components (server and client) for the normal case (without transcoding), three components (server, proxy and client) for the first request (caching) and two components (proxy and client) for the case after data is cached on the proxy. These measurements are done in 30 seconds period, similar to the tests done for the evaluation of the services.

In our model, another assumption is that there is no traffic in our server. The scenario for our model is as follow: Client(n) requests the web page and after 30 seconds passed, client(n+1) requests the same web page. No idle time is passed between these 30 seconds.

Thus, the inputs of our model is as follow:

A: Total Energy with Transcoding and the Source is Already Cached

B: Total Energy without Transcoding

C: Total Energy of First Request for Transcoding

A = Proxy Energy + Client Energy

B = Server Energy + Client Energy

C = Server Energy + Proxy Energy + Client Energy

n = Number of Clients

Based on the above explanations, the formula should be as follow to have overall energy efficiency with n clients:

$$n * B > C + n * A$$

The results of the above formula for the concatenation service is shown in Table 5.1. The values in the table are the mean values for 17 web pages out of 19. No energy efficiency is obtained for 2 of the web pages and this is why 17 web pages instead of 19 web pages is used to calculate

Table 5.1: Measurement Inputs for the Model and the Result for Number of Clients

Energy without Transcoding (mWh)	Client	33.542
	Proxy	0
	Server	19.974
	Total (B)	53.516
Energy with Transcoding and the Source is Already Cached (mWh)	Client	24.431
	Proxy	22.366
	Server	0
	Total (A)	46.797
Energy during First Request for Transcoding (mWh)	Client	55.964
	Proxy	40.963
	Server	19.974
	Total (C)	116.302
n		18

the average values. The average number of clients is calculated as 18 to achieve overall system energy efficiency (see Table 5.1 and Figure 5.1).In Figure 5.1, the energy consumption of the overall system for n clients with and without transcoding is showed. According to the figure, it can be seen that if the number of clients is more than 18, the energy consumption of the system with transcoding is less than the energy consumption of the system without transcoding. Therefore, after 18 clients request the same web page, energy efficiency is achieved. The rest of the result can be seen in Appendix D.

The power measurements over the client is done with Intel Power Gadget [86] and s-tui [36] is used over the proxy and the server. The reason of different tools used during the evaluation is that the operating system of the client is Windows10 and the proxy and the server is Ubuntu 16.04. We could not find the same tool that measures the power on both of the operating systems. To see the consistency between these tools, we have done some benchmarking. Same benchmarks are tested on Windows and Ubuntu using Intel Power Gadget and s-tui and the standard error between these tools are calculated as 5.2% in average.

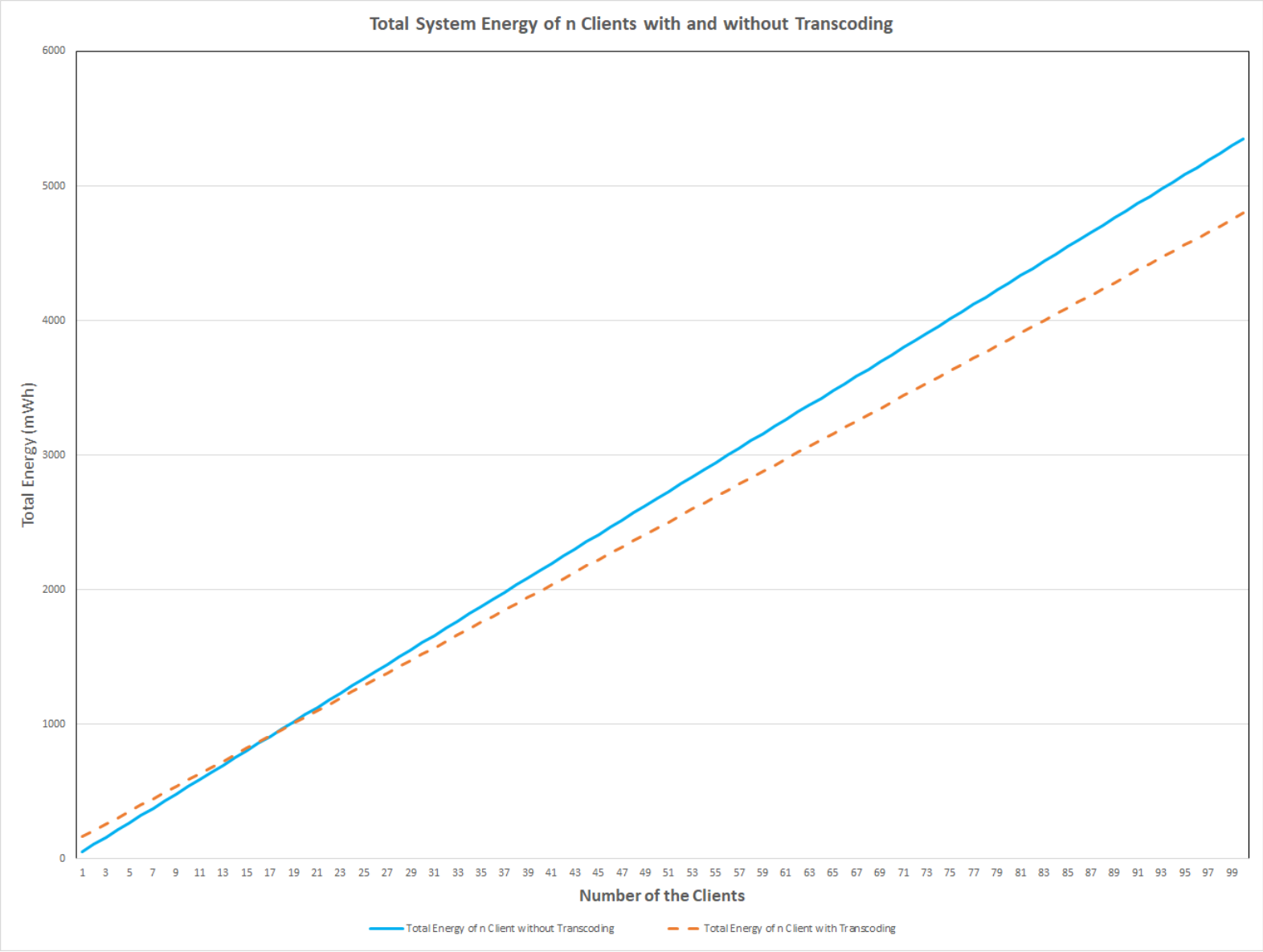

Figure 5.1: Total System Energy of n Clients with and without Transcoding

For testing purposes, the caching process for our model is not done automatically based on an algorithm. The web page source is saved to the proxy manually. We assumed that the source is cached over the proxy after the first request and when a client requests the same web page, the source is sent over the proxy instead of the server. Here, manually saved source is delivered to the client by the proxy. As no caching mechanism is structured over our proxy, the average power of the proxy and the client during the first request (caching) is based on some calculations. The formulas to calculate average power of the proxy and the client during the first request is as follow:

Total Energy Consumption during First Request (Proxy) = Total Energy Consumption to Transcode the Web Page (Proxy) + Total Energy Consumption to Request the Web Page without Transcoding (Client) - Total Idle Energy Consumption (Client)

*Total Energy Consumption during First Request (Client) = Total Idle Energy Consumption (Client) + 2 * (Total Idle Energy Consumption (Client) - Total Energy Consumption to Request the Transcoded Web Page without Caching (Client))*

In summary, we have described our basic model to see whether energy efficiency in our overall system can be achieved or not. This model is evaluated with concatenation service and the result shows that overall system energy efficiency can be achieved in 17 of 19 web sites and the mean client number that is needed to request the web page is calculated as 18.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Our world is threatened with the effects of the global warming. One of the most important cause of global warming is the carbon emission. There are different sectors that cause carbon emission with different rates and Information and Communication Technologies is one of the highest contributors. Mobile and fixed telecommunication has high impact on carbon emission under Information and Communication Technologies.

In the past, the number of mobile devices were not very high, people were using desktop computers. Over the years, this tendency has changed. Now, mobile devices are more often used than desktop computers. There are lots of things that we do with the mobile devices and browsing is one of them. As the number of mobile devices increased, people started to browse web pages from mobile devices rather than desktop computers.

Web pages have also evolved. Their average size and average number of requests increased. The evolutions on web pages is not a problem for desktop computers; however, mobile device users may not be satisfied with browsing from mobile devices. There are two reasons: the web pages that are not designed ideally and the limitations of mobile devices such as the processor, the battery, lack of memory, the size of the screen and limited bandwidth. Thus, web pages may be loaded with latency on mobile devices and it may increase the energy consumption and battery dies quickly.

A web page consists of different elements such as HTML code, images, videos, stylesheets and

scripts. Scripts and stylesheets have high energy consumption between these elements. In the literature, there are two main ideas behind reducing the energy consumption of the web pages: (1) reducing the number of requests and (2) reducing the size of the components. Thus, in this thesis, we attempted to reduce the energy consumption of web pages by transcoding web pages without modifying their look&feel and without adding extra load to the client or the server side with the goal of energy saving via scripts and stylesheets optimization. Two techniques were proposed and implemented in the proxy server: (1) concatenating external JavaScript and CSS files to reduce the HTTP request and (2) minification of CSS and JavaScript to reduce the size.

There are different ways to access Internet from mobile devices but the HTTP request is same for each access way. The 80% of this loading time is passed during the sources are downloaded and the process on the client side. The remaining 20% time is passed during the content is showed. Mobile users may access web via mobile web applications, native applications, hybrid applications, browsers and widgets. Browsers are the most suitable way for our research as they work on different platforms and many web pages can be reached by the browsers.

There are different studies in the literature that handle energy efficiency. In this thesis, we grouped them into three: hardware, network and software. In hardware level, studies are mentioned that focus on different areas such as reducing the energy consumption of the processor, embedding renewable energy resources into mobile devices, analyzing the role of the processor in energy consumption of mobile web browser and reducing the energy consumption of the screen. In network level, studies that focus energy consumption on network level such as analyzing the energy consumption of the mobile network, the impact of different communication technologies on mobile devices, using localization and caching to improve energy saving in mobile networks are given. In software level, there are different studies that handle the software in order to reduce the energy consumption. These studies handle the effect of the operating system and the programming language, refactoring, code obfuscation and the effect of the networking protocol. There are also some studies related with the energy consumption of web sites and the effect of different JavaScript libraries on energy consumption. Energy efficiency is also investigated by the browsers. However, as web sits between these three levels, it is important that to

evaluate the energy consumption under these three levels.

In the literature, there are various guidelines for performance improvement and energy saving in web pages. There are seven guidelines from both sector and academic studies included. These guidelines include suggestions related with images, redirects, scripts, stylesheets and cookies. There are many JavaScript and CSS related guidelines and there are some implementations related with these suggestions. These guidelines are summarized in a Table 2.4 with the location of transcoding, whether it modifies the look&feel of the page or not and the contribution to energy saving. We have seen that although there are some implementations on these suggestions, most of them were not evaluated in terms of energy saving.

In this thesis, we propose a system that does not put extra load on the client side or the server side and does not modify the look&feel of the web page with the goal of energy saving on the client side based on stylesheets and scripts optimization. Two services are implemented: (1) concatenating external JavaScript and CSS files to reduce the HTTP request and (2) minification of CSS and JavaScript to reduce the size. These services are implemented on a proxy server to prevent putting extra load on the client or on the server. The only setting on the client side is setting the IP address of the proxy over browser settings.

In our system, when a user requests a web page, the response first comes to our proxy, where Squid Caching Proxy is used. Squid Proxy acts like ICAP client and it passes the response to the GreasySpoon ICAP server. There are two services work on GreasySpoon: Minification Service and Consolidation Service. These services are connected to Minification Module and Consolidation Module works on the computer used as a proxy. During minification, each external CSS and JavaScript is found from HTML and downloaded. Then, they are minified and served from the proxy. HTML is also modified during this process. During concatenation, each external JavaScript and CSS call is found from the HTML and these files are downloaded. Then, JavaScript files are concatenated into one and CSS files are concatenated into one file. The external JavaScript and CSS calls are deleted from the HTML and the new concatenated files are called from the HTML. Client receives the transcoded content from the proxy.

The evaluation of these services are done with two different architectures to see the real impact of our system. In the first architecture, client requests the web page from the server directly. In the second architecture, there is a proxy sits between the server and the client. The transcoding is done on this proxy and the transcoded content is served from the proxy. Both of the architectures are local to eliminate external factors that may affect the results. The dataset used in the evaluation are chosen from Alexa Top 100 to ensure that real web pages are used in the study.

Evaluation is done over 3 levels: concatenation, minification and concatenation+minification. The tests are done on 25 web pages to evaluate minification service and 19 web pages for concatenation and concatenation+minification. The number of the requests is reduced up to 9.7% and the total size is reduced up to 26% in average after transcoded. Our results show that the services implemented achieved statistically significant energy saving. Our three services (concatenation, minification and concatenation+minification) reduced the cumulative processor energy (mWh) by 12.26%, 7.57% and 9.87% and the average processor power (Watt) 11.35%, 6.39% and 9.01% respectively. The statistical tests show that there is a significant improvement and the effect size is large.

The evaluation over the mobile client shows that our concatenation, minification and concatenation+minification services achieved 3.94%, 2.50% and 3.34% saving in average power, respectively. The efficiency over mobile client can be seen as less compared with the efficiency over desktop client. One possible reason behind this difference may be that different evaluation intervals are used over the mobile and the desktop client.

One aim of this thesis is to increase the battery life of the client device while browsing. Thus, we analyzed the impact of this improvement on the battery of a laptop. The result of the battery life calculation shows that our concatenation, minification and concatenation+minification services can achieve 13%, 7% and 9% improvement in the battery life on the client side with the assumption of processor power equals to total system power and 9%, 5% and 7% improvement in the battery life on the client side with the assumption of processor power equals to 70% of the total system power.

During the evaluation process, caching feature of the client and the proxy is disabled to eliminate the factors that may affect our results. However, this system can be used for large scale systems to respond many clients and caching on the proxy may be applied to reduce the number of requests between the proxy and the main server. Once the content is transcoded on the proxy, it can be served to many clients and this may decrease the energy consumption of the proxy and the load time of the web page. In particular, for pages that are not dynamic.

Our system achieves energy efficiency on the client side. However, if we consider sustainability, the energy consumption of the proxy cannot be negligible if we aim overall system energy efficiency. Thus, a basic model is constructed to see whether our system achieves overall energy efficiency or not. This model is used for our concatenation service to find how many clients is needed to request cached web page from the proxy. The result of our simple model shows that overall energy efficiency can be achieved if the web page is cached over the proxy and it is served to many clients.

To sum up, in this thesis we described our study that concatenates and minifies external JavaScript and CSS files with the goal of energy saving of the web pages on the client side without modifying their look&feel and without adding extra load to the client or the server side. The results show that there is a statistically significant improvement in energy consumption for accessing web pages.

6.1 Limitations

The current version of our system finds all .js and .css file calls by parsing the HTML. In this process, files with .js extension are found from the calls start with *"<script>"*, end with *">"* and include *src* tag. Similarly, files with .css extension are found from the calls start with *"<link"*, end with *"/>"* and include *"href"*. However, there may be external JavaScript and CSS files that are called from a script. Our system does not handle these files.

6.2 Future Work

The literature review shows that there are many guidelines that may improve the energy efficiency of the web sites. However, only two of them (reducing the number of redirects and resizing images) were implemented to save energy previously [91]. The study in this thesis reveals the fact that concatenating external files and minification can reduce the energy consumption of the web pages. Yet, there are many other suggestions or guidelines that may achieve energy saving. Our system can be improved by adding the implementation of these proposed techniques.

A simple model is constructed to evaluate the overall energy efficiency of our system. However, we did not build any caching mechanism over the proxy and the average power during caching is calculated instead of real measurements as our aim was to show that there is an evidence for overall energy efficiency. A caching mechanism can be integrated over the proxy and this model can be improved based on real measurements. We only evaluated concatenation service with the model. The overall efficiency of the other services can be modeled. Caching mechanism may also reduce the load time and this evaluation can be one of the another future works. The last possible future work for our model is integrating the network traffic as our model ignores the network traffic between the components.

In the evaluation of our services, 19 web pages are used to evaluate concatenation and concatenation+minification and 24 web pages are used to evaluate minification service. Future studies can be conducted with higher numbers of web pages.

REFERENCES

[1] Ajaxmin 5.14.5506.26202, https://www.nuget.org/packages/AjaxMin/

[2] Apache http server, https://httpd.apache.org/

[3] The asset pipeline, http://guides.rubyonrails.org/asset_pipeline.html

[4] Avoids document.write() — tools for web developers — google developers, https://developers.google.com/web/tools/lighthouse/audits/document-write

[5] Avoids document.write() — tools for web developers — google developers, https://developers.google.com/web/tools/lighthouse/audits/document-write

[6] Best practices for speeding up your web site, https://developer.yahoo.com/performance/rules.html, february, 2017

[7] Build websites faster and better, https://codekitapp.com/

[8] Chrome web store, https://chrome.google.com/webstore/category/collection/accessibility

[9] Combine css — pagespeed service — google developers, https://developers.google.com/speed/pagespeed/service/CombineCSS

[10] Combineandminify 2.1.0, https://www.nuget.org/packages/CombineAndMinify/

[11] Concatenation of js and css files, https://www.ibm.com/support/knowledgecenter/en/SSZH4A_6.0.0/com.ibm.worklight.help.doc/devref/c_optimizing_apps_concatenation.html

[12] Content adaptation, https://wiki.squid-cache.org/SquidFaq/ContentAdaptation

[13] Data saver, https://developer.chrome.com/multidevice/data-compression

[14] Dust-me selectors by brothercake, https://addons.mozilla.org/en-US/firefox/addon/dust-me-selectors/

[15] Facts&figures, http://www.atag.org/facts-and-figures.html

[16] Flatten css imports — pagespeed module — google developers, https://developers.google.com/speed/pagespeed/service/FlattenCssImports

[17] Google support, https://support.google.com/

[18] Greasyspoon, http://greasyspoon.sourceforge.net/index.html

[19] Grunt: The javascript task runner, https://gruntjs.com/

[20] Http client. Online, https://docs.microsoft.com/en-us/aspnet/web-api/overview/advanced/calling-a-web-api-from-a-net-client

[21] Http/1.1: Connections, https://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.1.4

[22] Icap, https://wiki.squid-cache.org/Features/ICAP

[23] Information and communication technology (ict) usage in households and by individuals, http://www.turkstat.gov.tr/PreTablo.do?alt_id=1028

[24] iphone 7 - technical specifications, https://www.apple.com/iphone-7/specs/

[25] Jawr - more than a javascript/css compressor, https://j-a-w-r.github.io/

[26] Jazzcat, http://jazzcat.mobify.com/

[27] Jscompress - the javascript compression tool, https://jscompress.com/

[28] Make the web faster — google developers, https://developers.google.com/speed/

[29] A modular minifier based on the postcss ecosystem., http://cssnano.co/

[30] The most popular browsers, https://www.w3schools.com/browsers/default.asp

[31] Pagespeed: Remove unused css, https://gtmetrix.com/remove-unused-css.html

[32] Percentage of households have devices connected to the internet, http://www.turkstat.gov.tr/

[33] Phonegap, https://phonegap.com/

[34] Prepros, https://prepros.io/

[35] Remove unused css - css optimizer, https://unused-css.com/

[36] s-tui, https://github.com/amanusk/s-tui

[37] Spdy: An experimental protocol for a faster web, https://www.chromium.org/spdy/spdy-whitepaper

[38] Sprite images, https://www.modpagespeed.com/doc/filter-image-sprite

[39] squid-cache.org, http://www.squid-cache.org/

[40] Trends, http://httparchive.org

[41] Visualizing web performance, http://www.strangeloopnetworks.com/assets/images/visualizing_web_performance_poster.jpg

[42] Website abandonment happens after 3 seconds, http://www.strangeloopnetworks.com/resources/infographics/web-performance-and-userexpectations/website-abandonment-happens-after-3-seconds

[43] Yui compressor, http://yui.github.io/yuicompressor/

[44] Network performance effects of http/1.1, css1, and png. Online (Jun 1997), https://www.w3.org/Protocols/HTTP/Performance/Pipeline.html

[45] Html 4.01 specification (Dec 1999), https://www.w3.org/TR/html4/cover.html#minitoc

[46] Icts and climate change, background paper for the itu symposium on icts and climate change. Tech. rep., ITU (06 2009), https://www.itu.int/dms_pub/itu-t/oth/06/0F/T060F00600C0004PDFE.pdf

[47] World's first 'smartphone' celebrates 20 years (Aug 2014), http://www.bbc.com/news/technology-28802053

[48] How apple.com will serve retina images to new ipads (Jul 2016), http://blog.cloudfour.com/how-apple-com-will-serve-retina-images-to-new-ipads

[49] css/csso (Nov 2017), https://github.com/css/csso

[50] Mobile fact sheet (Jan 2017), http://www.pewinternet.org/fact-sheet/mobile/

[51] Ahmadi, H., Kong, J.: Efficient web browsing on small screens. In: Proceedings of the working conference on Advanced visual interfaces. pp. 23–30. ACM (2008)

[52] Amrutkar, C., Traynor, P., Van Oorschot, P.C.: An empirical evaluation of security indicators in mobile web browsers. IEEE Transactions on Mobile Computing 14(5), 889–903 (2015)

[53] Asakawa, C., Takagi, H.: Transcoding. In: Web Accessibility, pp. 231–260. Springer (2008)

[54] Balasubramanian, N., Balasubramanian, A., Venkataramani, A.: Energy consumption in mobile phones: A measurement study and implications for network applications. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. pp. 280–293. IMC '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1644893.1644927

[55] Barr, K.C., Asanović, K.: Energy-aware lossless data compression. ACM Transactions on Computer Systems (TOCS) 24(3), 250–291 (2006)

[56] Basques, K.: What's new in devtools (chrome 59) — web — google developers, https://developers.google.com/web/updates/2017/04/devtools-release-notes

[57] Ben Abdesslem, F., Phillips, A., Henderson, T.: Less is more: energy-efficient mobile sensing with senseless. In: Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds. pp. 61–62. ACM (2009)

[58] Bengtsson, P.: peterbe/mincss (Nov 2017), https://github.com/peterbe/mincss

[59] Bui, D.H., Liu, Y., Kim, H., Shin, I., Zhao, F.: Rethinking energy-performance trade-off in mobile web page loading. In: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking. pp. 14–26. MobiCom '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2789168.2790103

[60] Bunse, C.: On the impact of code obfuscation to software energy consumption. In: From Science to Society, pp. 239–249. Springer (2018)

[61] Bunse, C., Rohdé, A.: Software development guidelines for performance and energy: Initial case studies. In: Advances and New Trends in Environmental Informatics, pp. 25–35. Springer (2017)

[62] Buzzi, S., I, C., Klein, T.E., Poor, H.V., Yang, C., Zappone, A.: A survey of energy-efficient techniques for 5g networks and challenges ahead. IEEE Journal on Selected Areas in Communications 34(4), 697–709 (April 2016)

[63] Cameron, C.: Chrome: Faster and more battery-friendly (Sep 2016), https://blog.google/products/chrome/chrome-faster-and-more-battery-friendly/

[64] Chen, J., Zhou, B., Shi, J., Zhang, H., Fengwu, Q.: Function-based object model towards website adaptation. In: Proceedings of the 10th international conference on World Wide Web. pp. 587–596. ACM (2001)

[65] Chi, C.H., Deng, J., Lim, Y.H.: Compression proxy server: Design and implementation. In: USENIX Symposium on Internet Technologies and Systems (1999)

[66] Chrome, G.: Power saver. Online (2014), https://chrome.google.com/webstore/detail/power-saver/jfhfaediacibhobdfjaemhdgljfgemki

[67] Crockford, D.: Jsmin the javascript minifier (Apr 2003), http://crockford.com/javascript/jsmin

[68] Dhiraj: Image sprites â how to merge multiple images, and how to split them (Sep 2013), https://dhirajkumarsingh.wordpress.com/2012/06/24/image-sprites-how-to-merge-multiple-images-and-how-to-split-them/

[69] Etoh, M., Ohya, T., Nakayama, Y.: Energy consumption issues on mobile network systems. In: Applications and the Internet, 2008. SAINT 2008. International Symposium on. pp. 365–368. IEEE (2008)

[70] Evans, D.: The internet of things: How the next evolution of the internet is changing everything. CISCO white paper 1(2011), 1–11 (2011)

[71] Everts, T.: Rules for mobile performance optimization. Commun. ACM 56(8), 52–59 (Aug 2013), http://doi.acm.org/10.1145/2492007.2492024

[72] Everts, T.: The average web page is 3mb. how much should we care? (Aug 2017), https://speedcurve.com/blog/web-performance-page-bloat/

[73] Fainberg, L., Ehrlich, O., Shai, G., Gadish, O., Amitay, D., Berger, O.: Systems and methods for acceleration and optimization of web pages access by changing the order of resource loading (Feb 3 2011), uS Patent App. 12/848,559

[74] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol–http/1.1. Tech. rep. (1999)

[75] Firefox, M.: Save battery power, https://support.mozilla.org/tr/kb/save-battery-power#w_using-power-saver

[76] Firtman, M.: Mobile html5 (November 2017), http://mobilehtml5.org/

[77] Firtman, M.: Programming the mobile web. " O'Reilly Media, Inc." (2010)

[78] Flinn, J., Satyanarayanan, M.: Energy-aware adaptation for mobile applications. SIGOPS Oper. Syst. Rev. 33(5), 48–63 (Dec 1999), http://doi.acm.org/10.1145/319344.319155

[79] Frain, B.: Responsive web design with HTML5 and CSS3. Packt Publishing Ltd (2012)

[80] Giakki: giakki/uncss (Nov 2017), https://github.com/giakki/uncss

[81] Gochman, S., Mendelson, A., Naveh, A., Rotem, E.: Introduction to intel core duo processor architecture. Intel Technology Journal 10(2) (2006)

[82] Gruntjs: gruntjs/grunt-contrib-concat (Apr 2016), https://github.com/gruntjs/grunt-contrib-concat

[83] Haines-Young, M.: Minqueue, https://wordpress.org/plugins/minqueue/

[84] Huy, N.P., vanThanh, D.: Evaluation of mobile app paradigms. In: Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia. pp. 25–30. MoMM '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2428955.2428968

[85] Iaccarino, G., Malandrino, D., Scarano, V.: Personalizable edge services for web accessibility. In: Proceedings of the 2006 international cross-disciplinary workshop on Web accessibility (W4A): Building the mobile web: rediscovering accessibility? pp. 23–32. ACM (2006)

[86] (Intel), M.Y.: IntelÂ® power gadget (Jun 2016), https://software.intel.com/en-us/articles/intel-power-gadget-20

[87] Interactive, L.: Merge minify refresh, https://en-ca.wordpress.org/plugins/merge-minify-refresh/

[88] Jobe, W.: Native apps vs. mobile web apps. International Journal of Interactive Mobile Technologies 7(4) (2013)

[89] Jugo, I., Kermek, D., Meštrović, A.: Analysis and evaluation of web application performance enhancement techniques. In: International Conference on Web Engineering. pp. 40–56. Springer (2014)

[90] KaÅºmierczak, B.: Why we challenge microsoft's battery test (Jun 2016), https://www.opera.com/blogs/desktop/2016/06/over-the-edge/

[91] Koksal, E.: Twisting web pages for saving energy. In: International Wireless Communications Expo (IWCE) (2017)

[92] Kotelnytskyi, Y.: Js & css script optimizer, https://wordpress.org/plugins/js-css-script-optimizer/

[93] Lai, P.P.: Efficient and effective information finding on small screen devices. In: Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility. p. 4. ACM (2013)

[94] Lam, H., Baudisch, P.: Summary thumbnails: readable overviews for small screen web browsers. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 681–690. ACM (2005)

[95] Liu, Y., Guo, L.: An empirical study of video messaging services on smartphones. In: Proceedings of Network and Operating System Support on Digital Audio and Video Workshop. p. 79. ACM (2014)

[96] Lohr, S.: Impatient web users flee slow-loading sites (Feb 2012), http://www.nytimes.com/2012/03/01/technology/impatient-web-users-flee-slow-loading-sites.html

[97] Looper, J.: What is a webview? (Nov 2015), https://developer.telerik.com/featured/what-is-a-webview/

[98] Lose, T., Thinyane, M.: A transcoding proxy server for mobile web browsing (09 2011)

[99] Marilyn: Manage the safari power saver feature. Online (2013), http://mac-fusion.com/manage-the-safari-power-saver-feature/

[100] Matsudaira, K.: Making the mobile web faster. Commun. ACM 56(3), 56–61 (Mar 2013), http://doi.acm.org/10.1145/2428556.2428572

[101] McArthur, S.: Css remove and combine, https://chrome.google.com/webstore/detail/css-remove-and-combine/cdfmaaeapjmacolkojefhfollmphonoh?hl=en-GB

[102] Microsoft: Create a custom power plan. Online (2017), https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/create-a-custom-power-plan-technicalreference

[103] Miettinen, A.P., Nurminen, J.K.: Analysis of the Energy Consumption of JavaScript Based Mobile Web Applications, pp. 124–135. Springer Berlin Heidelberg, Berlin, Heidelberg (2010), https://doi.org/10.1007/978-3-642-16644-0_12

[104] Moth, D.: Slow-loading websites cost retailers Â£1.73bn in lost sales each year (May 2012), https://econsultancy.com/blog/9790-slow-loading-websites-cost-retailers-1-73bn-in-lost-sales-each-year

[105] Muhtaroglu, A., Yokochi, A., Von Jouanne, A.: Integration of thermoelectrics and photovoltaics as auxiliary power sources in mobile computing applications. Journal of Power Sources 177(1), 239–246 (2008)

[106] Mullie, M.: Minify, css and javascript minifier, https://www.minifier.org/

[107] Nagy, Z., Nakov, O., Vasek, V., Naaji, A.: Improved speed on intelligent web sites pp. 215–220 (01 2013)

[108] Naveh, A., Rotem, E., Mendelson, A., Gochman, S., Chabukswar, R., Krishnan, K., Kumar, A.: Power and thermal management in the intel core duo processor. Intel Technology Journal 10(2) (2006)

[109] Nicolaou, A.: Best practices on the move: Building web apps for mobile devices. Queue 11(6), 30 (2013)

[110] Norris, C.A., Soloway, E.: Learning and schooling in the age of mobilism. Educational Technology 51(6), 3 (2011)

[111] Nottingham, M.: Caching tutorial (May 2013), https://www.mnot.net/cache_docs/

[112] Oliveira, W., Oliveira, R., Castor, F.: A study on the energy consumption of android app development approaches. In: Proceedings of the 14th International Conference on Mining Software Repositories. pp. 42–52. MSR '17, IEEE Press, Piscataway, NJ, USA (2017), https://doi.org/10.1109/MSR.2017.66

[113] Opera: Battery saver. Online, http://www.opera.com/tr/computer/features/battery-saver

[114] Opera: Opera mini (December 2015), http://www.opera.com/tr/mobile/mini/android

[115] Paul, K., Kundu, T.K.: Android on mobile devices: An energy perspective. In: Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on. pp. 2421–2426. IEEE (2010)

[116] Pentikousis, K.: In search of energy-efficient mobile networking. IEEE Communications Magazine 48(1) (2010)

[117] Pérez-Castillo, R., Piattini, M.: Analyzing the harmful effect of god class refactoring on power consumption. IEEE software 31(3), 48–54 (2014)

[118] Perrucci, G.P., Fitzek, F.H.P., Sasso, G., Kellerer, W., Widmer, J.: On the impact of 2g and 3g network usage for mobile phones' battery life. In: 2009 European Wireless Conference. pp. 255–259 (May 2009)

[119] Perrucci, G.P., Fitzek, F.H.P., Widmer, J.: Survey on energy consumption entities on the smartphone platform. In: 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring). pp. 1–6 (May 2011)

[120] Rodriguez, A., Mateos, C., Zunino, A.: Improving scientific application execution on android mobile devices via code refactorings. Software: Practice and Experience 47(5), 763–796 (2017)

[121] Rosenbaum, R., Schumann, H., Tominski, C.: Presenting large graphical contents on mobile devices-performance issues (2004)

[122] Saccomani, P.: Native, web or hybrid apps? whats the difference? (Jan 2018), https://www.mobiloud.com/blog/native-web-or-hybrid-apps/

[123] Sahin, C., Wan, M., Tornquist, P., McKenna, R., Pearson, Z., Halfond, W.G., Clause, J.: How does code obfuscation impact energy usage? Journal of Software: Evolution and Process 28(7), 565–588 (2016)

[124] Sailhan, F., Issarny, V.: Energy-aware web caching for mobile terminals. In: Proceedings 22nd International Conference on Distributed Computing Systems Workshops. pp. 820–825 (2002)

[125] Sin, D., Lawson, E., Kannoorpatti, K.: Mobile web apps-the non-programmer's alternative to native applications. In: Human System Interactions (HSI), 2012 5th International Conference on. pp. 8–15. IEEE (2012)

[126] SLACK, R., GRONOW, J., VOULVOULIS, N.: Hazardous components of household waste. Critical Reviews in Environmental Science and Technology 34(5), 419–445 (2004), https://doi.org/10.1080/10643380490443272

[127] Slegg, J.: Google mobile first index: Page speed included as a ranking factor (Mar 2017), http://www.thesempost.com/google-mobile-first-index-page-speed-ranking/

[128] Song, R., Liu, H., Wen, J.R., Ma, W.Y.: Learning block importance models for web pages. In: Proceedings of the 13th international conference on World Wide Web. pp. 203–211. ACM (2004)

[129] Sorber, J., Banerjee, N., Corner, M.D., Rollins, S.: Turducken: Hierarchical power management for mobile devices. In: Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services. pp. 261–274. MobiSys '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1067170.1067198

[130] Souders, S.: High-performance web sites. Communications of the ACM 51(12), 36–41 (2008)

[131] Souders, S.: Even faster web sites: performance best practices for web developers. " O'Reilly Media, Inc." (2009)

[132] Takagi, H., Asakawa, C., Fukuda, K., Maeda, J.: Site-wide annotation: reconstructing existing pages to be accessible. In: Proceedings of the fifth international ACM conference on Assistive technologies. pp. 81–88. ACM (2002)

[133] Telekosmos: Remove duplicates from js array (es5/es6), https://gist.github.com/telekosmos/3b62a31a5c43f40849bb

[134] Thiagarajan, N., Aggarwal, G., Nicoara, A., Boneh, D., Singh, J.P.: Who killed my battery?: Analyzing mobile browser energy consumption. In: Proceedings of the 21st International Conference on World Wide Web. pp. 41–50. WWW '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2187836.2187843

[135] Thong, C.: A survey on internet content transcoding for universal access. Department of Computer Science, Kent State University (2003)

[136] Titcomb, J.: Mobile web usage overtakes desktop for first time (Nov 2016), http://www.telegraph.co.uk/technology/2016/11/01/mobile-web-usage-overtakes-desktop-for-first-time/

[137] Turkyilmaz, S., Kulah, H., Muhtaroglu, A.: A development tool for design and analysis of mems based em energy scavengers. In: Energy Aware Computing (ICEAC), 2010 International Conference on. pp. 1–2. IEEE (2010)

[138] Vitousek, P.M.: Beyond global warming: ecology and global change. Ecology 75(7), 1861–1876 (1994)

[139] Walsh, J.: Granule (May 2011), https://github.com/JonathanWalsh/Granule

[140] Weber, J.: Get more out of your battery with microsoft edge (Sep 2016), https://blogs.windows.com/windowsexperience/2016/06/20/more-battery-with-edge/#MCyQIXD4topvxSdz.97

[141] Weber, J.: Microsoft edge now gets even more out of your battery (Sep 2016), https://blogs.windows.com/windowsexperience/2016/09/15/edge-battery-anniversary-update/#C56OkPtxC3uPdrAy.97

[142] Wilke, C., Piechnick, C., Richly, S., Püschel, G., Götz, S., A, U.: Comparing mobile applications' energy consumption. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. pp. 1177–1179. SAC '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2480362.2480583

[143] Wong, R.: Verizon's 5g internet coming in 2018 could replace your home internet. Online (Nov 2017), https://mashable.com/2017/11/30/verizon-5g-home-internet-2018/#OOkz9bQ4Lqqz

[144] Work, S.: How loading time affects your bottom line, https://blog.kissmetrics.com/loading-time/

[145] X-Team: Dependency minification, https://wordpress.org/plugins/dependency-minification/

[146] Xgvargas: js-css-min-django (Apr 2014), https://github.com/xgvargas/js-css-min-django

[147] Xiao, Y., Kalyanaraman, R.S., Yla-Jaaski, A.: Energy consumption of mobile youtube: Quantitative measurement and analysis. In: 2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies. pp. 61–69 (Sept 2008)

[148] Yesilada, Y., Harper, S., Eraslan, S.: Experiential transcoding: an eyetracking approach. In: Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility. p. 30. ACM (2013)

[149] Yin, X., Lee, W.S.: Using link analysis to improve layout on mobile devices. In: Proceedings of the 13th international conference on World Wide Web. pp. 338–344. ACM (2004)

[150] Zakas, N.C.: The evolution of web development for mobile devices. Queue 11(2), 30:30–30:39 (Feb 2013), http://doi.acm.org/10.1145/2436696.2441756

[151] Zhu, Y., Halpern, M., Reddi, V.J.: The role of the cpu in energy-efficient mobile web browsing. IEEE Micro 35(1), 26–33 (Jan 2015)

APPENDIX A

SUMMARY OF THE GUIDELINES

Table A.1: Summary of the Guidelines [6, 28, 130, 131, 71, 100, 150]

	References						
Best practices	[6]	[28]	[130]	[131]	[71]	[100]	[150]
Adapt to the network connection					✓		
Add expires or cache control header	✓		✓	✓	✓		
Avoid 404s / bad requests	✓	✓					
Avoid and minimize iframes	✓			✓			
Avoid and minimize redirects	✓	✓	✓	✓	✓	✓	✓
Avoid CSS @import	✓	✓					
Avoid CSS expressions	✓	✓	✓	✓			
Avoid document.write		✓					
Avoid empty image src	✓						
Avoid filters	✓						
Combine external CSS and JavaScript		✓			✓		✓

Combine images with CSS scripts	✓	✓			✓	✓	
Configure Etags	✓		✓	✓			
Defer or split Javascript payload		✓		✓	✓		
Defer parsing of Javascript	✓	✓			✓		
Defer rendering below-the-fold content					✓		
Develop Smart Event Handlers	✓						
Do not block the UI thread				✓			
Do not scale images in HTML	✓	✓					
Embed resources in HTML for first time use					✓		
Enable Gzip compression	✓	✓	✓	✓	✓	✓	
Establish device profiles for APIs						✓	
Flush buffer / document early	✓			✓			
Inline scripts before CSS				✓			
Keep components under 25 KB	✓						
Leverage browser caching		✓			✓	✓	
Leverage local storage					✓	✓	
Leverage proxy caching		✓			✓	✓	
Load scripts asynchronously				✓			
Make ajax cacheable	✓		✓	✓			

Make favicon.ico small and cacheable	✓						
Make fewer HTTP requests	✓		✓	✓	✓	✓	✓
Make Javascript and CSS external	✓		✓	✓			
Make landing page redirects cacheable		✓					
Minify CSS and Javascript	✓	✓	✓	✓	✓		
Minify HTML		✓			✓		
Minimize DOM access	✓						
Minimize request size		✓					
Minimize uncompressed size				✓			
Optimize images	✓	✓		✓	✓		
Pack components into a multipart document	✓						
Parallelize downloads across domains	✓	✓		✓			
Postload components	✓						
Prefer asynchronous resources		✓					
Prefetch and cache data						✓	
Preload components	✓						
Put scripts at bottom	✓	✓	✓	✓			
Put stylesheets at top	✓	✓	✓	✓			
Reduce cookie size	✓					✓	
Reduce DNS lookups	✓	✓	✓	✓		✓	
Reduce the number of DOM elements	✓						

Remove duplicated scripts	✓		✓	✓	✓		
Remove unused CSS		✓					
Replace click events with touch events					✓		
Serve resources from a consistent URL		✓					
Serve static content drom a cookieless domain	✓	✓					
Simplify pages with HTML5 and CSS 3.0					✓		
Simplify and use efficient CSS selectors		✓		✓			
Specify a character set		✓					
Specify image dimensions		✓					
Support partial response and partial update						✓	
Support responsive images					✓	✓	
Support the SPDY protocol					✓	✓	
Use Content Delivery Network (CDN)	✓		✓	✓	✓		
Use CSS animations and CSS transitions instead of Javascript							✓
Use CSS instead of images						✓	✓
Use data URIs for images inline to minimize extra requests						✓	
Use GET for ajax requests	✓						

Use HTML5 server-sent events					✓		
Use HTTP pipelining						✓	
Use nonblocking I/O						✓	
Use the HTML5 Web Worker Spec for multithreading					✓		
Write efficient Javascript				✓			

APPENDIX B

MEASUREMENTS FOR EACH WEB PAGE OVER DESKTOP

Table B.1: Total Number of Requests/Responses, Total Size and the Load Time Comparison of Each Web Page over Desktop for Concatenation Service

		# of Requests		Total Size (KB)		Load Time (s)	
		Before	After	Before	After	Before	After
3	facebook.com	49	40	404	404	1.08	1.40
5	wikipedia.org	8	7	80.2	80.2	0.46	0.66
6	qq.com	122	113	1638.4	1638.4	3.47	3.77
9	yahoo.com	111	88	1126.4	1126.4	2.37	3.49
10	sohu.com	152	145	598	598	1.59	2.60
14	twitter.com	10	8	357	357	0.84	1.21
15	amazon.com	122	118	1024	1024	2.15	2.69
19	360.cn	153	140	3481.6	3481.6	7.19	7.12
43	imdb.com	96	88	1433.6	1433.6	2.83	3.24
45	microsoft.com	28	27	192	193	0.65	0.92
55	naver.com	80	75	1331.2	1331.2	2.76	2.06
56	office.com	32	24	349	349	0.88	0.92
59	wordpress.com	20	18	128	128	0.53	0.65
66	paypal.com	17	12	402	402	0.96	0.90
69	microsoftonline.com	18	17	180	180	0.64	0.84
72	stackoverflow.com	23	21	301	301	0.87	0.90
74	github.com	55	51	3072	3072	6.04	5.87
87	xinhuanet.com	365	329	2150.4	2150.4	4.58	4.59
89	roblox.com	46	38	1331.2	1331.2	2.09	2.30
100	quora.com	29	28	1126.4	1126.4	1.81	2.14

Table B.2: Total Number of Requests/Responses, Total Size and the Load Time Comparison of Each Web Page over Desktop for Minification Service

		# of Requests		Total Size (KB)		Load Time (s)	
		Before	After	Before	After	Before	After
3	facebook.com	49	49	404	330	1.08	3.84
4	baidu.com	11	11	115	115	0.33	0.60
5	wikipedia.org	8	8	80.2	80.2	0.46	0.72
6	qq.com	122	122	1638.4	1638.4	3.47	4.45
9	yahoo.com	111	111	1126.4	1024	2.37	6.24
10	sohu.com	152	152	598	593	1.59	4.29
14	twitter.com	12	12	357	326	0.84	3.59
15	amazon.com	122	122	1024	1024	2.15	2.77
19	360.cn	153	153	3481.6	3.31024	7.19	7.82
25	blogspot.com	61	61	1228.8	1126.4	2.55	2.96
32	alipay.com	12	12	720	720	1.44	1.74
43	imdb.com	96	96	1433.6	1228.8	2.83	4.44
45	microsoft.com	28	28	192	192	0.65	4.11
52	tumblr.com	64	64	1638.4	1536	2.94	8.27
55	naver.com	80	80	1331.2	863	2.76	3.24
56	office.com	32	32	349	205	0.88	2.68
59	wordpress.com	22	22	128	93.1	0.53	1.49
66	paypal.com	19	19	402	216	0.96	1.75
69	microsoftonline.com	18	18	180	180	0.64	1.73
70	soso.com	8	8	26.3	26.3	0.16	0.39
72	stackoverflow.com	23	23	301	195	0.87	3.19
74	github.com	55	55	3072	2764.8	6.04	7.84
89	roblox.com	46	46	1331.2	928	2.09	4.15
95	bbc.com	53	53	203	195	0.76	2.42
100	quora.com	29	29	1126.4	985	1.81	9.43

Table B.3: Total Number of Requests/Responses, Total Size and the Load Time Comparison of Each Web Page over Desktop for Concatenation+Minification Service

		# of Requests		Total Size (KB)		Load Time (s)	
		Before	After	Before	After	Before	After
3	facebook.com	49	40	404	322	1.08	2.86
5	wikipedia.org	8	7	80.2	80.2	0.46	0.68
6	qq.com	122	113	1638.4	1638.4	3.47	4.3
9	yahoo.com	111	88	1126.4	1024	2.37	5.58
10	sohu.com	152	145	598	592	1.59	5.1
14	twitter.com	10	8	357	321	0.84	3.7
15	amazon.com	122	118	1024	1024	2.15	2.74
19	360.cn	153	140	3481.6	3379.2	7.19	7.42
43	imdb.com	96	88	1433.6	1228.8	2.83	4.63
45	microsoft.com	28	27	192	176	0.65	1
55	naver.com	80	75	1331.2	866	2.76	2.99
56	office.com	32	24	349	200	0.88	2.16
59	wordpress.com	20	18	128	93.5	0.53	1.05
66	paypal.com	17	12	402	202	0.96	1.7
69	microsoftonline.com	18	17	180	180	0.64	1.15
72	stackoverflow.com	23	21	301	194	0.87	3.28
74	github.com	55	51	3072	2764.8	6.04	7.77
87	xinhuanet.com	365	329	2150.4	1945.6	4.58	4.94
89	roblox.com	46	38	1331.2	915	2.09	4.01
100	quora.com	29	28	1126.4	1126.4	1.81	2.78

Table B.4: Cumulative Processor Energy (Joules), Cumulative Processor Energy (mWh) and Average Processor Power (Watt) Comparison of Each Web Page over Desktop for Concatenation and Concatenation+Minification Service

		Cumulative Processor Energy (Joules)			Cumulative Processor Energy (mWh)			Average Processor Power (Watt)		
		Before	Conc	Conc+Min	Before	Conc	Conc+Min	Before	Conc	Conc+Min
3	facebook.com	121.24	99.14	108.47	33.68	27.54	30.13	3.98	3.27	3.64
5	wikipedia.org	88.75	78.25	71.30	24.65	23.14	21.00	2.93	2.41	2.15
6	qq.com	133.72	120.52	129.96	37.15	30.13	36.10	4.47	3.99	4.20
9	yahoo.com	136.42	121.34	132.06	37.89	33.68	35.90	4.44	3.97	4.38
10	sohu.com	124.88	120.38	122.12	34.69	33.43	33.52	4.16	3.82	3.96
14	twitter.com	87.27	82.75	83.00	24.24	22.13	22.01	2.96	2.85	2.87
15	amazon.com	129.81	122.85	128.51	36.06	34.12	35.97	4.26	3.95	4.20
19	360.cn	154.66	139.75	120.28	42.96	31.36	26.79	5.22	4.55	4.07
43	imdb.com	135.92	108.00	117.20	37.75	31.36	31.61	4.46	3.72	3.74
45	microsoft.com	115.68	103.78	107.62	32.13	28.83	29.90	3.82	3.58	3.56
55	naver.com	139.02	101.31	108.00	38.61	28.14	30.02	4.54	3.34	3.40
56	office.com	93.48	91.62	92.01	25.98	25.45	26.01	3.07	3.02	3.03
59	wordpress.com	100.52	93.59	95.07	27.92	25.99	26.40	3.29	2.75	3.10
66	paypal.com	95.88	94.91	88.02	26.63	26.36	24.45	3.18	3.13	2.92
69	microsoftonline.com	107.96	101.83	97.63	29.99	28.28	27.12	3.39	3.38	3.20
72	stackoverflow.com	99.79	94.05	96.39	27.72	26.12	26.78	3.27	3.11	3.20
74	github.com	151.18	137.25	146.41	41.99	38.12	40.67	5.08	4.57	4.81
89	roblox.com	140.06	111.96	126.81	38.91	31.10	35.22	4.62	3.68	4.21
100	quora.com	113.92	100.80	101.14	31.65	28.03	28.74	3.75	3.30	3.50

Table B.5: Cumulative IA Energy (Joules), Cumulative IA Energy (mWh) and Average IA Power (Watt) Comparison of Each Web Page over Desktop for Concatenation and Concatenation+Minification Service

		Cumulative IA Energy (Joules)			Cumulative IA Energy (mWh)			Average IA Power (Watt)		
		Before	Conc	Conc+Min	Before	Conc	Conc+Min	Before	Conc	Conc+Min
3	facebook.com	39.25	25.89	30.25	10.90	7.19	8.41	1.29	0.86	1.02
5	wikipedia.org	20.51	18.82	15.82	5.70	5.23	5.19	0.68	0.62	0.61
6	qq.com	48.85	41.78	44.45	13.57	11.60	12.80	1.63	1.38	1.56
9	yahoo.com	52.54	46.32	50.80	14.59	12.44	13.85	1.71	1.48	1.67
10	sohu.com	43.82	41.28	41.70	12.17	11.00	10.98	1.46	1.38	1.40
14	twitter.com	21.65	19.78	21.70	6.01	5.50	5.71	0.74	0.64	0.64
15	amazon.com	46.19	42.24	45.77	12.83	11.20	11.80	1.51	1.35	1.46
19	360.cn	59.99	45.05	36.10	16.67	12.51	10.03	1.71	1.47	1.22
43	imdb.com	56.40	40.31	42.54	15.67	11.19	12.12	1.85	1.34	1.51
45	microsoft.com	37.03	29.47	34.28	10.29	2.18	5.74	1.22	1.02	1.17
55	naver.com	52.30	32.72	37.10	14.53	9.09	10.33	1.71	1.08	1.25
56	office.com	22.34	25.02	26.00	6.45	6.30	6.40	0.77	0.70	0.71
59	wordpress.com	25.71	19.91	28.74	7.14	5.55	7.12	0.84	0.64	0.85
66	paypal.com	25.80	22.90	18.64	7.17	6.36	5.18	0.87	0.76	0.62
69	microsoftonline.com	30.31	27.80	24.04	8.42	7.72	6.68	0.95	0.92	0.79
72	stackoverflow.com	26.78	22.93	23.07	7.44	3.37	5.55	0.88	0.75	0.76
74	github.com	60.69	49.96	51.63	16.85	13.88	14.34	2.04	1.67	1.70
89	roblox.com	50.34	35.20	39.80	13.98	9.77	11.06	1.66	1.16	1.32
100	quora.com	36.08	30.77	35.32	10.02	8.54	9.40	1.19	1.01	1.11

Table B.6: Cumulative Processor Energy (Joules), Cumulative Processor Energy (mWh) and Average Processor Power (Watt) Comparison of Each Web Page over Desktop for Minification Service

		Cumulative Processor Energy (Joules)		Cumulative Processor Energy (mWh)		Average Processor Power (Watt)	
		Before	Min	Before	Min	Before	Min
3	facebook.com	121.24	115.79	33.68	32.16	3.98	3.83
4	baidu.com	85.40	86.21	23.72	23.74	2.80	2.87
5	wikipedia.org	88.75	88.90	24.65	24.74	2.93	2.88
6	qq.com	133.72	134.00	37.15	37.20	4.47	4.47
9	yahoo.com	136.42	130.27	37.89	36.10	4.44	4.40
10	sohu.com	124.88	121.50	34.69	33.50	4.16	3.90
14	twitter.com	87.27	83.40	24.24	23.32	2.96	2.87
15	amazon.com	129.81	129.82	36.06	36.08	4.26	4.26
19	360.cn	154.66	143.05	42.96	31.80	5.22	4.67
25	blogspot.com	126.10	121.00	35.03	34.07	4.19	4.02
32	alipay.com	101.59	93.05	28.22	25.84	3.34	3.05
43	imdb.com	135.92	112.00	37.75	30.16	4.46	3.61
45	microsoft.com	115.68	116.80	32.13	34.90	3.82	4.17
52	tumblr.com	123.12	116.00	34.20	27.40	4.08	3.24
55	naver.com	139.02	116.43	38.61	32.35	4.54	3.80
56	office.com	93.48	91.96	25.98	25.54	3.07	3.03
59	wordpress.com	100.52	94.28	27.92	26.19	3.29	3.13
66	paypal.com	95.88	94.17	26.63	26.16	3.18	3.09
69	microsoftonline.com	107.96	98.26	29.99	27.29	3.39	3.21
70	soso.com	98.41	83.48	27.34	23.19	3.27	2.80
72	stackoverflow.com	99.79	97.07	27.72	26.96	3.27	3.23
74	github.com	151.18	145.63	41.99	40.45	5.08	4.85
89	roblox.com	140.06	114.21	38.91	31.72	4.62	3.78
95	bbc.com	93.73	92.00	26.04	25.20	3.08	3.05
100	quora.com	113.92	100.82	31.65	28.10	3.75	3.33

Table B.7: Cumulative IA Energy (Joules), Cumulative IA Energy (mWh) and Average IA Power (Watt) Comparison of Each Web Page over Desktop for Minification Service

		Cumulative IA Energy (Joules)		Cumulative IA Energy (mWh)		Average IA Power (Watt)	
		Before	Min	Before	Min	Before	Min
3	facebook.com	39.25	34.86	10.90	9.68	1.29	1.15
4	baidu.com	18.19	16.56	5.05	4.60	0.60	0.55
5	wikipedia.org	20.51	19.54	5.70	5.71	0.68	0.69
6	qq.com	48.85	48.86	13.57	13.60	1.63	1.63
9	yahoo.com	52.54	51.10	14.59	14.01	1.71	1.68
10	sohu.com	43.82	41.27	12.17	10.91	1.46	1.38
14	twitter.com	21.65	21.50	6.01	5.78	0.74	0.66
15	amazon.com	46.19	46.20	12.83	12.80	1.51	1.50
19	360.cn	59.99	49.65	16.67	13.80	1.71	1.63
25	blogspot.com	43.50	42.21	12.22	11.56	1.46	1.34
32	alipay.com	27.68	20.52	7.69	5.70	0.91	0.67
43	imdb.com	56.40	43.13	15.67	11.97	1.85	1.42
45	microsoft.com	37.03	37.72	10.29	11.14	1.22	1.31
52	tumblr.com	44.78	35.26	12.44	8.19	1.48	1.23
55	naver.com	52.30	39.96	14.53	11.10	1.71	1.31
56	office.com	22.34	24.75	6.45	6.41	0.77	0.69
59	wordpress.com	25.71	25.80	7.14	7.00	0.84	0.82
66	paypal.com	25.80	21.98	7.17	6.10	0.87	0.72
69	microsoftonline.com	30.31	24.18	8.42	6.71	0.95	0.80
70	soso.com	26.02	15.49	7.23	4.30	0.86	0.52
72	stackoverflow.com	26.78	23.80	7.44	6.61	0.88	0.79
74	github.com	60.69	53.97	16.85	14.99	2.04	1.80
89	roblox.com	50.34	34.95	13.98	9.71	1.66	1.16
95	bbc.com	26.09	25.98	7.24	7.20	0.92	0.89
100	quora.com	36.08	27.72	10.02	7.70	1.19	0.91

APPENDIX C

MEASUREMENTS FOR EACH WEB PAGE OVER MOBILE

Table C.1: Average Power (mW) Comparison of Each Web Page over Mobile for Concatenation and Concatenation+Minification Service

		No Transcoding (mW)	Concatenation (mW)	Concatenation+Minification (mW)
3	facebook.com	655	615	634
5	wikipedia.org	638	618	606
6	qq.com	663	633	645
9	yahoo.com	663	634	659
10	sohu.com	658	638	645
14	twitter.com	638	634	634
15	amazon.com	660	641	656
19	360.cn	676	629	589
43	imdb.com	663	617	615
45	microsoft.com	652	640	637
55	naver.com	664	589	587
56	office.com	640	638	638
59	wordpress.com	644	619	634
66	paypal.com	642	640	630
69	microsoftonline.com	645	645	636
72	stackoverflow.com	643	636	640
74	github.com	673	638	653
89	roblox.com	666	606	637
100	quora.com	651	628	637

Table C.2: Average Power (mW) Comparison of Each Web Page over Mobile for Minification Service

		No Transcoding (mW)	Minification (mW)
3	facebook.com	655	645
4	baidu.com	635	638
5	wikipedia.org	638	635
6	qq.com	663	663
9	yahoo.com	663	660
10	sohu.com	658	640
14	twitter.com	638	634
15	amazon.com	660	660
19	360.cn	676	631
25	blogspot.com	658	647
32	alipay.com	644	629
43	imdb.com	663	602
45	microsoft.com	652	674
52	tumblr.com	657	601
55	naver.com	664	610
56	office.com	640	638
59	wordpress.com	644	635
66	paypal.com	642	637
69	microsoftonline.com	645	635
70	soso.com	643	619
72	stackoverflow.com	643	641
74	github.com	673	655
89	roblox.com	666	603
95	bbc.com	640	639
100	quora.com	651	626

APPENDIX D

MEASUREMENT INPUTS FOR THE MODEL AND THE RESULT FOR NUMBER OF CLIENTS FOR EACH WEB SITE

Table D.1: Measurement Inputs for the Model and the Result for Number of Clients for Each Web Site

	Energy without Transcoding (mWh)				**Energy with Transcoding and the Source is Already Cached (mWh)**				**Energy during First Request for Transcoding (mWh)**				
	Client	**Proxy**	**Server**	**Total (B)**	**Client**	**Proxy**	**Server**	**Total (A)**	**Client**	**Proxy**	**Server**	**Total (C)**	**n**
facebook.com	33.167	0.000	21.342	54.508	24.767	23.417	0.000	48.183	38.925	42.500	21.342	102.767	17
wikipedia.org	24.417	0.000	21.850	46.267	19.833	24.192	0.000	44.025	21.058	32.558	21.850	75.467	34
qq.com	37.250	0.000	20.567	57.817	26.642	23.031	0.000	49.673	49.942	46.208	20.567	116.717	15
yahoo.com	37.000	0.000	20.350	57.350	27.950	22.638	0.000	50.588	66.167	44.600	20.350	131.117	20
sohu.com	34.667	0.000	20.483	55.150	23.225	22.714	0.000	45.939	63.667	44.092	20.483	128.242	14
amazon.com	35.500	0.000	18.308	53.808	24.733	20.854	0.000	45.587	65.833	41.492	18.308	125.633	16
360.cn	43.500	0.000	18.283	61.783	25.058	20.802	0.000	45.860	75.833	50.792	18.283	144.908	10
microsoft.com	31.833	0.000	20.642	52.475	27.783	23.041	0.000	50.824	59.667	37.875	20.642	118.183	72
naver.com	37.833	0.000	21.425	59.258	26.783	23.855	0.000	50.638	55.667	45.225	21.425	122.317	15
wordpress.com	27.417	0.000	18.383	45.800	21.175	20.842	0.000	42.017	45.833	32.783	18.383	97.000	26
paypal.com	26.500	0.000	17.392	43.892	20.133	19.932	0.000	40.065	52.167	33.683	17.392	103.242	27
microsoftonline.com	28.250	0.000	22.225	50.475	23.933	24.660	0.000	48.593	56.333	34.842	22.225	113.400	61
stackoverflow.com	27.250	0.000	19.650	46.900	24.608	22.111	0.000	46.719	51.833	34.425	19.650	105.908	585
github.com	42.333	0.000	20.375	62.708	23.975	22.769	0.000	46.744	76.167	51.033	20.375	147.575	10
roblox.com	38.500	0.000	18.233	56.733	26.975	20.594	0.000	47.569	61.333	45.700	18.233	125.267	14
quora.com	31.250	0.000	20.083	51.333	23.317	22.408	0.000	45.725	55.000	37.600	20.083	112.683	21

TEZ FOTOKOPİSİ İZİN FORMU

PROGRAM

SEES ☐

PSIR ☐

ELT ☐

YAZARIN

Soyadı: Ünlü

Adı: Hüseyin

Bölümü: Sürdürülebilir Çevre ve Enerji Sistemleri

TEZİN ADI (İngilizce) : Energy Efficient Mobile Web via Scripts&Stylesheets Based Transcoding

TEZİN TÜRÜ: Yüksek Lisans ☐ Doktora ☐

1. Tezimin tamamından kaynak gösterilmek şartıyla fotokopi alınabilir. ☐
2. Tezimin içindekiler sayfası, özet, indeks sayfalarından ve/veya bir bölümünden kaynak gösterilmek şartıyla fotokopi alınabilir. ☐
3. Tezimden bir bir (1) yıl süreyle fotokopi alınamaz. ☐

TEZİN KÜTÜPHANEYE TESLİM TARİHİ: