TRANSCODING WEB PAGES FOR ENERGY SAVING ON THE CLIENT-SIDE

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

BY

EDA KÖKSAL AHMED

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

FEBRUARY 2016

Approval of the Board of Graduate Programs

Prof. Dr. M. Tanju Mehmetoğlu
Chairperson

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

Assoc. Prof. Dr. Ali Muhtaroğlu
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.

Assist. Prof. Dr. Yeliz Yeşilada Yılmaz
Supervisor

Examining Committee Members

Assist. Prof. Dr. Yeliz Yeşilada Yılmaz	Computer Engineering Prog. METU NCC	
Assoc. Prof. Dr. Ali Muhtaroğlu	Electrical and Electronics Engineering Prog. METU NCC	
Dr. Simon Harper	School of Computer Science The University of Manchester	

ABSTRACT

TRANSCODING WEB PAGES FOR ENERGY SAVING ON THE CLIENT-SIDE

Ahmed, Eda Köksal
M.Sc., Department of Sustainable Environment and Energy Systems Program
Supervisor : Assist. Prof. Dr. Yeliz Yeşilada Yılmaz

February 2016, 120 pages

Mobile devices are essential to everyday life in order to have access to information everywhere. However, browsing web pages can be inconvenient on these devices because of the various constraints, such as limited battery size, processing power and device memory. As a result of these limitations, while browsing a web page, the battery of the mobile devices is drained. To solve this inconvenience and save energy while browsing, this thesis presents transcoding techniques that can be applied to web pages without modifying the look&feel of the web pages and without adding extra load on the client or the server. These techniques are implemented as part of our system called Twes+ (**T**ranscoding **W**eb Pages for **E**nergy **S**aving). The software architecture of Twes+ includes two main services: (1) Redirect transcoding and (2) image transcoding. This thesis first explains Twes+ in depth and then presents its evaluation. The evaluation results show that there is a statistically significant energy saving with Twes+. Finally, this thesis also discusses limitations and the future work.

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

ÖZ

WEB SAYFALARININ KULLANICI TARAFINDA ENERJİ TASARRUFU İÇİN DÖNÜŞTÜRÜLMESİ

Ahmed, Eda Köksal
Yüksek Lisans, Sürdürülebilir Çevre ve Enerji Sistemleri Programı
Tez Yöneticisi : Yrd. Doc. Dr. Yeliz Yeşilada Yılmaz

Şubat 2016, 120 sayfa

Mobil cihazlar günlük hayattaki birincil cihazlar olmaya başladı. Ancak mobil cihazlar bazı kısıtlamar sebebiyle internete erişim anlamında pratik olamayabilirler. Bu kısıtlamalar batarya boyutları,işlemci güçleri ve bellek boyutu... Bu kısıtlamaların sonucu olarak, web sayfalarına erişim esnasında mobil cihazların şarjları tükenir. Bu çalışmada sayfanın görünümünü değiştirmeden ve kullanıcıya ya da sunucuya artı yük vermeden ve bu kısıtlamaları çözmek ve internete erişim esnasında enerji tasarrufu sağlamak için web sitelerini dönüştüren teknikler sunuyoruz. Bu teknikler Twes+ adı verdiğimiz sistemimizde uygulandi. Twes+'in yazılım mimarisinde 2 servis: (1) yönlendirme dönüşümü (2) resim dönüşümü bulunur. İlk olarak Twes+'i detaylı olarak anlatıp ve devamında da değerlendirmesini anlatacağız. Değerlendirme sonuçlarına göre Twes+ kullanılarak istatiksel olarak kayda değer bir enerji tasarrufu sağlanabiliyor.

Anahtar Kelimeler: Dönüştürme, enerji tasarrufu, Tarama, Mobil Web, Web mühendisliği

To my parents, Aysın and Baki Köksal,
Without their support, advising and sacrifices none of my success would be possible

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor Yeliz Yeşilada for all her full support, her mentorship, understanding, and encouragement. Without her supervision, my thesis work would have been a frustrating pursuit. I owe her a lot of my learnings during my bachelor studies and throughout my master.

Beside my advisor, I would like to thank the examining committee members: Ali Muhtaroğlu and Simon Harper for their precious time and effort reading my thesis and for their valuable comments.

My special thanks to Ali Muhtaroğlu and my supervisor Yeliz Yeşilada, who helped me to go to an internship at Intel as I learned a lot in this internship that helped me during this study.

Thanks also go to my colleague Şükrü Eraslan, who helped me to improve this study with his valuable feedback. I would also like to thank Enver Ever for his assistance with the network of Twes+.

Finally, I would like to thank specially to my father Baki Köksal for his guidance with the technical setup of Twes+. To my family Baki Köksal, Aysın Köksal, Ali Köksal and Sefa Köksal and my husband Karim Ali Ahmed, I am extremely grateful for helping, comforting, motivating, understanding and encouraging me during my study and not letting me give up.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

3G	Third-Generation Mobile Network
ACL	Access control lists
ACPI	Advanced Configuration & Power Interface
CDN	Content Delivery Network
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DNS	Domain Name System
DOM	The Document Object Model
FOM	Function-based Object Model
GIF	Graphics Interchange Format
GPS	Global Positioning System
GPU	Graphics Processor Unit
GSM	Global System for Mobile Communications
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IA	Intel Architecture
IBM	International Business Machines Corporation
ICAP	Internet Content Adaptation Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
JIT	Just-in-time compiler
JPEG/JPG	Joint Photographic Experts Group
JS	JavaScript
JVM	Java Virtual Machine
LTE	Long-Term Evolution
MEMS	Microelectromechanical systems

PAN	Personalizable Accessible Navigation
PNG	Portable Network Graphics
RWD	Responsive Web Design
SPDY	SPeeDY Protocol
SSD	Small Screen Devices
TCP/IP	Transmission Control Protocol/Internet Protocol
TDP	Thermal Design Power
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAI	Web Accessibility Initiative
WCAG	Web Content Accessibility Guidelines
WWW	World Wide Web

CHAPTER 1

INTRODUCTION

The World Wide Web (WWW) has been a major source of information and it was accessed from the desktop browsers, but nowadays the popularity of the mobile devices is increasing. Mobile devices are becoming a big part of our life. However, these devices have some constraints when they are compared with the desktops. As a result of these constraints, the user is facing battery limitation while browsing a web page. The aim of this research is to reduce the power consumption and improve the battery life of mobile devices while browsing.

1.1 Motivation and Problem Statement

The WWW (the web) was only accessed by people via the desktop machines. By the time, the first mobile device (the forerunner of the modern smartphone named Simon produced by IBM) went on sale August 16, 1994 [60], and a new category of technology was started with this first mobile device. The users have then started to access the web via these mobile devices. There are some statistics that show the importance of mobile devices in our life. The number of the mobile devices ($\tilde{4}$.8 billion) is approximately same as the number of people who has access to drinkable water [62]. The result of another survey conducted between 2006-2015 indicates that ownership of mobile devices is increasing rapidly, see Figure 1.1 [39]. The similar survey conducted in the Republic of Turkey shows the number of mobile devices increased from 81 thousand to 72 million from 1994 to 2015 [38].

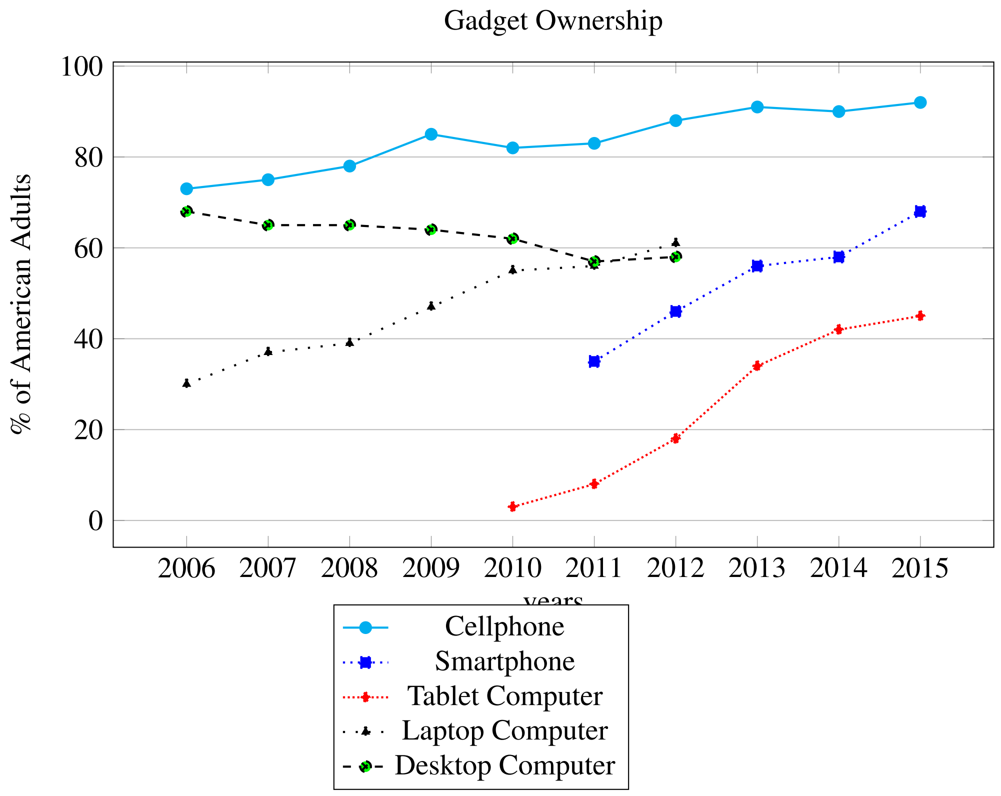


Figure 1.1: Gadget Ownership, Pew Internet Surveys 2010-2015 [39]

The previous forecast of Mary Meeker, the leader of the Global Tech Research Team of Morgan Stanley, was ”we believe more users may connect to the Internet via mobile devices than desktop PCs within five years” in 2008, see Figure 1.2 [54] and this forecast became true.

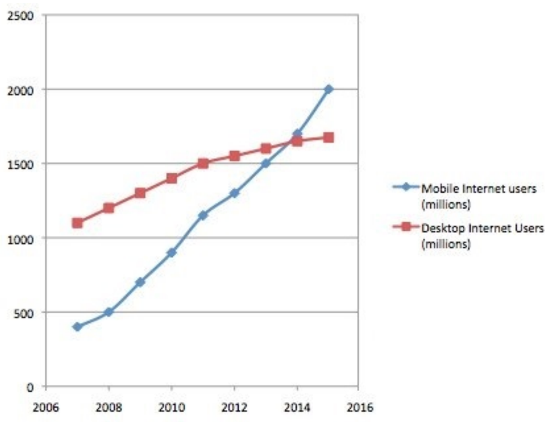


Figure 1.2: Mobile Internet Users versus Desktop Internet Users [62]

The number of people that access the web via traditional devices also dropped from 79.18% to 69.91% between the second quarter of 2013 and the second quarter of 2014 (see Table 1.1) [70]. Moreover, the number of people that access the web via smartphones increased from 8.66% to 16.25% during the same duration. However, tablets do not show a constant trend, generally, it increased from 12.16% to 13.84%. As a result, mobile devices are becoming primary devices to reach the Internet and the number of mobile devices in the market is increasing [37].

Table 1.1: Web Site Visits by Devices (Q:quarter) [57]

Web Sites visits by devices	**Q2 2013**	**Q3 2013**	**Q4 2013**	**Q1 2014**	**Q2 2014**
Traditional	79.18%	76.20%	73.42%	72.50%	69.91%
Tablet	12.16%	13.30%	14.69%	15.14%	13.84%
Smartphone	8.66%	10.50%	11.89%	12.36%	16.25%

Another research about the number of connected devices versus the population increase shows that in 2015 the ratio reaches to 3.47 and the expected ratio for 2020 is 6.58 (see Table 1.2) [17]. The number of Internet subscribers has also started to increase from 1998 and it has changed from 229 thousand to 42 million in the Republic of Turkey (see Figure 1.3) [38].

Table 1.2: The Number of Connected Devices versus the Population [17]

	2003	**2010**	**2015**	**2020**
World Population (Billion)	6.3	6.8	7.2	7.6
Connected Devices (Billion)	0.5	12.5	25	50
Connected Devices per Person	0.08	1.84	3.47	6.58

At the same time, computers, web pages, and network technologies have also been improved. Current computers have a powerful processing power, almost endless memory, big screens, power access, fast network connections. Web pages are getting bigger to make the user experience better and the quality of the images and videos are increasing, etc. According to HTTP Archive, the average web page size is more than 1MB and it contains 80 resources. Expected size according to the current rate of growth was 2MB by 2015 and now current average web page size is more than 2MB. There is a trending increase in the average web page size and the average total number of requests required to load a web page. The average web page size increased from

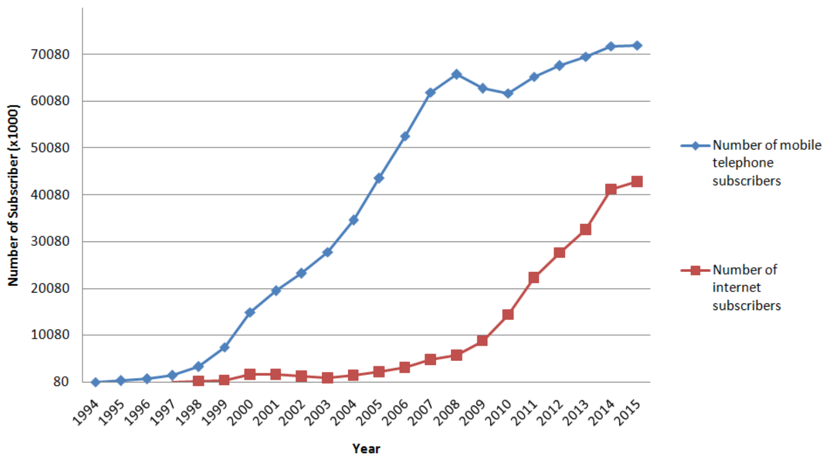
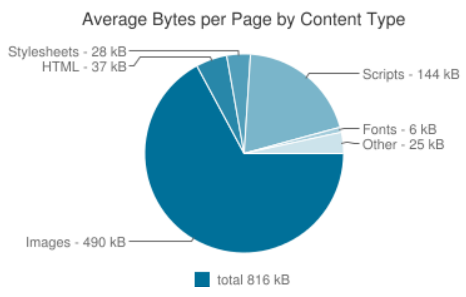


Figure 1.3: The Number of Internet Subscriber and the Mobile Devices 1994-2015 [38]

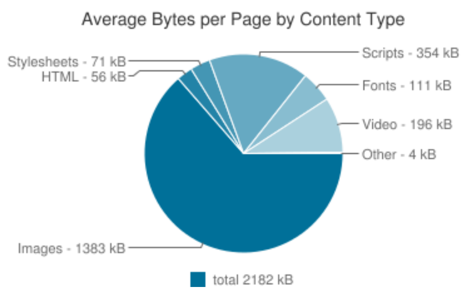
831kB to 2135kB between Sept 1st, 2012 and Sept 1st, 2015 and the total number of requests increased from 86 to 101 during the same period [26, 4].

Over the years, mobile devices have been also improved and now mobile devices are powerful as an average desktop [95]. Although the expectation of users from this new technology is better performance, the user experience is still not sufficiently good as expected. If these new technologies are used in ideal condition with an average web page, they might give better results. However, the web and the Internet connection have also improved. Web pages and applications are larger and they need more resources. As a result, there are more requests and responses in the traffic. Figure 1.4 illustrates the comparison of the content distribution of web pages between 2011 and 2015.

Mobile devices are able to connect to the Internet through a wireless network or a cellular network. The delays, the throughput and the power consumption comparison have been done between the cellular network and three different scheduling algorithms (IEEE 802.11, Ideal Link Scheduling, and Ideal Flow Scheduling) for the ad-hoc wireless networks. Figure 1.5a shows the mean end-to-end throughput of the four networks. As it can be seen from the figure, the throughput of ad-hoc wireless networks is higher than the cellular network for all power levels [36]. Fig-



(a) Content Distribution September 2011



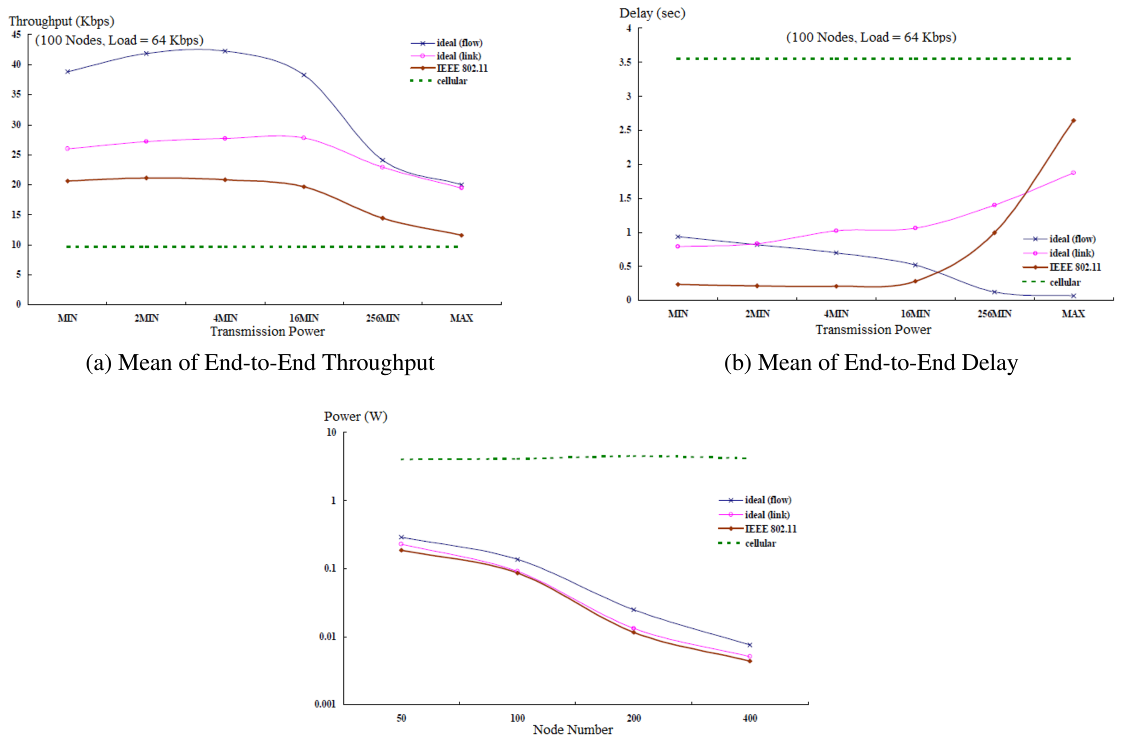
(b) Content Distribution September 2015

Figure 1.4: Comparison of Regular Images and Responsive Images Usage over Different Devices [4].

ure 1.5b shows the mean end-to-end delay for the four networks and again for all power levels, the cellular network performs worse than other three ad-hoc wireless networks [36]. Figure 1.5c shows the mean of power consumption of the four networks with different node numbers. In this figure, the results are the same as previous graphs. The cellular network consumes 18 times more power than others [36]. While the results show the possible power consumption of the connection with the cellular network for browsing, it is used by a large number of people [36].

To sum up, when the user browses a web page, the battery of the device drains and browsing web pages on mobile devices becomes inconvenient. The reason of these drawbacks is the various constraints of mobile devices such as battery size, the bandwidth of the connection, processing power, limited device memory, the smaller number of simultaneous connections, special browsers, and screen size [86, 26, 52]. The constraints of the mobile devices cause more energy consumption while browsing. Therefore, the user typically can not get the expected result from their experience over Internet [95].

Large-scale web pages improved back-end (server side) to answer millions of requests without latency (delay). Although it might solve the issue for the desktops, there are still delay and results opposite to the client expectations for some users. The reason is the limitations of front-end (client side). For example, when the user requests *'www.google.com/ig'*, the user waits for



(a) Mean of End-to-End Throughput

(b) Mean of End-to-End Delay

(c) Mean of End-to-End Power Consumption

Figure 1.5: The Delay, the Throughput and the Power Consumption of Different Networks [36].

the page to be loaded. The 9% of page loading duration is passed on the back-end and 91% of page loading duration is passed on the front-end [26, 75]. Mobile devices are not similar to laptops or desktops, they have some limitations as mentioned earlier. As a result, the developer has to consider who is accessing their applications and what device is used and the device limitations [95].

The number of connections is one of the limitations as mentioned before. In 1999, by Internet Engineering Task Force (IETF) in HTTP/1.1 Protocol, the number of simultaneous connections has been decided to prevent overloading web servers and Internet congestion [28]. By using different techniques, this limit is changed, so the browser can load the web page faster and it means the reduction in latency [76, 41]. As a reason for this change, computer browsers can send many requests because they do not have limits for simultaneous connections. However, mobile devices still have limited simultaneous connection. It means that if an HTML page needs lots of requests, this creates the latency issue [52].

The size of the mobile devices is much smaller than the desktop computers (see Figure 1.6). The lack of space leads to the lack of processing power because it is impossible to include similar hardware of desktop computers inside a mobile device for now. This makes the mobile devices slower and their performance is lower than the desktop computers [71].

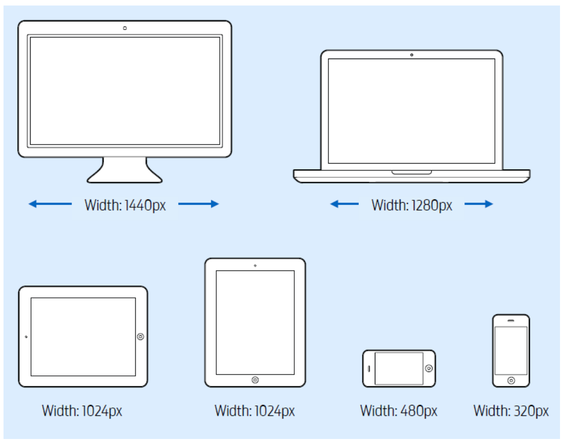


Figure 1.6: The Size Comparison of Devices [50]

Mobile devices are connected to the Internet via Wi-Fi or cellular connection. These connections are slower than the wired connection. As described before, connecting to Wi-Fi or cellular network consumes more power and if a web page loads slowly and requires more request/response round trips, it drains the battery [52].

JavaScript Engine performance can vary between different type of browsers and the platforms. JavaScript engine of the browsers over mobile devices is slower than desktop or laptop version of it (see Figure 1.7). The important point is that JavaScript engine performance has cost. Code execution is done over CPU. CPU uses power and processing power of mobile devices is slower and weaker than the processing power of computers. The time during the execution of code is directly proportional to use of power. As a conclusion, executing JavaScript code drains the battery faster [95]. Older mobile devices parse and execute a web page 100 times slower than desktops and new mobile devices are ten times slower than desktops [61].

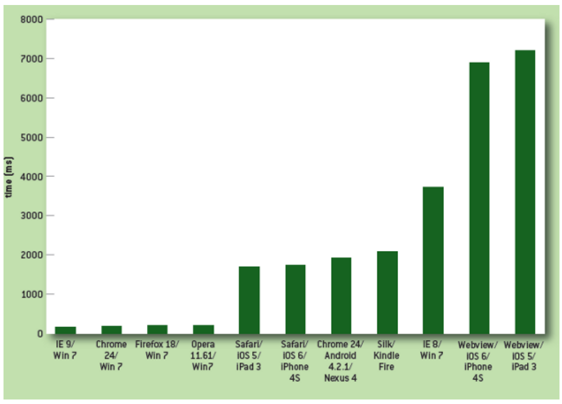


Figure 1.7: JavaScript Engine Performance Comparison of Browsers [95]

Less Memory capacity of mobile devices is another constraint. Images cause big concern about memory because of their sizes. Images that are loaded into the DOM (objects are represented as a tree). If the number of images is so much, it might cause a crash or slow experience because of the limited memory [95].

To improve the user experience, web page content is increased and web pages are designed for large screens of desktop computers. However, the screen size of the mobile devices is much smaller than computers, see Figure 1.6. It means less space to display the data and images. The content of the web page is requested and rendered but as there is not enough space to display, the client is not able to see the web page as it should be. To overcome this constraint, some web pages designed the mobile version but still the user can access the desktop version of the same web pages.

According to the research on online e-commerce sites, accepted latency for a web page response time is around two or three seconds. Two seconds is the threshold for acceptable web page response and 40% of users close the web page if the web page response takes longer than three seconds to load. Mobile devices are becoming primary devices for users and expected experience is equal to or better than a desktop computer. However over mobile devices, real web page response is 11.8 seconds with the Third-Generation Mobile Network (3G) [26]. Another problem is that a client may be using cellular data connections which are billed based on monthly total traffic. It increases the cost of the bill. Moreover, all mobile devices are using a battery for power. Mobile devices try to communicate with the Internet during page load and they send and receive a lot of requests/responses to get the content of the web page. If Wi-Fi connection is considered, there are different signals and getting the signal that client wants to connect when there are a bunch of other signals needs more power and it consumes the power of the mobile device. Connecting to the signal that user wants and then receiving data with latency means connecting to Wi-Fi longer duration and this means draining the battery. If the client uses the Global System for Mobile Communications (GSM), the Long-Term Evolution (LTE) or 3G, mobile devices are connecting to the strongest signal, again connecting to a base station means power consumption. Sending data over cellular means consuming more power and as mentioned before cellular data connections are billed [48].

1.2 Objective

The objective of this thesis is to transcode web pages for energy saving. The main reason for the energy consumption might be the payload and/or the number of HTTP requests/responses. The surveys show that the number of mobile devices is increasing and they have limitations that are not always considered by web page developers. Our goal is to transcode web pages according to mobile devices limitations so that the power consumption decreases over the mobile devices, but the look&feel of the web page is not modified and also the web page developers do not need to modify their web page. At the final stage, by saving the energy of mobile devices, we might make mobile devices more green so we might decrease the contribution of individuals to global warming. To achieve our objective, we ask the following research question: *"How can the web pages be transcoded without modifying their look&feel and without adding extra load to the client or the server with the goal of energy saving?"*

1.3 Contributions

The contributions of the Twes+ are listed below.

- **Novelity:** Some transcoding techniques have been suggested in the literature to improve user experience on web pages (see Chapter 2). However, these techniques were not combined and implemented together with the goal of saving energy. As the Twes+ implements three of these techniques on a proxy for energy saving, it is a novel system from this perspective.

- **Improving the battery life of mobile devices:** When a web page is loading over a mobile device, the most of the loading duration are spent for downloading the required resources from the main server. Moreover, to download a resource, the mobile device has to be connected to the Internet and send requests and receive responses. This process consumes power and drains the battery of the mobile device. As the Twes+ is designed to reduce the number of requests/responses, it saves energy over the client-side and it can help to improve the battery life of mobile devices. In particular, the Twes+ detects and eliminates the redirects of web pages. Moreover, it detects the images on web pages and allows to download the images with the same type combined.

- **Automated transcoding:** In the literature, there are several techniques to reduce the payload of the content and the number of requests/responses so the client experience gets better and the power consumption is improved (see Chapter 2). However, these techniques usually need to be applied by the web page developer. The problem is that there are still lots of web pages developed without considering these methods and some developers are non-programmer. The Twes+ allows to automatically transcode web pages on a proxy.

- **No extra load:** Twes+ has been implemented on a proxy to save energy, thus, there is no extra load on the client and the server. Although there are tools to save energy over the mobile devices, since they are working on the device, they also consume energy. However, Twes+ is on the proxy, so it does not add extra load to the client-side. In addition to this, Twes+ does not require modifications from the web developers since it takes into account the non-programmer developers.

1.4 Thesis Outline

The layout of this thesis is as follows:

Chapter 2: Related Work In this chapter, the mobile web is reviewed. There are many ways to access the Internet from the mobile devices and each way has its own benefits and challenges. Thereafter, the current studies related to energy saving in different fields are discussed. It is followed by what transcoding (i.e.,adaptation) is and how transcoding is used. It is followed by what the existing and possible transcoding techniques are to save energy.

Chapter 3: Research Methodology This chapter presents our research approach about how and which transcoding techniques can be applied systematically to save energy. Next, the description of the proposed architecture is provided. The last part is related to the implementation of the Twes+.

Chapter 4: Evaluation Method In order to validate whether the proposed Twes+ approach saves energy, we ask two research questions: *"Does Twes+ allow saving energy by applying redirect transcoding to reduce the number of redirects?"* and *"Does Twes+ allow saving energy by reducing the number of requests/responses between client and server and/or reducing the payload between client and server?"*. We conduct technical measurements over the desktop and the mobile device to answer these questions. The chapter describes the materials, the equipment and the tools. The results are presented, and discussed.

Chapter 5: Conclusion and Future Work We conclude the thesis in this chapter. The limitations of the current implementation of Twes+ and the future work are discussed.

CHAPTER 2

RELATED WORK

The overall goal of this research is to decrease the energy consumption of mobile devices while accessing the web through a browser. By decreasing the energy consumption of mobile devices during browsing, mobile devices might be more convenient for everyday use as their battery can last longer duration. The main reason for the energy consumption issue is the limitations of mobile devices. This research proposes a solution to achieve energy saving over mobile devices by transcoding (i.e., adaptation) the web pages in different ways. As a summary, this chapter reviews the related technologies and the related work in detail. This chapter is organized as follows: First, Section 2.1 summarize the ways to access the web. Section 2.2 discusses energy with respect saving energy in different fields and then Section 2.3 concludes with existing transcoding techniques.

2.1 Mobile Web

The meaning of mobile web is accessing the Internet through mobile devices instead of the desktop computers. It uses same protocols like HTTP, Transmission Control Protocol/Internet Protocol (TCP/IP), etc. The mobile devices can display the desktop web pages and the mobile web pages. Their main difference is mobile web is designed by considering the constraints of mobile devices. Before moving to the variations of the mobile web, the key characteristics of mobile devices is important. Mobile devices are portable, personal, easy to use and have access to network connection [29]. There are several ways to access the Internet via mobile devices but we can categorize them into four: Accessing the Internet via a native application or via a mobile web application or a browser or a widget.

Native Applications They are developed for specific mobile platform, firmware and operating system. These applications are stand-alone, this means the entire component runs on the device or might have their core component on the mobile device and communicate with web servers. The client needs to download native applications from an application store like Google play or Apple store and install on the platform [37]. The advantage of native applications is that they have access to the hardware of the mobile devices, so they can make use of the true power of the underlying hardware. This makes them better in user experience [40]. However, their main drawback is that they are compatible with only one platform and they require modification to be used over another platform [37].

Mobile Web Applications They are combination of HTML5, Cascading Style Sheets (CSS) and JavaScript (JS) [40]. They are designed for cross platforms and it is an advantage because to be displayed over a different mobile device, mobile web applications need only a browser that supports HTML5, JS and CSS [18]. They are using the power of the platform and hardware as much as possible and also, their development process is more straightforward. This takes the attention of non-programming developers [18, 37]. Mobile web applications are also able to create an icon over mobile devices. This makes them more like native applications and increases their usability as the user does not need to use bookmarks [18]. Although they are executed in a web browser on the mobile devices, their performance is getting closer to the native applications but at the end, their performance is lower than native applications as there is extra browser layer [18, 37]. Another drawback is that mobile web applications are dependent on the Internet connection. Without connecting to the Internet, they cannot be used [18].

Mobile Browsers A web browser is a software that is used to show the information resources on the WWW. These resources are accessed by using Uniform Resource Identifier (URI). The first web browser named WorldWideWeb, later named Nexus was developed by Tim Berners-Lee the director of the World Wide Web Consortium (W3C) in 1990 [11]. However, today there are different types of browsers. They can be divided into groups according to the devices that they are developed for. It is hard to list all the browsers but according to W3C, the most popular browsers are Google Chrome Browser, Internet Explorer, Mozilla Firefox, Apple Safari and Opera. The statistics show that Google Chrome browser is the most popular browser with 66.5% in October 2015 [91]. There are also different browsers for mobile devices, see Table 2.1 [30, 3]. These browsers are developed for the different operating systems but energy saving is becoming a concern for the browser developers. The energy column of Table 2.1 shows the existing plugins, add-ons, and modes for browsers that have an energy saving concerns and options. They can be enabled to save energy for the devices during browsing by blocking Flash contents or reducing performance or compressing over the cloud or reducing the payload by not transferring images.

There are also two different type of web pages which are desktop web pages and mobile web pages. The desktop web pages are designed according to the computer specifics like big screens, fast connection, etc. Although they are desktop versions, they can be accessed through a mobile browser. Sometimes they are designed by using responsive web design concepts so that it works on the mobile browsers as desktop web pages with a different layout. The main goal is having another version of the desktop web page for different devices. The mobile web pages on the other hand designed specifically for the mobile devices and can be created for a specific device as well. Mobile web pages and the responsive web pages work as similar. If the client tries to access the desktop version of a web page, website server checks the user-agent identifier of the request. If the server recognizes a mobile device, the web server sends the specific version of the web page or the one that is designed by using responsive design concepts [40].

Table 2.1: Browsers Energy Saving Details [30, 3]

Device Category	Browser Name	Energy	Ref.
Desktop	Google Chrome	Plug-in Power Save Mode	
	Internet Explorer	Power Plan of Windows OS	[55]
	Opera	Off-Road mode	[64]
	Apple Safari	Plug-in Safari Power Saver	[51]
	Mozilla Firefox		
	...		
Mobile Devices	Apple Safari		
	Google Chrome	Google Chrome Data Saver	[22]
	Opera Mini and Mobile	Data Saving Mode	[65]
	Mozilla Firefox Mobile	Power Saver Mode	[79]
	Amazon Silk	Amazon's Cloud	[58, 6]
	Blackberry Browsers		
	Nokia Browser		
	Internet Explorer Mobile		
	...		

Widgets Mobile widgets are small applications and they are task-specific. They are executed on widget engines and each type of widget needs its own engine. Samsung TouchWiz can be an example for widget engine. They are stand-alone and they communicate with a dedicated web server. Their advantage is that they are light-weighted and easy to use. On the other hand, they have a drawback like native applications. They are not cross-platform so they need modification to be used on different devices [37].

Because all these categories are using the HTTP in the background to communicate with the Internet, the HTTP is mentioned briefly at the end of the section. Since 1990, it has been in use by WWW global initiative. This protocol is based on requests and responses. Basically, the client establishes a connection and requests a web page content from a server and server responses this request and send the content of the web page. There can be intermediary forms like a proxy, gateway, etc. between the client and the server. If there are intermediary forms between them, request and response should pass through them as well. As a summary, client and server need requests/responses to communicate with each other and to send the content of a web page or data [28].

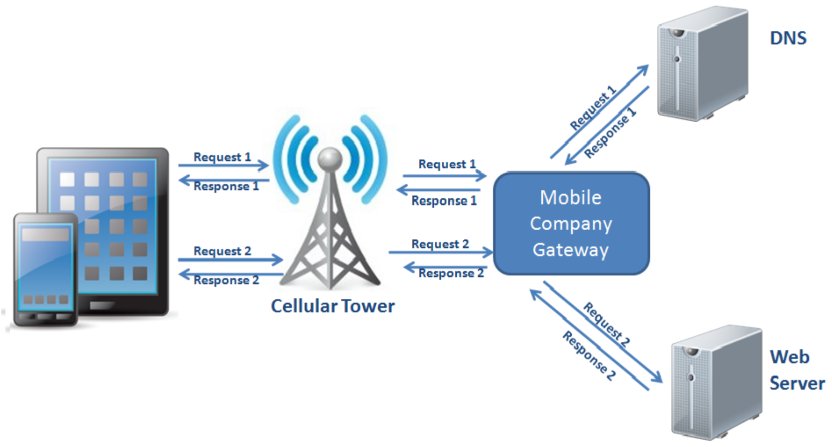


Figure 2.1: HTTP Request from a Mobile Device

The path of a HTTP request from a mobile device to a web server has several steps, see Figure 2.1. When a client visits a web page the first time, the device sends two requests. The first request is for the Domain Name System (DNS) lookup. In this step, the domain name that user entered will be found and response of this request will be the IP address of the domain. The second request will be sent to this IP address to get the HyperText Markup Language (HTML) of the web page. Typical web pages require more requests to get all resources and it means more DNS lookup requests and more HTTP requests. Each request has to pass through the following path. Firstly, the request needs to reach from mobile device to the nearest cellular tower. After that request goes to the mobile company gateway to reach the Internet. This gateway assigns an IP address and it sends the DNS lookup request and HTTP requests. Each response comes back from the same path. The response gets back to the gateway of the mobile company, cellular tower and as final mobile devices.

For an HTML web page, dozens of requests and responses are needed and the client will wait during this time. With low bandwidth and high latency, mobile device battery drains. Loading time of a web page has two parts. Displaying the content and loading the resource. The 20% of loading time is used for displaying the content for both desktop and mobile devices and the rest of the time is used for loading the resources and performing client-side processing [26]. Narendran et al. [59] show the impact of this HTTP traffic over the 3G radio. The measurement shows that for downloading or uploading, there are an approximately 12 Joules for connection establishment. The more interesting result is that the required energy for downloading data is not connected to the size of the data (if the size of the data is below 256KB). The same situation does not apply to the uploading data case as its energy consumption increasing with the data size. There is another interesting finding related to the traffic. They uploaded 8KB at 1KB per iteration and directly at once. The required total energy for iteration is 5%. It is more than the required energy to upload at once. This shows us the impact of the HTTP traffic over a mobile device and a web browser needs HTTP request/response traffic just to display a web page. According to Zakas [95], to improve the web performance, reduce the latency and improve the battery life of mobile devices, the best thing is reducing the HTTP requests and the size of the resources. The next section covers the energy saving concerns of different fields. If the key characteristics of mobile devices are considered, the battery life of mobile devices is crucial for them.

To sum up, this section summarizes the ways of accessing the Internet. If we categorize them, there are totally four categories to access the Internet. From these categories, the native applications and widgets communicate with the server over the Internet in the background for a specific purpose. Mobile web applications are cross platform version of native applications and although it is using the browser and connects to the Internet, it again works for its own specific purpose. From these four categories, only mobile browsers are a way to connect to the Internet, not for only a specific purpose. Mobile browsers are also the only way that is not blocked with applications securities. This makes mobile browsers a good candidate for research and it can be seen from the existing energy saving implementations for mobile browsers like plug-ins. In this section, how a web page is requested and the impact of resource transfer over the Internet from the servers to the clients is also discussed. The impacts show that each request and the latency that occurs when a resource of a web page is loaded from the server, it consumes energy.

Having many requests will obviously drain the battery. In the next section, proposed solutions of different fields to save energy will be introduced.

2.2 Energy

This section explains the research in different fields that focus on reducing the power consumption or use sustainable energy sources for mobile devices. There are different approaches to reduce the footprint of the technology, save energy on different sections of communication and improve the battery life of mobile devices. The literature review shows that research on energy has been done in the software, the hardware and in the network fields. The following sections introduce these approaches.

Software

The power measurements for Android (allows only JAVA) and Angstrom Linux (allows JAVA and native C) on Beagle Board are presented [67]. The comparison is between Angstrom (Sun embedded Java Virtual Machine (JVM) with Just-in-time compiler (JIT)) - Android (Dalvik JVM) and Angstrom (Sun embedded JVM without JIT) - Android (Dalvik JVM). The power results are between Angstrom (Sun embedded JVM with JIT) and Android (Dalvik JVM) as follows; Heap sort algorithm total energy consumption for Angstrom is 681.64 watt-sec and for Android is 5064.79 watt-sec, Quicksort algorithm total energy consumption for Angstrom is 10.56 watt-sec and for Android is 31.07 watt-sec. Their conclusion is with a good JIT compiler for Dalvik JVM Android can have better energy numbers and low execution time.

Google has also developed a protocol called SPeeDY Protocol (SPDY). SPDY is an application layer protocol. It is using the Transmission Control Protocol (TCP) as transport layer so it does not require a change in network infrastructure. It only requires changes in the client user agent and web server applications. The main change in the protocol is that it allows multiple requests to be sent on a TCP session. It is compressing the header of the request so content size is less than regular requests. The streams are bi-directional and it can be initialized by server or client so the server can initiate a communication and push data to the client without client request. To solve the possible bottleneck, the client can block the request or prioritize them. The main purpose of developing SPDY protocol is to reduce the latency. It is an application layer protocol. After the comparison between HTTP & SPDY up to 64% reduction is observed in page load duration [20].

There are also experiments related to energy consumption of browsers when displaying a web page. By using top web pages, the energy consumption of the browser is measured while displaying those pages [59]. As mention in Section 2.1, for 3G setup required energy is approximately 12 Joules. Their results show that for Apple [1] required energy is approximately 46 Joules and for the mobile version of Wikipedia [2] required energy is approximately 36 Joules. This shows that web browsers are consuming a serious amount of energy for mobile devices. About ten Joules of Wikipedia was consumed by downloading and rendering the JavaScript. On the other hand, their experiment results are by shrinking JavaScript and using only required functions can even reduce the energy consumption. For the Apple web page, 12 Joules of total consumed energy is used to download and render CSS files. By just removing none used functions, they reduce the energy consumption about five Joules. These numbers show that by reducing JavaScript and CSS, energy saving can be achieved.

Network

There are several approaches to reduce the footprint of mobile devices. The energy consumption of a customer for a terminal and for the mobile network was investigated. The results show that 0.83Wh/day for a terminal and 120Wh/day for the mobile network. Their main concern is about the mobile network and their results show that the net electricity consumption of data center per day for one mobile phone user is about 70Wh. Although the impact of terminals can be

[1] `apple.com`
[2] `m.wikipedia.org`

negligible when it is compared with the mobile network, terminals face energy starvation issue because of the battery limitations of mobile devices [25].

When the operational duration of mobile networked devices increases, it has an impact on the battery life of the device. To show the difference, the battery life comparison of the iPhone 3G can be an example. On standby, it is up to 300 hours but on 3G data connection it is up to five hours and on Wi-Fi connection it is up to six hours. The proposed solutions are changing the network paradigms and adopting an information-centric approach and the other proposed solution is multi-access mobile networking. In principle, it uses surrounding networks together and its simulation results are mentioned as promising but this solution needs infrastructural changes [68]. To optimize the localization service of mobile devices, a system called Senseless is proposed. Instead of using the only Global Positioning System (GPS), it combines accelerometer, GPS and 802.11 access point in location-aware services. Their early results show that their design can extend the battery life of mobile devices from nine hours to 22 hours [9]. Another study takes the attention to the energy efficiency requirement of localization. Localization is needed for emergency cases and tracking the mobile device and sending this data to a server have to be energy efficient so it does not drain the battery. They propose a sensor management, trajectory updating protocol and time to send the data. The power breakdown of their proposed tracking system is minimizing the 3G radio connection and GPS power, so the battery of the mobile device does not drain during localization [8].

Hardware

Including renewable energy sources into the mobile devices is proposed to improve the battery life of mobile devices. There are two methods mentioned as a source. One of the methods is integrating thermoelectric generator to the CPU heat pipe and using the waste heat of the processor. The other method is integrating photovoltaic unit to the devices and generating power from environmental illumination [2]. Moreover, a tool is developed to estimate the output power level of the Microelectromechanical systems (MEMS) that converts low-frequency vibration to electrical energy.The MEMS Energy Scavenger Module is for mobile computing and wireless sensor applications [87]. Processor wise energy saving is a crucial topic. Intel Corporation developed the first mobile processor that is using Core-Multi-Processor microarchitecture to achieve high performance with low power consumption, named Intel Core Duo. Normally power consumption of processors is controlled with Advanced Configuration & Power Interface (ACPI). The system defines different sleep states so that when the system is idle, the processor goes into deep sleep states and when the system is active, the processor goes to the active state. For Intel Core Duo, there are two core and these two cores have their individual power states and there is combined package state. This way the performance improvement is achieved where the processor is using the same power as single core previous generation processor [73].

To sum up, the literature review shows that energy consumption is a concern of different fields. Hardware field is making major changes to reduce the energy consumption of mobile devices at the hardware, low level and also to find sustainable sources for the devices as the battery of the mobile devices is one of the main limitations of the mobile devices. By defining power states, the hardware components are pushed into sleep states. However, without solving the interruptions of software and network, the device continues still consuming more energy. From the network perspective, there are some studies to improve the network connections as the battery life of mobile devices drains while devices are connected and also data centers consumes a serious amount of energy when they are answering plenty of connections.

In the software field, there are some studies to improve the code execution, but the studies specific to the web is very limited. When the key characteristics of mobile devices are considered, mobile devices should be portable. It means they should be on battery and mainly they should be able to access the Web. So far, Google proposes a new experimental protocol to improve the request/response between the device and the web. There are also recommendations for developers to fix the design of their web pages to eliminate unnecessary energy consumption. However, as the web is standing between network and software, there are not enough improvements related to the web except few plug-ins. Moreover by just concentrating only hardware or network or software, improving the energy consumption of mobile devices does not seem to solve the battery life issue of mobile devices. The next section covers what is transcoding stands for and what are the existing transcoding techniques in the literature.

2.3 Web Page Transcoding

Transcoding is the process of transforming web pages into alternative forms to make improvements in a certain way. It can be used to improve the web accessibility for disabled people, the performance and the usability of mobile devices. Transcoding can be done at three locations namely server-side, proxy-side and client-side [86]. Working principle is as the client requests the web page from a web server and this request passes through transcoding server. Transcoding server receives the response from the web server and applies certain transcoding techniques to the content of the web page according to the type of device, network bandwidth, etc. After this transformation, modified web page content is sent to the client by transcoding server [12].

The server-side transformation means transformation is done by the web server and the client receives the transcoded content. It is invisible to the client since it is a part of the web page [5]. The user requests the starting page of the web page. For example, the URI of the user is `www.ibm.com` and the URI of the modified page is `http://access.accu.org/cbi-bin/access?Aesu=1&Au=www.ibm.com`. This means on the server-side *cbi-bin/access* script is processed to transform the content [5]. The advantages of server-side transcoding are that it requires the minimum bandwidth as transfers the minimum amount of data and also no required changes on client-side. The disadvantage of server-side transformation is that web servers are private servers so a limited number of the web page might have the transcoding property [93].

On the other hand, the client-side transformation means that transformation is done on the client side. The client is receiving web page content as it is and doing the transformation on the device itself [86]. Transcoding the content is done on the browser [5]. The main purpose is modifying the web pages according to the mobile device limitations, so the device usage can be more convenient while browsing. This way the user uses the web page easier. The advantage of client-side transcoding is that the web pages are transcoded according to the preferences of the client, as a result, the transcoded web page can fit better. The disadvantage of client-side transcoding is that transcoding is done on the device so the device should be powerful enough. Moreover, this option does not reduce the bandwidth usage. If the constraints of mobile devices are considered, client-side transcoding is not useful enough [93].

The third option is applying the transformation on the proxy. Proxy-side transformation stands between the client and the server. Originally transcoding was referring only Proxy-side transcoding [86]. It is not totally transparent to the user. The address of transcoding server for the client browser should be set by the client. After this setting, transcoding is transparent to the user [5]. The mobile device sends a request to the proxy server. If the request is URI, the proxy server requests overall web page. It assigns a unique ID to the client and returns the overall page to the client. If the request is an event, the proxy application selects the next visual element according to the event and sends it to client [93]. The drawback is connection speed between the client-the proxy and the proxy-the server [93].

There can be another option for the transcoding location like instead of using one type of transcoding, the combined version can be used. For existing video messaging services, the client-side transcoding can be used to reduce the upload traffic and the server-side transcoding can be used to deliver a lower quality version to the recipient [49]. So it can be combined and network traffic that is generated by upstream and downstream transmission can be lower down. For this part, the client-side transcoding gives flexibility to the user to have different choices for recording and transmission. High-quality video can be recorded by the device and video can be transformed to a lower quality video. It reduces the transmitted data across the network. It leads to a reduction in the network cost, battery usage, and latency. One of the concern is the constraints of mobile devices for the client-side transcoding. According to the test results of a study, that is done on Google Nexus 4 (1.5 Ghz CPU), transcoding is slow [49]. The result is

30 seconds to transcode ten seconds video. So it might not be a good option for battery usage wise but for sure it reduces the amount of data that is transferred and the latency. For server-side transcoding, it reduces the traffic, network cost, downloading latency and battery usage of the recipient. The recipient has more options for slow network and less cellular data cost [49].

The transcoding techniques can also used to improve accessibility of the web pages. The web is universal and designed for all the people. To make the web accessible by everyone including people with disabilities, the Web Accessibility Initiative (WAI) developed Web Content Accessibility Guidelines (WCAG) 1.0 [10] and W3C also provides guidelines [5]. However, there are web pages that are developed without following those guidelines. Transcoding is used to improve the usability of the devices that are used to make web pages accessible to the people with a disability. In addition to this, transcoding also modifies and transforms the web pages into an accessible form [5]. The other purpose is to improve the performance and the usability of mobile devices. Web pages are typically created for desktops in mind with complex design to improve the user experience. However, the decoding capability of mobile devices does not match with the web page requirements because of the constraints of mobile devices. To solve this issue, the web pages are transformed into a matching format for mobile devices and it is called transcoding [12].

In order to analyze existing techniques of transcoding, we have developed a scheme as shown in Table 2.2. In this scheme, whether or not these techniques are used for energy saving without modifying the look&feel of the web page and without adding extra load on the client and the server is investigated. Existing techniques are categorized into 13 groups according to their goal. For each technique, the references of literature and some of the existing implementation of these techniques are listed. One of the important criteria is the location of the transcoding technique to be applied. As the server-side transcoding have to be done with the knowledge of the developer, the client-side transcoding have to be done with the knowledge of the user and also the client-side transcoding increases the load on the mobile devices, this criterion is important. Another criterion is whether the client is viewing the web page same as before transcoding technique is applied or not. In the previous sections, it is mentioned that the user likes to view the web pages as original. The effect of transformation on web accessibility is also categorized under these three criteria.

Some techniques are used or proposed to solve the screen size limitation of the mobile devices and some techniques are used to improve the web accessibility of people with special needs(p/wSN). Some of these techniques also have an impact on the battery life of the mobile devices as well. In the next section, some of the techniques are described, but there are also text magnification, color scheme changes, and so on. The details of these techniques are not described as they are totally irrelevant to energy saving. For example, text magnification is suitable for the server-side and the client-side transcoding. The goal is changing the font-size for the sighted users. Color scheme changes technique is useful for the people who has some eye conditions like the color-vision deficiency [5].

Alternative Text Insertion

This technique is needed for the screen readers as they are not able to access images. By using alternative text insertion, images can be accessible [5]. Google Chrome browser has an extension that works on the client-side browser and transcodes the web pages. It is called Image Alt Text Viewer and it replaces images with their alternative text and if the user wants to see the original way, there is an undo button [33].

The Personalizable Accessible Navigation (PAN) also have a similar technique by using proxy-side transcoding. It replaces the images with link information and gives a link to the original image [88]. In HTML 4.0 Guidelines for Mobile Access, using alternative text for the mobile devices is mentioned under the client-side image maps [42]. Assuming that images are always rendered on mobile devices is not right, so this technique can be also used for the mobile usability. As it removes the images from the web page and a link is added for each image, it solves the screen size limitation and battery life limitation. Images are not transferred so the

Table 2.2: Transcoding Methods [§P/wSN: People with Special Needs, S: Server-Side, C: Client-Side, P: Proxy-Side, *: Reverse-Proxy]

Technique	Implementation	Reference	Location			Modify the Look&Feel	Adaptation for P/wSN §	Screen Size Adaptation	Battery Life Improvement
			S	P	C				
Text Magnification	IGoogle, Browser	[5, 34]	✓	χ	✓	✓	✓	✓	χ
Color Scheme Changes	Color Enhancer, High Contrast, PAN	[5, 88, 33]	χ	✓	✓	✓	✓	χ	χ
Alternative Text Insertion	Image Alt Text Viewer, PAN	[5, 33, 88]	χ	✓	✓	✓	✓	✓	✓
Page Rearrangement	BETSIE	[86, 5, 82, 74]	✓	✓	χ	✓	✓	✓	χ
Simplification	FOM, [94]	[5, 13, 94]	✓	✓	χ	✓	χ	✓	✓
Summarization	SSD Browser, [46]	[5, 86, 1, 46, 47]	χ	✓	✓	✓	χ	✓	✓
Image Consolidation	Google PageSpeed	[44, 26, 21, 27]	✓	✓*	χ	χ	χ	χ	?
Responsive Image	Apple Retina, Mobify, Google PageSpeed, HTML5 & CSS	[32, 26, 35, 52, 21, 56]	✓	χ	✓	χ	χ	✓	?
Image Inlining	[31]	[52, 53]	✓	χ	χ	χ	χ	χ	χ
Data Compression	Google Data Saver, Research by using Squid Proxy	[7, 26, 22, 14, 43]	✓	✓	✓	χ	χ	χ	✓
Concatenation of external files	Google Apache Module, Mobify Jazzcat	[26, 95, 84]	✓	✓	χ	χ	χ	χ	✓
Reducing the number of Redirects	Apache Reverse Proxy and mapping rules	[95, 92, 83]	✓	✓*	χ	χ	χ	χ	?
Expiration Header	Content Delivery Network, Proxy-Cache of Corporations	[26, 63]	✓	✓	✓	χ	χ	χ	✓

number of requests/responses is less than the original web page and also images are not seen on the mobile devices screen. However, It changes look&feel of the web page, as the images are not displayed to the user.

Page rearrangement

The web page layout is in the grid form, so the content can be divided into columns and the page can be reorganized. As a result, the important parts of the web pages can be moved to the upper part of the web page [5, 82]. For the voice browsers, this technique is used so that the visual element are converted to non-visually distinguishable. BETSIE is the first system that applies page rearrangement for voice browsers [5] to solve the screen size issue [86]. By transcoding the web page over a proxy server, it can be accessed easier over a mobile device. This technique is also used to sort the visual elements of the pages, as a result, important elements can be seen first [74]. Another way to use this technique is like the page can be rearranged by referring to the role of each component and it can be used for lazy-loading.

Simplification

It can be called also selective representation or page clipping. By removing non-vital, irrelevant and unnecessary elements, it reduces the size of the content and the time to reach important contents. It is used for the usability of the mobile device as mobile devices have screen size limitation [5]. For example, Function-based Object Model (FOM) is proposed and it uses this idea for irrelevant images and removes them from the web page by using a proxy server [13]. Another project is also proposed that is following the opposite path. It finds the important elements instead of irrelevant elements [94]. The problem with this technique is that it modifies the look&feel of the web page and some elements are not accessible at all.

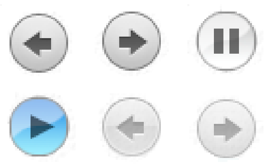
Summarization

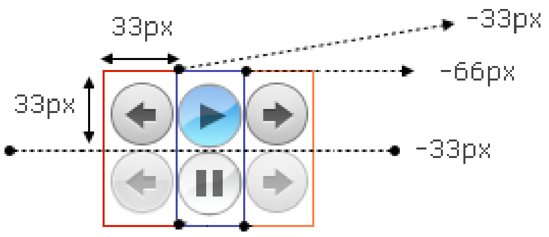
It stands for summarizing the web page as a table of content by moving the contents to a new page and giving a link to them from main page [5, 86]. This technique is applied in different ways but it is mainly used to increase the usability of mobile devices. For example, the Small Screen Devices (SSD) browser is developed and this technique is applied to create the table of content of the web pages [1]. Moreover, different adaptations are generated for different needs. As a result, the adapted versions are rendered by the browser faster than the original version [46]. As a different example, the thumbnail interface of the web page and the thumbnail summary of the web page are applied. The results of these two methods show that the user prefers the thumbnail summary interface instead of the thumbnail interface [47]. However, this technique also modifies the look&feel of the web page.

Image consolidation

All the external images require a request/response round trip. Image consolidation technique is merging these images. The images can be consolidated into one single CSS background image and transmitted this way. The client receives one image and by using CSS background positioning(splitting technique), individual images can be accessed, see Figure 2.2 [44]. This technique is proposed for the small images of the web pages [27] to increase the performance. Everts [26] also proposes image consolidation to reduce the number of requests/responses of the mobile devices for performance optimization. Normally, the client requests the each image and each image request/response has overhead of TCP handshake. Moreover, during each request, extra HTTP header, and image header is transfered [27].

Benefits of image consolidation technique are saving the network bandwidth, reducing the number of requests/responses and reducing the payload of the web pages. In addition to those benefits, it can increase the parallelism of the web pages [21]. There are two options to use this technique. The first one is implementing this technique manually over the server, so basically


(a) Six Requests/Responses.



(b) One Request/Response.

Figure 2.2: Comparison of Regular Images and Responsive Images Usage over Different Devices [44].

web page is designed by using this technique. Another one is that image consolidation technique can be applied by the client and the server when they are uploading images [27].

There are two constraints about this technique. First one is the web page that contains dynamic images. For example, the homepage of Facebook, it contains different images for each user and it is regularly updating itself. Only background images are the constants images. To solve this problem, the CSS sprite can be designed such a way that it can differentiate static images of the web page. However, the systems except the web server do not have the knowledge of the type of the images. The second problem is about the decision of how many images should be put into each sprite. It is important because till the entire consolidated image is downloaded, the client cannot see those images. As a result, the fewer image is a better option and it can be done by separating the images according to their types or the time when they will be visible to put them into the container [27].

PageSpeed module by Google applies this technique over a reverse proxy that needs to be setup by the original server. It contains rewrite CSS filters. It still has certain limitations like only supports Portable Network Graphics (PNG) and Graphics Interchange Format (GIF) images and CSS backgrounds and images should be smaller than the background images. Developers are also expecting possible issues like browser caching might not be effective anymore, server load might increase, the CSS files might get bigger as the Uniform Resource Locator (URL) of the combined images is longer so the CSS files might need compression and the web page might consume more cache on clients side after decompressed bitmap images, see Table 2.2 [21].

Responsive Image or Resize Image

This is part of the Responsive Web Design (RWD) approach. The web pages were designed for desktop computers with large screens, a powerful processing unit, almost endless memory. However, checking this kind of web pages over mobile devices is frustrating. The user has to zoom in-out all the time. This brings the idea of the RWD. It has different names like the fluid design, elastic layout, cross-device design, etc. The main idea is changing the web page design based on the device [32]. The design can be done by using ready HTML5 and CSS3 features. To compare regular images and responsive images see Table 2.3.

(a) Regular Images (b) Responsive Images

Figure 2.3: Comparison of Regular Images and Responsive Images Usage over Different Devices [77]

The responsive image is a solution for the high-resolution images. The high-resolution images are generally big and transmitting them increases the payload. However, because of the screen size constraints of mobile devices, those high-resolution images do not look as they are expected. As a result, those images are a waste of resources. There are two options that can be considered. The first option is that the high-resolution images can be resized dynamically by the client side and smaller version of these images can be used on the web page [26, 56]. The second option is keeping the different size and resolution of the same image on the web server. According to the device type and the connection type of the device, the matching version of the image can be used [52].

Apple has used this technique for a different purpose. For marketing the IPAD retina, the device is loading the regular images and replacing this images with the high-resolution version to show the screen resolution feature of the device, see Table 2.2 [35]. As a summary, according to the way of implementing this technique, rendering the page can be faster and bandwidth, memory, battery consumption and latency might reduce.

Image Inlining

This technique works by decoding the image and turning it into a string. This string is embedded into the source (HTML or CSS). The server sends this source to the client, so request/response is removed. This technique is to embed images into web pages themselves, so when the web page is compressed, the image is compressed and transmitted. When the client receives this source, browser decodes this image URI and show the image [52]. There are online tools to create data URI or by using PHP images can be converted to data URI on-the-fly (dynamically), see Table 2.2 [31]. Although by using data URI the number of requests/responses decreases so energy saving might be achieved, there are some drawbacks. The first drawback is decoding this image over a mobile device, as decoding uses Central Processing Unit (CPU) and it increases the battery consumption. The second drawback is when the image is converted into data URI, the size of the image is increasing. Another problem is if the source is not cached, the image needs to be transferred every time [53]. According to the results of some measurements, the data URI usage is six times slower than the external linking over the mobile devices [53]. As a result, when the drawbacks are considered, this technique is good for the infrequent small images.

Data Compression

It can be an option to reduce the payload of web pages. According to a research, transmitting one bit over a wireless connection requires 1000 times more energy than one 32-bit computation. If the energy required to compress a web page is less than the required energy for the transmission than there is saving. It increases the battery life of mobile devices. Moreover, the payload of the web page decreases so the transmission time decreases, see Figure 2.4 [7]. This technique can be implemented by the server and modern browsers support this feature [26]. This technique is implemented in the proxy server to test different compression techniques. The results are 30% bandwidth saving with about 1% transfer time overhead [14]. In addition to this research, Google developer group implements this technique for mobile devices as well, see Table 2.2. Compression is applied with other techniques by Google and 70% response size reduction is observed [26]. However, as it is found in previous studies, the compression reduces the payload but it adds the compression and the decompression overhead for the server and the mobile devices [26]. In this case, this technique needs a comparison between the decompression and the latency energy consumption to see this feature is suitable or not.

Original Data
Original Data Download Time
Original Data
Original Data
Original Data
Decompressed
Compressed Data Download Time
Compressed
Original Data

(a) Original Data. (b) Compressed Data.

Figure 2.4: Comparison of Original Data and Compressed Data Effect Over the Bandwidth

Concatenation of external files

It can be also referred as consolidating resources. For each external resource, one request and one response are required. If external JS and CSS are concatenated, the number of the requests/responses decreases. According to Everts [26], during the development having multiple style sheet and scripts are better and easy for the developers. However, the ideal web pages should have one request for each style sheet and each script. Google released apache module that requires the server support. This module uses special URL format to download multiple files using 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
http://www.example.com/assets/js??main.js,utils.js,lang.js
```

Here this double question marks indicated to the server that this URL is concatenated. It setups the concatenation mod on the server side [95].

There is also another backend service called Jazzcat. It stands between the client and the server. It receives a list of resources and requests all those resources on behalf of the client. Thereafter, it encodes those resources and sends one response [84]. This response goes to the next point which is a Content Delivery Network (CDN) module. There, the response is cached, compressed and sent to the client, see Table 2.2 [84]. However, there are some concerns about external file concatenation. Without concatenation, if there is an error in Javascript, the browser stops compiling it and passes to the next external JS source. If all external JS files are combined and there is an error, there might be nothing to execute and parse. This means before combining, JS sources should be error free. For CSS sources, there are more concerns such as different encoding techniques, conversion of relative URLs to absolute URLs and import parts as they only work at the beginning of the code. Combining the CSS sources is also hard [84].

Reducing the number redirects

Redirects have the status code as 3xx. All necessary information for the redirect are not inside the body, moreover, the body is empty [92]. If a web page has a redirect, it means client requests the web page but the server returns a response with a header and the new URL of the web page content. Pages cannot be rendered or download can not be done. In addition to that, each request has the overhead of a full request and each request adds extra latency [95]. The most wasteful redirect is missing slash punctuation. If the original web page is `www.yahoo.com/astrology/` and the client writes it like `www.yahoo.com/astrology`, web page response has status code 3xx. However, the most frequent reasons are connecting a new web page from the old URL and the mobile version of the web pages [92]. The Web page developers create the mobile version of their web pages to improve the mobile user's experience *(original:`ncc.metu.edu.tr`, mobile version:`m.ncc.metu.edu.tr`)*. When the developers are developing their mobile web pages, they made an assumption that the user enters the full domain. The problem is when the user enters only hostname (ncc metu), the server decides where to redirect. The server receives the request and the server redirects the user to the mobile web page [95].

Apache has two types of traffic server that stand between the client and the server. Forward proxy caching type is like a traffic server requests on behalf of the client. Reverse proxy caching type is like the client see the traffic server as the original server and the traffic server stands on behalf of the original server [83]. Forward Proxy Caching Traffic Server works like a gateway between the client and the web server. When it receives the request from the client, it filters or applies the specified rules to the request and forward it to the web server. When the web server sends the response back to the client, it again applies the specified modifications or caches the content and forwards the content to the client. It can be transparent to the client or by browser setting it can be set up between the client and the web server. Reverse proxy caching traffic server is between the client and the web server but it acts like an original server setup by the web server. By using mapping rules, HTTP requests and redirects can be assigned [83].

Expiration header and caching

Caching means saving for the future usage and for this research saving for future requests. When a request is made by the client, as a first step, the cache is controlled. If data is in the cache, it is called cache hit. If data is not in the cache, it is called cache miss. Storage can be done inside and outside of client system. All browsers even have a local storage. Expires header is used especially for the caching. This header shows how long data can be cached. When a request is done, the response can be cached. If the same request is sent, instead of asking it from the web server, cache responses on behalf of the server. When data is expired, data is removed from the storage. If it is not expired and it is a cache hit, the number of requests/responses decreases, because the client does not send a request to the server [26].

This technique is related to caching part. When a request is done, the response can be cached in the web cache between client and web server. If the same request is sent, instead of asking it from the web server, web cache can response it on behalf of the web server. This reduces the latency and the network traffic. Expiration header can be used for those data that are cached. There are three kinds of web caches; browser caches, proxy caches and gateway caches. The browser web cache is not applicable for the mobile devices because of the mobile device constraints. The proxy caches are mainly set up by large corporations on their firewall or as a standalone system. On this proxy web cache, data can be saved according to certain rules and expiration date. The gateway caches are owned by web server owners to make their web page more reliable, scalable and better performing. As an example for gateway caches, content delivery networks can be considered [63].

From Table 2.2, it can be seen that data compression, concatenation of external files and expiration header usage are implemented on a proxy server so the client and the server side do not know that there is a middle layer. Moreover, they are not modifying the look&feel of the web page and they have energy saving. As they are already implemented, these techniques are not further considered. We also eliminate the first six techniques in the Table 2.2 as they are modifying the look&feel of the web page. As a result, our investigation shows that there are four techniques that are not modifying the look&feel of the web page, but they are not implemented over the proxy-side. The battery life criterion shows that image inlining does not provide any improvement in the energy saving for the mobile devices. The final techniques that are suitable to our criteria and that are promising for the energy saving are image consolidation, responsive image and reducing the number of redirects. These three techniques are not used to improve the battery life of mobile devices when the client is accessing the Internet through the browser. In Twes+, these three techniques are implemented to save energy over a proxy server that the client and the server do not have any setup process and also the look&feel of the web page remains same as before Twes+ transcode the web pages.

This chapter starts with a summarization of the ways to access the Internet from the mobile devices. The mobile web looks similar as the desktop versions but when the limitations of mobile devices are considered, these categories make a difference. There are four categories to access the Internet by using the mobile devices which are native applications, mobile web applications, mobile web pages and the widgets. The advantages and the disadvantages of this four categories are mentioned. The next section covers the different approaches of other fields towards energy saving for the mobile devices. Other fields are categorized as software, network, and hardware. The last section is about the web page transcoding and the existing techniques what are used for the people with special needs and the mobile usage. There are lots of techniques applied at different locations but these techniques have certain drawbacks. The next chapter describes our proposed approach and architecture to overcome some of these drawbacks and save energy over mobile devices.

The goal of this study is saving energy for the mobile devices during browsing but without putting extra load on the client device or the developers. In this chapter, though the browser developers develop tools to improve the energy consumption by compressing the data or caching or removing parts, they do not try to improve the source code of the web pages. Moreover, it is clear that mainly energy improvements are done on the hardware or the network level and the software wise improvements are not enough. There are suggestions for the developers to improve their web pages. However, many non-programmer developers do not follow these suggestions and there are many web pages developed without following these suggestions. For these reasons, there are some works to adapt transcoding techniques that are mainly used for the web accessibility to the energy saving area. However, as it can be seen the works related to images are server dependent or modifying the look&feel of the web page and there are no independent works related to the redirects. In this study, Twes+ is proposed, since It works independently from the client and the server and transcode the web pages to save energy.

CHAPTER 3

RESEARCH METHODOLOGY

In this chapter, we present our novel approach called Twes+ [1] (stands for **T**ranscoding **W**ebpages for **E**nergy **S**aving) that transcodes web pages to decrease energy consumption during browsing. Our approach is intercepting with the network traffic between the client and the server and transcode the source code of the web pages and the elements of the web page to reduce the number of requests/responses and to reduce the payload of the web pages.

This chapter includes three sections: Section 3.1 explains the techniques that are employed in Twes+. Section 3.2 gives brief information about the system architecture and Section 3.3 covers the implementation of the techniques over this architecture.

3.1 Research Approach

In Chapter 2 Table 2.2, the possible transcoding techniques are discussed. The desired way to display the web page is without modifying the look&feel of it hence, text magnification, color scheme changes, alternative text insertion, page rearrangement, simplification and summarization transcoding techniques are not suitable for our work. For Twes+, our goal is to reduce the energy consumption of mobile devices so Twes+ should not run on the client because it will increase the energy consumption of the mobile devices. Moreover, there are non-programmer

[1] Twes+ is pronounced as Twist

developers. In addition to that, there are web pages without considering the mobile limitations. Therefore, transcoding should not be applied on the server. These two goals lead us to apply transcoding over the proxy-side. There are some techniques that are not applied over the proxy without knowledge of server and client sides. These techniques are image consolidation, responsive image-image resizing, image inlining and reducing the number of redirects. The research that is done about image inlining shows that it is not saving energy, actually, it is increasing the energy consumption [53]. At the end, three techniques are applied which are image consolidation, responsive image-resizing images and reducing the number of redirects. In this chapter, the details about these three techniques are covered.

Image consolidation: This technique reduces the number of requests/responses and also the payload of the web page. However, the options stated in literature for using this technique depends on the developer of the web page. This technique is applied by the developers over their web server. This option that works on the web server might reduce the number of requests/responses of the client but it can work only on the web page that applied it. In addition to that, Google PageSpeed Module can be used as well to implement this technique over a reverse proxy server. This reverse proxy server is communicating with the web server and preparing the consolidated version of the images. The reverse proxy stands on behalf of the specific web server and it applies this technique. As a result of the characteristic of the reverse proxy, it only works for the specific web page. Moreover, PageSpeed module has some limitations. Some of those limitations are as follows:

- It only works on CSS background images.
- It supports PNG and GIF image formats.
- The dimensions of the images should be equal or bigger than the CSS dimension declaration. Moreover, the CSS must have explicit dimension declaration in the same declaration as the image URL.

In this research, this technique is used because it reduces the round trip between the client and the server when it is applied by the reverse proxy or the web server. As a result of this observation, reducing the number of requests/responses might be achieved. If the cache size of a mobile device like Samsung Galaxy S3 is 32KB for Level 1 Cache that is used by processor and 1MB for Level 2 Cache is considered [72], caching images of a desktop version web pages do not change much for mobile devices as the size of images is big. Figure 3.1 is an image from a desktop version of xkcd-Sandwich [2]. This image is used for mobile devices as well and its size is 11.5KB. When the size of the image is considered, reducing the round trips can be useful.

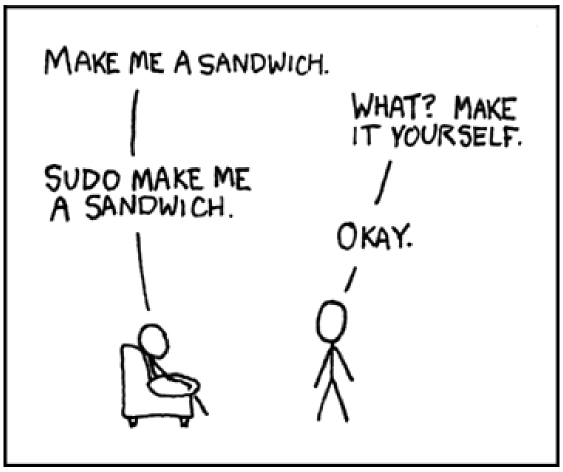


Figure 3.1: Size of the Desktop Version of Image is 11.5 KB from xkcd-Sandwich

To reduce the round trips, the external images can be prefetched from any web server by the proxy server. Those images can be consolidated over the proxy server and this consolidated image can be sent to the mobile device. This way, the number of HTTP requests/responses decreases and the payload reduces as the header of HTTP requests/responses and the header of the image are removed.

Responsive Image or Resize Image: There are three ways for implementing responsive image as it is mentioned in Chapter 2. The first way is to resize images dynamically by the

[2] `http://xkcd.com/149/`

mobile device. This way of implementation solves the screen size constraint of mobile devices. The client does not keep using the device navigation to view the web page. However, this way of implementing does not reduce the network usage of mobile devices, as the high-resolution image is transferred in any case. Moreover, resizing images over the mobile device consumes more energy. As a result, it is not useful for this research goal. The second way is called as responsive web design. On the web server, there are different sizes and the resolution of the same images on the server. The server responses the same HTML source code to all devices. The CSS portion of the web page has options for each type of device, so the device requests the images according to device type. This way, page loading time decreases, the payload decreases, page rendering is faster, bandwidth, memory, battery consumption and latency decrease [26]. However, the second way has to be done by website developers as the server-side should support the different version of the same image. The third way of implementing responsive image is a client-side adaptation. The content of the web page is modified by a JavaScript. The file is downloaded by the browser from the development server. The development server compiles this JavaScript file dynamically for each request. The problem is that client-side adaptation still uses the resources of the mobile device [56].

To use this technique and solve the limitations of each implementation way, the resizing image can be done on the proxy server. Images can be requested by the proxy server from any web server. The response of the server can be resized by the proxy server and sent to the client. By implementing this technique on the proxy server, the client does not have to resize images on the device or apply client-side adaptation for the images. Moreover, the web server does not have to serve the different size and the resolution of the same image for the each type of devices. Although the responsive web design has already benefits as it is mentioned, the goal of this research is to consider the web pages that are developed by the non-programmer developers or the web pages that does not consider the energy consumption of their web page.

Reducing the number redirects: The content of an HTTP message is the status-line, header and the body. The status line includes the HTTP protocol version, the status code and the corresponding explanation of the status code. The status code is three digits integer and the most significant digit of the integer give the class of the response, see Table 3.1 for the status classes [28]. As it can be seen from the Table 3.1, there are multiple status codes for the redirect. The list of the status codes exist for the redirect and their meaning is as Table 3.2.

Table 3.1: The Status Class [28] [MSD:The Most Significant Digit]

MSD	Role	Status Codes
1	Informational	100
2	Success	200, ..., 206
3	Redirection	300, 301, 302, 303, 304, 305, 307
4	Client Error	400, ..., 417
5	Server Error	500, ..., 505

Table 3.2: The Status Code [28].

Status Code	Reason-Phrase
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect

These status codes for the redirects are used for different purposes. The following gives information about these status codes. When a resource is moved and the response has a list of locations for the content that the client chooses the most appropriate one, the status code ”300 Multiple-Choices” is used. Example usage can be the existence of different format or extension of a file. As this status code might change the version of a content, eliminating this redirect is not suitable. The status code ”301 Moved Permanently” indicates that the location of the resource is changed to another one permanently. The response has the new URI, and the client updates the old URI and the bookmarks. This status code has a one-time negative impact but when the client is trying to access the same web page, the client has the new address. The ”304 Not Modified” status code works with the request header as the request header sends the details about the content whether it is modified or not. If it is not modified, the client uses the content that is already downloaded before. This status code reduces the number of requests/responses in a different way. If the response requires a proxy, the server sends a response that indicates the URI of proxy in the location field with status code ”305 Use Proxy”. As there is a proxy usage requirement, eliminating this redirect is not suitable as well. The ”See Other 303” status code is used to show the correct location of the correct response. It does not indicate that web page is moved. These status codes are modifying the responses of the web server. Even they have a negative impact on the client or not, they are making changes. On the other hand, the status code ”302 Found” and ”307 Temporary Redirect” are similar. They indicate that the content is available under a different URI temporarily, accordingly, the reference is not updated to the new URI. The only difference is the request method cannot be changed if status code 307 is used [28]. These two status codes can be reduced as they do not have changes that can affect the client. There is also another redirection which does not have a specific code but it is a problem for mobile devices. It is called redirection loop and these types of loops create network traffic. These types of loops can occur because of wrong redirections. The client side should detect and stop them. This detection is done by the browsers by setting a limit on the number of redirects.

From all above redirects, the temporary redirects are the one that can be eliminated to reduce the round trips. These redirects do not affect any options given by the browser. If a web page has a temporary redirect, the body content of the response is also empty. The header includes a status code and the new location of the web page. Normally when a browser requests a web page and the server responses with a redirect message, client browser checks the status code. The status code indicates that there is a redirect, so the browser takes the new address and sends a new request to the new address [28]. As the negative impact of redirect is discussed in Chapter 2 in Section 2.3, reducing the number of redirects is a good solution to save energy. In this case, by transcoding the web page redirects can be eliminated. When the client requests a web page from a server and if there is a redirect, the proxy can send the rest of the requests instead of the client browser. To support the status code ”302 Found” and ”307 Temporary Redirect”, the request method can be used without changing it. When the proxy receives the actual content, it can response only the actual content to the client. For the infinite redirects, these redirect cases can be tracked and if there are more than a certain number of redirect repetition, then the proxy can return a warning message.

Before finishing the section, there are some statistics related to the number of images and redirections. According to the statistics of HTTP Archive between December 28, 2010, and January 1, 2016, the number of image requests indicates a slightly upward trend and the size of images indicates an upward trend, see Figure 3.2 [4].

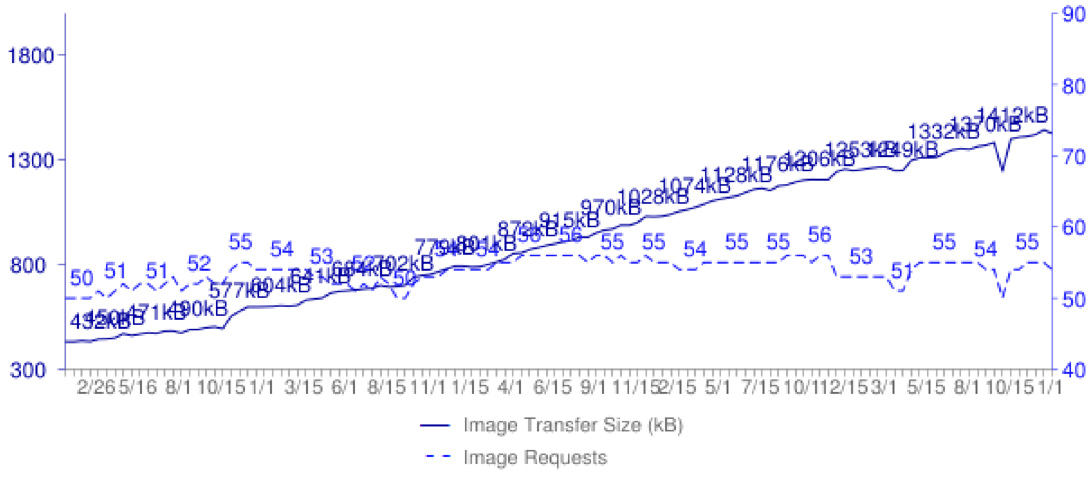


Figure 3.2: Trend of Size of the Images and the Number of Image Requests/Responses [4]

Another interesting statistics is related to the number of web pages includes redirect. Results show that the number of web pages that includes redirect is increasing between December 28, 2010, and January 1, 2016. More interesting thing is that already the number of web pages that has redirect is high, like the end of 2010 the percentage is about 61% and it is still increasing (see Figure 3.3 [4]).

As a summary, by transcoding the web pages over a proxy server, these techniques can be applied to save energy over the mobile devices. Transcoding over the proxy server leads to remove the client and the server limitations and also save energy for the mobile devices. In the next section, the architecture of Twes+ is provided.

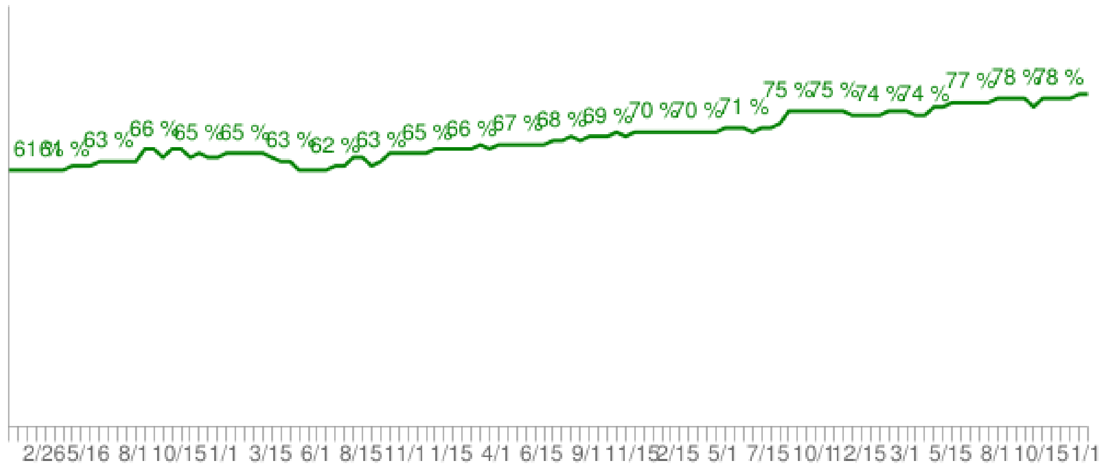


Figure 3.3: Trend of Pages with Redirects(3xx) [4]

3.2 Architecture

The overall deployment architecture of Twes+ has three main components, which can be summarized as follows:

Server: The main purpose of the server is serving web pages for the system. For example, in our Evaluation Chapter, the web pages are downloaded and located on this server (see Chapter 4).

Client: The client is a component that requests web pages from the server. It requires a browser so the user can request the web pages from the server.

Proxy: Implementation of Twes+ is located on this proxy server. The main purpose of this component is to keep tracing the Internet flow between the client and the server. When a response is coming back to the client, it applies the techniques and sends the response on behalf of the server.

The overall network architecture that shows how Twes+ works is illustrated in Figure 3.4. The client, proxy, and the server are connected to each other locally by using a router. When

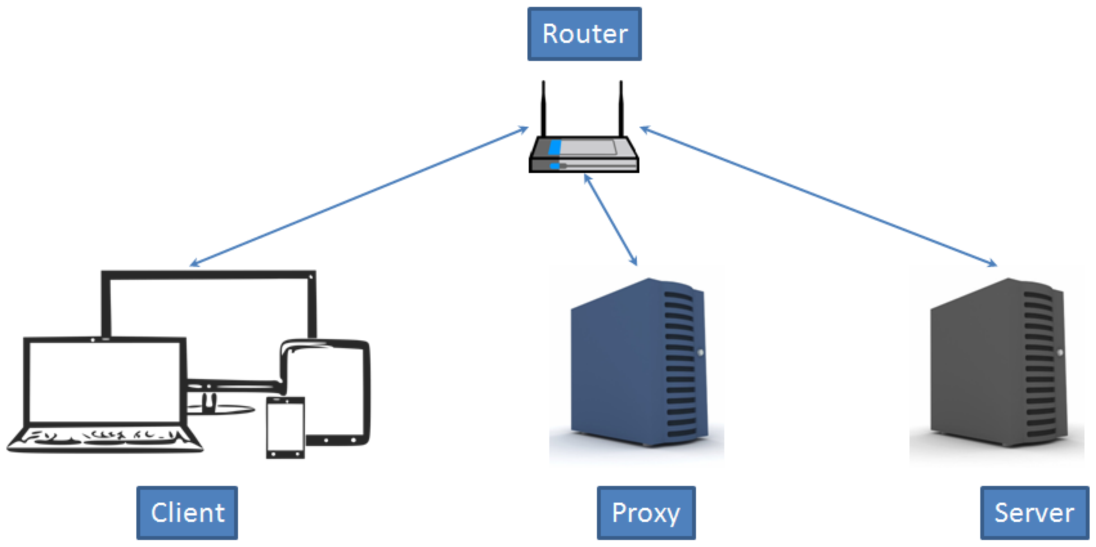


Figure 3.4: The Network Architecture

the client requests a web page from the server, this request goes through the router, then the proxy and finally it reaches to the server. When the server sends the response, the response goes through the router and the proxy. The web page is transformed over the proxy server and goes to the router and then reaches to the client.

3.3 Implementation

In this section, we give the details about the implementation of Twes+. The software architecture of Twes+ is presented in Figure 3.5. The proxy is where Twes+ is placed. For the proxy server, Squid Caching Proxy (A in Figure 3.5) is used. It is an open source caching proxy and it runs over Linux OS. It is developed to cache the frequently used web pages content so response time is faster and payload is less [23]. Although caching is a good option to reduce the number of requests/responses, for Twes+ project the main goal is to intercepting the network and modify the flowing contents. There are plenty of options in squid configuration files and there are different examples that show how to intercept the network and modify the content by using squid proxy server.

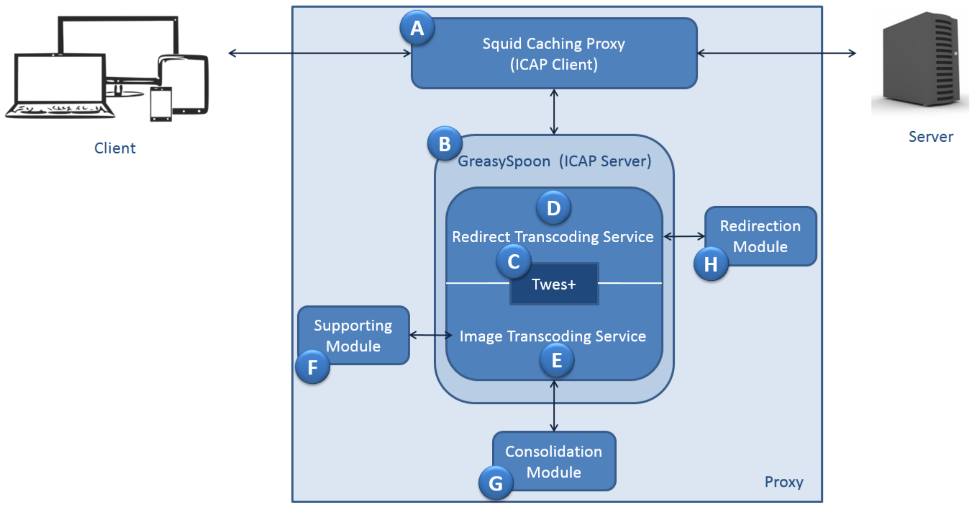
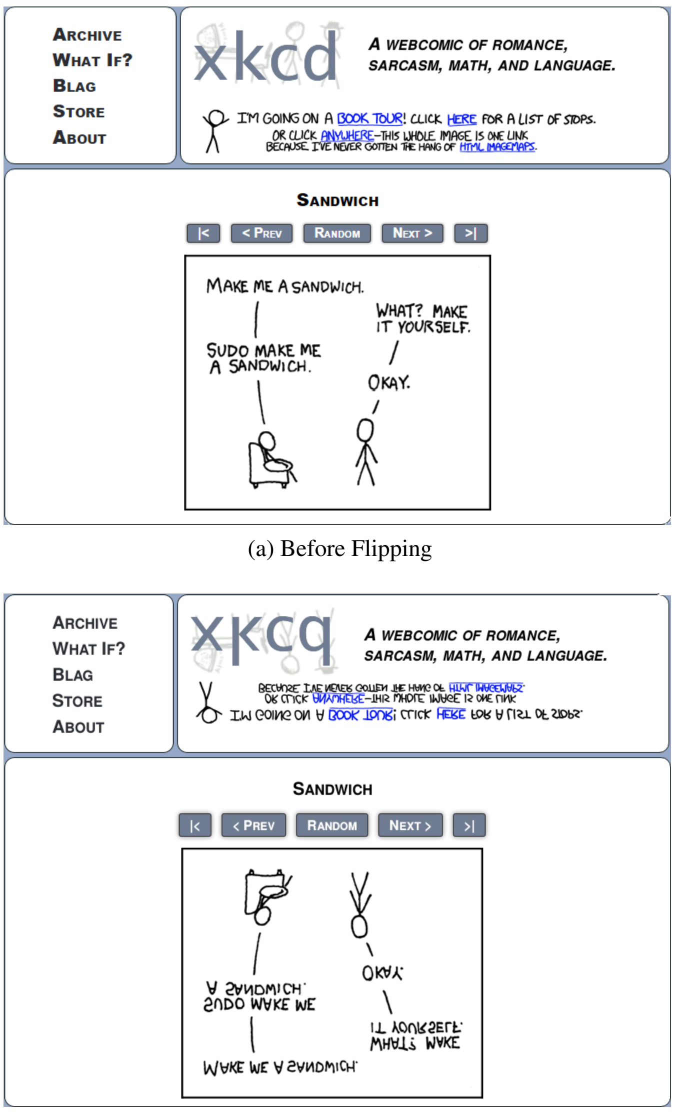


Figure 3.5: The Software Architecture of Twes+

One of the famous example of the content adaptation feature of Squid Caching Proxy over the Internet is flipping images upside-down while the client is browsing. It is developed as an April Fool's prank and it is not related to Twes+ but a useful example of intercepting the traffic, modifying the images and serving them back to the client transparently. It requires a proxy server, a web server, in this case, Apache Server and a script to flip images. By using *url_rewrite_program* configuration of Squid Caching Proxy, Squid Caching Proxy intercepts the traffic and sends the traffic to the script. The script downloads and flips the images and returns the images back to the client. Output of this example can be seen in Figure 3.6 [78, 15].

Although this example intercepts the network and modifies the images, Squid Caching Proxy supports also different content adaptation mechanisms like the Client Streams, the (eCAP), the Access control lists (ACL), the Code hacks and the Internet Content Adaptation Protocol (ICAP) [23]. Table 3.3 gives details about these different content adaptation mechanisms.

Table 3.3 shows that Client Streams and ACLs have limitation related to the modification. To have the request and the response modifications feature, these two options are not eliminated as they do not support. Another criterion to decide which content adaptation mechanism is the


Archive
What If?
Blag
Store
About
xkcd
A webcomic of romance, sarcasm, math, and language.
I'm going on a book tour! Click here for a list of stops.
Or click anywhere—this whole image is one link because I've never gotten the hang of HTML imagemaps.
Sandwich
|<
< Prev
Random
Next >
>|
Make me a sandwich.
What? Make it yourself.
Sudo make me a sandwich.
Okay.

(a) Before Flipping

(b) After Flipping

Figure 3.6: Intercepting the Network and Flipping Images of Requested Web Pages [15].

proxy dependence. To be able to use Twes+ with different systems, it needs to be independent, hence, the ICAP is chosen. The ICAP uses an HTTP proxy as an ICAP client (A in Figure 3.5) and the algorithm over the ICAP server (B in Figure 3.5) can work in different environments. The ICAP is a protocol that is supported by the popular proxies and its main purpose is to receive HTTP messages from the ICAP clients and do the transformations over an ICAP server. It can also modify non-HTTP contents as well. The ICAP client is responsible for the connection to

Table 3.3: The Content Adaptation Mechanisms Supported by Squid Proxy [*:Different Between Versions] [23]

Mechanisms	Request		Response		Proxy Dependence
	Header	Body	Header	Body	
Client Streams	χ	χ	✓	✓	✓
eCAP	✓	✓	✓	✓	✓
ACL	✓	χ	*	χ	✓
Code hacks	✓	✓	✓	✓	✓
ICAP	✓	✓	✓	✓	χ

the ICAP server and the ICAP server is same as the HTTP server [24]. Moreover, the relation between the ICAP client and server is a many-to-many relation.

There are two ways to modify the flowing content. They are request modification and response modification. When a request is directed to the ICAP server by the ICAP client, the ICAP server can take three actions. The first action is to modify the request and send back to the ICAP client and the ICAP client contacts the web server with the modified request. The second action is to send a response to the client without contacting the web server. The last action is returning an error. Some ICAP client can bypass the error so the client does not notice or it can retry the adaptation. There are two actions for the response modifications. The ICAP client sends the response to the ICAP server and ICAP server might send the modified content or an error message. The error action of response modification is similar to the request modification [24].

Squid Caching Proxy is used as the ICAP client. However, an ICAP server is required. In the web page of Squid Caching Proxy, there are some suggested ICAP servers that accept custom adaptations. These are C-ICAP (in C), Traffic Spicer (in C++), ICAP-Server (in Python), POESIA (in Java) and GreasySpoon (in Java and JavaScript) [23]. Three of these options are open-source which are C-ICAP, ICAP-Server, and GreasySpoon. Greasyspoon (B in Figure 3.5) is used in Twes+ because of the language preference and also it is able to support other languages as well like Ruby and Python with the installation. GreasySpoon is inspired from the Firefox GreaseMonkey extension which works and applies the content adaptation over the client. It is also transparent and does not require anything from the client side. This makes GreasySpoon more attractive as well because it is independent of the client platform. It intercepts on-the-fly (i.e., dynamically when content is transferred from the server to the client) content and applies the services to modify the content. Twes+ is a service that works over the GreasySpoon to modify the content to save energy for mobile devices [45].

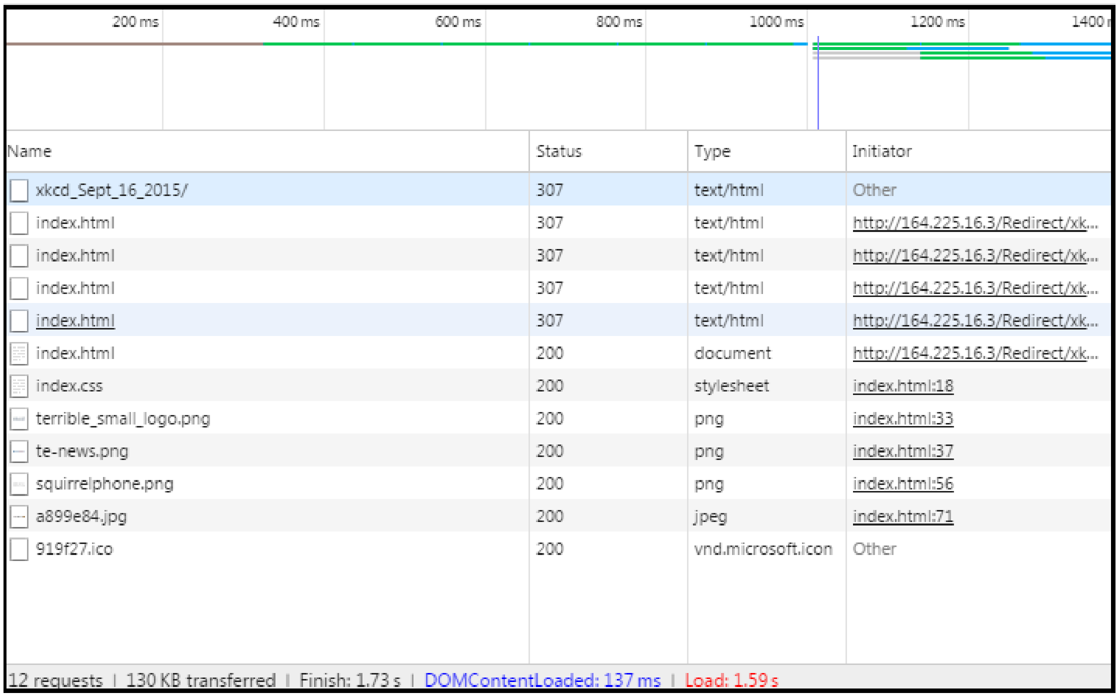
Normally Squid Caching Proxy comes as ICAP feature enabled but otherwise it needs to be built. The connection between GreasySpoon and Squid Caching Proxy is controlled by the Squid Caching Proxy. In the configuration part of the Squid Caching Proxy, ICAP should be enabled and also services, address and port should be specified for both request and response.

Overall dependence of Twes+ is mainly the Internet Content Adaptation Protocol. The input of Twes+ is the response content. Twes+ applies the required transcodings and forwards it as a response to the client. Receiving and sending the content is done with ICAP. Therefore, to be able to use Twes+, the ICAP setting should be done. The ICAP needs a client and a server. During the implementation process of Twes+, minimum dependence is considered so with an ICAP-client and an ICAP-server that allows custom adaptations, Twes+ can run. Twes+ (C in Figure 3.5) includes two different services which are redirect transcoding and image transcoding. In the following paragraphs, these two services are explained.

Redirect Transcoding Service - (D in Figure 3.5)

As it is mentioned in Chapter 2, each redirect carries the overhead of a request/response without the content of the requested web page. These redirects can occur because of different reasons. Temporary redirects are considered for Twes+ as they do not include any modification for the client bookmarks and also do not include an option for the user. Over Twes+ Redirect Transcoding Service, a filter is used for status code 307 as it represents Temporary redirect and if the response comes with status code 307, it goes through the modification. This modification service extracts the redirected address from the location field of the header and sends this address to the Redirection Module (H in Figure 3.5). The Redirection Module takes the address and tracks the new possible redirects. If there are more redirects, then instead of the client, the Redirection Module over the proxy follows the next addresses.

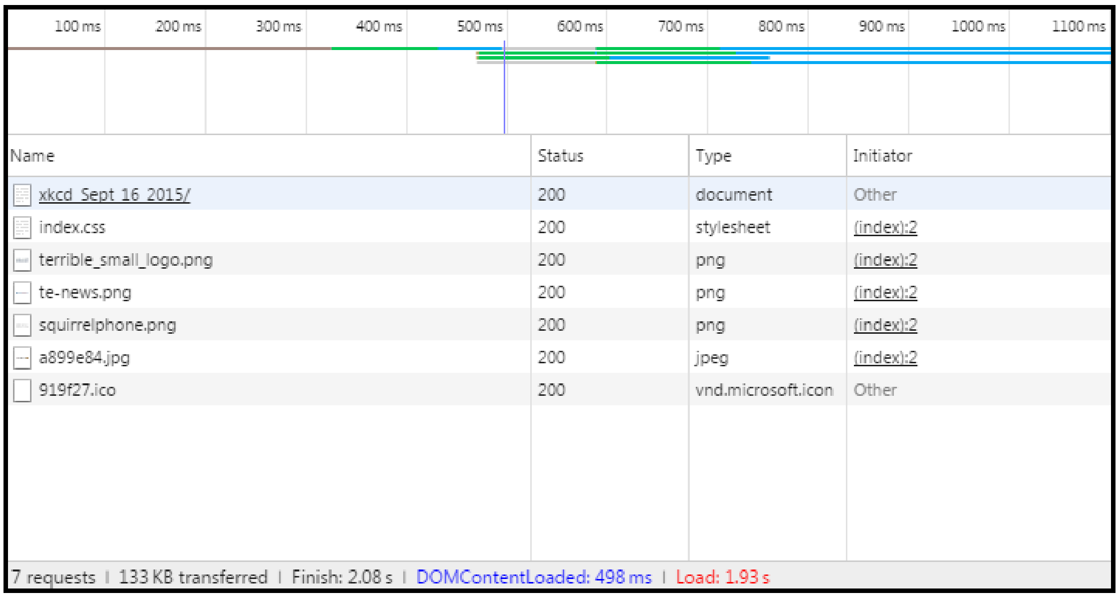
Nowadays, there are no limits on the number of redirects but if limitations of mobile devices are considered, having limitless redirection consumes energy and drains the battery. Previously suggested the maximum number of redirect was five [28]. During Twes+ implementation, this suggestion is used, so, it creates two cases. If the total redirect number is less than six, the Redirection Module returns the address. The body and the header of the first response are modified over Twes+ Redirect Transcoding Service and the modified response is sent to the client. Another case is total redirect number exceeds six and the Redirection Module returns negative. This time, Twes+ returns a simple body that mentions the redirect problem. For both of the case, the client only requests once and receives one response. This way the number of request and response decreases.



Name	Status	Type	Initiator
xkcd_Sept_16_2015/	307	text/html	Other
index.html	307	text/html	http://164.225.16.3/Redirect/xk...
index.html	307	text/html	http://164.225.16.3/Redirect/xk...
index.html	307	text/html	http://164.225.16.3/Redirect/xk...
index.html	307	text/html	http://164.225.16.3/Redirect/xk...
index.html	200	document	http://164.225.16.3/Redirect/xk...
index.css	200	stylesheet	index.html:18
terrible_small_logo.png	200	png	index.html:33
te-news.png	200	png	index.html:37
squirrelphone.png	200	png	index.html:56
a899e84.jpg	200	jpeg	index.html:71
919f27.ico	200	vnd.microsoft.icon	Other

12 requests | 130 KB transferred | Finish: 1.73 s | DOMContentLoaded: 137 ms | Load: 1.59 s

Figure 3.7: The Network Traffic of a Web Page that Contains Five Redirects without Twes+



Name	Status	Type	Initiator
xkcd_Sept_16_2015/	200	document	Other
index.css	200	stylesheet	(index):2
terrible_small_logo.png	200	png	(index):2
te-news.png	200	png	(index):2
squirrelphone.png	200	png	(index):2
a899e84.jpg	200	jpeg	(index):2
919f27.ico	200	vnd.microsoft.icon	Other

7 requests | 133 KB transferred | Finish: 2.08 s | DOMContentLoaded: 498 ms | Load: 1.93 s

Figure 3.8: The Network Traffic of a Web Page that Contains Five Redirects with Twes+

For example, if a web page contains five redirects, the browser of the client receives the original contents after these redirects. Figure 3.7 shows the network traffic without Twes+. The total number of responses are 12 and the status code of the first five responses is ”307 Temporary Redirect”. The size of each response is shown as well. The total duration for redirects and for loading the HTML is 998ms. Figure 3.8 shows the network traffic with Twes+. The total number of responses are seven and there is no response with the status code ”307 Temporary Redirect ”. The duration to load each response is shown as well. The duration to load the HTML is 494ms which is less than the total duration for redirects and loading the HTML without Twes+.

Image Transcoding Service - (E in Figure 3.5)

The impact of images is also mentioned in Chapter 2. These days web pages consist of lots images to make the web page more attractive and the size of images are also increasing (see Figure 3.2). The problem is that each request is adding extra load and caching them over the mobile device is not an important consideration when we consider the cache limitation of mobile devices.

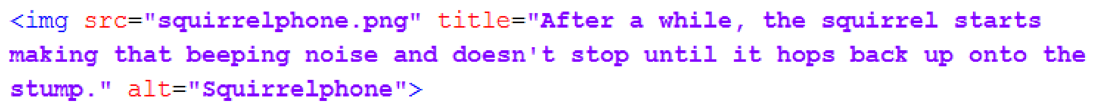
Image transcoding service consists of several different steps but the first step is always the same: The client requests the web page from a web server and this request goes through the proxy. The response is sent by the web server and the proxy receives the response. Thereafter, the proxy (A in Figure 3.5) sends this response to Twes+ that is located in the ICAP server (B in Figure 3.5). For the Image Transcoding Service (E in Figure 3.5), it only filters the content type HTML. This way any HTML source that is sent to the client and either it is external or not can be received by Twes+. If the HTML of the web page does not include any image tag <*img* >, these web pages are directly sent back to the proxy without an adaptation. However, if the web page contains image tags, all the image tags are tracked in the HTML to extract the location of the requested images. Before explaining the next step, brief information about image tags is given in the following, which is useful as the information inside the image tag is used for the transcoding.

Image tag of HTML is used to define images inside the HTML and it requires at minimum two attributes which are the URL source <*src* > of the image and an alternate text <*alt* > for the image. There are other attributes that can be included in the image tag, but only two of these attributes are required for Twes+. These attributes are the dimensions of the image which are the width and the height. The metric of these two attributes are pixels, see below, for an example image tag from xkcd-Squirrelphone [3]. After extracting the image tag from the HTML, three attributes are searched which are *src*, *width*, and *height*. As dimensions are not a required attribute, their default value is set to zero.

Before explaining how these dimension attributes are used in Twes+, CSS also can be used to define the dimensions of an image. In CSS, the dimension of any element of HTML can be defined. If the dimension of the image is not defined in the image tag of HTML, it means either the dimensions of the image is as required or it is defined by using CSS. However, in CSS, every element inherits all the inheritable properties from the parent element [69]. The dimensions are inheritable properties, thus, if the dimensions of an image are defined by using the CSS, it is

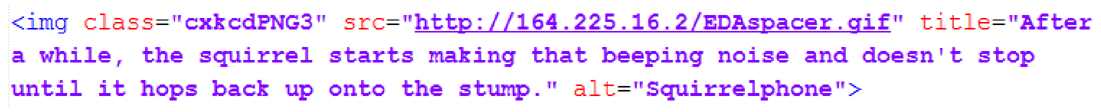
[3] http://xkcd.com/1578/

hard to find from where these dimensions are coming. Moreover, the CSS can be external so the image can get its dimensions from another file. Hence, this unpredictability of CSS, its current implementation Twes+ does not support the CSS.



```
<img src="squirrelphone.png" title="After a while, the squirrel starts
making that beeping noise and doesn't stop until it hops back up onto the
stump." alt="Squirrelphone">
```

(a) The Original Image Tag from xkcd-Squirrelphone



```
<img class="cxkcdPNG3" src="http://164.225.16.2/EDAspacer.gif" title="After
a while, the squirrel starts making that beeping noise and doesn't stop
until it hops back up onto the stump." alt="Squirrelphone">
```

(b) The Modified Image Tag from xkcd-Squirrelphone

Figure 3.9: The Original and the Modified Version of the Image Tag

After extracting the three attributes, these attributes are sent to the Supporting Module (F in Figure 3.5). The Supporting Module is written in Shell Script. In the Supporting Module, images are downloaded from the source to the proxy. The dimensions of the images are controlled with ImageMagick. ImageMagick is a free software for images. The same tool is also used for flipping the images upside down. The dimensions of the downloaded images and the image tag are compared. This comparison leads to some possible cases. If there is a specified dimension in the image tag, it is the required dimensions for the image to make it fit into the layout of the web page. So in case dimensions of image tag are not zero and they are not equal to the original image dimensions, it means the dimensions of the image should be fixed. To change the dimensions of the original image, ImageMagick is used. If the dimensions of image tag are not zero and they are equal to the original image dimensions, no modification is required. If dimensions of image tag are equal to zero, the original image is kept as it is downloaded from the web server.

In its current implementation, Twes+ supports three type of images which are Joint Photographic Experts Group (JPEG/JPG), PNG and GIF. It only supports these formats because they are the most popular formats, see Figure 3.10. In the Supporting Module, the format of the image is found by using ImageMagick. Normally, expected format of the image is what is written in the *src*. However, in reality, the images that are downloaded from the web server might have a different format or might be non-image files. To solve the possible errors, for each image always format is checked. According to the format of the images, each image gets a new name with the image format, website name and a number based on the order of this format appears in the HTML. The reason for changing the name of the images is explained in the Consolidation Module. In the Supporting Module, there is one more important point that needs to be considered. If images with GIF format do not have animation, they can be easily consolidated. However, if they are animated, it means that these images are a combination of multiple images. Twes+ does not support animated images as these images cannot be consolidated. Animated images do not have fixed number of images to be combined to create the animation. Figure 3.11 shows 28 JPEG formatted images that are used to create an animated GIF image. It is clear that there are no common dimensions. Moreover, another animation can be a combination of a different number of JPEG image.

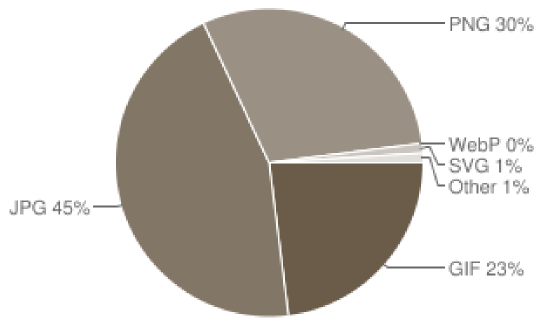


Figure 3.10: The Format Distribution of Requested Images [4]

At the end, the Supporting Module (F in Figure 3.5) fixes the dimensions of the images if it is required and changes the name of the image according to the format of the image. If any error occurred, the Supporting Module returns negative so this image tag is not modified over the HTML. Otherwise, the image tag in HTML is modified for the Consolidation Module (G in Figure 3.5). According to the new name of the images and the number in the image name, a name is created for each image for the next step. After all the image tags in HTML are checked, modified and images are downloaded with the Supporting Module, the Consolidation Module is

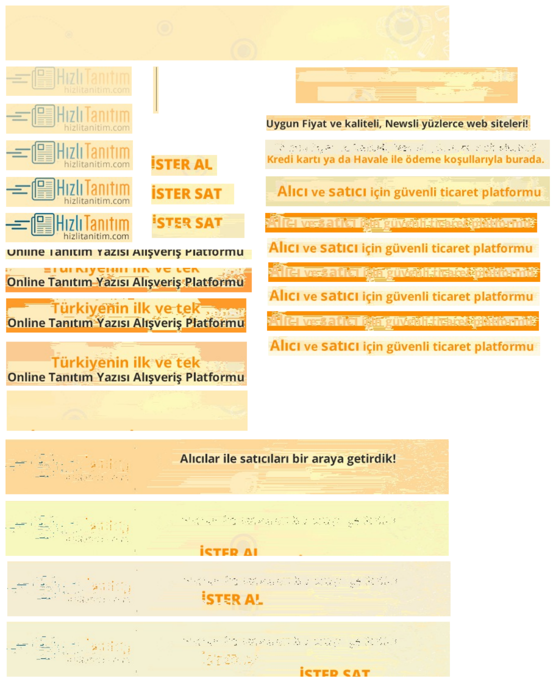

HızlıTanıtım
hizlitanitim.com
İSTER AL
İSTER SAT
Uygun Fiyat ve kaliteli, Newsli yüzlerce web siteleri!
Kredi kartı ya da Havale ile ödeme koşullarıyla burada.
Alıcı ve satıcı için güvenli ticaret platformu
Online Tanıtım Yazısı Alışveriş Platformu
Türkiyenin ilk ve tek
Alıcılar ile satıcıları bir araya getirdik!

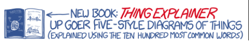
Figure 3.11: The GIF Formatted Image Consist of 28 JPEG Formatted Images

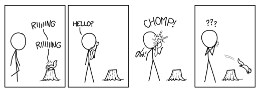
called. Modified version of the original image tag in Figure 3.9a is like the image tag in Figure 3.9b. The IP address *164.225.16.2* is the IP address of the proxy itself and *EDAspacer.gif* is a 1x1pixels transparent image to control the HTML layout only.

The Consolidation Module (G in Figure 3.5) is used to concatenate images by only controlling the HTML source codes of the web pages. As it is mentioned before that Twes+ supports JPEG/JPG, PNG and GIF image formats because of their popularity (see Figure 3.10). It is already tested that converting all images on a web page into JPEG/JPG format saves energy for mobile devices, like on Facebook 30% and Amazon 20% energy saving [59]. This saving occurs because JPEG/JPG is better in compression and rendering then PNG and GIF. The JPEG/JPG uses lossy data compression but PNG and GIF use lossless compression [59]. However, Twes+ keeps the format of the images because of some feature of PNG and GIF formats. For example, the PNG and the GIF images might be transparent and when they are converted to JPEG/JPG, they lose their transparency. The format of the images is not converted because the goal of

Twes+ is not to modify the look&feel of the web pages. Another important thing is that the number of images that are concatenated. Having one image for each format is not logical because if a web page contains too many images, this makes the concatenated image size too large and sending a large image is not safe in case of any failure. As a result, at maximum ten images are concatenated into one image. Ten images are chosen because in HTML the corresponding image is found by mode ten and also in the Consolidation Module Wildcards are used and they work on base ten. According to these concerns, the Consolidation Module uses Imagemagick and combines the images according to their format ten by ten. It gives another decoded name like *Out*, the format of the image, name of the web page and a number based on the division of concatenated images number by ten.

(a) The First Image


NEW BOOK: THING EXPLAINER
UP GOER FIVE-STYLE DIAGRAMS OF THINGS
(EXPLAINED USING THE TEN HUNDRED MOST COMMON WORDS)
(b) The Second Image



(c) The Third Image

Figure 3.12: The Individual Images Before Concatenation from xkcd-Squirrelphone

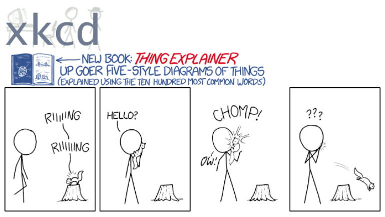

xkcd
NEW BOOK: THING EXPLAINER
UP GOER FIVE-STYLE DIAGRAMS OF THINGS
(EXPLAINED USING THE TEN HUNDRED MOST COMMON WORDS)
RIIIING
RIIIIING
HELLO?
CHOMP!
OW!
???

Figure 3.13: The Concatenated Version of All Images in Figure 3.12 [Width:679 pixels Height:371 pixels]

Figure 3.12 contains three PNG formatted images from xkcd-Squirrelphone [3]. The dimensions of these images: the first image width:185pixels x height:83pixels, the second image width:483pixels x height:64pixels and the third image width:679pixels x height:224pixels. These images are concatenated by the Consolidation Module (G in Figure 3.5) and Figure 3.13 is created. The dimensions of this image are width:679pixels x height:371pixels. Images are concatenated vertically and for splitting the images from the concatenated version inside the CSS, the exact position of the images is required. For each ten consolidated image, cumulative height is used and a CSS style is created for each image. For example, the exact start location of the second image is the height of the first image where exact start location of the third image is the height of first image plus the height of the second image and so on.

At the end of the image transcoding, the images of the web pages are found by the image transcoding service and each image is downloaded by the Supporting Module. Image tags are replaced by the new ones (like Figure 3.9b) and created CSS style for those image tags are included into the HTML. Including the CSS style into HTML is a better option than including as an external resource because, in that case, no extra round trip is added and HTML is compressed, thus, the payload is less.

There is also image resizing or responsive image included into the image transcoding service. For this technique, the type of the device is found at the beginning. Resizing and making the images smaller is not a good option for the desktop but it is required for the mobile devices. At the beginning, the device model number (i.e., GT-I9505 is for Samsung Galaxy S4) is searched in the header of the request. According to the device type, a different percentage is set. For the desktops, the percentage is kept as default (100%) but for the mobile devices, the percentage is set to 50%. This percentage is used in the Supporting Module (F in Figure 3.5) and if the percentage is not equal to 100%, each image is resized 50% smaller. After resizing operation, the process continues as above.

3.4 Pilot Study

After the implementation, a pilot study is conducted. The materials are picked from Alexa. The first five suitable web pages are selected from the top 100 web pages for Turkey. These pages are located on our local server and Image Consolidation Service is tried with these five web pages. Some issues occurred during this pilot study. After these issues are solved, the evaluation of the Twes+ is conducted. The first issue is the calling these local web pages with their domain name instead of their IP address and location. For the updated name of the images, the name of the web page is extracted from the request header. However, as we use a local network, the name of the web pages was the IP address of the server and the folder names. After this issue, all the web pages are served by their local domain name.

The second issue is iframe tag of the web pages. The web pages might include other HTML files inside an iframe tag as a frame. The image consolidation service tracks any HTML source code that is sent as a response from the web servers. If the web page includes multiple HTML source codes, these source codes proceeds as an individual web page. They have their own consolidation process. The problem is again the domain name of the web pages. At first, we extracted the first part of the domain name but to solve this problem we extract all the components in the domain name. For example, the original domain name is `www.example.com` and the iframe is `www.example.component.com`. This case for the name variable of the main source file is *example* while the name variable of the iframe source is *examplecomponent*.

The last issue is the images of the web pages. Sometimes, the developers are adding non-image sources inside an image tag. This created a crash on the service and service was by-passing the web page without modifying the source code. Inside the supporting module, a control section is included for these web pages to prevent the further processes.

3.5 Summary and Open Data

This chapter has described our study called Twes+, transcoding web pages to save energy during browsing. The literature review shows that three techniques have the potential to reduce the round trips or the payload of the web pages (see Chapter 2, Section 2.3). These techniques are: (a) Reducing the number of redirects, (b) image consolidation and (c) image resizing. In the literature, those techniques are not implemented at all or they are implemented over the client or the server sides or the reverse proxy. Our approach is implementing services that stand on a proxy server and applying the three transcoding techniques over the proxy. The local system architecture is used and Twes+ is simulated over this local system. The overall advantage of those techniques is that they reduce the number of requests/responses of the client and the payload of the web pages. The source code of the implementation can be downloaded from `https://github.com/EdaKoksal/Twest.git`. In the next chapter, the effect of those techniques on energy saving while browsing web pages is investigated.

CHAPTER 4

EVALUATION

In the preceding chapters, the transcoding techniques and Twes+ to save energy while browsing the web pages on the mobile devices is described. In order to validate energy saving of Twes+, two separate evaluations with and without using Twes+ are conducted over two different devices, the desktop and the mobile device. Evaluation is repeated with and without Twes+ to see the impact of Twes+ over both devices. In this chapter, the tools, the evaluation methods and the materials are described and the results of these experiments are presented and discussed.

This chapter is organized as follows: First, Section 4.1 presents our research questions. Section 4.2 presents the materials that are used during the evaluation process and the selection of these materials are discussed. The next section describes the details about the used equipment during the measurements. The following section is about the tools and their details. Section 4.5 describes our methodology used for evaluation. Section 4.6 presents the results of the experiments. Finally, Section 4.7 presents the discussion of the results.

4.1 Research Questions

This study aims to investigate whether or not Twes+ can save energy during browsing by reducing the number of requests&responses and/or reducing the number of redirects and/or resizing the images. Thus, the following questions are asked:

1. Redirect Transcoding Service: The goal is to validate whether or not Twes+ Redirection Transcoding service can reduce the number of redirects and Twes+ can save energy. That's why we ask *"Does Twes+ allow saving energy by applying redirect transcoding to reduce the number of redirects?"*

2. Image Transcoding Service: We would like to validate whether or not Twes+ can save energy by applying image transcoding techniques. Thus, we ask *"Does Twes+ allow saving energy by reducing the number of request/response between client&server and/or reducing the payload between client&server?"*

4.2 Materials

The web page that is used to validate the reduction in redirection number is xkcd [1]. For this test, only this web page is used because the content of the page is not important. The web page is mainly used as a control mechanism and it can be any web page. The same web page is used for all the cases that are created to test redirect transcoding service. The important difference between the cases is the number of redirects to access the web page. By using this web page three cases are created:

Case 1: The case is created by using the web page without any redirects, so when the client requests the web page, the web page can be accessed directly.

Case 2: The suggested maximum number of redirect is five [28, 81], hence more than five redirection indicates an infinite redirection loop. We follow the suggestion and by using the same web page we create five redirects. To create the redirects, *.htaccess* file of apache server over the server is used.

Case 3: For the last case, again the same web page is used and an infinite redirection loop is created over the server. In this case, the client does not receive the web page that is used as a control mechanism. Instead of the content of the web page, a warning page is returned back to the client to inform the client about the infinite redirection.

[1] http://xkcd.com/1578/

The web pages that are used in image transcoding service validation experiment are selected from the most popular 100 web pages from Alexa [2] on August 20, 2015. However, some web pages are not suitable for the tests according to the criteria. The first criterion is the language of the web page. If the language of the web page is not English, these web pages are not suitable. It is important to be sure the look&feel of the web page is not modified, for example, the language of Sina [3] is Chinese. Another criterion is whether or not the web page is a duplicate of an existing web page. For this criterion Yahoo Japan [4] or Google India [5] can be an example. If a web page is repeated in the top 100 list, these duplicates are eliminated.

Another criterion is the security of the web page. If a web page is using HyperText Transfer Protocol Secure (HTTPS), it means that the data transfer between the client and the server has security. For the measurements, the web pages need to be served locally and if the web page is using the HTTPS, the web server does not allow downloading some contents. As a result, the HTTPS pages are not suitable to be able to serve the materials locally with the same look&feel as the original one. The final criterion is the need for an account. If a web page requires an account verification to be able to show its content, these web pages are eliminated. During the measurements, everything over the browsers is deleted so the account information cannot be saved, for example, VKontakte [6] requires an account. According to these criteria, 30 web pages left and from these 30 web pages, three web pages are eliminated because of their contents. The content of two web pages is pornographic and one of them is a torrent web page. The rest of the 27 web pages are downloaded.

After the first eliminations, the web pages are downloaded to the local web server. Several different tools are used to download these web pages. After the web page is downloaded, the look&feel of the original web page that is served by the original web server and the web page that is served by the local web server are compared. If the look&feel of any downloaded version

[2] alexa.com
[3] `Sina.com.cn`
[4] `Yahoo.co.jp`
[5] `Google.com.in`
[6] `Vk.com`

Table 4.1: The List of the Web Pages from Alexa on August 20, 2015. [^{a}: Does not Have External or Supported Image Format, ^{b}: Does not Look as the Original Web Page]

Rank	Name	URL
17	Ebay	Ebay.com
24	Bing[a]	Bing.com
27	Msn[b]	Msn.com
28	Microsoft	Microsoft.com
29	Ali Express	Aliexpress.com
33	Ask	Ask.com
38	Onclick[a]	Onclickads.net
43	Imgur	Imgur.com
47	Imdb	Imdb.com
48	Apple[a]	Apple.com
50	Fc2	Fc2.com
52	Google add services[a]	Googleaddservices.com
55	Stackoverflow	Stackoverflow.com
57	Craigslist[b]	Craigslist.org
59	Diply[b]	Diply.com
61	Odnoklassniki	Ok.ru
62	Alibaba[b]	Alibaba.com
71	Outbrain[b]	Outbrain.com
72	Booking	Booking.com
75	Nicovideo	Nicovideo.jp
76	Flipkart	Flipkart.com
77	Cnn[b]	Cnn.com
79	Go[b]	Go.com
85	BBC	Bbc.co.uk
95	Dailymotion	Dailymotion.com
96	Wikia	wikia.com
97	China Daily	chinadaily.com.cn

of the web page does not look like the original web page, these web pages are not used. Another issue that is faced, the web pages with the unsupported format of images. These web pages do not go through the image transcoding service, thus, these pages are also not suitable for the evaluation. As the last control over these web pages is the errors. When the web page is opened by a browser over the system itself, the browser does not try to connect any external link and page load finishes quickly. However, during the validation test of Twes+, there is a local network and the web pages on the server are accessed by the client. Because of this local network, the browser over the client tries to access external links and tries to load all the files that give errors. All the web pages source codes are checked by using the Google Chrome Browser Developer Tool. The errors are fixed, the missing files are downloaded, links that are not local are changed to their local version and the external links like Google Analytics are removed from the web pages. After these changes, if the web page does not look like as the original web page, these web pages are removed. After all these, 16 web pages are used in the evaluation (see Table 4.1). While downloading and preparing the web pages, some issues occurred. The details about these issues are discussed in Appendix C.

4.3 Equipments

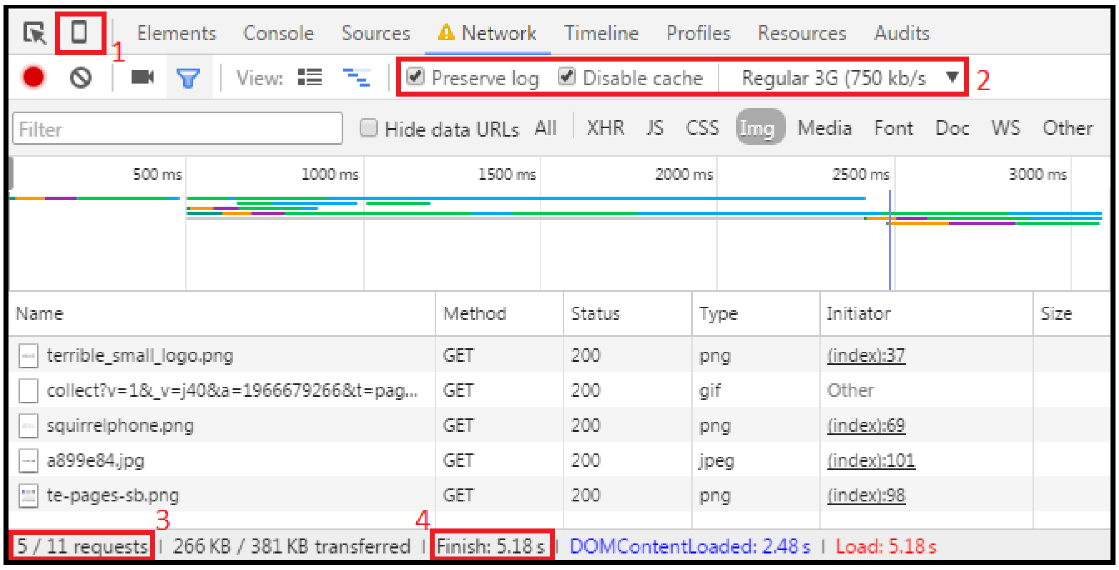
In order to do measurements, the overall architecture needs to be controlled. There are a proxy server, a web server and a client (see Figure 3.4). The server, the proxy, and the client are identical three desktops for the hardware component wise. Their processor is Intel Core i7, model: 4770. The base frequency and the turbo frequency are 3.4 GHz and 3.9 GHz, respectively. Installed memory is 16 GB and the processor has 4 cores. The specified TDP power is 84 Watts as it is mentioned in Section 4.4 [16]. The brightness of the screens is set to 50%. Ubuntu 14.04 operating system is used for the server and the proxy and Windows 7 is used for the client.

As a switch between these three desktops, Cisco Catalyst 2960-X Series is used and its speed is 80 Gbps (Gigabits per second) [80]. Some of these devices are used only for the measurements over the desktop. For the measurements over the mobile device, the client desktop is replaced with Samsung Galaxy Ace (GT-S5830i). Its processor is Qualcomm Snapdragon S1 MSM7227-1 Turbo and it is frequency is 800MHz. It is Level 1 Cache is 16Kb and Level 2 Cache is 256KB. The display resolution is 320x480 and the brightness is 50% and it is unified over all the measurements. Only a 4GB microSD card and Powertutor application are installed. The operating system over the mobile phone is Android version 2.3.6 Gingerbread. Hence, the mobile device can connect to the local network via Wi-Fi, the switch needs to be also replaced with Tp-Link Wireless Router (TL-WR840N) and its speed is 300 Mbps (Megabits per second).

4.4 Tools

The evaluation part of Twes+ is done by measuring the power and/or the energy consumption and the number of request/response round trips of the mobile device before and after transcoding. There are different tools like YSlow, Powertop, etc. but the following tools are used for specific reasons.

Devtool: It stands for the Google developer tools and it is developed by Google developer team [19]. There are alternative tools to Devtool, but for the experiments, Devtool is used to simulate 3G network. This tool can simulate and support the 3G network. Devtool is built into the chrome browser and it allows to record the loading process of the web pages. Inside this tool, there are eight subcategories; elements, resources, network, sources, timeline, profiles, storage, audits and console panels. During the Twes+ development, elements panel is used to inspect the HTML and CSS of the web pages. The network panel is also used, thus, it can show the duration of each request/response over the network. To investigate the number of requests/responses and the duration of the page load, the network panel is used and Figure 4.1 shows a screenshot of the panel. The rectangle number one is for enabling the device mode over the browser. This module is used to emulate the different devices as there is a device selection option. During the development of Twes+, the device mode is used to test image resizing feature of image transcoding service. The rectangle number two is to preserve the timeline logs and also to disable caching as during all the tests the cache needs to be disabled. From this part network throttling is also selected. There are several options but for the Twes+ evaluation, Regular 3G is used. The 3G network is used instead of Wi-Fi, as 3G network has larger coverage, while for Wi-Fi, the user has to be in certain range. Moreover, considering the portability of mobile devices, 3G network is preferred for the tests over the desktop by using the network throttling. The rectangle number three in Figure 4.1 represents the number of image responses vs. the number of total responses that is required to load the web page. The rectangle number four represents how long it takes to load the page. During the power and the energy measurement, the network throttling is set and the caching is disabled. The timeline feature of the tool is not used as it affects the power and the energy measurement. To sum up, by using Google Devtool, a web page can be investigated and assessment of the research can be done [19].



☑ Preserve log ☑ Disable cache Regular 3G (750 kb/s

☐ Hide data URLs

Name	Method	Status	Type	Initiator	Size
terrible_small_logo.png	GET	200	png	(index):37	
collect?v=1&_v=j40&a=1966679266&t=pag...	GET	200	gif	Other	
squirrelphone.png	GET	200	png	(index):69	
a899e84.jpg	GET	200	jpeg	(index):101	
te-pages-sb.png	GET	200	png	(index):98	

5 / 11 requests | 266 KB / 381 KB transferred | Finish: 5.18 s | DOMContentLoaded: 2.48 s | Load: 5.18 s

Figure 4.1: The Google Developer Tool - The Network Panel

Intel Power Gadget (version 3.0.2): As the processor of the desktops is Intel, a tool is required that is allowed to read the data from the Intel processor. Intel Power Gadget is a tool developed by Intel Corporation for Windows, Linux, and Mac operating systems. It supports Intel Core processors from 2nd Generation up to 6th Generation Intel Core processors but still it does not support Atom processors that are mainly used for the mobile devices. The purpose of developing this tool is power estimation without any hardware instrumentation. It estimates power data by reading the data of registers and energy counters of the processor, so it can only work for Intel processors. The GUI of the tool is as Figure 4.2. This figure contains three graphical sections. The first section (A in Figure 4.2) shows the current package power consumption (i.e., Cores, GPU, Cache ...etc.) and the limit of the current average package power which refers to the Thermal Design Power (TDP) of the processor installed. The TDP is the average power of processor when it works at Base Frequency for a high-complexity workload. The next section (B in Figure 4.2) shows the frequency of the package and the base frequency of installed processor. The frequency of operation affects the systems energy. By tracking the operating frequency, the sleep states can be figured out, as, during the sleep state, the frequency of operation is set to the lowest possible. For the evaluation of the Twes+, these two sections are used to see whether the system is at the stable state to start the measurement or not. The last section (C in Figure 4.2) is for the temperature of the package and the max temperature that system can reach [89, 16]. At the end of the measurement, the tool creates a log file that contains the cumulative Intel Architecture (IA) and processor energy and the average IA and processor power. This tool is used for the measurements that are done on the desktop client.

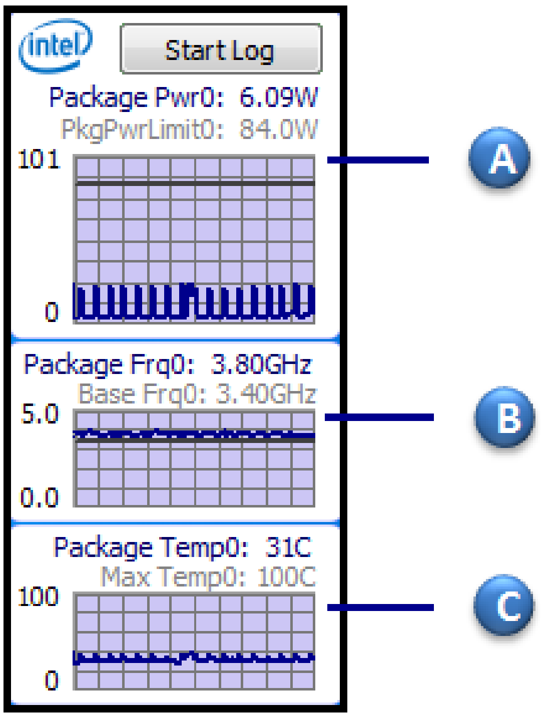


Figure 4.2: The Intel Power Gadget

PowerTutor: It is a tool to monitor power consumption of Android mobile devices. In our knowledge, this is the only tool that gives the detailed power consumption of the hardware components. It is released 2011 and it shows the power consumption of major components like processor, network interface, display, etc. for the system. For measurement, the tool is using device power management states with accuracy ± 5%. PowerTutor is using the power state of components to estimate the power consumption of the mobile devices. For instance, the tool is using the CPU utilization and the frequency level to determine the power consumption of the processor, the brightness level for display, the data rate and transmitted packets per second for Wi-Fi, the transmitted packets per second and the power states for 3G [85]. When the total power and the CPU power is stable, this tool is used for the measurements on the mobile device.

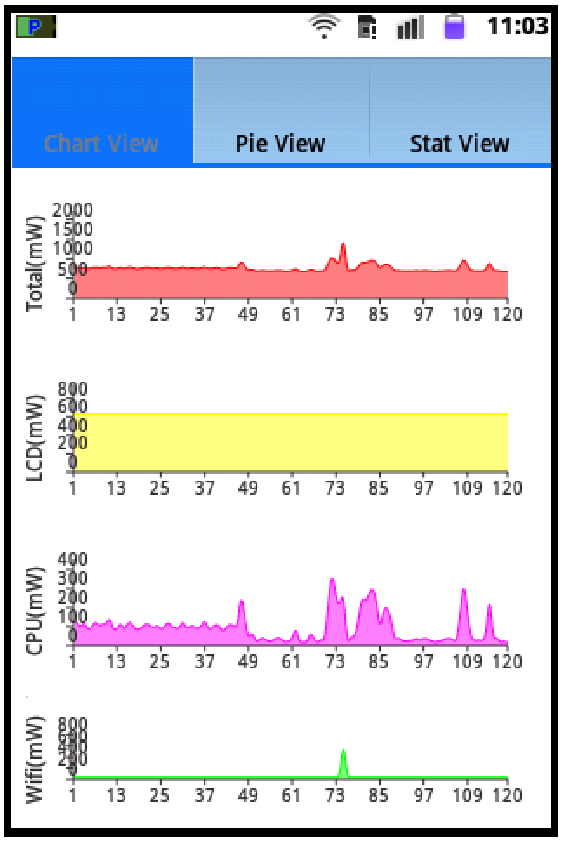


Figure 4.3: The PowerTutor Application (Colors are inverted)

4.5 Methodology

The assessment of this research includes two divisions, which are experiments (measurement) without Twes+ and with Twes+. The evaluation metrics are decided according to research questions. The first research question is about redirect transcoding. The redirect transcoding service purpose is to reduce the number of request/response round trip to save energy. Therefore, the number of requests, page load duration and power/energy are chosen as metrics. The second research question is about the image transcoding. The image transcoding purpose is reducing the request/response round trip and also reducing the payload by resizing the images to save energy. For this research question, the size of the web pages is not included as it is obvious that the size of web pages drops. To evaluate the impact of payload over the energy saving, the power consumption is measured with and without resizing the images. As a summary, our measurement metrics are the number of requests, the page load duration, the power/energy.

Figure 4.4 represents the setup for the experiments without Twes+. In this setup, as a regular case, there are the client and the server over a local network. The second setup is represented in Figure 4.5. In this setup, Twes+ stands between the client and the server. All the network traffic between the client and the server goes through the Twes+. As it is mentioned at the beginning of Chapter 4, experiments are done for two different clients. The first client is a desktop and the second client is a mobile phone. The measurements are done on the desktop because if there is an improvement over the desktop which has a powerful processor and has a limitless source, there will be an improvement over the mobile device which has a less powerful processor and limitations. In order to illustrate the improvements of Twes+, measurements are done on the desktop as well.

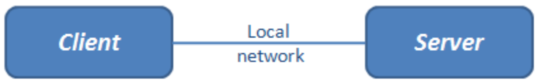


Figure 4.4: The Architecture without Twes+

When the client is a desktop, the web pages are requested from Google Chrome Browser with network throttling which is used to simulate Regular 3G (750 kb/s 100ms RTT). To measure the number of requests/responses and the duration, Google Chrome Browser Developer Tool and to measure the power and energy consumption, Intel Power Gadget are used and the measurement procedures are different. The cache & the history of the browser, the cache & the data of the

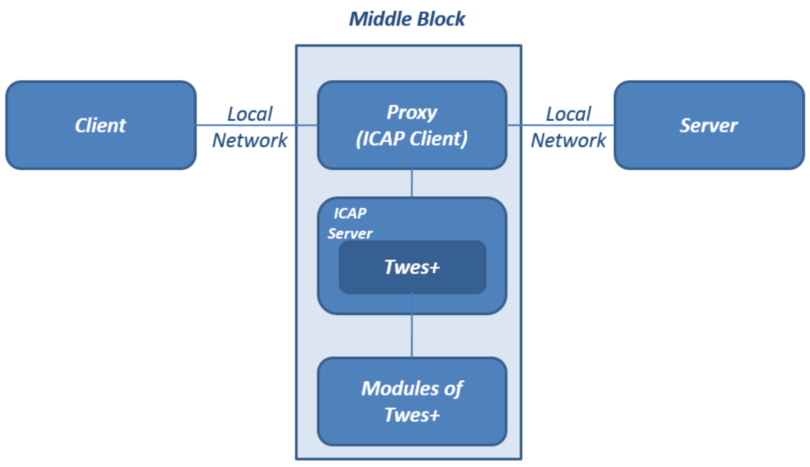


Figure 4.5: The Architecture with Twes+

proxy and the cache of the server are cleared between each web page requests. The power and the energy measurement is done when the system totally idle and during the measurement, only Google Chrome Browser and Intel Power Gadget are active. Duration of the measurement is 30 seconds. The first ten seconds are idle measurements and the next 20 seconds are the web page loading duration. Page load duration is mostly shorter than 20 seconds but to have a common pattern 30 seconds is used. For the measurements that are done on the Google Chrome Browser Network panel, there is no time limitation as network panel itself is taking time.

When the client is a mobile device, the web pages are requested from the built-in browser of Samsung called Internet over Wi-Fi. Powertutor application is used to measure the power consumption during browsing. The measurement procedure is as follows. The cache and the history of the browser are cleared between each web page request. This measurement duration is two minutes. When Powertutor shows the entire system idle, the browser is launched and waited for 30 seconds. After waiting 30 seconds, the web page is requested and during one minute there is no other interaction with the mobile device. After one minute, we switch back to the Powertutor and wait for another 30 seconds and pause the measurement. If there is any fractionation occurred, the test is repeated. During the measurements over the mobile device, the device is connected via Wi-Fi because we do not want the Internet to affect our measurements.

Twes+ is developed to be an additional tool for saving energy on the client-side. It is not the alternative to any other techniques or the browser plugins. Therefore, a comparison-based evaluation is performed between with and without Twes+. For example, during the comparisons, caching features are disabled to see only the real impact of Twes+ on the energy saving. However, the caching or other options might be enabled to increase the saving. In addition, to the reasons behind the comparison evaluation of Twes+, for Image Transcoding Service testing in the wild is preferred instead of the controlled experiment. This way the real impact of Twes+ on the energy saving can be reflected by using the real web pages that are used by the normal users.

4.6 Results

In this section, the results are presented based on the two research questions that are posed in Section 4.1.

Redirect Transcoding Service

"Does Twes+ allow saving energy by reducing the number of redirects?"

Three cases are implemented: a web page without a redirect, a web page with six finite redirects and a web page with an infinite redirect. The total number of requests/responses and the page load duration differences are represented in Table A.1 in Appendix. The difference in the number of requests/responses is also shown in Figure 4.6. The shortening *w/o Twes+* represents the data of requesting each page without Twes+ and the shortening *w/ Twes+* represents the data of requesting each page with Twes+. These data are extracted from Google Developer Tool. For example, case 2 results show that the total number of requests/responses without Twes+is 12 and with Twes+ it remains 7. This difference is coming from the redirects that occur. The duration of loading the web page is changing between with and without Twes+, see Table A.1. For the case 2, the web page is loaded in 2.08 sec with Twes+ and it is loaded in 1.73 sec without Twes+.

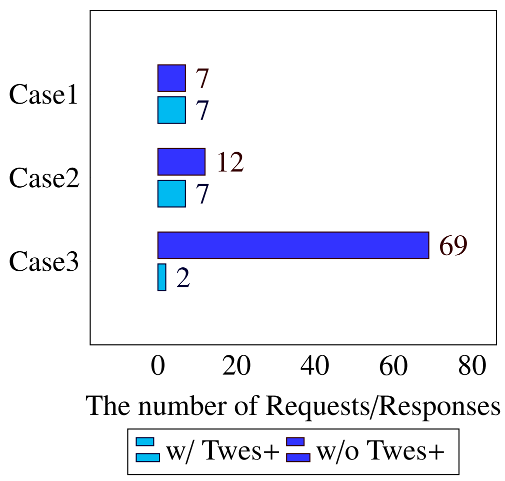


Figure 4.6: The Total Number of Request/Response with and without Twes+ [w/:with, w/o:without]

For the energy impact of redirect transcoding service, the same web pages are requested by the client without having any other application on the background except Intel Power Gadget and Google Chrome Browser for *w/o Twes+* and *w/ Twes+*. In Table A.2 in Appendix, the cumulative processor& IA energy and the average processor& IA power are represented for three cases. Figure 4.7 represents the cumulative processor energy. For example, the cumulative processor energy for case 2 is decreasing from 80.39 mWh to 61.7 mWh with Twes+. The average processor power for case 2 is decreasing from 9.53 Watt to 7.31 Watt with Twes+, see Figure 4.8.

To see the impact, same web pages are requested from a mobile phone. While the test is running, only Powertutor tool and the browser is active on the background and duration of the test is 120 secs. The first 30 seconds is the duration that shifting from Powertutor GUI to the browser and waiting for the system to reach the stable power level occur. After the first 30 seconds, the web pages are requested and even the web page is loaded, without interaction with the device one-minute measurement is done. After this one minute, shifting back to the GUI of Powertutor and waiting another 30 seconds occurs. So all the power screenshots of the mobile device include 120 seconds time chart for Total power, LCD power, CPU power and W-Fi Power.

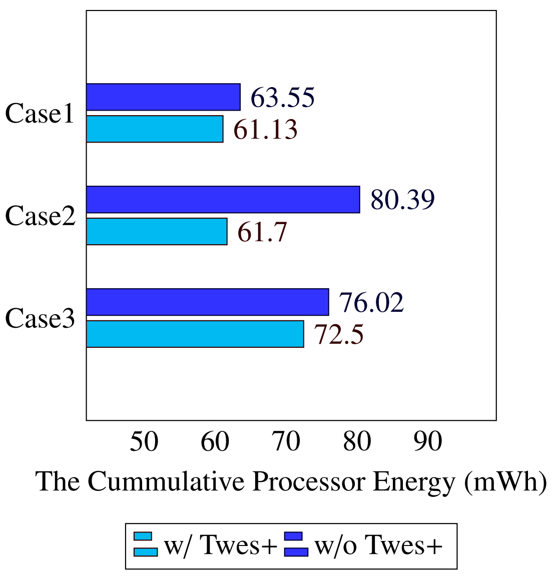


Figure 4.7: The Cummulative Processor Energy Comparison for Three Cases with and without Twes+ [w/:with, w/o:without]

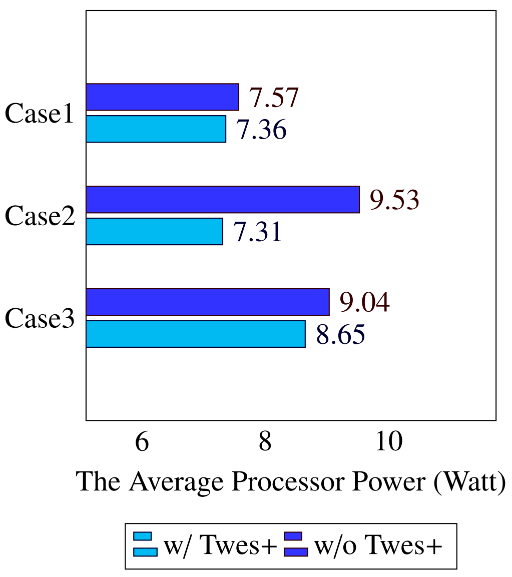


Figure 4.8: The Average Processor Power Comparison for Three Cases with and without Twes+ [w/:with, w/o:without]

For example, case 2 measurement over the mobile phone without Twes+ is presented as A in Figure 4.9 and with Twes+ is presented as B in Figure 4.9. As the LCD brightness does not change, its power consumption remains same but other three graphs have a peak. The peak occurs as it is the time the web page is requested. Any fluctuation between 0 to 30secs and 90 to 120secs should be ignored as their reason is the switching process between the browser and the Powertutor. The total power, CPU power and Wi-Fi power decrease when Twes+ is used. Other power results of the measurements over the mobile device are in Appendix Section A.3.

Image Transcoding Service

"Does Twes+ allow saving energy by reducing the number of request/response between client&server and/or reducing the payload between client&server?"

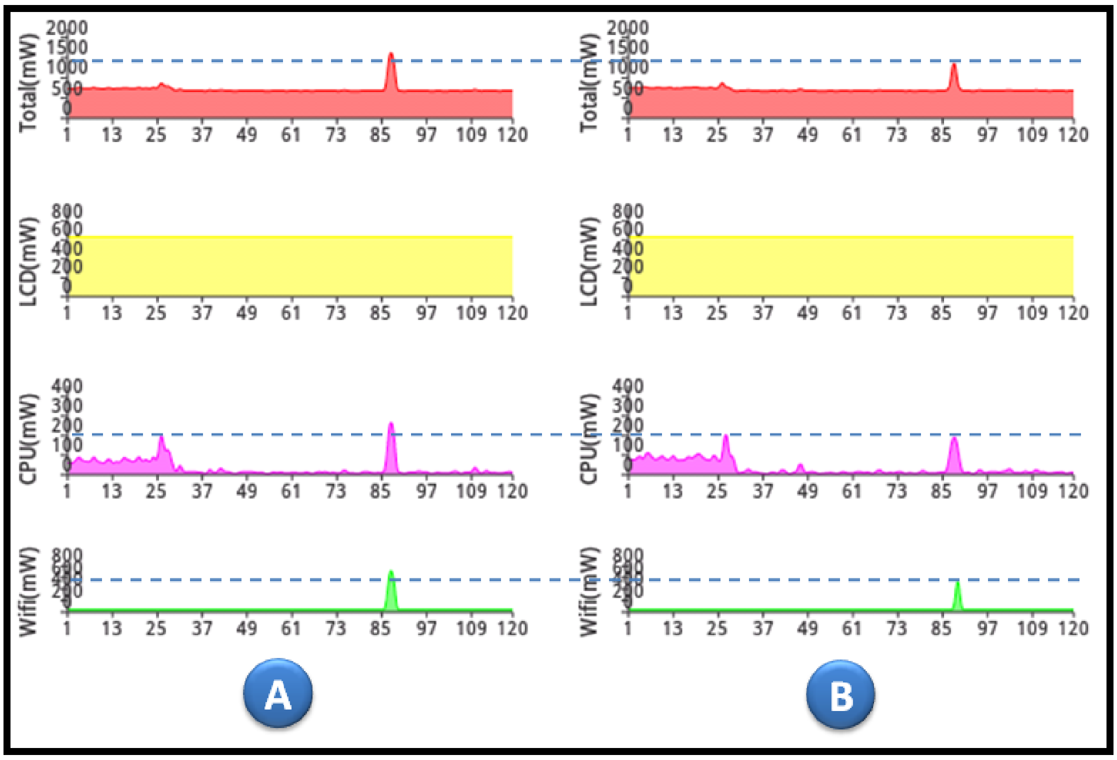


Figure 4.9: Power Consumpiton during Browsing Case 2 without Twes+ (A) and with Twes+ (B)

The web pages that are listed in Table 4.1 are requested by the client with Twes+ and without Twes+. Their total number of requests/responses, the total number of image requests/responses and the page load duration are extracted from Google Developer Tool. These data is represented in Table B.1 in Appendix B.

First of all, to evaluate the number of requests/responses comparison results, the mean, median and standard deviation of each sample are calculated, see Table 4.2. This table shows that with Twes+, although Twes+ increases the page load duration, the number of total requests/responses and the number of total image requests/responses are lower than without Twes+. For example, the results of the web page Ebay shows that although the load duration is increased from 19.35secs to 22.49secs, the total number of requests/responses between the client and the server decreases from 170 to 33 with Twes+, see Figure 4.10. To determine whether or not the

Table 4.2: Mean, Median and Standard Deviation of Each Sample of the Number of Requests/Responses, the Number of Image Requests/Responses, the Load Duration(Finish) [w/:with, w/o:without]

	The Number of Requests/ Responses		The Number of Image Requests/ Responses		Load Duration-Finish(s)	
	w/o Twes+	w/ Twes+	w/o Twes+	w/ Twes+	w/o Twes+	w/ Twes+
Mean	82.13	36.63	67.13	21.44	19.35	22.49
Median	68.00	29.00	50.50	15.00	13.98	18.52
Standard Deviation	60.44	32.5	56.90	19.69	19.78	21.83

difference between with and without Twes+ is significant, the Paired-Dependent T-Test is applied. Same web pages are measured with and without Twes+ and since the design is repeated measures so the Paired-Dependent T-Test or its non-parametric alternative Wilcoxon Signed Rank Test is applied. To decide either Paired-Dependent T-Test or its non-parametric alternative Wilcoxon Signed Rank Test is needed to be applied, the normality of the difference between the results with and without Twes+ are checked by using Shapiro-Wilk (SW) test. The significance of the Paired-Dependent T-Test and Wilcoxon Signed Rank Test are found from the probability value. If the probability value (p) is less than .05, it means that there is a significant difference in the mean (for the Paired-Dependent T-Test) and in the median (for the Wilcoxon Signed Rank Test) between with and without Twes+. If the probability value is more than .05, it means that there is no significant difference between with and without Twes+. This .05 value is coming from the 95% Confidence Interval of the Difference. If the probability value is more than .05, it can be said that there is a significant difference between with and without Twes+ measurements. However, to find the magnitude of the effect, Eta Squared Statistics for the Paired-Dependent T-Test and finding r value for the Wilcoxon Signed Rank Test can give us a hint. Interpreting the Eta Squared value is as follows; value .01 small effect, value .06 moderate effect, value .14 large effect. Interpreting the r value is as follows; value .1 small effect, value .3 medium effect and value .5 large effect [66]. All statistical calculations are performed by using IBM SPSS Statistics to evaluate the data.

For the data in Table B.1, Shapiro-Wilk (SW) test is applied for the normality. The difference between the results of the number of total requests/responses and the number of total image requests/responses with and without Twes+ are not normally distributed so the Wilcoxon Signed Rank Test is applied. The Wilcoxon Signed Rank Test results show that there is a statistically significant reduction in the total number of requests/responses and the total number of image requests/responses with Twes+, $z = -3.408$, $p = .001$, with a large effect size ($r = .6025$) for the total number of requests/responses and $z = -3.5198$, $p < .0005$, with a large effect size ($r = .6221$) for the total number of image requests/responses. The median score on the total number of requests/responses decreases from 68 to 29 and the median score on the total number of image requests/responses decreases from 50.50 to 15, see Table 4.2.

The Shapiro-Wilk (SW) test is applied to the load duration (Finish column of Table B.1) difference between with and without Twes+ and it is normally distributed so the Paired-Dependent T-Test is applied to evaluate the impact of the Twes+ in time. The result of the Paired-DependentT-Test shows that there is a statistically significant increase in Time with Twes+, $t(15) = 2.532$, $p = .023$. The mean and the standard deviation score on time is increasing from ($M = 19.35$ $SD = 19.77$) to ($M = 22.49$ $SD = 21.83$), see Table 4.2. The eta squared statistics (.30) indicates a large effect size.

To be able to evaluate whether Twes+ allows saving energy by reducing the number of requests/responses between the client and the server, first the results of the number of requests/responses are evaluated. The Wilcoxon Signed Rank Test results show that there is a statistically significant reduction in the total number of requests/responses and the total number of image requests/responses with Twes+. The next step, the power and the energy consumption are measured by using Intel Power Gadget. These data is presented in Table B.2 in Appendix. As a first step to evaluate these power and energy numbers, the mean, median and standard deviation of each sample are calculated, see Table 4.3 and Table 4.4. The mean and the median values indicate that with Twes+ the cumulative energy and the average power consumption decrease. To investigate whether or not the difference between with and without Twes+ is significant, the

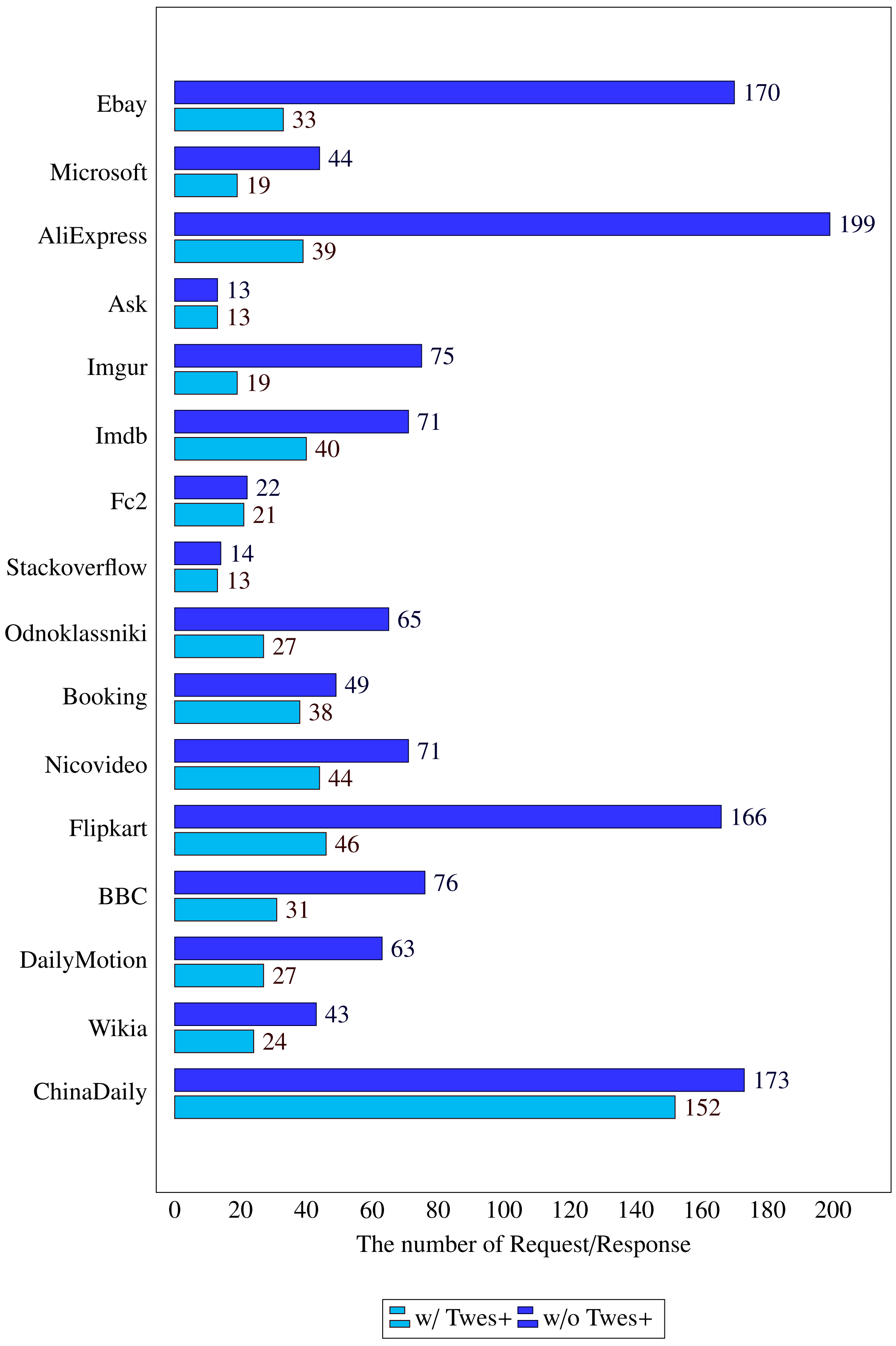


Figure 4.10: The Total Number of Requests/Responses Comparison with and without Twes+ [w/:with, w/o:without]

Paired-Dependent T-Test or its alternative Wilcoxon Signed Rank Test is applied. To decide which one is needed to be applied, the Shapiro-Wilk(SW) test is used to determine the difference between results with and without Twes+ is normally distributed or not. The results of each sample are normally distributed so the Paired-dependent T-Test is applied to all.

Table 4.3: Mean, Median and Standard Deviation of Each Sample of the Average Power and Cumulative Energy of Processor [w/:with, w/o:without]

	Cumulative Processor Energy (Joules)		Cumulative Processor Energy (mWh)		Average Processor Power (Watt)	
	w/o Twes+	w/ Twes+	w/o Twes+	w/ Twes+	w/o Twes+	w/ Twes+
Mean	280.23	259.61	77.84	72.11	9.33	8.62
Median	278.06	260.86	77.24	72.46	9.34	8.53
Standard Deviation	24.93	24.90	6.92	6.92	0.79	0.82

Table 4.4: Mean, Median and Standard Deviation of Each Sample of the Average Power and Cumulative Energy of IA [w/:with, w/o:without]

	Cumulative IA Energy (Joules)		Cumulative IA Energy (mWh)		Average IA Power (Watt)	
	w/o Twes+	w/ Twes+	w/o Twes+	w/ Twes+	w/o Twes+	w/ Twes+
Mean	135.42	117.72	37.62	32.70	4.51	3.91
Median	131.45	116.42	36.51	32.34	4.46	3.81
Standard Deviation	17.82	17.24	4.95	4.79	0.57	0.57

The results of the Paired-Dependent T-Test and the Eta Square statistics are represented in Table 4.5. Interpretation of these results is that there is a statistically significant energy saving and a decrease in power consumption with Twes+. To find the magnitude of the effect, Eta Squared statistics applied and the results indicate a large effect size.

Table 4.5: The Results of the Paired-Dependent T-Test and the Eta Square Statistics for the Cumulative Energy and the Average Power Test of Image Transcoding Service

	Cumulative Processor Energy (Joules)	Cumulative Processor Energy (mWh)	Average Processor Power (Watt)	Cumulative IA Energy (Joules)	Cumulative IA Energy (mWh)	Average IA Power (Watt)
t	-6.019	-6.019	-5.898	-7.445	-7.445	-7.22
df	15	15	15	15	15	15
p	<0.0005	<0.0005	<0.0005	<0.0005	<0.0005	<0.0005
Eta Squared	0.71	0.71	0.70	0.79	0.79	0.78

To investigate the effect of Twes+ over the mobile device, the same web pages are requested by using the mobile device. The Powertutor tool is used to save the power consumption for two minutes interval. The first 30 seconds is the duration of switching from the Powertutor GUI to the browser. The web pages are requested and measurement is done for a one-minute duration. After that, 30 seconds measurement is the switching duration from the browser to the GUI of Powertutor. The image resizing feature of image transcoding service of Twes+ is only measured for the mobile devices as device mode of the Google Chrome Browser adds an overhead because entire developer tool works on the side. A in Figure 4.11 shows the total power consumption, the CPU power consumption and the Wi-Fi power consumption while browsing Ebay without Twes+. B in Figure 4.11 shows the impact of Twes+ but the resizing feature is not active. C in Figure 4.11 shows the power results with Twes+ with the resizing feature is active. It means that all the supported images dimensions are reduced by 50%. These figures show that there is a decrease in power consumption when Twes+ is active and also there is more decrease in the total power consumption and the Wi-Fi power consumption when the resizing feature of Twes+ is active. The rest of the results are in Section B.2 in Appendix.

In this section, the results are present to answer the research questions. For both of the research questions, the results are presented according to two devices as a client. These devices are a desktop and a mobile device. The measurements that are done on the desktop includes the total number of requests/responses, the load duration, the cumulative energy and the average power. The measurements that are done on the mobile device covers the power consumption during

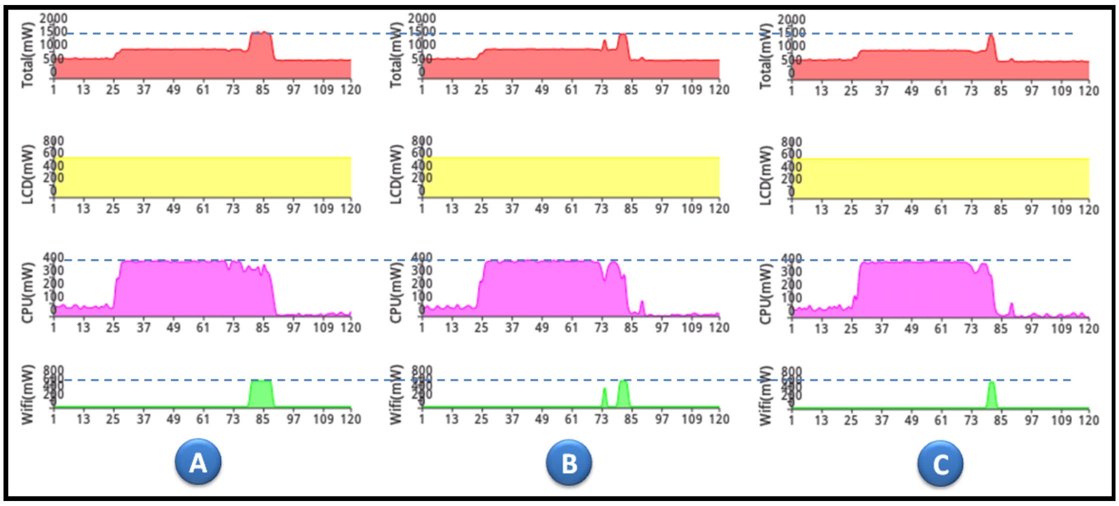


Figure 4.11: Power Consumption While Browsing Ebay without Twes+ [A], with Twes+ with Resizing Feature [B], with Twes+ without Resizing Feature [C]

browsing. For the image transcoding service, the measurement over the mobile device includes the impact of resizing feature of Twes+ as well.

For the measurements that are done on the desktop, some statistical tests are applied to see whether there is a statistically significant improvement or not and if there is, then what is the magnitude of its effect. The results show that with Twes+ there is a statistically significant improvement in the number of requests/responses, the cumulative energy and the average power consumption while browsing a web page.

4.7 Discussion

Here we discuss our major findings with respect to two research questions posed in Section 4.1.

Redirection Transcoding Service

The evaluation of redirect transcoding service indicates that Twes+ can reduce the number of redirect round trips by transcoding the web pages and therefore with Twes+ energy saving can be achieved. In Table A.1 in Appendix, the case 1 represents the web page without any redirect. The number of requests/responses with and without Twes+ remains same. The case 2 represents the web page with a finite number of redirects. The number of a web page with redirect is showing an uprising trend, see Figure 3.3. In this case, the client requests the web page and the rest of the redirect parts are done over the Twes+ so the client does not have any knowledge about these redirects and the client receives the web page as case 1. For case 2, Twes+ saves about 12 mWh energy and reduces the average processor power about two Watts. For the mobile device test, Twes+ is reducing the Wi-Fi and the CPU power for case 2.

Case 3 represents the web page with infinite redirects. Mainly infinite redirects occur because of missing files or wrong redirection. The tests over the desktop, the browser is prevented from interfering with any requests/responses so the test does not finish without Twes+. The browser is prevented because the browser blocks the timeline and does not show the real requests/responses. However, for Google Chrome browser the maximum number of allowed redirect is 20. So if the browser is not prevented from interfering with the redirects, Twes+ is still giving much better results. The cumulative energy of processor results indicates that Twes+ is able to reduce it from 76mWh to 72.5mWh (i.e.,5% reduction). Twes+ is also able to reduce the average power of the processor. For the mobile version of this test, as there is no option to prevent the browser from interfering with the redirects. At the certain point during the infinite redirection loop, the browser stops the redirects and gives an infinite redirection warning to the client. Twes+ is reducing the Wi-Fi and the CPU power for the case 3 and the reduction is more than half. Therefore, it can be concluded that with the Twes+ Redirection Transcoding Service, there is an energy consumption improvement during the browsing.

Image Transcoding Service

The evaluation of image transcoding service indicates that Twes+ can save energy by reducing the number of requests/responses between the client and the server. The number of requests/responses are shown with and without Twes+ in Table B.1 in Appendix. Although some web pages do not have a reduction or do not have a large number of reduction in the number of requests/responses, Twes+ is still working on those images. These web pages are Ask [7], Fc2 [8] and Stackoverflow [9]. The detailed requests/responses of these web pages are in Section B.3 in Appendix. The number of images transcoded in these web pages is low because some of the images are inline images or background images or requested by the CSS file or the format of the image is not supported by Twes+. As a low reduction in the number of requests, there is a small amount of reduction in the cumulative energy of processor and the average processor power. The reason of reduction is in some cases, the decrease in the size of transferred images or the decrease in the duration to load those images or the reduction in the number of requests/responses. The power results from the mobile devices of these web pages show similar or slightly lower power results during the measurement without Twes+, with Twes+ (resizing active) and with Twes+ (resizing is not active). As a result, if the web page does not include enough image tag, as Twes+ is not able to reduce the number of requests/responses, the energy saving does not occur or it is slightly low.

For the other web page results, their number of requests/responses with Twes+ is always lower than without Twes+. Half of the web pages show 50% reduction in the number of requests/responses. Moreover, all the web pages are showing a reduction in the cumulative processor energy consumption and the average processor power except AliExpress [10], see Table B.2. When it is compared with the closest candidates according to the number of requests/responses and the

[7] `http://ask.com/`
[8] `http://fc2.com/`
[9] `http://stackoverflow.com/`
[10] `http://www.aliexpress.com/`

content of the web page (online shopping): Ebay [11] and Flipkart [12], the reason might be the size of the images. The overall size of AliExpress is 7.3MB while the overall size of Ebay is 2.2MB and for Flipkart it is 1.8MB. The cumulative processor energy consumption and the average processor power of both web pages indicate, with Twes+ for Ebay 10 mWh (12%) saving and one Watt (11%) reduction in the average processor power and for Flipkart 11.2 mWh (13%) saving and 1.5 Watt (15%) reduction in the average processor power can be achieved. If the web pages are ranked according to the maximum energy saving, the order is Flipkart (11.2mWh), BBC [13] (11.1mWh) and Booking [14] (9.7mWh). When the web pages are examined from the look&feel perspective, we observe that the pages with the large images have lower energy saving than the pages with the moderate size of images.

If the power results of AliExpress is checked over the mobile phone, even there are some fluctuation a reduction in Wi-Fi power consumption and in overall power consumption are achieved. For Ebay and Flipkart, there is a clear reduction in the Wi-Fi power, the CPU power and the overall power. If the web pages that the most energy saving is achieved over the desktop are checked for the mobile devices: Booking web page has changes in the CPU power consumption over the mobile device and BBC shows similar results over the mobile device because it has integrated mobile version. Moreover, this mobile version is like after applying summarization transcoding (see Chapter 2 Section 2.3) to the original web page. Chinadaily [15] shows an interesting result. This web page has an active section, hence actually, it increases the power consumption with a running JavaScript. This JavaScript is regularly changing the images in the largest area of the web page. The pages that show the max reduction are Odnoklassniki [16], Imgur [17] and Nicovideo [18]. As a conclusion, from the desktop measurements it shows that with Twes+ the number of requests/responses reduction can be achieved. From the desktop results, it can be said

[11] http://www.ebay.com/
[12] http://www.flipkart.com/
[13] http://www.bbc.com/
[14] http://www.booking.com/
[15] http://www.chinadaily.com.cn/
[16] http://ok.ru/
[17] http://imgur.com/
[18] http://www.nicovideo.jp/

that the power consumption also decreases and energy saving can be achieved. From the mobile devices, it can be concluded that power reduction is achieved in most of the pages. The resizing feature of Twes+ is also used for the mobile tests. The web pages that are mentioned above even AliExpress, there is a reduction in power when the resizing feature is active.

To see the impact of the energy saving on the battery life, we assume that our desktop client works on the battery. The battery specifications of a closest laptop (TOSHIBA Satellite P70) is: the capacity is 4400 mAh and the voltage is 10.8 V. The average processor power of results is not equal to the total system average power. Therefore, the analysis is done on two assumptions which are the average processor power is 70% of the entire system and the average processor power is 100% of the entire system. Both analyses are done for Redirect Transcoding Service and Image Transcoding Service. The results show that there is always improvement in the battery life of the device. The Redirect Transcoding Service results show that even with the average processor power is 70% of the entire system assumption, for case 2 there is 48 mins increase in the battery life. The Image Transcoding Service results indicate that again with the average processor power is 70% of the entire system assumption, the battery life of the client-side is increasing more than 30 mins for Booking, Flipkart and BBC web pages.

Table 4.6: Battery Life Analysis with the Assumption of Total System Power for Redirect Transcoding Service Results [w/:with, w/o:without]

Name	Reduction %	w/o Twes+ (hr)	w/o Twes+ (mins)	w/ Twes+ (100%)(hr)	w/ Twes+ (100%)(mins)	Difference btw w/o Twes+ and w/Twes+ (100%) (mins)
Case 1	2.83	6.3	376.5	6.5	387.4	10.9
Case 2	23.23	5.0	299.1	6.5	389.6	90.5
Case 3	4.33	5.3	315.3	5.5	329.6	14.3

In conclusion, the results are discussed in this section. The results show that with Twes+ energy saving can be achieved. The type of the web page, the size of the images and the number of images change the reduction percentage but we achieve our goal. Our battery life analysis also shows that there is an increase in the battery life of the client-side.

Table 4.7: Battery Life Analysis with the Assumption of 70% of Total System Power for Redirect Transcoding Service Results [w/:with, w/o:without]

Name	Reduction %	w/o Twes+ (hr)	w/o Twes+ (mins)	w/ Twes+ (70%)(hr)	w/ Twes+ (70%)(mins)	Difference btw w/o Twes+ and w/Twes+ (70%) (mins)
Case 1	2.83	6.3	376.5	6.4	383.9	7.4
Case 2	23.23	5.0	299.1	5.8	347.7	48.6
Case 3	4.33	5.3	315.3	5.4	324.9	9.6

Table 4.8: Battery Life Analysis with the Assumption of Total System Power for Image Transcoding Service Results [w/:with, w/o:without]

Name	Reduction %	w/o Twes+ (hr)	w/o Twes+ (mins)	w/ Twes+ (100%)(hr)	w/ Twes+ (100%) (mins)	Difference btw w/o Twes+ and w/Twes+ (100%) (mins)
Ebay	11.36	5.2	314.4	5.9	354.6	40.3
Microsoft	7.15	4.2	253.3	4.5	272.8	19.5
AliExpress	0.91	5.0	302.0	5.1	304.8	2.8
Ask	4.85	5.9	353.3	6.2	371.3	18.0
Imgur	0.86	5.1	305.7	5.1	308.4	2.7
Imdb	7.43	5.4	325.5	5.9	351.6	26.1
fc2	5.05	5.0	300.6	5.3	316.6	16.0
Stackoverflow	4.72	5.0	300.8	5.3	315.7	14.9
Odnoklassniki	3.54	5.1	304.6	5.3	315.7	11.2
Booking	17.19	5.3	317.8	6.4	383.7	66.0
NicoVideo	9.15	5.0	298.5	5.5	328.6	30.1
Flipkart	15.67	4.8	286.5	5.7	339.7	53.2
BBC	14.82	5.2	309.6	6.1	363.5	53.9
DailyMotion	9.83	5.3	315.2	5.8	349.5	34.4
Wikia	1.53	6.0	357.3	6.0	362.8	5.5
ChinaDaily	7.48	4.6	274.8	4.9	297.0	22.2

In addition to these results, Twes+ can be used for large-scale systems like our university network and it can answer a lot of clients. By enabling the caching feature with Twes+, clients of our university network can access the web pages of the university with less energy consumption and sustainable energy consumption can be achieved. However, these measurements are done for a single client on a normal desktop computer as a proxy server. For large-scale systems, with a normal desktop computer, the client will suffer from the latency as the desktop computer is not powerful enough for large scale systems. Therefore, the proxy system requires hardware improvement for the large-scale system usage.

Table 4.9: Battery Life Analysis with the Assumption of 70% of Total System Power for Image Transcoding Service Results [w/:with, w/o:without]

Name	Reduction %	w/o Twes+ (hr)	w/o Twes+ (mins)	w/ Twes+ (70%)(hr)	w/ Twes+ (70%)(mins)	Difference btw w/o Twes+ and w/Twes+ (70%) (mins)
Ebay	11.36	5.2	314.4	5.7	339.4	25.0
Microsoft	7.15	4.2	253.3	4.4	266.0	12.7
AliExpress	0.91	5.0	302.0	5.1	304.0	1.9
Ask	4.85	5.9	353.3	6.1	365.2	12.0
Imgur	0.86	5.1	305.7	5.1	307.6	1.8
Imdb	7.43	5.4	325.5	5.7	342.4	16.9
fc2	5.05	5.0	300.6	5.2	311.3	10.6
Stackoverflow	4.72	5.0	300.8	5.2	310.8	9.9
Odnoklassniki	3.54	5.1	304.6	5.2	312.1	7.5
Booking	17.19	5.3	317.8	5.9	356.0	38.2
NicoVideo	9.15	5.0	298.5	5.3	317.7	19.1
Flipkart	15.67	4.8	286.5	5.3	317.9	31.4
BBC	14.82	5.2	309.6	5.7	341.7	32.1
DailyMotion	9.83	5.3	315.2	5.6	336.9	21.7
Wikia	1.53	6.0	357.3	6.0	361.1	3.8
ChinaDaily	7.48	4.6	274.8	4.8	289.1	14.4

4.8 Summary

In this chapter, the evaluation process of Twes+ is presented. We ask two research questions that are investigated in our evaluation. Our evaluation process details including the materials, the technical specifications of our equipment and the tools are introduced. Thereafter, we present our evaluation metrics, the software architecture of Twes+ and evaluation procedure. The next section, the results and the analysis that are used to answer our research questions over the desktop and the mobile devices are discussed.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In early days of the web, the users are used to access the web from desktop machines. Therefore, the web page developers used to consider the desktop browsers. In time, the desktops are improved and the design of the web pages follow this improvement. In the 90s, mobile devices are introduced and later they become a popular medium for accessing the web. Over the years, the number of mobile devices increased and they start to replace the desktops. Nowadays, the number of mobile Internet users is more than the number of desktop Internet users and the mobile devices become the primary devices to access the Internet. However, the mobile devices still have limitations. Their size is smaller than the desktops and this limitation decreases the power of the components. When we compare the current desktops and the current mobile devices, the mobile devices are suffering from the battery size, the bandwidth of the connection, the processing power, the limited memory, the smaller number of simultaneous connection, the power of their browsers and the smaller screen size. While browsing web pages over the mobile devices, the user suffers from the battery, since the battery of the mobile devices is draining fast. The reason for this problem is the conflict between the limitations of mobile devices and the design of the web pages. There are three types of web page design. The desktop version of web pages is designed for the desktops which have almost limitless power and faster connection. When the user requests those pages over the mobile device, because of the limits of the mobile devices, the number of required requests/responses round trips for the web page and the size of the web pages, the battery of the mobile devices are draining fast. The desktop version of web pages that are developed as responsive web pages for mobile devices generally considers the screen size limitation of the mobile devices so the layout of the web page changes but the requests/responses are usually as the desktop version. Sometimes they reduce the payload by including the different version of the images. The mobile version of the web pages also require many requests/responses round trips but their main pattern is creating the summary of the web page so the look&feel of the web page becomes as a table of content. For each section, the user needs to request it from the server to load it.

In the literature, there are four categories to access the Internet: native applications, mobile web applications, widgets and the browsers. The first three categories are developed for a single-purpose task, but the browser is the category that anyone can access any web pages as it is not a single-purpose task. As a result of this feature of the browsers, plug-ins or external methods are developed to save energy during the browsing so the battery of the mobile devices does not drain. These plug-ins or external methods work on saving energy by blocking the power-hunger contents or reducing the performance or compressing the data or reducing the content of the web pages. Their common characteristic is the user or the developer needs awareness. If the developments in other fields (i.e., network and hardware) are considered, energy efficiency and sustainability are major concerns. However, the developments in the software field are not enough. Some studies have highlighted the increase in energy consumption during browsing web pages but there are not enough research to solve them.

The literature review shows that there are several transcoding techniques to save energy and improve the battery drain issue of the mobile devices but there are some drawbacks. The look&feel of the web pages are modified to make the web page simple so that the number of requests/responses decreases. However, this makes web pages like a summary of the original web page. Another concern about existing techniques is that if it runs on the client, it consumes more energy to save energy and also the user needs to be aware and needs to deal with the process, but there are all types of users, thus, expecting them to deal with the configuration of these settings can be tricky. Moreover, expecting from all developers to consider power consumption of their web pages can be wrong, thus, nowadays there are many of non-programmer web page developers. As a result, many web pages are developed without considering power consumption. Our literature review shows that there are three transcoding techniques that do not modify the look&feel of the web pages, do not put extra load on the server or the client. These techniques are compressing the data to reduce the bandwidth usage, concatenating the external sources to reduce the number of requests/responses and adding expiration header again to reduce the requests/responses. In the literature review, however, there are three more techniques that are very promising guidelines for the web page developer to save energy while browsing and they are not applied. These techniques are image consolidation, responsive images, and reducing the number of redirects.

In this thesis, we present our novel approach which focuses on transcoding web pages for energy saving, called Twes+. The overall architecture of Twes+ has two main services: redirect transcoding service and image transcoding service. In redirect transcoding service, our research question is whether or not Twes+ can save energy by reducing the number of redirects. Twes+ monitors the traffic and when it receives a redirect response, it sends this redirection address to the Redirection Module. This module tracks the responses. If the number of redirects is less than the limit, it receives the content on behalf of the client and sends the content to the client. This way the client does not receive any redirect response or does not track any of redirection round trips. If the number of redirects is more than the limit, Twes+ assumes that there is an infinite redirection so it returns a message to the client.

In image transcoding service, the research question is whether or not Twest+ can save energy by reducing the number of requests and/or reducing the payload between the client&the server. Twes+ monitors the HTML responses that are sent by the web server. In the received HTML files, it finds all image tags and extract the dimensions and the location of the image. Twes+ sends these extracted data to Supporting Module. This module downloads the images from the server and checks the dimensions of the images to find required resizing operation. If there is a resizing operation that the client needs to apply, they are resized by the module. Supporting module returns the new name & the dimensions of the image to Twes+ and the image tag in the HTML is updated. If the module finds any error, it returns negative so that image tag remains as the original. After all the images are downloaded, Twes+ calls the Consolidation module which takes images and consolidates them ten by ten according to the format of the image. After Consolidation module finishes all the images and creates the concatenated versions it returns to Twes+. The last task is adding the CSS section of the consolidated images to find the right image dimensions in the consolidated image. When these steps are done, Twes+ returns the updated HTML to the client.

For example, the web page only contains ten PNG format image. When the client browser requests the images from the web server, without Twes+ the clients sends ten requests and receives ten responses and with Twes+ the client requests once and receives one response, thus total requests/responses reduce from 20 to two. The payload reduction occurs in the Supporting module. There are two way of reduction: resizing of the images that are supposed to be resized by the client browser and resizing the images to 50% of their size if the client is a mobile device. The first case of resizing is done according to the dimensions specified in the image tag of the HTML file. The second case of resizing depends on the device. The device model is extracted from the request header and if it is a mobile device, resizing applied to all images in Supporting module.

This thesis also presents the technical evaluation of the proposed transcoding techniques over the desktop and the mobile device. The measurements are done with and without Twes+. The evaluation of the redirect transcoding service shows that Twes+ is able to reduce the number of redirect round trips. After that power and energy measurements are done on the desktop and the results show that Twes+ can save energy and reduce the average power consumption. With the same cases, the same process is repeated for the mobile device and it shows that there is a reduction in the power with Twes+. The evaluation of the image transcoding service indicates that with Twes+ the number of requests/responses of the web pages that are selected from Alexa decreases. When the cumulative energy consumption and the average power is measured, the results show that with Twes+ there can be reduction and energy saving over the desktop. The same measurements are repeated over the mobile device with and without resizing feature. The general results show that there is a trending reduction in the Wi-Fi power and/or in the CPU power and/or the total power.

During the evaluation of Twes+, the caching feature is disabled. All the results show the pure impact of Twes+. However, if caching is considered, it further improves the impact of Twes+. For the redirect transcoding service of Twes+, when the client requests the same page that has redirection, Twes+ can check the final URL instead of the requested URL. If the redirection is changed, Twes+ can check the requested URL but otherwise, Twes+ has the original URL and more Twes+ has the content of the web page. If the image transcoding service is considered, the results do not include any impact of the caching. When caching is enabled for the image transcoding service, the images that are consolidated before might be available for the next requests. For example, in our university, there are certain pages that are used by lots of people regularly (i.e., Intranet of METU NCC [1], Academic Planning Unit of METU NCC [2]). These pages can be cached after passing through the Twes+. The user of these web pages can directly receive the transcoded content from Twes+. By enabling caching, a sustainable consumption solution can be achieved for browsing over the mobile devices with Twes+, because the load of the proxy, the load of the web server and the load over the network decrease. Moving forward, the limitations of current Twes+ implementation are discussed.

Our goal is to save energy by transcoding the web pages while not modifying the look&feel of the web pages and not adding extra load on the client or the web server. Twes+ is independent of the client and the server, it means that Twes+ can work with any client and any server. For the look&feel modification concern, generally, we achieve our goal. However, we face some limitations that affect the look&feel of the web pages. In the next section, the limitations of Twes+ are presented. Thereafter, we state our future work.

[1] https://intranet.ncc.metu.edu.tr/

[2] https://apu.ncc.metu.edu.tr/

5.1 Limitations

In this section, we discuss the five limitations encountered during the development and measurements. They are CSS limitations, the resizing limitations, non-supported image formats and animated GIF images.

CSS Limitation: The first limitation is finding the dimensions of an image in the entire resource files of a web page. There are two ways to define the dimensions of an image: Using the HTML width and height attributes or using the CSS width and height properties. There is another way which is not defining the dimensions at all as the dimension of an image is not actually essential. From these three options (defining by HTML attributes, defining by CSS properties and not defining), for faster image display defining by HTML attributes is suggested. The reason behind this suggestion is when the client browser finds the image element, it requests the image but it continues to the rest of the HTML document. When the image is totally loaded, it moves the content of the web page and put the image to the right place, thus, the look&feel of the web page is changing. This can slow down the page loading as well. Because of the impact, it is suggested to give the dimension details of the image inside the HTML so the client browser can allocate the right spot for the image and other contents do not change the location inside the HTML file [90]. Moreover, the HTML attributes and the CSS properties can be used for resizing over the client browser. This case is adding extra work to the client, hence, the energy consumption of the client increases. If the image size is bigger than the required image, it is also a waste of bandwidth as a large image that not used is going to be transferred. Resizing over the client is opposing to the goal of this research. However, in reality, these three options are mixed and combined and this creates several results.

1. If the real dimensions of the image and the required dimensions of the image are same and there is no defined HTML attributes and CSS properties for dimensions, it means the page load might be longer as explained above. Twes+ takes the image tag from the HTML. The Supporting Module of Twes+ finds the dimensions of the image and these

dimensions are added into the HTML. As a result, the client browser knows the size of the image so loading might be faster without modification in look & feel of the web page.

2. If the real dimensions of the image and the required dimensions of the image or the defined dimensions using CSS or HTML are same, Twes+ works for this type of web pages without a modification in look&feel of the web page.

3. If the real dimensions of the image and the required dimensions of the image are not the same. It means there will be resizing done by the browser. This resizing can be done according to the HTML attributes of the image or the CSS properties. If resizing is done according to the HTML attributes of image and there is no defined CSS properties for the image, Twes+ can convert this case into the previous case 2. Twes+ takes the image tag from the HTML and sends this to the Supporting Module with the dimension attributes inside the HTML. The image is downloaded by the Supporting Module and it is resized over the proxy. This way the real dimensions of the image is changed to be same as the required dimensions of the image. The resizing task of the client browser is removed from the browser and done over the proxy.

4. If the real dimensions of the image and the required dimensions of the image are not the same and resizing is done according to the CSS properties of the image and there is no defined HTML attribute for the image, Twes+ takes the same action as the first case.

5. If the real dimensions of the image and the required dimensions of the image are not same. Moreover, the HTML attributes and the CSS properties are defined. For this case, Twes+ takes the same action as the third case.

As a summary, the current version of Twes+ does not support the CSS properties. There are several reasons behind this. The first reason, the CSS properties can be defined inside the HTML file at different locations, like at the beginning of the HTML document. Moreover, these CSS properties can be defined inside an external CSS file. Another and more important issue about CSS properties usage is that dimension properties are inheritable properties. It means that every element inherits all the inheritable properties from the parent element [69]. All these variations make finding the CSS properties for the dimensions hard and it increases the load on the proxy, so the energy consumption of the proxy increases and the waiting time of the client increases as well. Because of all these considerations, currently, it is not supported. For some web pages, if the web page is applying case 4 or 5, Twes+ is still working but it is modifying the look & feel of the web page. If the difference between CSS properties and the dimension of the image are close enough, the modification is not noticeable. For some cases, there is a noticeable change in the entire or portion of the web page.

Resizing Limitation: The second limitation is about the resizing feature of Twes+. Twes+ can resize images if the request is coming from a mobile device. Resizing is modifying look&feel of the image but images are in the same position with a smaller version. However, if the CSS properties are used, it interferes with the result of Twes+. It repeats the same image or shows the neighbor images in consolidated version.

Non-supported Image Formats Limitation: The third limitation is which images are covered with Twes+. Images can be added into the web page inside the external CSS file, as a background image inside a CSS section or external CSS file or as a background image inside HTML file. These images might be consolidated images by the web server and include their own splitting pattern. Moreover, differentiating whether it is a single image or consolidated image is hard and it requires search operation inside the CSS. Because of this possibility, Twes+ does not cover the images inside the CSS and the background images inside the HMTL and the CSS. This implementation of Twes+ only makes changes for image tags inside the HTML file.

Animated GIF Limitation: The fourth limitation is about the animated GIF images. If an image is animated, it means that this image actually contains several images (see Figure 3.11) and these images are combined. When consolidation is tried for animated images, the combination gets divided. As a result, animated images are not supported and they are returned negative by the Supporting Module and they are not modified.

Dynamic Web Pages: The last limitation is about the dynamic web pages. If the web page changes the images using JavaScript, this dynamic structure is not modified with Twes+ as there are a lot of variation in dynamic image tags. To be able to overcome this limitation, the JavaScript needs to run and the created new image tag should be used for Twes+. Without using the created new image tag, Twes+ might modify the look&feel of the web page.

In conclusion, the research presented in this thesis contributes three effective transcoding techniques. These techniques can reduce the number of redirects, reduce the number of requests/responses and the resize images automatically without adding extra load to the client or the web server and without modifying the look&feel of the web pages to save energy over the mobile devices. The results of this work show that if the web pages are developed with consideration of their energy consumption, energy saving can be achieved for each individual client-side. There are guidelines for the developers to improve their web pages but as the number of non-programmer developers are high, there is a need for a systematic alteration. This alteration might be improvements in the HTML and CSS syntax, for example, developers should not be able to include any non-image files into an image tag. With this improvement, still non-programmer developers might develop web pages while more systematic adaptations can be applied to the web pages to save energy as there is still a gap to be fixed in the source code of the web pages.

5.2 Future Work

The major limitation of the current version of Twes+ is the CSS limitation. As this limitation is affecting the look&feel of the web pages. The main concern during the development of the current version is while finding the CSS of the web page, the duration or the energy consumption of the web page should not increase. Our future work plan is to work on this limitation by considering the key points. The second major limitation is the resizing limitation as it changes also the look&feel of the web pages, thus, a new feature can be added to make the resized images to fit in the position. For other two limitations, the range of supported image formats and the animated GIFs are planned to be included in the next version of Twes+.

In this study, the evaluation of Twes+ is done with 16 web pages. Although the results of the evaluation are very promising, a study can be conducted with more web pages in our future plan. Moreover, more mobile tests can be added with different mobile devices and operating system. In order to show the sustainability impact of Twes+, caching might be included with Twes+ and locally test it within the university network.

In the current implementation of the Twes+, three transcoding techniques are applied. Another plan is extending the number of techniques to be able to save more energy in the next version of Twes+. Moreover, a report for the server-side can be sent to the web server to show the weak or the problematic points of their code for improvement. In addition to the server-side extension, Twes+ can be made as an application that the client can turn on and off using Twes+ with their mobile devices.

REFERENCES

[1] Hamed Ahmadi and Jun Kong. Efficient web browsing on small screens. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '08, pages 23–30, New York, NY, USA, 2008. ACM.

[2] Muhtaroglu Ali, Yokochi Alex, and von Jouanne Annette. Integration of thermoelectrics and photovoltaics as auxiliary power sources in mobile computing applications. *Journal of Power Sources*, 177:239–246, 2008.

[3] Chaitrali Amrutkar, Patrick Traynor, and Paul C van Oorschot. An empirical evaluation of security indicators in mobile web browsers. *Mobile Computing, IEEE Transactions on*, 14(5):889–903, 2015.

[4] Http Archive. Trends and statistics, September 2015. `http://httparchive.org/`.

[5] Chieko Asakawa and Hironobu Takagi. Transcoding. In Yeliz Yesilada and Simon Harper, editors, *Web Accessibility*, pages 231–260. Springer, 2008.

[6] Paramvir Bahl, Richard Y. Han, Li Erran Li, and Mahadev Satyanarayanan. Advancing the state of mobile cloud computing. In *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, MCS '12, pages 21–28, New York, NY, USA, 2012. ACM.

[7] Kenneth C. Barr and Krste Asanović. Energy-aware lossless data compression. *ACM Trans. Comput. Syst.*, 24(3):250–291, August 2006.

[8] Kjærgaard Mikkel Baun, Bhattacharya Sourav, Blunck Henrik, and Nurmi Petteri. Energy-efficient trajectory tracking for mobile devices. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 307–320. ACM, 2011.

[9] Fehmi Ben Abdesslem, Andrew Phillips, and Tristan Henderson. Less is more: energy-efficient mobile sensing with senseless. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, pages 61–62. ACM, 2009.

[10] Tim Berners-Lee. Web accessibility initiative (wai), 2014. `http://www.w3.org/WAI/`.

[11] Tim Berners-Lee. The worldwideweb browser, Dec 2015. `https://www.w3.org/People/Berners-Lee/WorldWideWeb.html`.

[12] Thong Chanchaem. A survey on internet content transcoding for universal access, May 2003. `http://medianet.kent.edu/surveys/DR03S-webxcoders/index.html`.

[13] Jinlin Chen, Baoyao Zhou, Jin Shi, Hongjiang Zhang, and Qiu Fengwu. Function-based object model towards website adaptation. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 587–596, New York, NY, USA, 2001. ACM.

[14] Chi-Hung Chi, Jing Deng, and Yan-Hong Lim. Compression proxy server: Design and implementation. In *USENIX Symposium on Internet Technologies and Systems*, 1999.

[15] Paul Cobbaut. Introduction to squid, May 2015.

[16] Intel Corporation. *Intel Core i7-4770 Processor (8M Cache, up to 3.90 GHz)*. Intel Corporation, March 2015.

[17] Evans Dave. The internet of things - how the next evolution of the internet is changing everything. *CISCO white paper*, 1:1–11, 2011.

[18] Sin David, Lawson Erin, and Kannoorpatti Krishnan. Mobile web apps - the non-programmer's alternative to native applications. In *Human System Interactions (HSI), 2012 5th International Conference on*, pages 8–15, 2012.

[19] Google Developers. Chrome devtools overview, October 2014. `https://developer.chrome.com/devtools`.

[20] Google Developers. The chromium project - spdy: An experimental protocol for faster web, September 2014. `http://www.chromium.org/spdy/spdy-whitepaper`.

[21] Google Developers. Pagespeed module - sprite images, August 2014. `https://developers.google.com/speed/pagespeed/module/filter-image-sprite`.

[22] Google Developers. Data compression proxy, January 2015. `https://developer.chrome.com/multidevice/data-compression`.

[23] Robert Collins Duane Wessels, Henrik Nordstrom and Amos Jeffries. Squid-cache.org optimising web delivery, 2013.

[24] J. Elson and A. Cerpa. Internet content adaptation protocol (icap), April 2003.

[25] Minoru Etoh, Tomoyuki Ohya, and Yuji Nakayama. Energy consumption issues on mobile network systems. In *Proceedings of the 2008 International Symposium on Applications and the Internet*, SAINT '08, pages 365–368, Washington, DC, USA, 2008. IEEE Computer Society.

[26] Tammy Everts. Rules for mobile performance optimization. *Queue*, 11(6):40:40–40:51, June 2013.

[27] L. Fainberg, O. Ehrlich, G. Shai, O. Gadish, A. Dobo, and O. Berger. Systems and methods for acceleration and optimization of web pages access by changing the order of resource loading, August 2 2010. US Patent App. 12/848,559.

[28] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Rfc2616 hypertext transfer protocol http/1.1, 1999.

[29] Maximiliano Firtman. *Programming the Mobile Web*, volume 1. OReilly Media, Inc., second edition, March 2010.

[30] Maximiliano Firtman. Mobile html5, December 2015. `http://mobilehtml5.org/`.

[31] Mike Foskett. Image to data uri converter web semantics, September 2012.

[32] Ben Frain. *Responsive Web Design with HTML5 and CSS3*. Packt Publishing, 2012.

[33] Google. Chrome accessibility, November 2015. `https://chrome.google.com/webstore/category/collection/accessibility`.

[34] Google. Google support, November 2015. `https://support.google.com/`.

[35] J. Grigsby. How apple.com will serve retina images to new ipads., 2012. Cloud Four Blog; http://blog.cloudfour.com/how-apple-com-will-serve-retina-images-to-new-ipads.

[36] Hung-Yun Hsieh and Raghupathy Sivakumar. Performance comparison of cellular and multi-hop wireless networks: A quantitative study. *SIGMETRICS Perform. Eval. Rev.*, 29(1):113–122, June 2001.

[37] Ngu Phuc Huy and Do vanThanh. Evaluation of mobile app paradigms. In *Proceedings of the 10th International Conference on Advances in Mobile Computing &#38; Multimedia*, MoMM '12, pages 25–30, New York, NY, USA, 2012. ACM.

[38] Turkish Statistical Institute. Percentage of households have devices connected to the internet, September 2015. `http://www.turkstat.gov.tr/`.

[39] Princeton Survey Research Associates International. Pew internet: Mobile. Pew Research Center's Internet Project, October 2013. `http://printinthemix.com/Fastfacts/Show/787`.

[40] William Jobe. Native apps vs. mobile web apps. *International Journal of Interactive Mobile Technologies*, 7:27–33, 2013.

[41] Troy A. Johnson and Patrick Seeling. Desktop and mobile web page comparison: Characteristics, trends, and implications. *IEEE Communications Magazine*, 52(9):14–151, 2014.

[42] Tomihisa Kamada, Takuya Asada, Masayasu Ishikawa, and Shinichi Matsui. Html 4.0 guidelines for mobile access. W3C, March 1999.

[43] Rachita Kothiyal, Vasily Tarasov, Priya Sehgal, and Erez Zadok. Energy and performance evaluation of lossless file data compression on server systems. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, SYSTOR '09, pages 4:1–4:12, New York, NY, USA, 2009. ACM.

[44] Dhiraj Kumar. Image sprites - how to merge multiple images, and how to split them, June 2012. https://dhirajkumarsingh.wordpress.com/2012/06/24/image-sprites-how-to-merge-multiple-images-and-how-to-split-them/.

[45] L3WS. Greasyspoon- open-source icap server factory for core network services, 2015. http://greasyspoon.sourceforge.net/.

[46] Pauli P. Y. Lai. Efficient and effective information finding on small screen devices. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, W4A '13, pages 4:1–4:10, New York, NY, USA, 2013. ACM.

[47] Heidi Lam and Patrick Baudisch. Summary thumbnails: Readable overviews for small screen web browsers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 681–690, New York, NY, USA, 2005. ACM.

[48] Charles Limonard. Electronic design, what drains the smart-phone battery, January 2013. http://electronicdesign.com/ed-europe/what-drains-smart-phone-battery.

[49] Yao Liu and Lei Guo. An empirical study of video messaging services on smartphones. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, NOSSDAV '14, pages 79:79–79:84, New York, NY, USA, 2014. ACM.

[50] Ethan Marcotte. Responsive web desing, 2010. http://alistapart.com/article/responsive-web-design.

[51] Marilyn. Manage the safari power saver feature, November 2013. http://www.mac-fusion.com/manage-the-safari-power-saver-feature/.

[52] Kate Matsudaira. Making the mobile web faster. *Queue*, 11(1):40:40–40:48, January 2013.

[53] Peter McLachlan. On mobile, data uris are 6x slower than source linking (new research), July 2013. http://www.mobify.com/blog/data-uris-are-slow-on-mobile/.

[54] Mary Meeker. The mobile internet report, 2008. www.morganstanley.com.

[55] Microsoft. Tips to save battery power, December 2015. http://windows.microsoft.com/en-us/windows-8/tips-save-battery-power.

[56] Mobify. Image resizing with mobify.js, 2013. https://www.mobify.com/mobifyjs/docs/image-resizing/.

[57] Monetate. Ecommerce quarterly eq2 2014: M-commerce today: Opportunity & challenge. Monetate, September 2014.

[58] Erica Naone. Amazon's cloud gives its new browser an unfair edge, October 2011. http://www.technologyreview.com/news/425617/amazons-cloud-gives-its-new-browser-an-unfair-edge/.

[59] Thiagarajan Narendran, Aggarwal Gaurav, Nicoara Angela, Boneh Dan, and Singh Jatinder P. Who killed my battery?: analyzing mobile browser energy consumption. In *Proceedings of the 21st international conference on World Wide Web*, pages 41–50. ACM, 2012.

[60] BBC News-Tech. World's first smartphone celebrates 20 years, August 2014. `http://www.bbc.com/news/technology-28802053`.

[61] Alex Nicolaou. Best practices on the move: Building web apps for mobile devices. *Queue*, 11(6):30:30–30:41, June 2013.

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

[63] Mark Nottingham. Caching tutorial, May 2013. `https://www.mnot.net/cache_docs/`.

[64] Opera. Faster browsing on slow networks with off-road mode, December 2015. `http://help.opera.com/opera/Windows/1116/en/fasterBrowsing.html`.

[65] Opera. Opera mini, December 2015. `http://www.opera.com/mobile/mini/android`.

[66] Julie Pallant. *SPSS Survival Manual A Step by Step Guide to Data Analysis using SPSS for Windows*. Open University Press, 3rd edition, 2007.

[67] Kolin Paul and Tapas Kumar Kundu. Android on mobile devices: An energy perspective. In *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology*, CIT '10, pages 2421–2426, Washington, DC, USA, 2010. IEEE Computer Society.

[68] K. Pentikousis. In search of energy-efficient mobile networking. *IEEE Communications Magazine*, 48:95–103, 2010.

[69] Jonathan Renoult. Inheritance and cascade, January 2016.

[70] Padley Richard. Html 5 bridging the mobile platform gap: mobile technologies in scholarly communication. *Serials: The Journal for the Serials Community*, 24(3), 2011.

[71] R. Rosenbaum, H. Schumann, and C. Tominski. Presenting large graphical contents on mobile devices - performance issues. In *IRMA International Conference - Innovations Through Information Technology*, 2004.

[72] Samsung. *Samsung Exynos 4 Quad (Exynos 4412) RISC Microprocessor*. Samsung Electronics Co., Ltd., San number: 24 Nongseo-Dong, Giheung-Gu Yongin-City, Gyeonggi-Do, Korea 446-711, 1.0 edition, October 2012.

[73] Gochman Simcha, Avi Mendelson, Naveh Alon, and Rotem Efraim. Introduction to intel core duo processor architecture. *Intel Technology Journal*, 10(2), 2006.

[74] Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma. Learning block importance models for web pages. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 203–211, New York, NY, USA, 2004. ACM.

[75] Steve Souders. High performance web sites. *Queue*, 6(6):30–37, October 2008.

[76] Steve Souders. Roundup on parallel connections, March 2008. `http://www.stevesouders.com/blog/2008/03/20/roundup-on-parallel-connections/`.

[77] Paul Stamatiou. Developing a responsive, retina-friendly site (part 2), March 2013. `http://paulstamatiou.com/responsive-retina-blog-development-part-2/`.

[78] Pete Stevens. Upside-down-ternet, December 2015.

[79] Mozilla Support. Save battery power, December 2015. `https://support.mozilla.org/en-US/kb/save-battery-power`.

[80] Cisco Systems. *Cisco Catalyst 2960-X Series Switches*, May 2015. `http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-2960-x-series-switches/qa_c67-728348.html`.

[81] H. Frystyk T. Berners-Lee, R. Fielding. Hypertext transfer protocol http/1.0, February 1996.

[82] Hironobu Takagi, Chieko Asakawa, Kentarou Fukuda, and Junji Maeda. Site-wide annotation: Reconstructing existing pages to be accessible. In *Proceedings of the Fifth International ACM Conference on Assistive Technologies*, Assets '02, pages 81–88, New York, NY, USA, 2002. ACM.

[83] Apache Traffic Server Team. Apache traffic server, January 2015. `https://docs.trafficserver.apache.org/en/latest/admin/index.en.html`.

[84] Mobify team. Introducing jazzcat: A javascript and css concatenation service, August 2012. `http://www.mobify.com/blog/introducing-jazzcat/`.

[85] PowerTutor Team. Powertutor - a power monitor for android-based mobile platforms, September 2014. `http://ziyang.eecs.umich.edu/projects/powertutor/`.

[86] Lose Thoba and Thinyane Mamello. A transcoding proxy server for mobile web browsing. In *The Southern Africa Telecommunication Networks and Applications Conference*, volume 2011. The Southern Africa Telecommunication Networks and Applications Conference, 2011.

[87] Serol Turkyilmaz, Haluk Kulah, and Ali Muhtaroglu. A development tool for design and analysis of mems based em energy scavengers. In *Energy Aware Computing (ICEAC), 2010 International Conference on*, pages 1–2. IEEE, 2010.

[88] Erra Ugo, Iaccarino Gennaro, Malandrino Delfina, and Scarano Vittorio. Personalizable edge services for web accessibility. *Universal Access in the Information Society*, 6:285–306, 2007.

[89] Jun De Vega. Intel power gadget, January 2014.

[90] W3C. Images in html, March 2014.

[91] W3School. Browsers, December 2015. www.w3schools.com/browsers/default.asp.

[92] Yahoo. Best practices for speeding up your web site, January 2015. `https://developer.yahoo.com/performance/rules.html`.

[93] Yeliz Yesilada, Simon Harper, and Sukru Eraslan. Experiential transcoding: An eyetracking approach. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, W4A '13, pages 30:1–30:4, New York, NY, USA, 2013. ACM.

[94] Xinyi Yin and Wee Sun Lee. Using link analysis to improve layout on mobile devices. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 338–344, New York, NY, USA, 2004. ACM.

[95] Nicholas C. Zakas. The evolution of web development for mobile devices. *Queue*, 11(2):30:30–30:39, February 2013.

APPENDIX A

REDIRECT TRANSCODING SERVICE

Case 1: No-Redirection

Case 2: Finite-Redirection

Case 3: Infinite-Redirection

A.1 Total Number of Request/Response and the Loading Duration Comparison

Table A.1: Total Number of Request/Response and the Loading Duration Comparison (*: Redirection was still going so web page never finished loading)

		Number of Request/ Response	Finish(s)
Case 1	w/o Twes+	7	2.06
	w/ Twes+	7	2.04
Case 2	w/o Twes+	12	1.73
	w/ Twes+	7	2.08
Case 3	w/o Twes+	69	*
	w/ Twes+	2	0.576

A.2 Power and Energy Comparison over Desktop

Table A.2: Power and Energy Comparison

		Cumulative Processor Energy (Joules)	Cumulative Processor Energy (mWh)	Average Processor Power (Watt)	Cumulative IA Energy (Joules)	Cumulative IA Energy (mWh)	Average IA Power (Watt)
	Idle	248.92	69.14	8.30	113.63	31.56	3.79
	Idle Browser	201.20	55.89	6.63	79.96	22.21	2.63
Case 1	Without Twes+	228.76	63.55	7.57	93.98	26.11	3.11
	With Twes+	220.06	61.13	7.35	90.57	25.16	3.03
Case 2	Without Twes+	289.42	80.39	9.53	137.01	38.06	4.51
	With Twes+	222.12	61.70	7.32	91.63	25.45	3.02
Case 3	Without Twes+	273.66	76.02	9.04	125.85	34.96	4.16
	With Twes+	260.99	72.50	8.65	120.64	33.51	4.00

A.3 Power Comparison over Mobile Phone

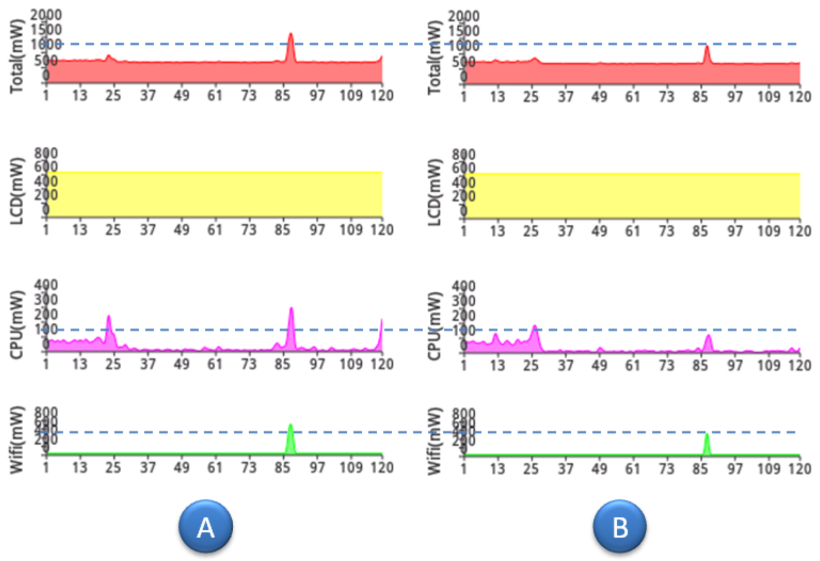


Figure A.1: Power Consumpiton During Case 1 without Twes+(A) and with Twes+ (B)

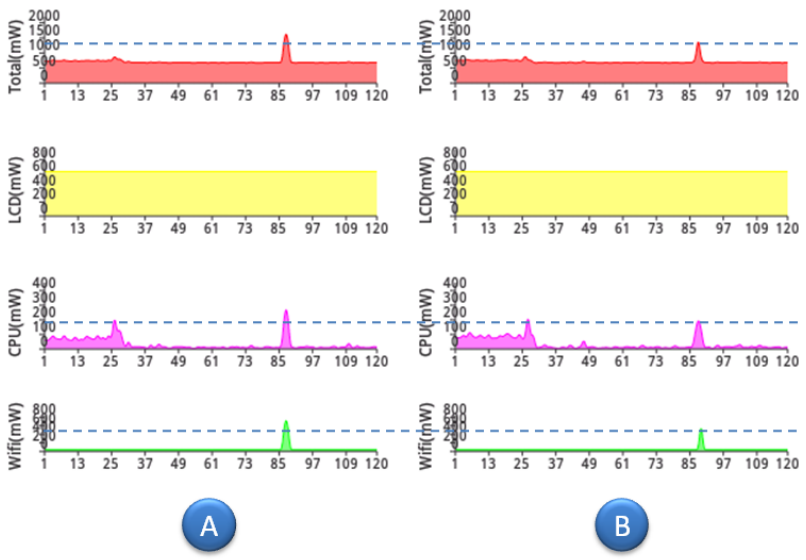


Figure A.2: Power Consumpiton During Case 2 without Twes+(A) and with Twes+ (B)

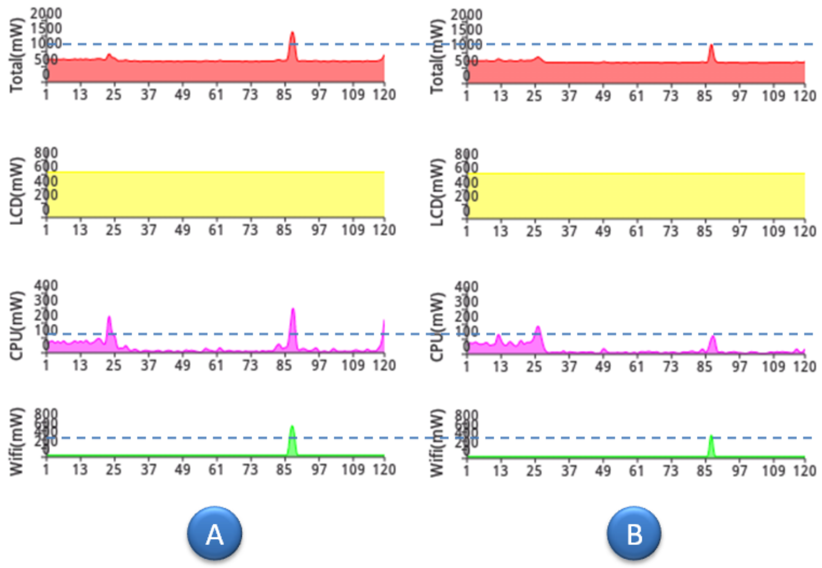


Figure A.3: Power Consumpiton During Case 3 without Twes+(A) and with Twes+ (B)

APPENDIX B

IMAGE TRANSCODING SERVICE

B.1 Total Number of Requests/Responses, the Loading Duration Comparison and Power and Energy Comparison over Desktop

Table B.1: Total Number of Request/Response and the Loading Duration Comparison for Image Transcoding Service

		Number of Request		**Number of Image Request**		**Finish(s)**	
Rank	**Name**	**w/o Twes+**	**w/ Twes+**	**w/o Twes+**	**w/ Twes+**	**w/o Twes+**	**w/ Twes+**
17	**Ebay**	170	33	165	28	25.57	30.15
28	**Microsoft**	44	19	30	5	17.92	20.6
29	**Ali Express**	199	39	192	31	84	96
33	**Ask**	13	13	6	5	3.4	3.68
43	**Imgur**	75	19	70	13	9.19	10.27
47	**Imdb**	71	40	61	30	10.46	12.49
50	**fc2**	22	21	11	10	4.08	8.5
55	**Stackoverflow**	14	13	9	8	4.08	4.23
61	**Odnoklassniki**	65	27	56	18	21.43	23.4
72	**Booking**	49	38	37	26	7.52	8.36
75	**NicoVideo**	71	44	43	16	7.43	21.61
76	**Flipkart**	166	46	149	29	21.48	28.95
85	**BBC**	76	31	59	14	17.03	16.43
95	**DailyMotion**	63	27	45	9	10.93	12.29
96	**Wikia**	43	24	33	14	28.72	33.25
97	**ChinaDaily**	173	152	108	87	36.39	29.65

Table B.2: Power and Energy Comparison

		Cumulative Processor Energy (Joules)	Cumulative Processor Energy (mWh)	Average Processor Power (Watt)	Cumulative IA Energy (Joules)	Cumulative IA Energy (mWh)	Average IA Power (Watt)
Ebay	w/o Twes+	274.26	76.18	9.07	126.56	35.15	4.19
	w/ Twes+	240.29	66.75	8.04	103.16	28.66	3.45
Microsoft	w/o Twes+	337.38	93.72	11.26	182.23	50.62	6.08
	w/ Twes+	320.47	89.02	10.45	159.81	44.39	5.21
AliExpress	w/o Twes+	274.44	76.23	9.44	129.91	36.09	4.47
	w/ Twes+	277.32	77.03	9.35	126.58	35.16	4.27
Ask	w/o Twes+	242.01	67.22	8.07	116.56	32.38	3.89
	w/ Twes+	229.04	63.62	7.68	95.30	26.47	3.20
Imgur	w/o Twes+	281.64	78.23	9.33	140.07	38.91	4.64
	w/ Twes+	272.10	75.58	9.25	128.25	35.62	4.36
Imdb	w/o Twes+	263.58	73.22	8.76	126.61	35.17	4.21
	w/ Twes+	243.29	67.58	8.11	107.68	29.91	3.59
Fc2	w/o Twes+	285.57	79.33	9.48	137.10	38.08	4.55
	w/ Twes+	267.27	74.24	9.01	125.34	34.82	4.22
Stackoverflow	w/o Twes+	280.93	78.04	9.48	138.04	38.35	4.66
	w/ Twes+	273.38	75.94	9.03	126.09	35.02	4.17
Odnoklassniki	w/o Twes+	278.43	77.34	9.36	132.21	36.73	4.45
	w/ Twes+	271.33	75.37	9.03	130.99	36.39	4.36
Booking	w/o Twes+	265.45	73.74	8.97	125.44	34.84	4.24
	w/ Twes+	230.27	63.96	7.43	96.79	26.89	3.12
NicoVideo	w/o Twes+	290.32	80.65	9.55	141.76	39.38	4.66
	w/ Twes+	261.87	72.74	8.68	116.29	32.30	3.85
Flipkart	w/o Twes+	300.17	83.38	9.95	148.33	41.20	4.92
	w/ Twes+	259.85	72.18	8.39	116.55	32.37	3.76
BBC	w/o Twes+	277.69	77.14	9.21	130.69	36.30	4.33
	w/ Twes+	237.72	66.03	7.84	102.79	28.55	3.39
DailyMotion	w/o Twes+	271.54	75.43	9.05	126.66	35.18	4.22
	w/ Twes+	242.36	67.32	8.16	110.87	30.80	3.73
Wikia	w/o Twes+	238.85	66.35	7.98	104.32	28.98	3.49
	w/ Twes+	236.27	65.63	7.86	100.29	27.86	3.34
ChinaDaily	w/o Twes+	321.43	89.29	10.38	160.22	44.50	5.17
	w/ Twes+	290.86	80.80	9.60	136.77	37.99	4.51

B.2 Power Comparison over Mobile Phone

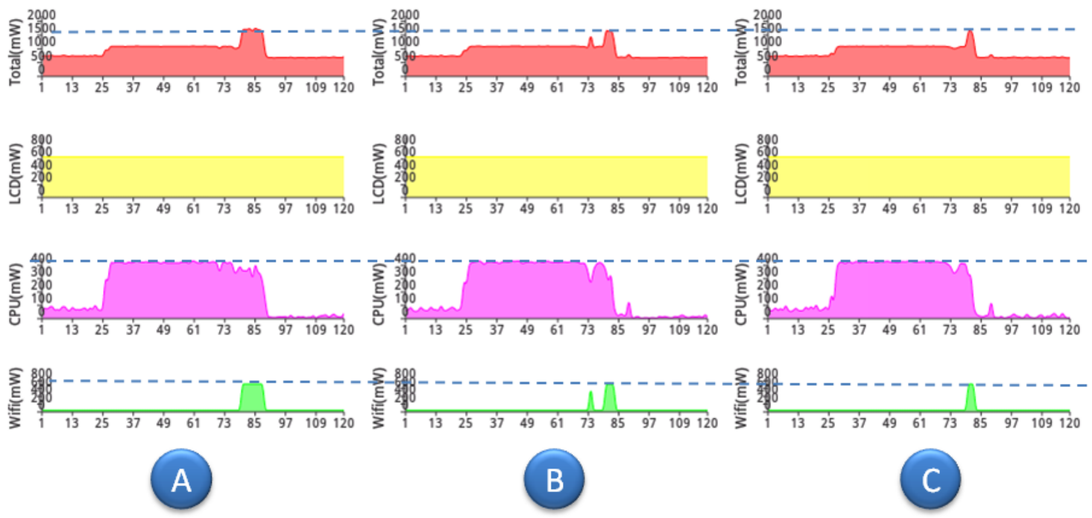


Figure B.1: Power Consumption While Browsing Ebay without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

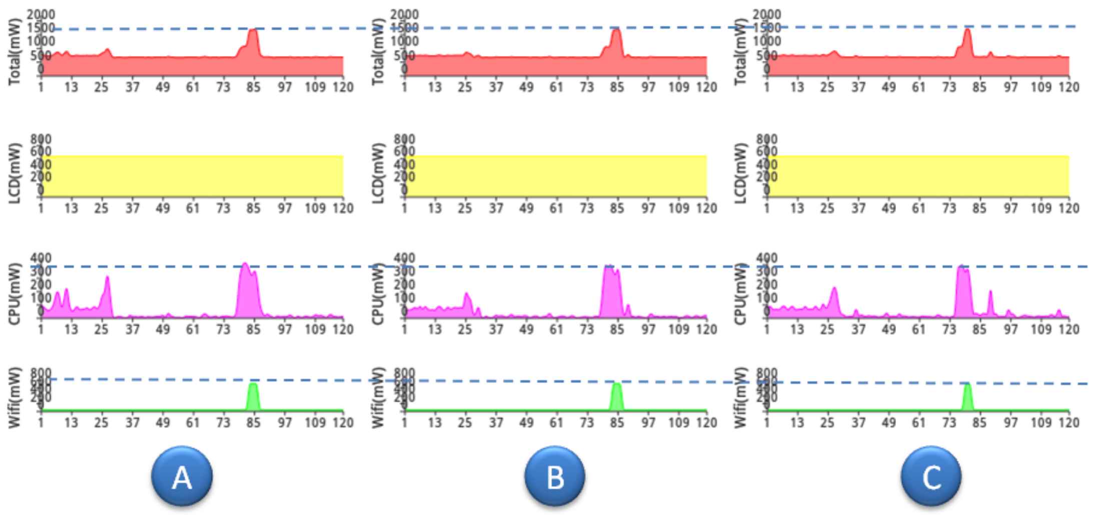


Figure B.2: Power Consumption While Browsing Microsoft without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

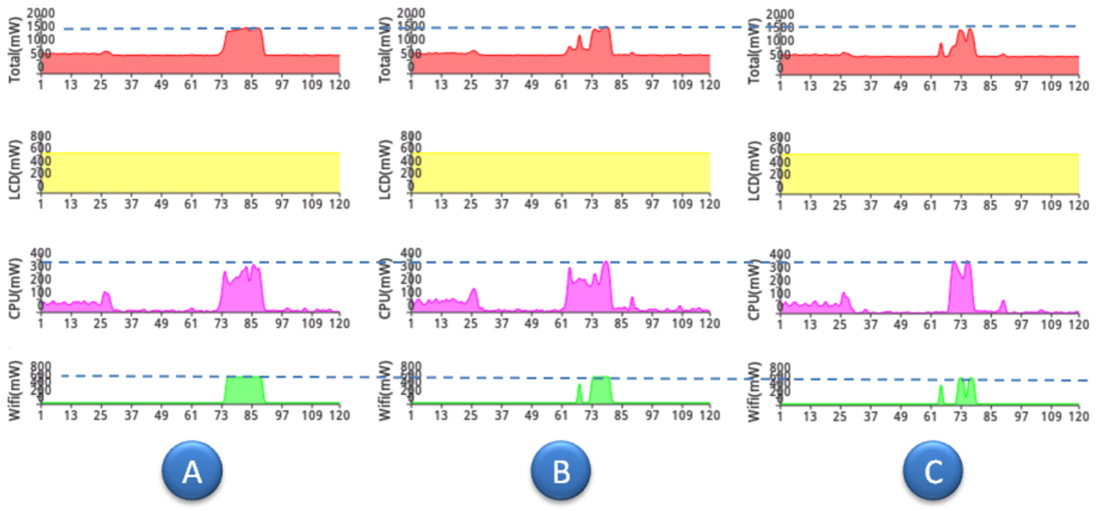


Figure B.3: Power Consumption While Browsing Aliexpress without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

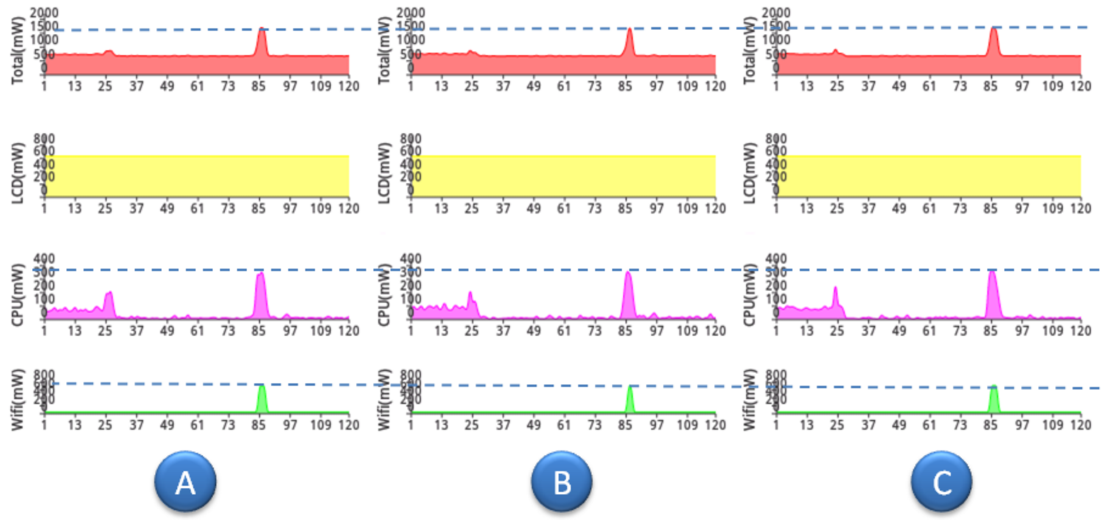


Figure B.4: Power Consumption While Browsing Ask without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

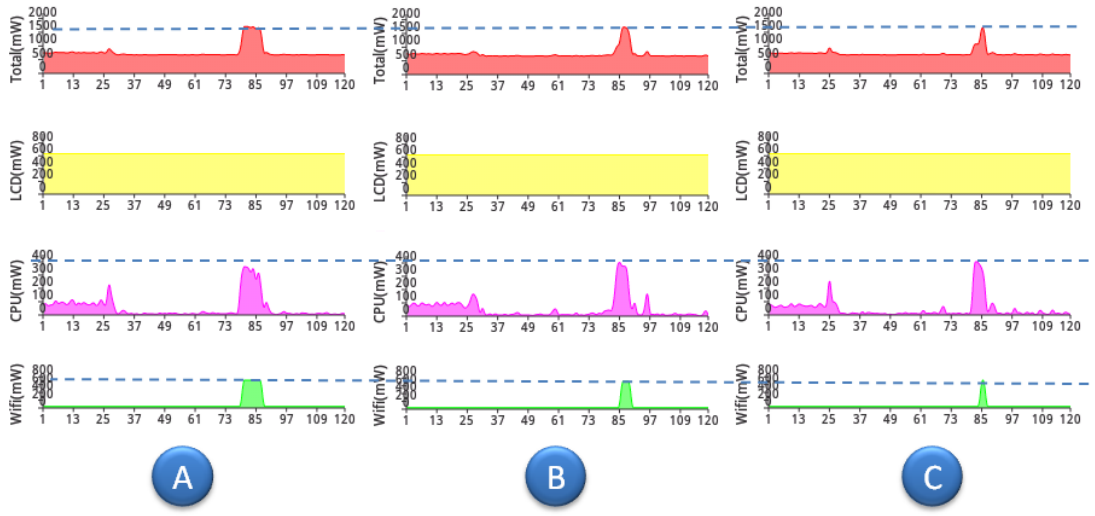


Figure B.5: Power Consumption While Browsing Imgur without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

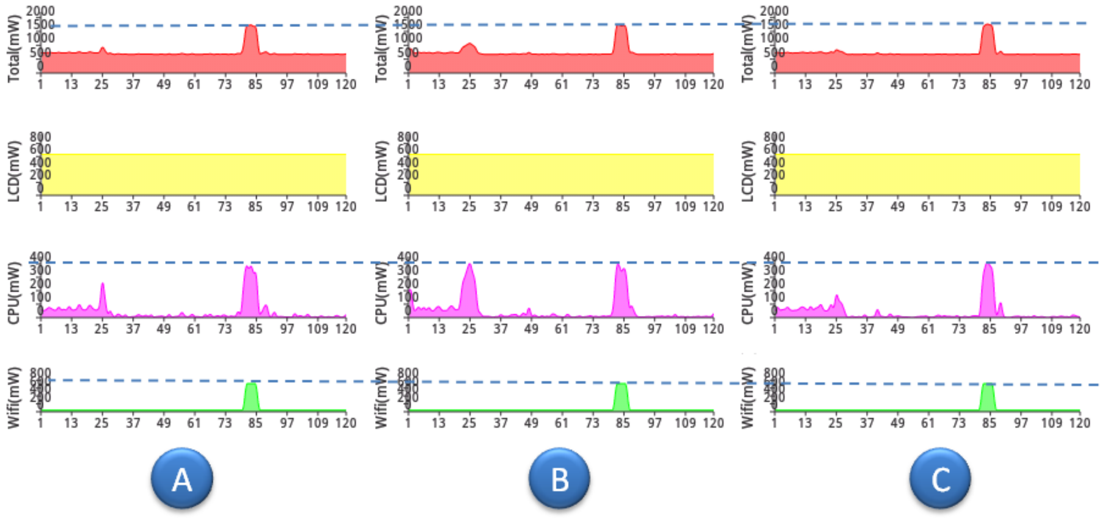


Figure B.6: Power Consumption While Browsing Imdb without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

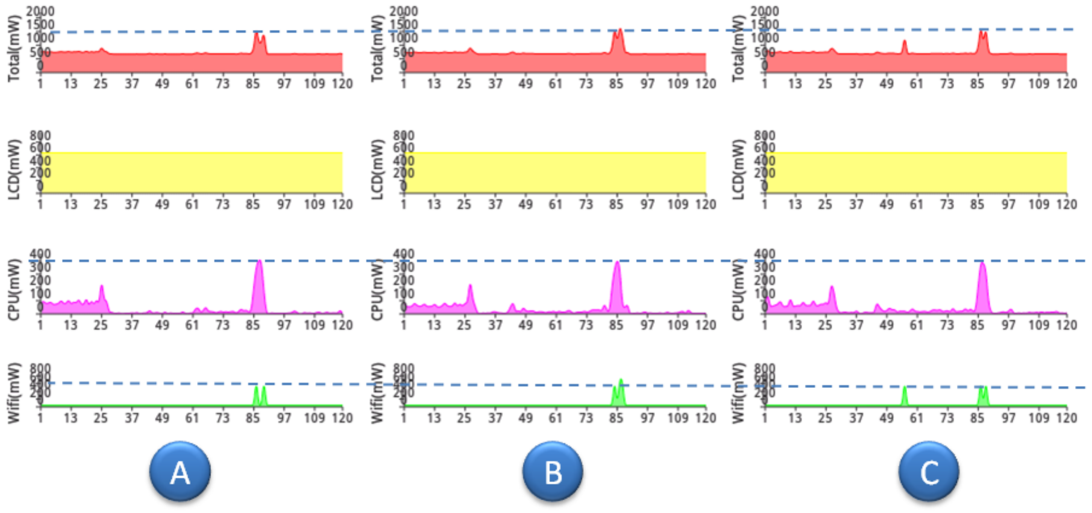


Figure B.7: Power Consumption While Browsing Fc2 without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

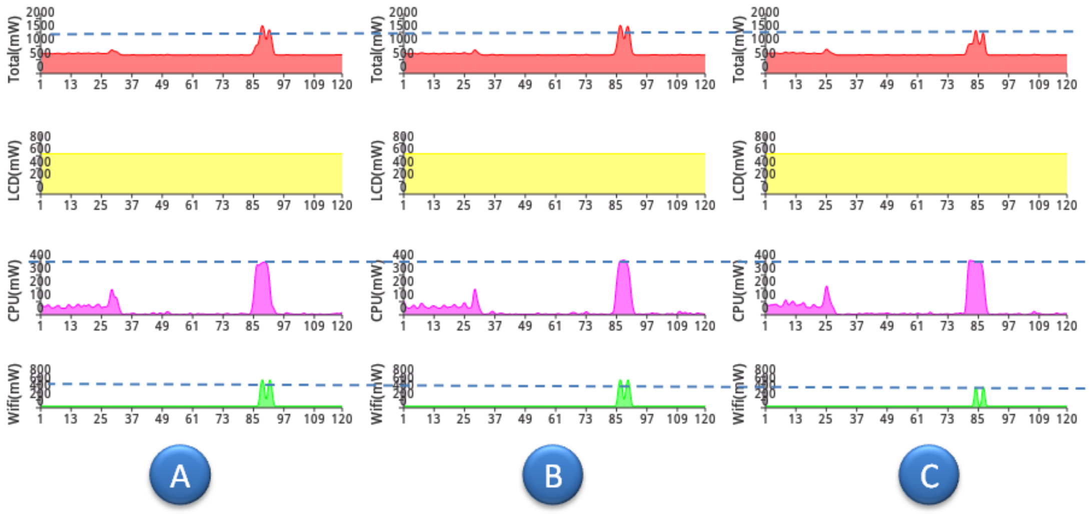


Figure B.8: Power Consumption While Browsing StackOverflow without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

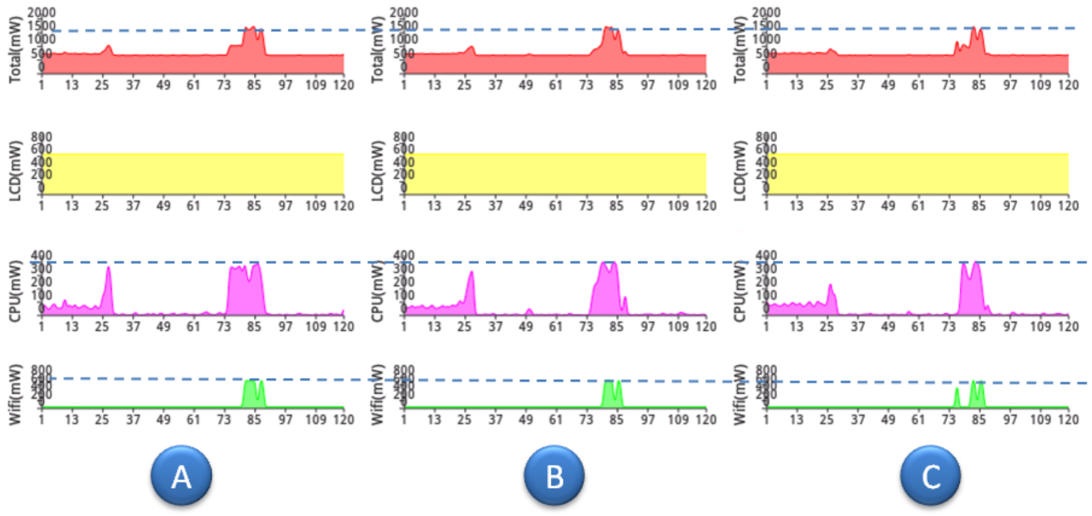


Figure B.9: Power Consumption While Browsing Odnoklassniki without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

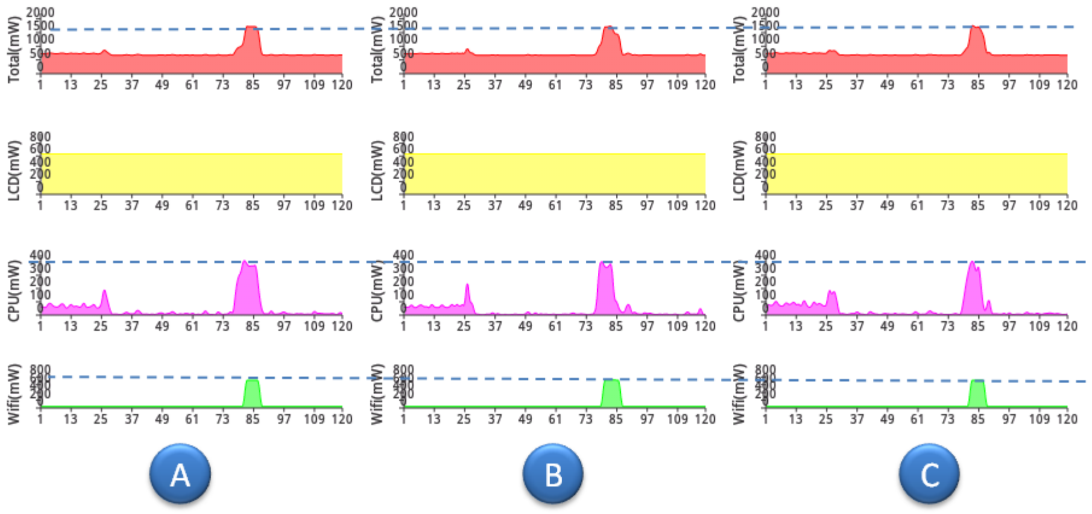


Figure B.10: Power Consumption While Browsing Booking without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

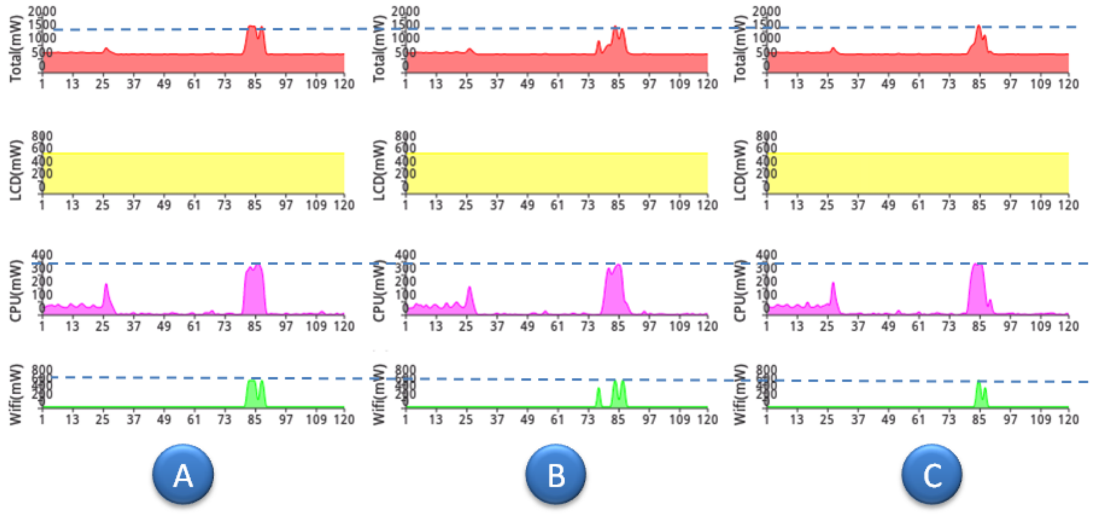


Figure B.11: Power Consumption While Browsing Nicovideo without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

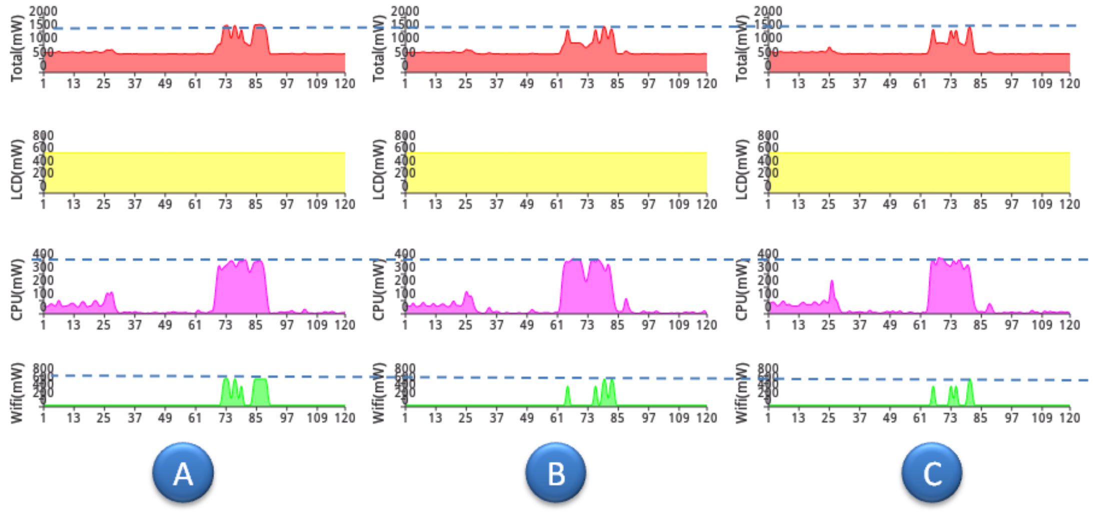


Figure B.12: Power Consumption While Browsing Flipkart without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

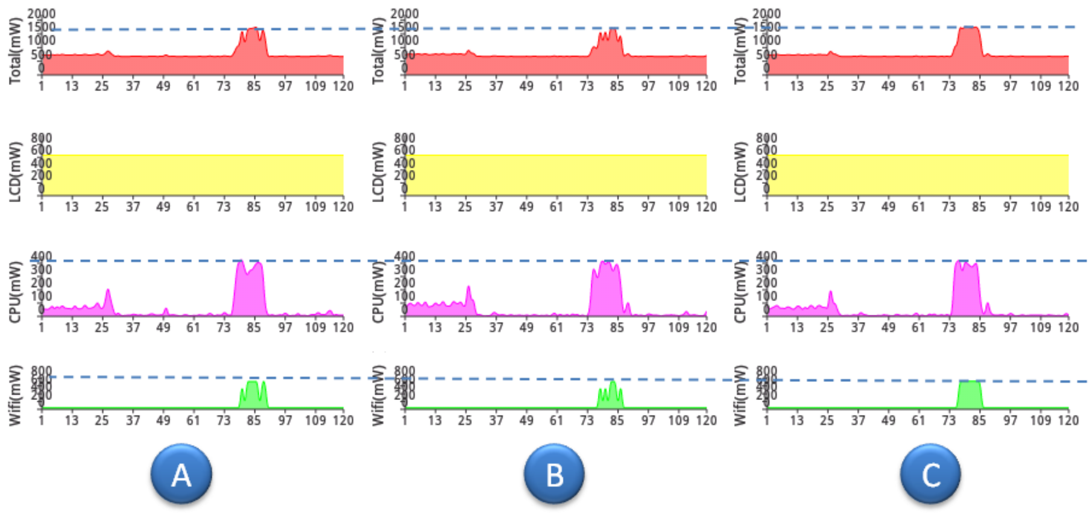


Figure B.13: Power Consumption While Browsing BBC without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

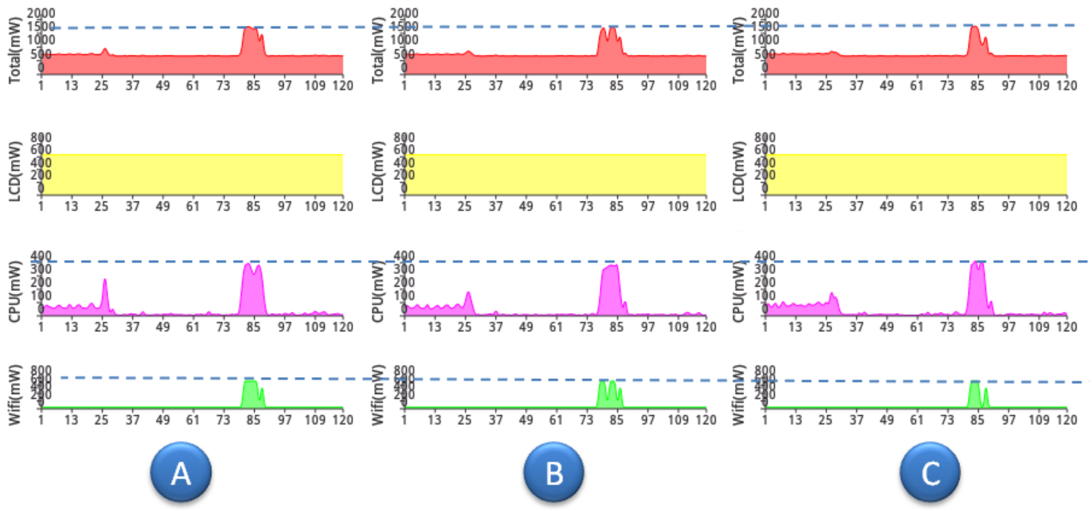


Figure B.14: Power Consumption While Browsing Dailymotion without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

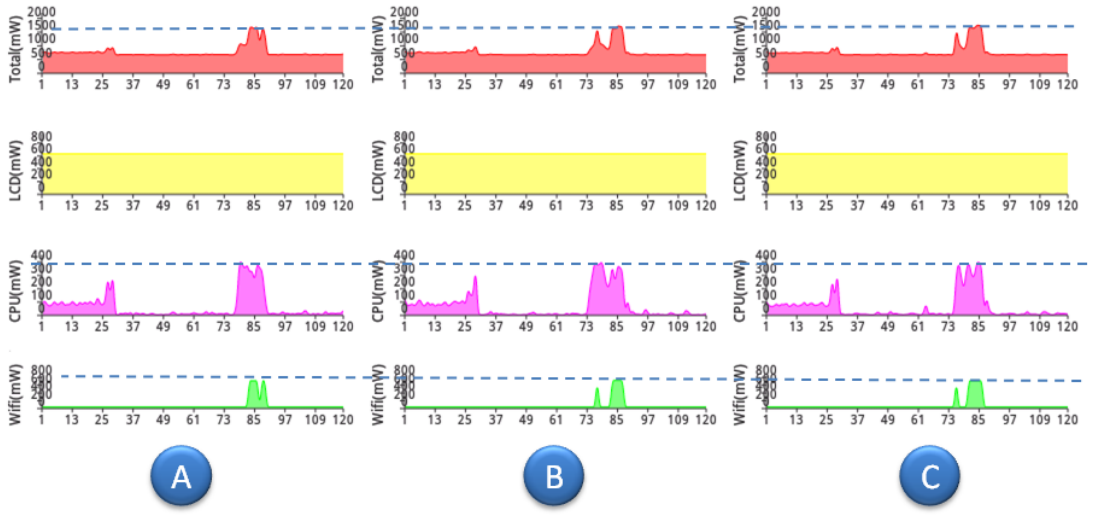


Figure B.15: Power Consumption While Browsing Wikia without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

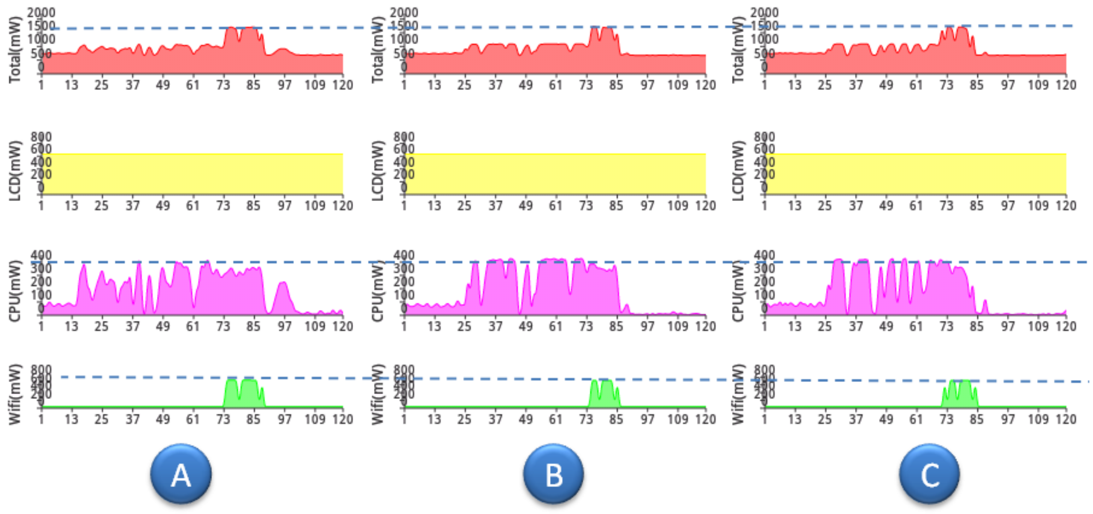
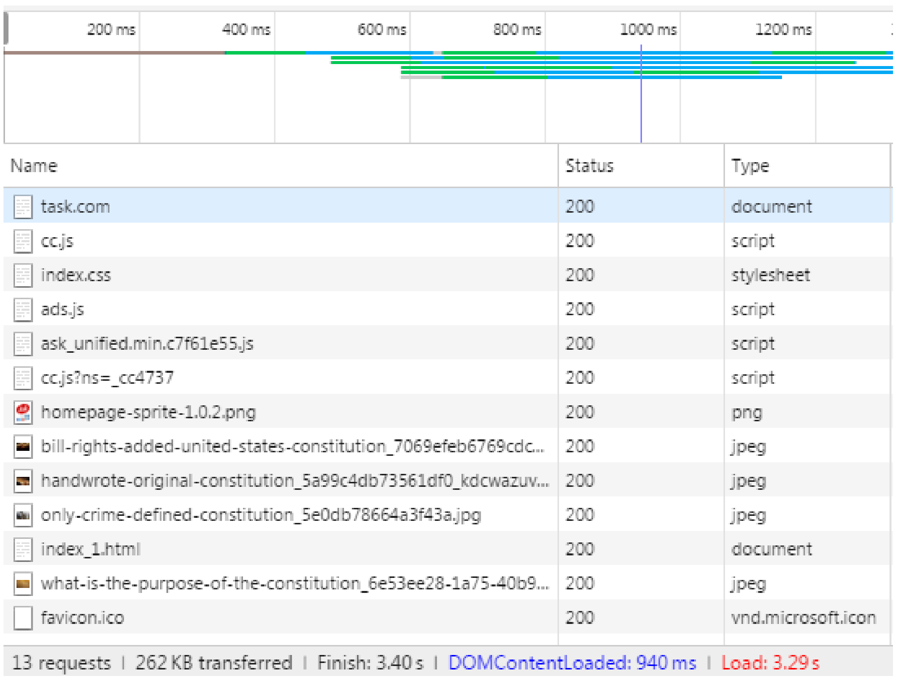


Figure B.16: Power Consumption While Browsing Chinadaily without Twes+(A), with Twes+ and without Resizing Feature (B), and with Twes+ and Resizing Feature (C)

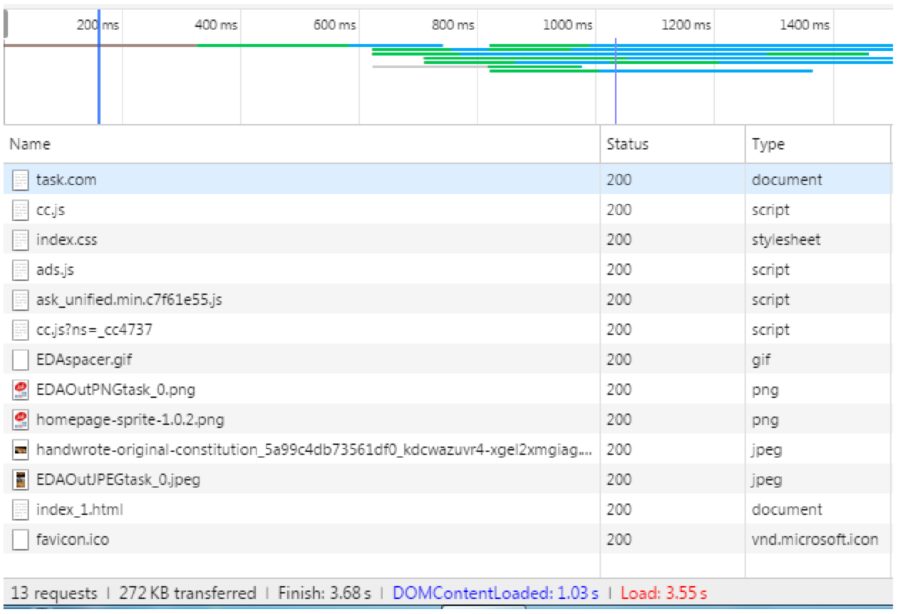
B.3 The Requests/Responses of Web Pages with and without Twes+



Name	Status	Type
task.com	200	document
cc.js	200	script
index.css	200	stylesheet
ads.js	200	script
ask_unified.min.c7f61e55.js	200	script
cc.js?ns=_cc4737	200	script
homepage-sprite-1.0.2.png	200	png
bill-rights-added-united-states-constitution_7069efeb6769cdc...	200	jpeg
handwrote-original-constitution_5a99c4db73561df0_kdcwazuv...	200	jpeg
only-crime-defined-constitution_5e0db78664a3f43a.jpg	200	jpeg
index_1.html	200	document
what-is-the-purpose-of-the-constitution_6e53ee28-1a75-40b9...	200	jpeg
favicon.ico	200	vnd.microsoft.icon

13 requests | 262 KB transferred | Finish: 3.40 s | DOMContentLoaded: 940 ms | Load: 3.29 s

(a) The Requests/Responses of Ask without Twes+

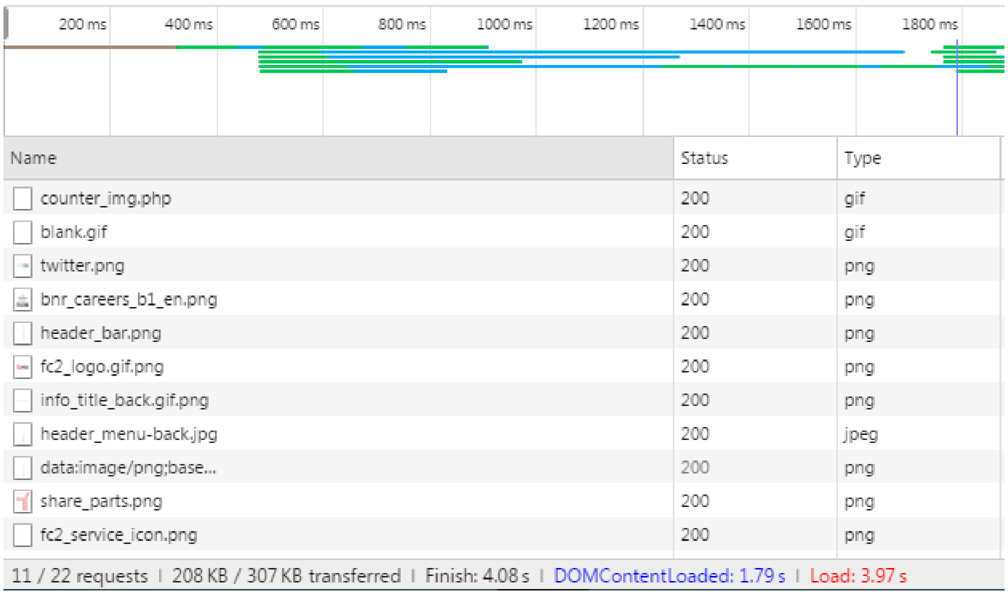


Name	Status	Type
task.com	200	document
cc.js	200	script
index.css	200	stylesheet
ads.js	200	script
ask_unified.min.c7f61e55.js	200	script
cc.js?ns=_cc4737	200	script
EDAspacer.gif	200	gif
EDAOutPNGtask_0.png	200	png
homepage-sprite-1.0.2.png	200	png
handwrote-original-constitution_5a99c4db73561df0_kdcwazuvr4-xgel2xmgiag...	200	jpeg
EDAOutJPEGtask_0.jpeg	200	jpeg
index_1.html	200	document
favicon.ico	200	vnd.microsoft.icon

13 requests | 272 KB transferred | Finish: 3.68 s | DOMContentLoaded: 1.03 s | Load: 3.55 s

(b) The Requests/Responses of Ask with Twes+

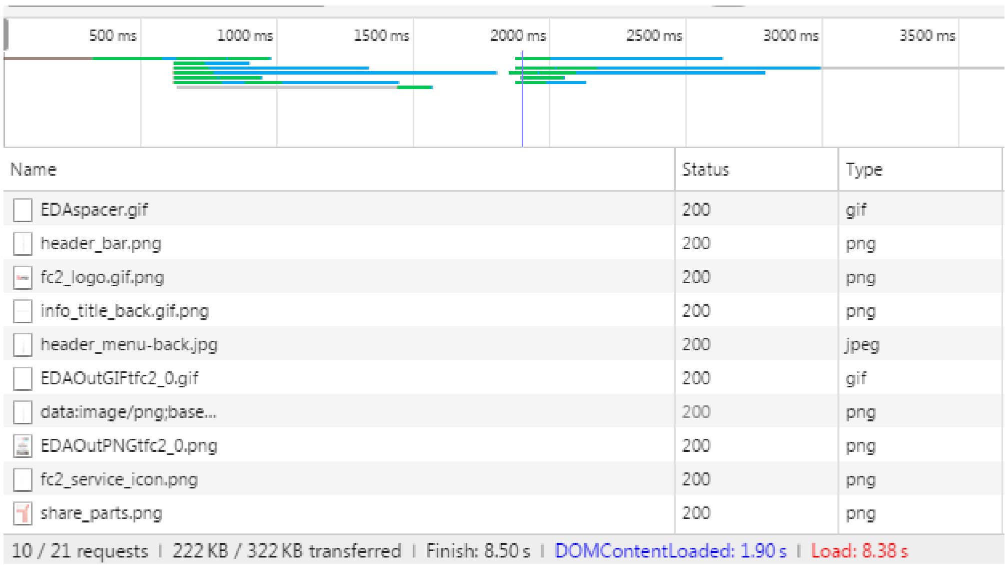
Figure B.17: The Requests/Responses of Ask with and without Twes+



Name	Status	Type
counter_img.php	200	gif
blank.gif	200	gif
twitter.png	200	png
bnr_careers_b1_en.png	200	png
header_bar.png	200	png
fc2_logo.gif.png	200	png
info_title_back.gif.png	200	png
header_menu-back.jpg	200	jpeg
data:image/png;base...	200	png
share_parts.png	200	png
fc2_service_icon.png	200	png

11 / 22 requests | 208 KB / 307 KB transferred | Finish: 4.08 s | DOMContentLoaded: 1.79 s | Load: 3.97 s

(a) The Requests/Responses of Fc2 without Twes+

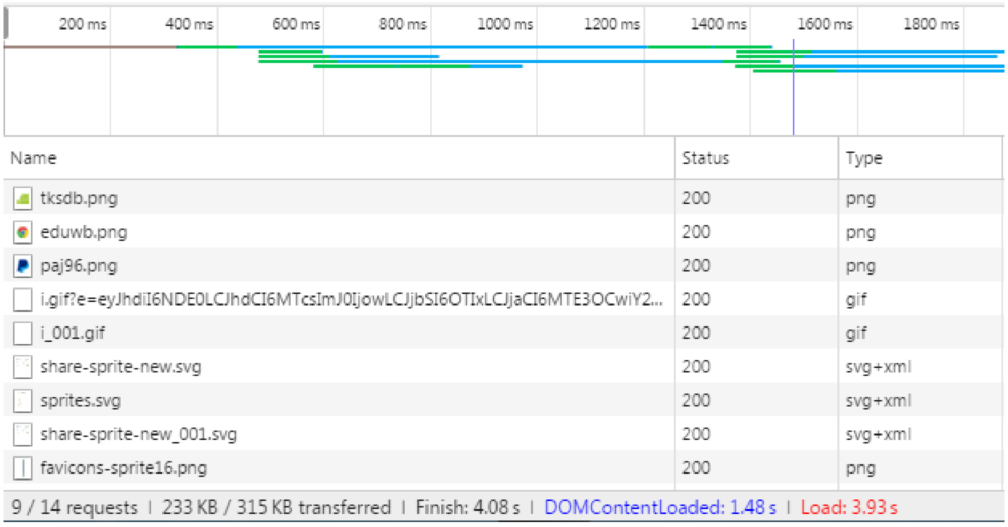


Name	Status	Type
EDAspacer.gif	200	gif
header_bar.png	200	png
fc2_logo.gif.png	200	png
info_title_back.gif.png	200	png
header_menu-back.jpg	200	jpeg
EDAOutGIFtfc2_0.gif	200	gif
data:image/png;base...	200	png
EDAOutPNGtfc2_0.png	200	png
fc2_service_icon.png	200	png
share_parts.png	200	png

10 / 21 requests | 222 KB / 322 KB transferred | Finish: 8.50 s | DOMContentLoaded: 1.90 s | Load: 8.38 s

(b) The Requests/Responses of Fc2 with Twes+

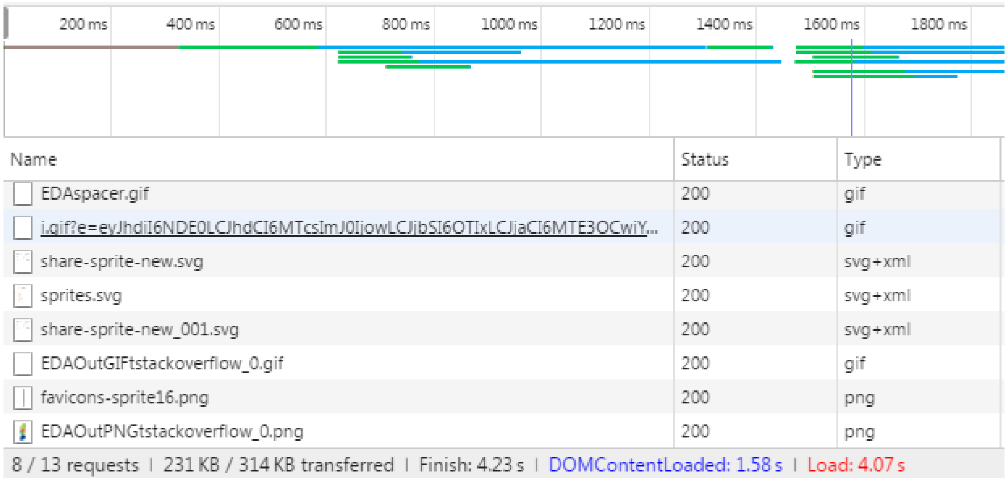
Figure B.18: The Requests/Responses of Fc2 with and without Twes+



Name	Status	Type
tksdb.png	200	png
eduwb.png	200	png
paj96.png	200	png
i.gif?e=eyJhdiI6NDE0LCJhdCI6MTcsImJ0IjowLCJjbSI6OTIxLCJjaCI6MTE3OCwiY2...	200	gif
i_001.gif	200	gif
share-sprite-new.svg	200	svg+xml
sprites.svg	200	svg+xml
share-sprite-new_001.svg	200	svg+xml
favicons-sprite16.png	200	png

9 / 14 requests | 233 KB / 315 KB transferred | Finish: 4.08 s | DOMContentLoaded: 1.48 s | Load: 3.93 s

(a) The Requests/Responses of Stackoverflow without Twes+



Name	Status	Type
EDAspacer.gif	200	gif
i.gif?e=eyJhdiI6NDE0LCJhdCI6MTcsImJ0IjowLCJjbSI6OTIxLCJjaCI6MTE3OCwiY...	200	gif
share-sprite-new.svg	200	svg+xml
sprites.svg	200	svg+xml
share-sprite-new_001.svg	200	svg+xml
EDAOutGIFtstackoverflow_0.gif	200	gif
favicons-sprite16.png	200	png
EDAOutPNGtstackoverflow_0.png	200	png

8 / 13 requests | 231 KB / 314 KB transferred | Finish: 4.23 s | DOMContentLoaded: 1.58 s | Load: 4.07 s

(b) The Requests/Responses of Stackoverflow with Twes+

Figure B.19: The Requests/Responses of Stackoverflow with and without Twes+

APPENDIX C

METHODOLOGY USED IN COLLECTING MATERIALS FOR THE EVALUATION

When we downloaded the experimental materials, we faced four major problems: Download problem, missing file issues, external links, JavaScripts. In this section, we discuss these problems and the taken actions.

Download Problem: In Chapter 4 Section 4.2, we discuss the web pages that are removed from the experimental material list. For downloading the web pages, four methods are used: Wget, HTTrack Website Copier, Mozilla Firefox Scrapbook and saving from the browser. We use multiple tools because sometimes because of the some problems. For example, to download Ebay, Wget and HTTrack Website Copier are tried. The problem is after three days, the download of the web page was still going. We switched to the Mozilla Firefox Scrapbook and just for 2 level we download the web page. The problem we face with Mozilla Firefox Scrapbook is during the download process tool does not download and stop at certain point. The reason is mainly web page has an authentication step that Mozilla Firefox Scrapbook could not pass for example the login parts of the web pages. The reason why we tried to save a web page from the browser is the failure of all other tools.

Missing File Issues: Some web pages we faced missing file issue. Sometimes web pages work without some files and actually browser gives an error during browsing these web pages.

However, these missing files are effecting our measurements because they extend the loading duration of the web page and delay loading process of other files. To solve this problem, we remove the file from the source code and if it changes the look & feel of the web page, we remove this web page from the list, for example, Disney web page [1]. Another problem is the different version of the web pages. When we use the tools not the browser, they download all the versions and it does not finish especially if the web page is changing frequently (i.e., the news pages, the shopping pages). This case we use the browser to download the web page and web page has the DNS restriction. We can access the Turkish version of the web page. If there is this restriction, the missing files are hard to access and the web page does not look as the original web page, for example, MSN [2].

External Links: Another problem is the external links that are not downloaded. These days pages are using the lazy-load feature. When the page is downloaded, instead of the real image, the temporary image for holding a place in the layout is downloaded and the link of the original image stays as an external link. We track these cases and download the original images from their links and update the external link. The same external link issue occurs with the fonts and script files as well especially if they are inside the other external files. For the tests, we download and fix the links for all these issues.

JavaScripts: Web pages are using also JavaScripts used by Google Analytics or Twitter or Facebook. These scripts are sending requests to outside network. We remove these scripts from all web pages. When we are preparing the materials for the evaluation, we face with these problems. If any changes occur in the look & feel of the web page, we remove these web pages from our list.

[1] `http://go.com/`
[2] `http://www.msn.com/tr-tr/`