UNIVERSITY OF OSLO
Department of Informatics

Analyzing the compression of Opera Mini ™ traffic

Stian Østen

Network and System Administration

Oslo University College

May 19, 2008

Analyzing the compression of Opera Mini ™ traffic

Stian Østen

Network and System Administration
Oslo University College

May 19, 2008

Abstract

Opera Mini ^{TM} is a mobile web browser. It was developed for mobile phones incapable of running regular web browsers and released globally in January 2006. It has gained great popularity and can be downloaded free of charge to your mobile phone. The Opera Mini client consists of a small Java MIDlet and the only requirement is that your mobile phone support Java ME applications. Opera Mini is a server-client technology. The main benefit of this compared to a regular mobile browser is that all web pages are being pre-processed and compressed in dedicated Opera servers before the information is sent to the mobile phones. This is to make the information more suitable for the small hand held devices as well as to reduce the amount of information transferred to the mobile phones. Opera claims that this technology increases the traffic rates by two to three times, which means better response time and lower cost for the end user. The fact that all traffic is handled by dedicated servers demands for extensive link and server capacity. Opera has a large cluster of Opera Mini servers connected with gigabit links. This is a huge cost for the company.

In this thesis we will analyze log files generated from Opera Mini traffic. We will keep main focus on parameters that affects the amount of data transferred to the mobile phones. Our motivation for this choice is that in a regular Opera Mini session, the tight link of the connection will often be the wireless connection between the mobile phone and the mobile operator [1]. This makes the level of compression of the data transferred from the Opera Mini server to the client of great importance for the user experience. Different data types will achieve different level of compression. Some data are more difficult to compress than other. Pictures, already compressed data and encrypted data will not be compressed as much as plain html files or text files [2]. The extensive use of images in web pages is the factor we believe have the greatest impact on the achieved level of compression. In most scenarios the user wants to view images, but the quality of the displayed images is of less importance. The regular Opera Mini user can choose between four different image quality settings in the browser. Our analysis reveal that these settings have impact on the results.

Acknowledgments

This thesis is the conclusion of the two year master's degree in network and system administration at Oslo University College in collaboration with the University of Oslo. First I would like to thank Professor Mark Burgess for giving me the opportunity to attend this exciting master's program and for the dedication he puts in his profession. I will thank all my fellow students for making these two years memorable. It has been fruitful to collaborate with nice people from all parts of the world and fun to be part of this class.

This thesis was encouraged by the system department at Opera Software. I am grateful for the nice atmosphere in this department and for you all letting me feel included in the group from day one. I will in particular thank Claudia Eriksen, Sven Ulland and Trond Aspelund for their assistance during the project.

I will thank my supervisor, assistant professor Hårek Haugerud for his nice conversations and great help during this stressful period. I will also send my gratitude to my 'second supervisor', Kyrre Begnum, for getting me back on track when I got lost.

To my 'wife' waiting for me at home, I love you!

Oslo, May 19 th, 2008

Stian Østen

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The World Wide Web is the greatest information resource in the world. The development in the field is rapid and the number of people with access to these resources are constantly increasing. This is not only due to construction of new infrastructure, but the development of new technology. A technology with great potential is mobile web browsers. The use of mobile devices is increasing and the devices are getting more advanced. This makes them suitable for other operations than calling. The users rapidly take advantage of the new possibilities and change their habits. This is a marked that is expected to grow further and cutting edge applications are of great importance for the vendors in the fight of market shares. The struggle to provide the best service gains the user and give them the opportunity to choose the product best suited for their needs.

In this thesis we will investigate traffic logs generated by the Opera Mini mobile browser technology. This is a browser made for mobile phones and other low capacity hand held devices. The benefit with Opera Mini compared to a regular mobile browser is that all web pages are being pre-processed and compressed in dedicated Opera servers before the information is sent to the mobile phones. The aim of this is to make the information more suitable for the small hand held devices as well as to reduce the size of the information needed to be transferred to the mobile client. Opera claims that this technology increase the traffic rate by two to three times, which means better user experience and lower cost for the end user, compared to normal web browsing [3].

The fact that all Opera Mini traffic is handled by dedicated servers demands extensive link and server capacity. This is a great cost for the company, but it makes the system easy to maintain. It gives Opera the opportunity to easily monitor and analyze the use of their service. They have access to all information which can be used for capacity planning, marketing and other means. Today Opera has gigabit connections and approximately 54 servers

dedicated to Opera Mini traffic. The traffic load will over seed these limits in the near future, and Opera is planning to spread the load by placing servers at other locations. To spread the load over several locations will increase the redundancy and give them the opportunity to run at reduced level if one location is down.

For this thesis we will take advantage of log files collected over a time period of two weeks and use off-line data mining to analyze the data. Data mining of log files and network monitoring are important tool for system administrators in order to know what is going on in their networks and to make sure that they keep their system optimized based on their needs. There are several ways to perform these tasks and you can choose amongst an enormous number of tools to help you both gathering and analyzing the information. Based on your policy you can define how much effort and cost you need to invest in this process. No tool covers all aspects, but based on your needs, a combination of tools can fulfill your demands. The complexities of networks are continuously increasing and a well planned network is important to maintain overview and control of your system.

With the network topology today wireless links are often the bottleneck in data communication with mobile devices. The transfer on the wireless link will in many cases cover a great part of the total transfer time. For this reason the level of compression of the transferred data between the server and the client is of great importance for the user experience. Different data types will achieve different level of compression. Some data are by far more difficult to compress than other. Pictures, already compressed data and encrypted data will not be compressed as much as plain html files or text files [2].

1.2 Problem statement

The data compression for the Opera Mini traffic is performed in transcoders outside the control of the users and most users will not offer this part of the technology any thoughts. It is common to use the default settings in a browser, not only for Opera Mini. Due to the high capacity in the desktop environment, the effort in creating low content web pages is limited. In Opera Mini the client can configure the quality settings for displaying downloaded images. The extensive use of images in web pages makes this setting of great importance if you look at the amount of data that is transferred to the client. In most scenarios the user wants to view images, but the quality of the displayed images is of less importance. The regular Opera Mini user can choose between four display options. No images, low quality images, medium quality images or high quality images. Low quality images is the default setting. We assume that this choice has great impact on the data transfer to the client and hence the user experience. This has made us come up with the following problem statement (PS):

PS *- What context affects the compression of the data?*

The context can be Meta data such as quality of images or file sizes. When we talk about compression, we mean the difference between the two parameters 'uncompressed' and 'compressed', found in the log files. 'Uncompressed' is the file size of the requested web page downloaded to the transcoder, while 'compressed' is the file size of the requested web page transferred to the client. The parameters give us the file size in bytes. By the word data we mean the web pages transferred and when we talk about images we mean images included in the download of these web pages.

In particular we will look at the following sub statments:

PS1 *- if there is a connection between the image display quality settings configured in the Opera Mini browser and the size of the files downloaded to the transcoders.*

We assume that the size of the web pages downloaded to the transcoder is equal for all cases where images are included in the download. We also assume that the size of the downloaded web pages is less for the case with image display setting configured to *no images* and the images are excluded from the download.

PS2 *- if there is a connection between the quality of the image transferred to the client and the achieved level of compression*

We assume that the amount of data transferred to the mobile user is decreasing as the quality of the images is reduced.

PS3 *- if there is a connection between the quality of the image transferred to the client and the time it takes to download and process the request by the transcoder*

The time it takes to download and process the request in the transcoder is stored in the log files as a parameter called 'total time'. It states the time it takes from the transcoder receives a request from a client to the web page is downloaded and processed by the transcoder, ready to be transferred to the client. We assume that the total time it takes to handle the request in the transcoder is increased as the quality level of the images is reduced. The low quality image demand for a more extensive file compression than the ones with high image quality and we believe that this will increase the 'total time'.

PS4 *- if there is a connection between the size of the files downloaded to the transcoder and the 'total time' in the transcoder*

We assume that the total time to handle the request by the transcoder will increase as the size of the downloaded files increase.

PS5 *- if there is a connection between the uncompressed file sizes and the compressed file sizes.*

We assume that the compressed files will increase as the uncompressed files increase and that this will follow a linear pattern.

1.3 Approach

The work with this thesis was encouraged by the the system department at Opera Software ASA. The Opera Mini browser experience constantly growing popularity and the service generates huge amounts of data traffic. This require extensive server and link capacities that produce potential work for the department. The original problem description was focusing on global site load balancing, but during the early period of the project this was re-written several times. The final description was focusing on log file analysis of the data generated by Opera Mini users.

We have been two students performing a study for Opera on the same data material, Valeri Cheremetiev and the author of this thesis. We have used this advantage in the first phase of the project that was consisting of exploratory work with the provided data material. We have shared knowledge in the process to find the importance of the different parameters in the log files and have had many useful discussions on how we can use the different parameters to clarify important problems related to the Opera Mini service. We have been collaborating in the attempt to figure out the optimal strategy on how to collect and process the selected information. Valeri has been the author of the scripts we used to extract the data from the log files and store it in a database. We agreed on different angles for the analysis and from the day the database was ready the work has been performed separately.

Opera Mini generates huge amounts of traffic and monitoring the system demands high performance equipment. The cluster has gigabit links and the average traffic is around 700Mbps. Capturing the traffic will rapidly lead to enormous amounts of data. We got access to all log files collected from the cluster with Opera Mini transcoders and even this was huge amounts of data for each day. We decided to use files from a two weeks period for our analyses. The first thing to do was to download the files and unpack them. The size of the log files from each of the 54 transcoders was approximately 400MB per day. This gave us a total of close to 300GB of log files.

We started by investigating the content and the format of the log files to get an overview of what information we got. The format of the log files consisted of 36 different parameters. All the parameters were not explicitly written and we had to discuss the actual meaning of some parameters with Opera Mini experts. When we had figured out the meaning of all parameters, we chose the ones we wanted to use for our analyses based on our problem statements. These where , 'Uncompressed file size', 'Compressed file size', 'Total time' and 'Image quality'. We created Python scripts to collect the relevant information from the log files and store it in a database. We used a widely deployed database engine, Sqlite [4]. To collect and quire the database is a time consuming task when we work with this large amount of data. The size of the database containing all 14 days reached 30GB. To present the results graphically we have used the plotting utility, Gnuplot [5].

1.4 Thesis outline

In this outline we describe briefly the chapters of the thesis.

Chapter 1 - In the first chapter we have been trying to give the reader basic information about the technologies described and our motivation for doing this project. We have explained shortly how we approached the task and what problems we wanted to address.

Chapter 2 - In this chapter we describe some relevant background information and previous work related to our project. We explain the basic theory of web browsing and try to make the reader familiar with the Opera Mini technology.

Chapter 3 Here we introduce the reader to different methodologies and the uncertainties of science. We compare four different methods and explain why we chose the one we did.

Chapter 4 - In this chapter we describe the hardware and software used in the project. All the equipment used have 'normal' specs and the software we have used for the work of this thesis has been free of charge.

Chapter 5 - Chapter 3 Here we introduce the reader to different methodologies and the uncertainties of science. We compare four different methods and explain why we chose the one we did.

Chapter 6 - In this chapter we analyze and discuss our results thoroughly. We try address the validity and repeatability of the project, in addition to revisit our problem statements.

Chapter 7 - At the end we draw some conclusions based on our findings and propose possibilities for further work.

Chapter 2

Background and literature

In this chapter we will introduce the underlying technology related to web browsing and look at some previous work done in the field.

2.1 Web browsers

A web browser is a software application that can be installed on a computer. It's main purpose is to display information from web pages stored on web servers on the world wide web. Web browsers make all kind of information easy available to a large number of people.

A web browser gets access to web pages through web servers. A web server is a computer program that communicate with a web browser using HTTP(hyper-text transfer protocol). The browser will request the server for web pages and the server will respond with the correct information. Each web page must have an unique URL(Uniform resource locator) that identifies the page. The user can write the URL in the web browser address bar to get access to the desired web page.

The information in a web page is primarily stored in HTML(hyper-text markup language) file format, but as web pages have become more advanced, browsers can display images and sound stored in other formats. HTML give you the possibility to easily define the layout of the document with a combination of text, pictures, animations and links.

There are many participants contributing to the web technology and several tools that can help you when you work with Web technology. There have been different standard with the different tools, but most new tools try to follow the W3C standards. W3C is a consortium where member organizations contribute to develop standards for World Wide Web. WC3 was founded by Sir Tim Berners Lee in 1994, five years after he invented the World Wide Web (The Web). The Web is a system with interlinked hypertext documents that can be accessed via the Internet. The first web site was a description by Berners Lee on how he made it. He used a NeXTcube both as the web server and to

write the first web browser, WorldWideWeb. On August 6, he posted a summary of the project on the alt.hypertext newsgroup and this date marked the beginning of the Web as a publicly available service on the Internet [6].

It took two years before a web browser managed to bring the Web to the 'people'. NCSA Mosaic web browser was released in September 1993 and the introduction of this browser lead to an explosion in the use of Web services. It was a graphical browser originally for the Unix platform, but it did not take long before it was running on Amiga, Apple Macintosh and Microsoft Windows as well. The leader of the Mosaic team quit to start a new company that one year later, in October 1994, released Netscape Navigator. Netscape Navigator soon became the most popular web browser with the major market share with a maximum of 86 per cent in 1996. By this time Microsoft had started to launch their operating system with their web browser, Internet Explorer, pre installed and managed to turn the marked share in their favor with a maximum at 92% in 2004. From this time other vendors have managed increase their share, but Microsoft is still the most used with approximately 75% [7]. From table 2.1 you can see the popularity of the most used browsers based on numbers from Net Applications, 09.05.2008 [8].

Web browser usage shares

Internet explorer	Firefox	Safari	Opera	Netscape	Mozilla	Opera Mini	MPIE 4.0
74.83%	17.76%	5.81%	0.69%	0.56%	0.16%	0.04%	0.02%

Table 2.1: Web browser usage shares based on numbers from Net Applications, 09.05.2008.

2.2 Mobile Web browsers

A mobile web browser is a web browser intended for small devices like mobile phones and PDA's. They take advantage of the same technology as a regular browser, but due to limited recourses in the hand held devices, they are stripped down to optimize the performance. In addition to low memory and low bandwidth capacities the main difference between a workstation and a mobile device is the size of the display. Regular web pages are not created with the intention of being displayed on small screens and the mobile web browser must be able to modify the layout of the pages to make them suitable for the different displays [9]. There are many browsers for mobile phones, but most of them rely on the WAP technology. The first mobile browser believed to supported HTML was called HitchHiker. It was a browser made by the British company, STNC, released in 1997. This company was bought by Microsoft in 1999 and HitchHiker changed name to Pocket Internet Explorer(PIE) 2.0. The next version PIE 3.0 added support for jScript and some secure protocols. The last version, PIE 4.0, was the first to support ActivX, CSS and VBScript. From the introduction of operating system Windows Mobile 5.0 in May 2005

the browser changed name to Internet Explorer Mobile. Because of the different names and different versions it is hard to figure out how large share of the marked the different Windows browsers have got. The only version we find on a list powered by Net Applications, is the PIE 4.0 which have 0.02% of the market. The first mobile browser we find on the list of the most used browsers is Opera Mini. With 0.05% of the market share it is located on seventh place (See table {tab:webtab. This number include all versions of Opera Mini. This project is based on traffic generated by Opera Mini users and the Opera Mini technology will be described later in this chapter.

When people browse the web with regular browsers they are often connected with fixed links charged at flat rates. This makes the pattern of the browsing include a lot of non interacting time periods [10]. People like to keep the service running to minimize the response time when they need the service. Mobile web services are often charged by connection time or amount of downloaded information. This affects most users in the way that they will terminate the connection as soon as they have achieved the requested information and restart the application at demand. This behavior will probably change when the prices for fixed rate prescriptions for mobile web services are reduced. In table 2.2 and 2.3 we present the cost of different prescriptions from the two major mobile providers in Norway, Telenor [11] and NetCom [12].

Prices - Telenor

Prescriptions	Description	Cost in Norwegian kroner
Connect Premium	Free usage	499 NOK per month
Connect Standard	Usage up to 500 MB	312 NOK per month
Connect Fritid	Free usage - 17.00-08.00 + weekends	99 NOK per month
Connect Basic	Usage up to 20 MB	99 NOK per month
No data prescription	20 NOK per MB	Max 20 NOK per day

Table 2.2: Prices for mobile data prescriptions from Telenor

Prices - NetCom

Prescriptions	Description	Cost in Norwegian kroner
Fri Bruk	Free usage	499 NOK per month
Kveld & Helg	Free usage - 17.00-07.00 + weekends	99 NOK per month
Litt Bruk	Pay for the time of usage (max 50 NOK per day)	60 NOK per month + usage
Betal for bruk	12.50 NOK per MB	Max 75 NOK per day

Table 2.3: Prices for mobile data prescriptions from NetCom

The Universal Law of Web Surfing says that the user patterns are regular and states the preference of short surfing session over longer ones. A study

[13] has addressed the issue and compared the pattern of mobile web surfing to regular web surfing. They have confirmed the generality of this law and concluded that the pattern of mobile surfing is closely related to the pattern of regular web surfing.

2.3 Wap browsers

WAP is a technology intended for mobile phones and PDA's. It's main goal is to provide Internet access to small devices through wireless communication. The main drawback with WAP is that the WAP browser do not understand regular web pages created in HTML. The WAP pages have to be created separately from the web pages in a WML(Wireless Markup Language) or you need a converter that can convert HTML to VML format. A limited number of companies have put the effort in creating WAP pages and this limits the amount of available information for the WAP user [14].

2.4 Opera Mini^{TM}

Opera Mini is a free web browser for mobile phones, smartphones and PDA's. It was developed for mobile phones that were incapable of running regular web browsers. The Opera Mini client consists of a small Java MIDlet that can be downloaded free of charge to your mobile phone. It requires that your mobile phone support Java ME applications, which most new phones do. Opera Mini will give you access to the hole world wide web, and not just the specialized WAP pages. The benefit of Opera Mini compared to a regular mobile browser is that all web pages are being pre-processed and compressed in dedicated Opera servers before the information is sent to the mobile phones. The aim of this is to make the information more suitable for the small hand held devices as well as to reduce the size of the information needed to be processed by the low capacity client.

Opera claims that this technology reduces the transferred information and increases the traffic rate by two to three times. This means better response time and lower cost for the end user, compared to normal web browsing [3]. Some would say that this could lead to a reduction in the earnings of mobile providers, but based on numbers from T-mobile [15] the introduction of their service 'web'n'walk' has increased the usage and hence the total data transfer by far. A study on energy loss in wireless transmissions [16] pinpoint another benefit for the mobile operators. They claim that it can be beneficial to perform additional computation to reduce the number of bits transferred due to the fact that wireless transmission of a single bit can require more than 1000 times energy than a single 32-bit computation.

Opera Mini was released August 10'th 2005 in Norway and globally January 24'th 2006 [17]. Seven months later one billion page requests had been handled by the Opera Mini servers. Opera claim that they have more than 27

million users around the world and that the amount of traffic generated by these users is approximated to be 10-20 per cent of all international traffic to and from Norway [18]. Most users are said to be located in Russia, followed by the US, India, Scandinavia and Great Britain. The five most visited pages are claimed to be Google.com, Gmail.com, Friendster.com, Myspace.com and hi5.com [19].

Today all servers are located at Opera's main office in Oslo, Norway. The structure of the cluster is shown in figure 2.1. The procedure when a Opera Mini user browse from the mobile phone is well described on a developer site at Sony Ericsson. When the Opera Mini user requests a web page the Opera Mini client installed on the mobile phone has a pre configured IP address that it uses to contact the the transcoder server at Opera. The requests first arrive at a LVS (Linux Virtual Server) which is a dumb load balancer. The LVS just forwards the requests using a Round Robin algorithm to smart load balancers (LB). These load balancers can either act as a server or handle the requests themselves or forward the requests to other servers based on information in the request itself, or information provided from the servers. The transcoder that handles the request fetches the requested web page and all related elements like images and style sheets. When everything is downloaded it transforms the page according to the Opera Small Screen Rendering (SSR) algorithm. This is to make the content suitable for the small display and to avoid horizontal browsing. Then it transforms the resulting layout into OBML (Opera Binary Markup Language). This is a compact binary format that is used to transfer the data from the transcoder to the end user. Next it partitions the file with regards to the specifications of the receiving device in order to not send more information than the mobile phone can handle. In general newer phones have more RAM available then older phones and can for that reason handle larger files. The last step is to fulfill the compression. This includes compress the data, re-size and re-encode all images according to display specifications and user configurations in the client. The client can choose between four image display options. No images, low quality images, medium quality images or high quality images. The result is transferred to the Opera Mini client that displays the information on the mobile phone [20].

The fact that all traffic is handled by dedicated servers demands for extensive link and server capacity. This is a great cost for the company, but it makes it easy to maintain. It also gives Opera the opportunity to easily monitor and analyze the use of their service. They have access to all information which can be used for capacity planning, marketing and other means. Today Opera has gigabit connections and approximately 100 servers dedicated to Opera Mini traffic. The traffic load will in the nearest future over seed these limits and Opera is planning to spread the load by placing servers at other locations.

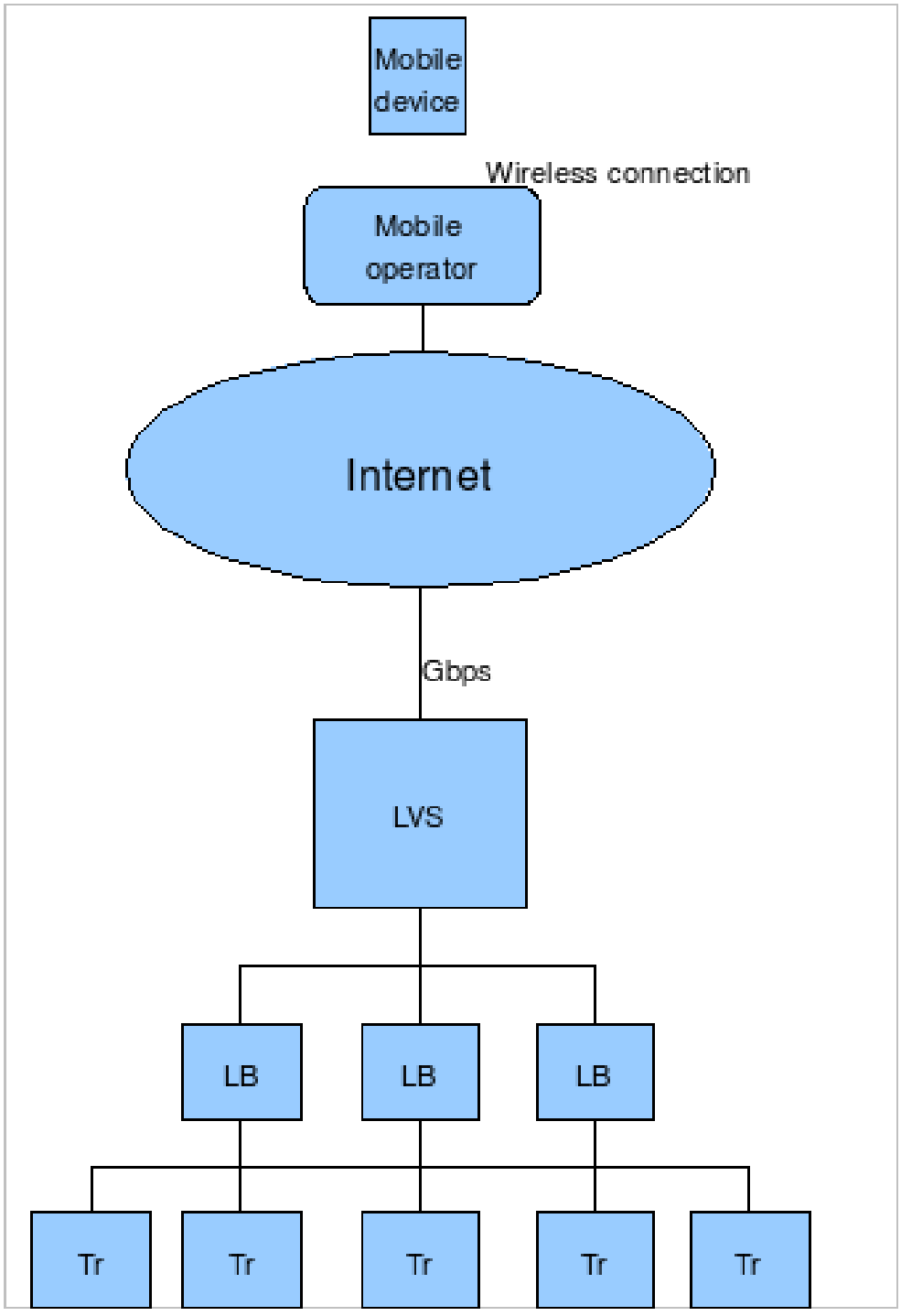

Figure 2.1: Structure of the Opera Mini cluster.

2.5 Internet censorship and privacy

The Internet contains information of all you can possibly think of, both good and bad. Internet censorship put limitations to what kind of information that can be published on the Web. In most parts of the world governments put restrictions on some kind of information that is forbidden to publish on web pages hosted in their country. Where to put the boundaries will differ due to the laws and regulations of the country. In Norway we believe that we are free to do everything we want, but there are actually some limitations forced by the government here as well. The major Internet service providers have DNS filters that blocks access to sites authorities claim are known to provide child pornography. Most people would say that this is a good thing, but who should actually decide which pages are better than other? When you filter out illegal information, you will probably filter out legal information as well. In 2006 the organization 'Reporters without borders' published a list with thirteen nations calling them 'Enemies of the Internet'. The nations on this list are known to abuse the censorship and use it to suppress their own citizens. They often censor material with political content, but some may also block e.g video-uploads from sites such as YouTube [21].

The Opera Mini technology may contribute to give people in these countries access to information that is blocked by the governments through a regular Internet connection. Since all traffic has to pass through the Opera Mini transcoders, the traffic seems to be requests to Opera instead of requests to the blocked sites and the traffic will not be blocked. There are probably better solutions for people working regularly with illegal content, like Freenet [22], but it can be a safe alternative to get access to information that is actually blocked.

The Opera Mini technology gives great benefits to the user by reducing the amount of data transferred to the mobile phone, but the fact that all traffic pass through dedicated Opera Mini servers give Opera access to all data and huge possibilities when it comes to monitoring and analyzing user information. The requested web pages get modified at Opera before they are transferred to the end user. The possibility to perform these modifications opens for a possibility that can be abused. There is an example from 2004 where technology with similar possibilities has been abused. Jason Calcanis was using a mobile html transcoder, called Skweezer. Skweezer modify the data before it is transferred to the end user, but the thing that started the debate was that blogger Calcanis was object to advertisements being placed on transcoded versions of blog content. This lead to a discussion on copyrights and about the legality of placing ads on other publishers content from proxy based services, like a transcoder. The subject is still under debate, but the owner of Skweezer, Greenlight Wireless, stopped placing ads on transcoded web pages in early 2005 [23].

In Norway people's privacy is protected by the Personal Data Act.

'The purpose of this Act is to protect natural persons from violation of their right to privacy through the processing of personal data. The Act shall help to ensure that

personal data are processed in accordance with fundamental respect for the right to privacy, including the need to protect personal integrity and private life and ensure that personal data are of adequate quality.' (Quote: Personal Data Act, 2000, section 1).

The Data Inspectorate is responsible for ensuring that the Personal Data Act is complied to. If companies want to process personal data, they have to apply to The Data Inspectorate. The Inspectorate keeps a public record of those who have got a license and those that have been refused. The Inspectorate should advice companies that are planning to process personal data as well as clearly state their opinion on cases of that concern. If someone is dissatisfied with a decision made by the Inspectorate, they can complain to the Appeals board. The Appeals Board can overrule the Inspectorate and make the final decision [24].

Chapter 3

Methodology

'The principal aim of science is to uncover the most likely explanation for observable phenomena.' (Quote: Mark Burgess, Analytical Network and System administration)

Science is about managing uncertainties. It does not necessarily give us the correct answer of a phenomena, but can help us to evaluate the likelihood that a given explanation is true. Uncertainties are caused by various errors and according to Mark Burgess there are three types of errors. Random error are usually small deviation from mean that occur by accident. Random errors can usually be represented by a 'normal distribution', with evenly numbers of errors around the mean value of the observation. Personal error is an error that is introduced by the use of specific software or through the interpretation of the data. Systematic error is a shift in the true value that runs throughout all the data. The systematic errors are often the most difficult to locate [25].

Measuring mobile user behavior can be performed in a various of methods. In a study performed at the Helsinki University of technology [26] they have compared five different methods for collecting information on mobile user behavior. The problem they try to address is to figure out where it will be most convenient to perform the measurements. Because all traffic generated by Opera Mini is handled by dedicated servers located at Opera it seems quite obvious that these servers would be the best place to perform the measurements in our case, but if you want to have control over all parts of the process you must use some other methods. In this chapter we will describe four different methods, discuss these and explain why we chose the one we did.

3.1 Off-line data mining of existing data

Off-line data mining of existing data is widely used for data analysis. The method take advantage of the benefit that the data is already collected. This gives great possibilities to save time and focus more on the analysis of the data at an earlier point of the project. The fact that all data are physically stored, makes it possible to do the analysis several times. This is beneficial if you want

to do modifications to your tests, it is easy to just run them ones more. It is also interesting that the data easily can be used by others, either to verify previous work or for new projects with different angles. A drawback with using off-line data is that the data is getting old and that might not be representative for the situation today. If the data is collected by a third party, you loose the control of the collection process. If this process not is documented it will be hard to verify if it has been done according to your demands and it can create uncertainty about the validity of your results.

Collecting information in log files is flexible and you can define which parameters you will include according to your needs. In software development testing it can be used to evaluate the results. It is described how it is possible to automate the error checking of the log files from the tests to validate if the test ran as expected [27]. In this paper they have described how you can use automatically generated log files in a log file analyzer that will check the file for error occurred during the test. System behavior, resource utilization and test performance are some of the main purposes. Log files analysis can give you great insight and help you to improve your system or tasks. The cost of performing the analysis can be saved from an efficiency increase due to your greater knowledge. There are various papers and books that cover traffic measurements and log file analysis. The tools and methods evolve as the complexity of the systems are increasing.

When you perform data mining it is of great importance that you know what kind of information you have available and the value the different parameters have for your analysis. In a survey addressing the interestingness of different measures [28] they classify the parameters from several perspectives, compare their properties, identify and give strategies for selecting appropriate measures. It would be useful to have a general framework for defining a subjective and semantics-based measures representing knowledge that is related to data mining. They conclude that choosing interestingness measures that reflect real human interest remains an open issue. Since user interactions are indispensable in the determination of rule interestingness, methods and tools need to facilitate the users involvement.

Another study [29]that focus on the huge amounts of data generated by measurements address the need of knowledge discovery in databases (KDD). The amount of data being collected in databases exceeds our ability to reduce and analyze the information without the use of automated analysis techniques. There is a distinction between data mining and KDD. Data mining is the process of extracting trends or patterns from the data while KDD takes the raw results from the data mining and transform them into useful and understandable information. Knowledge discovery is defined as the non-trivial extraction of implicit, unknown, and potentially useful information from data. One of the major premises of KDD is that the knowledge is discovered using intelligent learning techniques that goes through the data in an automated process.

3.2 Traffic measurements.

Another widely used method is to capturing traffic using network monitoring tools. There are several ways to measure traffic, active vs passive, off-line vs online and hardware vs software. By active measurements you add packets with known characteristics into the network and use these for your analysis. This is suitable for measuring infrastructure since you have full control over the measured traffic. This could be one alternative approach for measuring the delay between the mobile phone and the Opera Mini server. By passive measurements you only measure the real traffic. This is suitable if you want to find characteristics of the traffic. You do not apply any additional load to the network so you can not control the traffic that you measure. On-line measurements require that some of the data analysis is performed in real time. This is often necessary when you measure large amount of data and you don't need to store it all. These solutions can often be complex and costly, but for critical systems they can give you results that you can react right away. For off-line measurements you capture the traffic and store it for later analysis. There are several cheap and simple solutions to do this and it give you the possibility to run complex analysis on the data, or even run it several times if that is desirable. This is not suitable for time critical systems and if If you monitor networks with high traffic load it will demand huge amounts of storage capacity [30].

A study on performance monitoring [31] address the dilemma that when you collect huge amounts of data to find performance bottlenecks, the process of collecting the data might introduce serious data collection bottlenecks. They also focus on the difficulty of interpreting and presenting these huge amounts of data. The study present a new approach called W^3 Search Model that addresses both these problems. It combines dynamic on-the-fly selection of what data to collect, with decision support to assist users with the selection and presentation of the data.

3.3 Controlled experiment

Another approach to the project would be to do a controlled experiment. Controlled experiments are performed in various projects related to data analysis. It gives you the advantage of controlling greater parts of the information. One example could be to generate web pages with known content and fixed sizes. You could then use the Opera Mini browser to download these pages a numerous of times. When the logs where generated you could filter on the known web pages and only analyze this data. This could give you better insight in how different content affects the compression of the web pages in the Opera Mini servers.

This could create information of great importance for Opera Mini developers as well as developers of regular web pages. It could be an exiting project to perform as an follow up to the work in this project.

3.4 Testbed

The last method we will discuss is a testbed with all necessary equipment to simulate the Opera Mini traffic. This approach gives you the possibility to control all parts of the system and an opportunity to modify what kind of information you will store. You can create your own log schemes with the explicit form that suits for your analysis. This approach is the most complex one. There are several issues to take into account before it is possible to work further with this. For our project the time restriction is the first argument. To build the testbed would require a decent amount of time and resources. It would be possible to reduce the cost by using virtualization, but on the other hand this could lead to even greater complexity. In addition one would be heavily dependent on assistance from Opera Mini experts due to Opera specific configurations and copyright issues. There are strict restrictions on what kind of information related to the Opera Mini technology you will get access to. The main development of Opera Mini is performed in Sweden and this could lead to harder communication if you want to perform the project from another location, as in our case, from Opera's head office in Norway.

3.5 Summary of methods

All four methods we have discussed have pros and cons compared to the others. We got access to a span (Switched port analyzer) port covering all Opera Mini traffic. We ran a few tcpdumps to get the overview of what kind of information we could obtain and what amount of data the traffic generated. We limited the dump files to only include 60 bytes from each packet to reduce the size of the captured data. The benefit of this compared to using the log files would be that we where fully in charge of the information we wanted to store and that we could use data that where collected during the period of the project. This method has been used in the thesis of a former student at this education program, Håvard Wik Thorkildssen, that performed a study where ha compared a university network and an ISP network. He analyzed collected measurements looking for different characteristics of the two networks [32].

We decided that we had all the parameters we needed in the log files and came to the conclusion that we wanted to use off-line data mining of existing data as our method for the project. This would save us the time of the collection process and leave us with greater time to analyze the data. This decisions was agreed upon by Valeri Cheremetiev and me as we collaborated in the first phase of the project. There are no direct methods for analyzing the Opera Mini log files today. This makes our approach exploratory in the sense that we do not know what kind of results we will get. We will startup by getting the overview of the information. We will extract the relevant information from the log files and store it in a database. We will store the data in a database to make the access more efficiently. We will represent the results graphically to easily show the first results and then discuss and analyze the meaning of

the findings. We will investigate the relationships between the variables and hopefully find answers to our questions from the problem statements.

There are some benefits with the two latter methods, but non of them would be very interesting without having performed the first method. To make an web application that provides a good service in a real life scenario you need to now about user patterns, user configurations and other characteristics of your system. We believe it would be to start at the wrong end to perform the two latter methods without having any knowledge about the real life traffic information. In our case we believe it would have been easier to influence the Opera experts to include a new parameter in the existing log setup, than providing the resources to set up a entire testbed. When the information regarding the use of the Opera Mini service is available we would encourage that there are performed some controlled experiments. One example could be to look into how large share of the total download time is outside the control of Opera, meaning from the Opera cluster to the mobile phone, over the wireless connection.

Chapter 4

Equipment and tools

This thesis did not require any specific capacities from the equipment and could for this reason be repeated by any others. The main limitation will be to get access to the same log files we have used for the analysis.

4.1 Hardware

All work with this thesis has been performed on the laptop of the author or at a workstation provided by Opera Software ASA. The specifications of these computers can be found in table 4.1.

	Equipment	
Specifications	Laptop HP Pavillion 2017ea	Workstation Dell
Processor	Intel Core Duo T2050, 1.6GHz	Pentium 4, 3GHz
Ram	2 GB	1GB
Harddisk	100GB SATA	500GB
OS	Ubuntu 7.10 Gutsy Gibbon	Debian 4.0 Etch

Table 4.1: Specifications of the equipment.

We have used various software for the different processes of the thesis. All software we have used for the work of this thesis have been free of charge.

4.2 Python

”Python is a dynamic object-oriented programming language. It offers strong support for integration with other languages and tools and comes with extensive standard libraries.” Python is an open source product that runs on most OS’s [33]. It is influenced by the better known language, Perl, but is said to be easier both to learn and to read.

We used python to collect the information from the log files and store the information in a database.

If you want to learn more about python you can visit:
http://www.python.org/

4.3 Sqlite

SQLite is a C library that implements an SQL database engine. Programs that link with the SQLite library can have SQL database access without running a separate RDBMS process [4]. We mainly used command line interface when we quired the database. This was easily done with sqlite3. When we quired the database we could only run the command from the command line like the example below,

```
sqlite3 database.db  'select csize,usize from rq;'  >  file.dat
```

Here we quire the database called 'database.db' for the parameters 'csize' and 'usize' in table 'rq'. We write the result to the output file 'file.dat'.

You can find more information about sqlite on:
http://www.sqlite.org/

4.4 Gnuplot

We used Gnuplot to present our results graphically. Gnuplot is a command line plotting utility compatible with a various number of platforms. It supports plots in 2D and 3D and a you can set the style you want to use to represent your plot [5]. The ones we have used for our plots are linespoints, boxes and dots. We could either type the command in directly in Gnuplot, like:

```
gnuplot> plot 'dist_img.gp' using 1:2 with boxes fs pattern 1 /
         t 'Uncompressed data'
```

Here we plot the numbers in column one and two from file 'dist_img.gp'. We use boxes with a pre defined pattern and add the text 'Uncompressed data' to describe what kind of data we plot.

In most cases we would like to define some additional configurations for each plot, like labels and range. When you plot several plots with some modifications it is preferable to write the commands in a file and run this file with Gnuplot. You can do that from command line with the command: 'gnuplot filename'.

You can find Gnuplot documentation and examples at:
http://www.gnuplot.info/

Chapter 5

Results

In this chapter we will present the results achieved. All results are generated from log files of 54 Opera Mini servers collected over a fourteen days period. The results will be discussed thorough in next chapter.

5.1 Compression

We can use the cumulative distribution function, also called the probability function, to find the probability that the size of a file is lower or equal to a given value.

The compression achieved in the Opera Mini transcoder when you download a web page is a key criteria for the success of the Opera Mini browser. The Opera Mini transcoder reduce the information transferred to the mobile phone. This means better response time and lower cost for the end user. The user cannot affect the compression, but he can configure the image display settings in the browser and in this way affect the amount of data that is transferred to his mobile phone.

The notation of the image display settings differ between the log files and the Opera Mini browser (See table 5.1). In the Opera Mini web browser you can choose between four image display quality options. These are, no images, low-,medium- and high-quality images. In the log files the image quality settings are stored with numerical values from zero to six. One to four equals to the options you get in the browser, while zero, five and six is used for test and development purposes. There were a few request with these test values, but with regards to the minimal share and that this data is not generated by a regular user, we decided to not take this data into account for our analysis.

In table 5.1 we see some key numbers regarding compression collected from the log files. The first column shows the four different image display options that it is possible to choose from in the Opera Mini browser. The second column represent the average file size of the requested web pages downloaded to the transcoder. The third column is the size of the same files after they have been processed and compressed in the transcoder, ready to be transferred to

Image quality

Image display quality in log files	Image display quality in Opera Mini browser	Description
0		No images
1	No images, ALT text	No images, ALT text
2	Low	Low image quality
3	Medium	Medium image quality
4	High	High image quality
5		Insane image quality
6		Insane image quality

Table 5.1: Image display quality settings.

the mobile phones. The fourth column gives us the total number of requests for the different image qualities and the fifth present this number in per cent of the total number of requests. The last column tells us the average compression level in per cent achieved for the different image qualities.

We see from table 5.2 that almost 60 per cent of the requests are performed with the low image quality setting. This is the most common choice as we expected. We believe there are two major reasons for this large majority of image quality 2.

- it is the default value for the browser when downloaded on most phones.
- people want to see the images, but the download time is more important than the quality of the image.

We notice that more than 20 per cent of the requests are performed with the high image quality setting. These users must have actively configured the setting to this level. It is hard for us to tell why as many as 20 per cent need to view images in high quality. One reason could be that they once downloaded a image they wanted to display in high quality and that they forgot to change the display option back. Another reason could be that a great deal of the images downloaded are pornographic, and that the users downloading this kind of material prefer to view the images with high quality. If we reduce the image quality from high to low, it leads to almost 50 per cent reduction in the data transferred for these requests.

The numbers from table 5.2 is presented in figure 5.1. The figure shows both the uncompressed and the compressed values to easily see the effect of the compression. Each image quality is represented by the same color, but with different patterns for uncompressed and compressed values. The average file sizes are remarkably equal for all transcoders. The case with the most stable values is the one with the greatest number of requests, low image quality. We see from the graph that the size of the compression is almost the same for each image quality. In general the compression leads to an average reduction in the file size of 13000 bytes. The compression of the files with 'no images' gains great compression for the text. The reduction of the files with images is

Average file compression

Image quality	Uncomp. file size	Comp. file size	Number of req.	No. of req. in per cent of total	Relative comp.
1	19391.96	6335.23	38693785	10.61%	67.33%
2	30577.79	17229.93	215344980	59.07%	43.65%
3	36453.42	23684.43	32568495	8.93%	35.03%
4	45410.38	32928.48	77928644	21.38%	27.49%

Table 5.2: Average file sizes before and after compression, number of requests, number of requests in per cent of total and relative compression.

achieved from reducing the images in addition to the compression of the text. That the levels between the different qualities are the same can be due to limit settings of the quality areas decided by the Opera Mini developers. If we look at the achieved level of compression compared to the average file sizes, we see that the small files experience a much better compression if you compare with their original file sizes. This tells us that the lower image quality display setting we use, the better relative compression we achieve.

We see that the file size increase with the image quality. The size of the files with 'no images' are small, since they do not download any images to the transcoder. We expected that the file sizes where decreasing with lower image quality after compression since these files get more compressed. We see that this is not entirely true. As we stated earlier, the compression is almost the same in bytes, while the relative compression is better the lower the image quality. It is hard to explain why the file size of the files downloaded to the transcoder with images is larger as the image quality increase. We expected quite equal file sizes for the uncompressed files that included images. One explanation could be that the users with high image quality settings in their browser have chosen this configuration because they visit web pages with images of high quality. It would have been interesting to have the total number of images downloaded for each request as a parameter in the log files. This could have helped us to figure to what degree the number of pictures affects the the file size and processing time of a web page.

5.2 Total bytes

The total amount of data generated by Opera Mini browsers is a key number to have in mind in the struggle to maintain an optimized service. This number tells you how much data your servers must be able to handle and the capacity you need for your Internet connection. This will help you to verify your needs and your costs. These numbers can also be used in marketing and other commercial agreements, like pre-installation agreements with Mobile Operators.

Figure 5.2 presents the total amount of data handled by each of the 54 transcoders per day. The upper part of the graph shows the uncompressed data while the lower part of the graph present the amount of data transferred

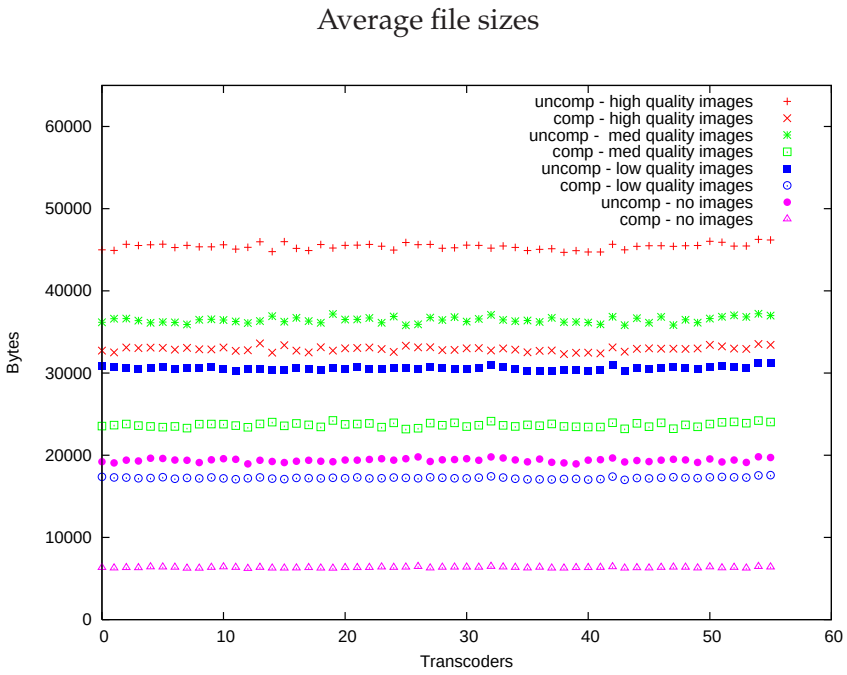

Figure 5.1: Average size of web pages before and after compression per transcoder for all image qualities.

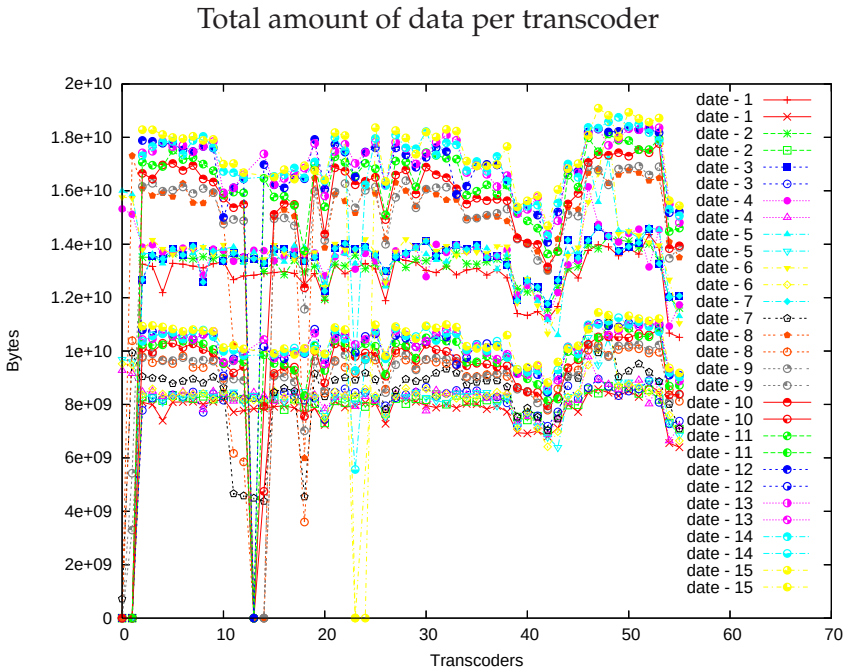

Figure 5.2: Total amount of data uncompressed and compressed per transcoder per day.

to the mobile devices after compression. We see that there were some instability in the behavior of the transcoders. Transcoder one, two, thirteen and fourteen was down most of the period, while some other were down only for shorter periods. We see that there is a dip from transcoder 38 to transcoder 44. These are handling less traffic than the other. This can be due to the fact that these transcoders also act as load balancers as well as transcoders. Another possibility is that they are newly rebooted or installed in the cluster. A device that request for a web page will be handled by the same transcoder for all requests until the mobile phone or transcoder is rebooted. With this load balancing feature it will take some time for a new transcoder to get as much traffic as an old transcoder that has built up a list of clients. We see that the traffic rate is increasing every day for all transcoders. The increase is much greater than we expected. To present this as simple as possible, we have used the total values for all transcoders per day in figure 5.3.

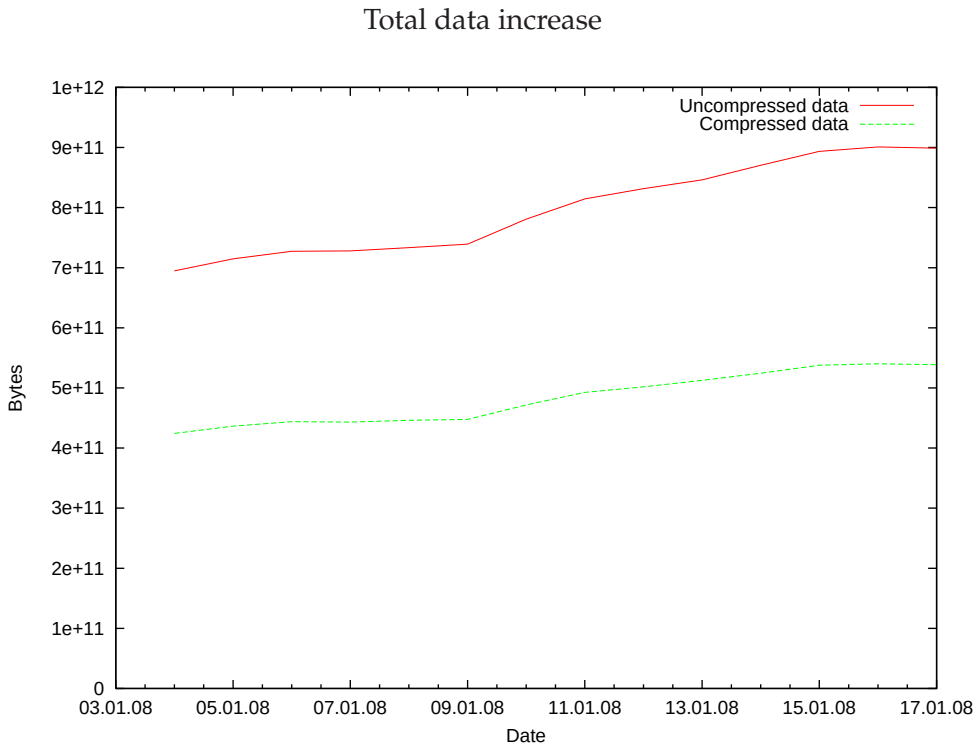

Figure 5.3: Total amount of data uncompressed and compressed for all transcoders per day.

Figure 5.3 display the increase of the total amount of transferred data, uncompressed and compressed, for all the 54 transcoders throughout the test period. We notice that there is an increase during the two weeks of approximately 186GB from 651GB to 838GB. This is an increase of more than 20 per cent for this short time period of two weeks. It looks like the graph is decreasing a bit the last day and it would be interesting to see the development of the traffic for a longer time period. It is hard to tell exactly why we see this

major increase during the test period. Our test period started 04.01.2008. This is close to New Year and a great deal of people where having holiday until Sunday 06.01.2008. One explanation could be that people are using the service more regularly in weekdays than in holidays. Another possibility is that there where some major events that affected the user behavior.

5.3 Distribution

It is important to know the distribution of the traffic with regards to different parameters. This gives you the opportunity to analyze your system which can help you to support your configurations based on real numbers.

5.3.1 Distributions with regards to image quality

There are four possible image quality display options in the Opera Mini browser. In figure 5.4 we present four different distributions that all is related to the image quality settings. We present these together to see the relationship between the figures. From first glance we see that the shapes of the figures have great similarities and that it look like the parameters are depending on each other.

The plot in the upper left shows the distribution of uncompressed data for all for image qualities. The plot in the upper right shows the distribution of the same files after compression We have used the same scale for these to plots to see the relationships between them. We see that the majority of the bytes transferred are with image quality setting two, meaning low quality images. This is as we expected since low image quality is the default setting for the Opera Mini browser. We see that the amount of data is reduced for all image qualities after compression, but the compression had different effect on the different qualities. We see that image quality one is reduced from 0.75 terabytes to 0.24 terabytes which is a reduction of two third of the original size during the compression. This is the setting with 'no images' and we clearly see that the compression works well for these files with out images. The next column which consists of data with low image quality is reduced from approximately 6.5 terabytes to 3.7 terabytes giving a reduction of more than forty per cent. The third column represent medium image quality. If we look at the figure in the left corner we see that this is the image quality with least requests. Anyway we see that the amount of data transferred for this quality has a higher value than image quality one, especially for the compressed files. The compression reduce the amount of data from approximately 1.2 terabytes to 0.8 terabytes, leaving us with a compression of one third. The last column is for the high image quality setting. We see that this is the second most popular setting when it comes to number of requests. When we look the difference between the uncompressed and the compressed values we see that this is the hardest one to compress. The data is reduced from 3.5 to 2.5 terabytes. This is a reduction of less than thirty per cent, a result about five per cent below the result of image quality three.

If we look at the figure in the lower right corner, we see the distribution of times it takes to handle the requested web pages by the transcoders. These are the total times that include download of the web page and compression of the data to the format suitable for the mobile phone of the end user. If we compare the figure of times with the figure of requests we see that these are almost the same. Since we operate with different units we can overlook the different scaling. The fact that these two figures have the same shape tells us that the time it takes to handle a request is the same four all image qualities. We can see from the plot in the upper left corner that the amount of data handled for image quality one is less the amount of data handled for image quality three. Anyway we see from the 'time' plot that the requests with image quality one use more time than the requests with image quality three. This tells us that the time it takes to handle a request to a great extent depends on the number of requests and not on the type of image quality or size of file.

The approach we have used to analyze the Opera Mini log files could easily have been implemented in other cases of data mining. The setup could have been used in a real life working environment, but it might be preferable to run the analysis once per day to keep decent time aspects for the tests. To query the database that covered data from the entire two weeks was a time consuming task. One example of this is that the queries we ran to get the information we used in section 5.3.2,with the distributions regards to file sizes, took approximately two days. The long runtime demands for good planning to make sure that the queries are correct from the beginning. The runtime could have been reduced if we had equipment with better processing capacities. Due to the fact that we were not sure what kind of analysis we were supposed to use and how many times we needed to request the different parameters, we decided to store the information in database. This makes it easier to keep the overview of all parameters and the access to the data more efficiently. The collaboration with Valeri Chermetiev in the preface of the project has given me great experiences and the discussions we have had has given me useful reflections on how it is to perform these kind of projects in a working environment.

5.3.2 Distribution of requests with regards to their size

To find the distribution of requests with regards to their file size we divided into areas of 5000 bytes up to 595000 bytes. We have only included the values up to 140000 bytes in the plot because the numbers where so small for file sizes larger than this. We present both the uncompressed and compressed files to see the relationship between these.

Image quality 1

Figure 5.5 shows the distribution of requests with image quality based on their file sizes. We see that there is most requests are below 5000 bytes. We notice that the number of compressed files in this area is more than 22 millions, approximately twice the number of uncompressed files. The difference we found

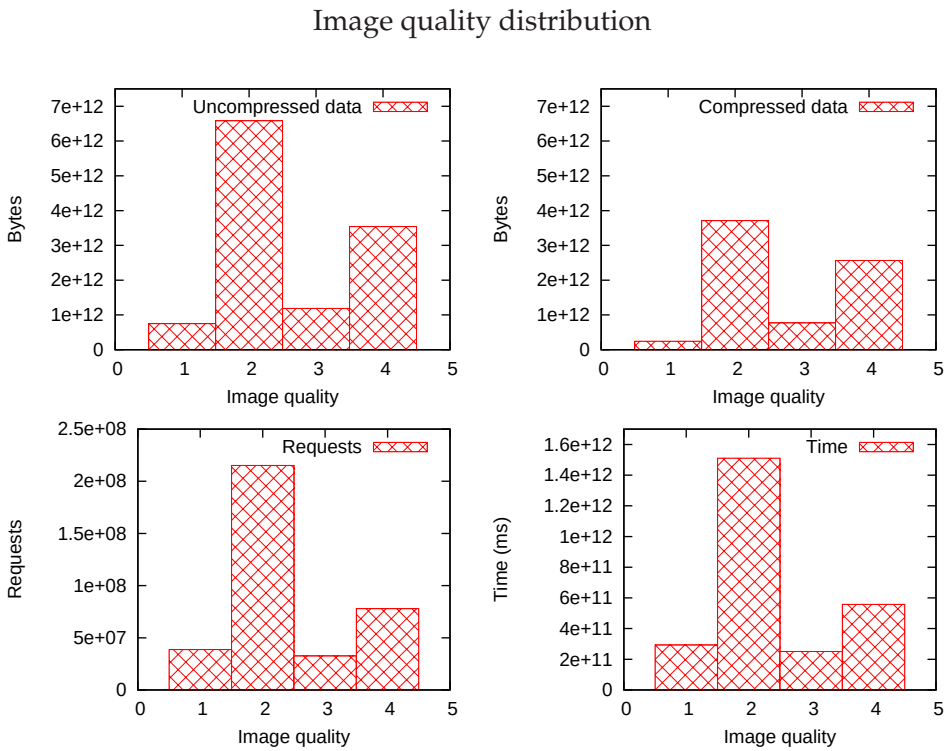

Figure 5.4: Distributions with regards to image quality.

in the first area of 11 million files, are spread over the scale and the uncompressed files have the greatest numbers for all areas except for the two first.

Image quality 2

We see the same shape for image quality two (figure 5.6) as we saw in the previous figure. We see that the numbers are much higher since this is the most common image quality with low quality images. The compressed files are in majority in the area with the smallest files. Then we see that the uncompressed have a few more for the next two areas, before the compressed score greatest in the three following. From here the difference from the first area is divided throughout the scale and the uncompressed files have higher values for the rest of the areas.

Image quality 3

From figure 5.7 we see the same shape as in the two latter figures. The number of compressed files are in majority in the area with the smallest files. We see that this is the only area with more compressed than uncompressed files. The huge difference from the first area is divided on the rest of the areas.

Figure 5.5: Distribution of requests with no images with regards to their size.

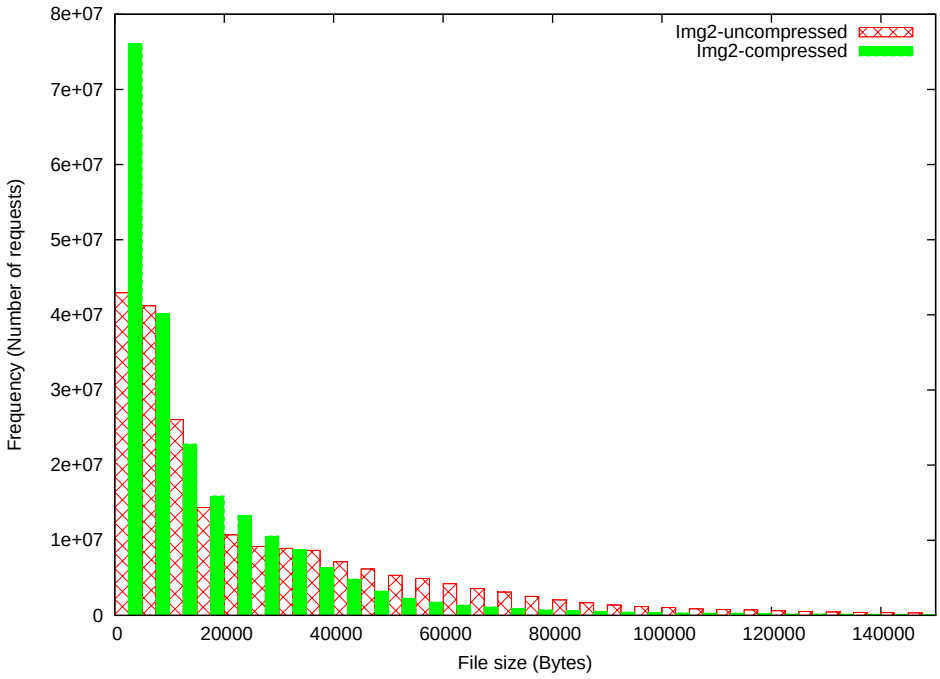

Figure 5.6: Distribution of low quality image requests with regards to their size.

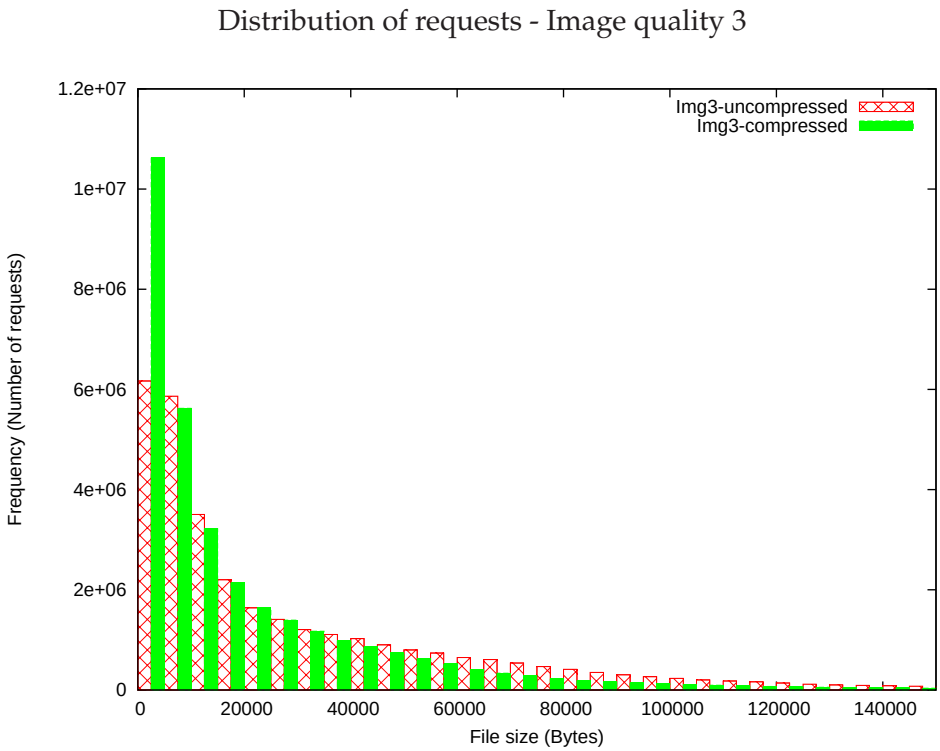

Figure 5.7: Distribution of medium image quality requests with regards to their size.

Image quality 4

Figure 5.8) has also the same shape as the previous. Also here the compressed files are in majority in the area with the smallest files. We see that this is also the case for four other areas between 20000 and 40000 bytes. The difference is then divided on the rest of the areas.

Sum of all image qualities

If we look at figure 5.9 we see that the pattern of the graphs are having great similarities, but that image quality two have a much higher rate of requests than the other. We see that all the image qualities have a high number of small files. We were expecting greater differences for these numbers. We assumed that image quality one would have a different pattern than image quality four. We were expecting a steeper curve for image quality one since we expected a high number of small files. The result was not far from what we expected, but for image quality four we did not expected such a steep fall. Here we expected to see a great number of large files due to the high image quality setting. The fact that there is such an overwhelming majority of compressed files in the areas of the smallest files is a good sign. This tells us that the compression works because many of these files have been swapped to a lower file size level after compression.

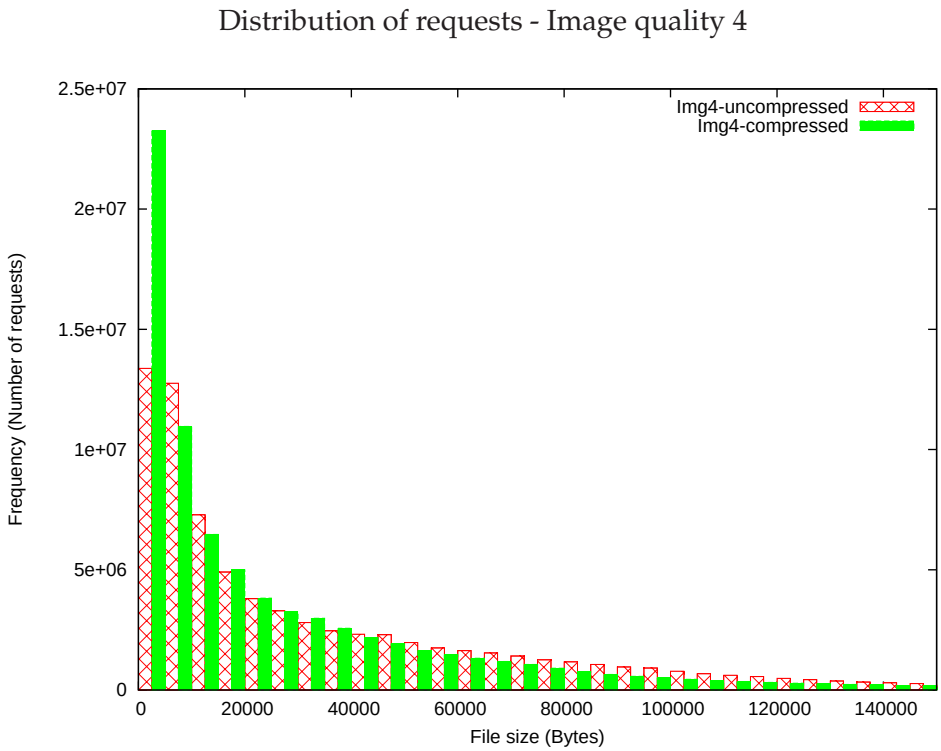

Figure 5.8: Distribution of high image quality requests with regards to their size.

5.4 Cumulative distribution

Cumulative distribution function, also called the probability function, describe the probability that a variable will have a value lower or equal to a given value [34]. We have used the cumulative distribution function to see how the variables distribute in per cent. The approach we have used to analyze the Opera Mini log files could easily have been implemented in other cases of data mining. The setup could have been used in a real life working environment, but it might be preferable to run the analysis once per day to keep decent time aspects for the tests. To query the database that covered data from the entire two weeks was a time consuming task. One example of this is that the queries we ran to get the information we used in section 5.3.2,with the distributions regards to file sizes, took approximately two days. The long runtime demands for good planning to make sure that the queries are correct from the beginning. The runtime could have been reduced if we had equipment with better processing capacities. Due to the fact that we were not sure what kind of analysis we were supposed to use and how many times we needed to request the different parameters, we decided to store the information in database. This makes it easier to keep the overview of all parameters and the access to the data more efficiently. The collaboration with Valeri Chermetiev in the preface of the project has given me great experiences and the discussions we have had has given me useful reflections on how it is to perform these kind of projects

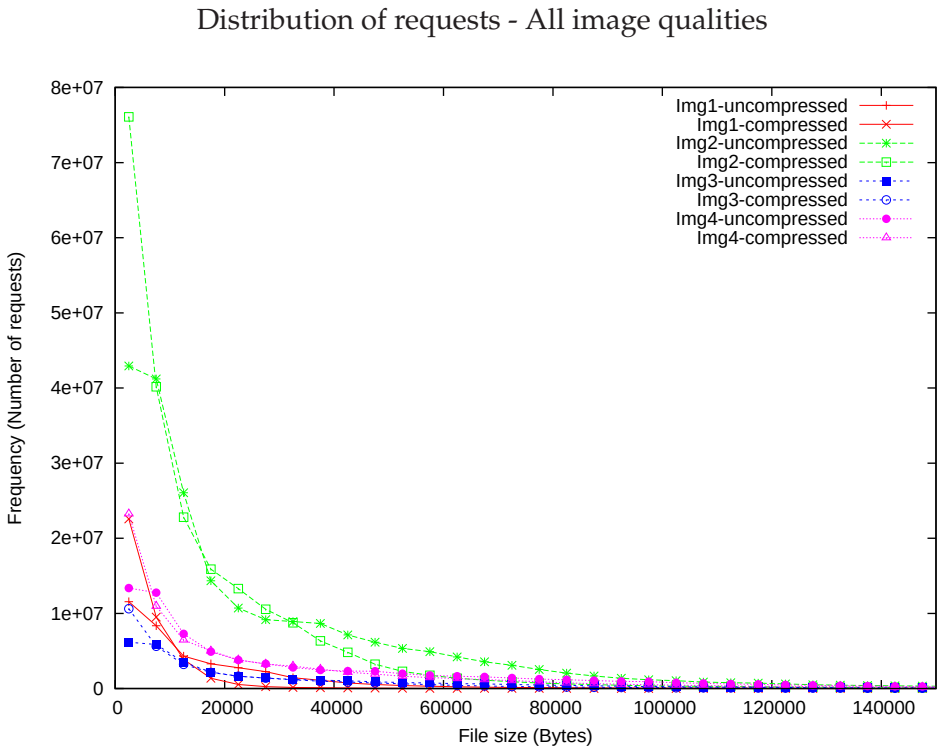

Figure 5.9: Distribution of requests, uncompressed and compressed, for all image qualities with regards to their size.

in a working environment.

5.4.1 Cumulative distribution of file sizes

To calculate the probability that a request will be within a given value, we have used the cumulative distribution function. We have plotted the distributions of both the uncompressed and compressed file sizes in figure 5.10. We see from the figure that the curve is experiencing a steep growth and that the majority of the files are small for all image qualities. For image quality one, more than 90 per cent of the files are smaller than 40000 bytes, while for image quality four we need 110000 bytes to cover the area with more than 90 per cent of the files. For the compressed files we see that, for image quality one, more than 90 per cent of the files are smaller than 15000 bytes, while we have to increase the area to 85000 bytes to cover 90 per cent of the files with image quality four. We notice that to cover 90 per cent of the requests the difference between image quality one and four has the same value, 70000 bytes, both for the uncompressed files and the compressed files. This supports the findings from section 5.1, that the compression is the same for all image qualities in case of reduction in number of bytes.

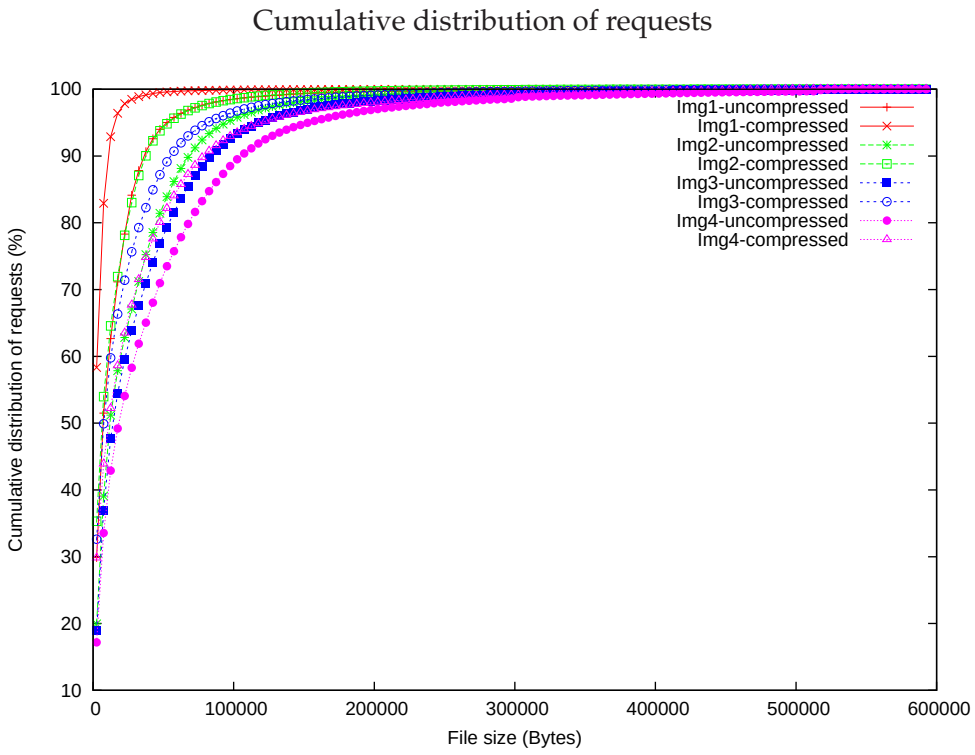

Figure 5.10: Cumulative percentage distribution of requests, uncompressed and compressed, for all image qualities with regards to their file sizes.

5.4.2 Cumulative time distribution

From figure 5.11 we see that the cumulative time distribution for all image qualities are identical. This supports our findings from section 5.3.1, that the image quality settings have no impact on the time it takes to download and process a web page in the transcoder. We see a steep increasing curve that tells us that that majority of the requests are handled within short time intervals.

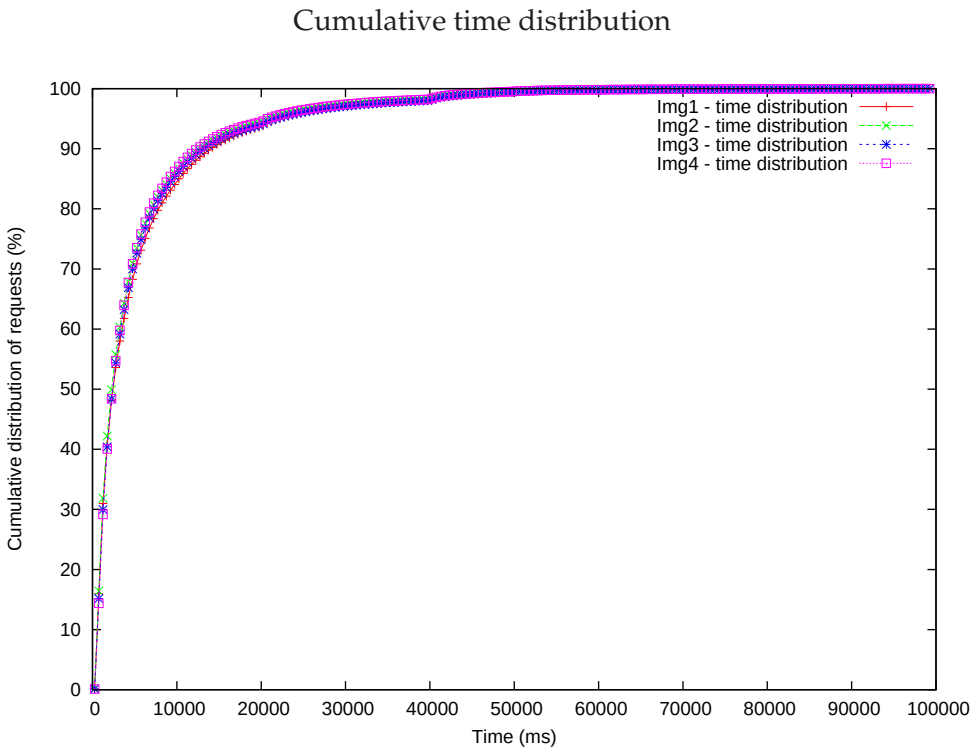

Figure 5.11: Cumulative percentage distribution of requests with regards to the total time in transcoder.

5.5 Scatter plots

In this section we will describe some scatter plots to see the relationships between our main parameters, uncompressed file size, compressed file size and time. We have made one plot for each of the four image qualities and one plot where we have merged these together. Due to the fact that gnuplot is incapable of plotting the enormous amount of points we would get if we used our entire database we only use the four first gigabytes of the database for these plots. This is still a representative selection with more than 41 million requests.

5.5.1 Compressed vs uncompressed file sizes

In the next five sections we will describe the relationship between the uncompressed and compressed file sizes for each image quality with scatter plots.

We have used the same scale on both axis for each individual plot to show that the straight line with a 45 degree angle appear when the x and y values are equal. We will refer to this line as the maximum line and the files that follow this line has the same size after compression as they had before, meaning that the compression algorithm in the Opera Mini servers have no effect on these files. If we had found any cases above this line it would have meant that the compression algorithm had increased the file.

Image quality 1

We see from figure 5.12 that there is a high density in the area with small files up to circa 50000 bytes uncompressed. This makes it difficult to predict what will happen in this area, as the results spread over the hole scale. When the file sizes increase we see that the gap between the maximum line and majority of the files increase. We see that there is an area with almost no files. This area seems to increase as the file sizes increase, meaning that the compression is getting more effective the larger files we get. We see that the area with relatively high density of files seems to cover an area from a new line with almost 30 degree angle, down to a line with only a few degrees angle above the x-axis. We notice that no files with uncompressed size above circa 550000 bytes follows the maximum line. This tells us that the effect of the compression increase with the file size and that all files above 550000 bytes uncompressed experience relatively good compression. We see that there is a cloud of dots with uncompressed file size of approximately 2.2 million bytes,more than 2MB. It is difficult to explain why we find such concentrations in this area. Image quality one are downloads with no images included and it is hard to believe that there is one popular web page with of this size with no images. These are large files to download to a cellular phone even if the compressed size is reduced by four times.

Image quality 2

From figure 5.13 we see that the area with uncompressed files sizes below 550000 bytes is dense. At this point we have a vertical line and it is not possible to tell what kind of compression you will achieve in the area below this line. Above the line there are suddenly a different pattern. Most of the files are far from the maximum line and score well in terms of compression. We see that there are two clouds of dots with uncompressed file size of approximately 2.2 million bytes, more than 2MB. It is difficult to explain why we find such concentrations in this area. These are large files to download to a cellular phone even if the compressed size is reduced by four times.

Image quality 3

From figure 5.14 we see that the pattern is almost the same for image quality 3 as for image quality 2. The same vertical line is located at uncompressed file sizes of approximately 550000 bytes and the area below this line is all dense.

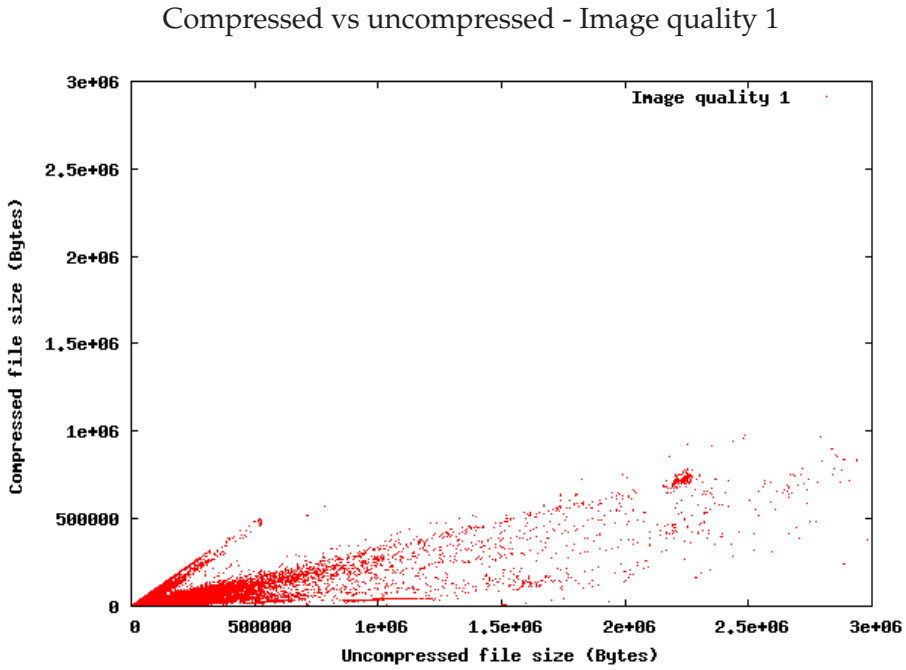

Figure 5.12: Scatter plot of uncompressed file sizes vs compressed file sizes for image quality 1.

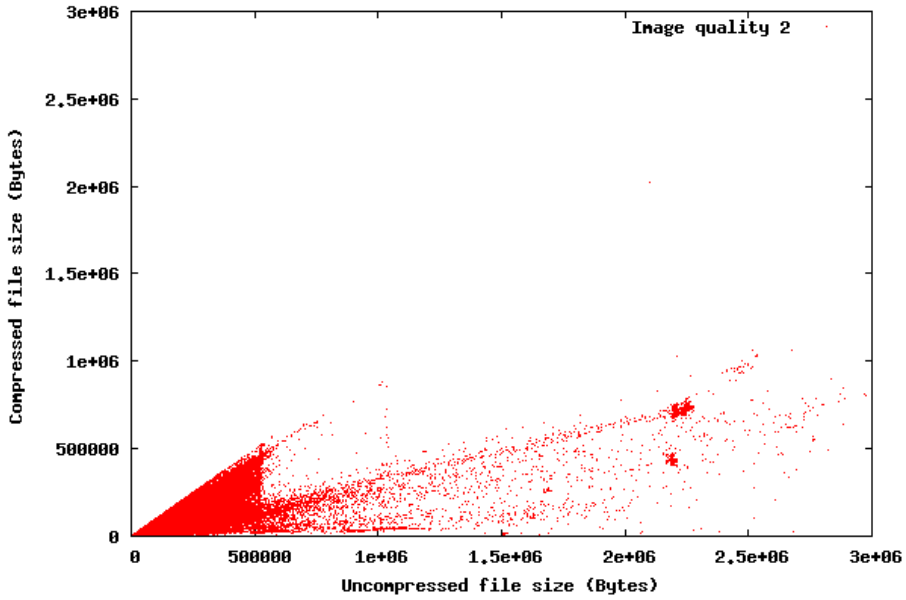

Figure 5.13: Scatter plot of uncompressed file sizes vs compressed file sizes for image quality 2.

Above the line we see that the files are far from the maximum line and that the majority of the files achieve good compression. We also notice the same clouds of dots as for image quality two.

Compressed vs uncompressed - Image quality 3

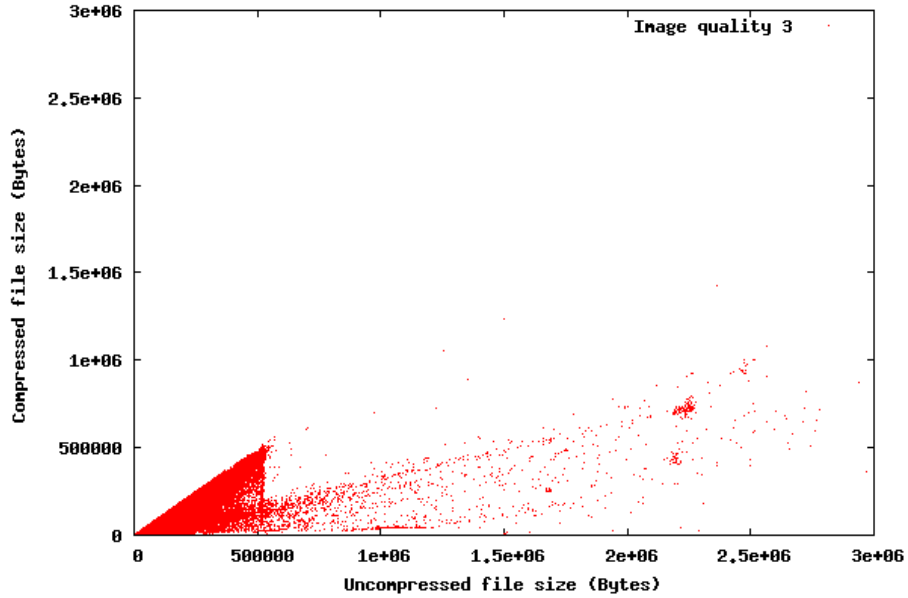

Figure 5.14: Scatter plot of uncompressed file sizes vs compressed file sizes for image quality 3.

Image quality 4

From figure 5.15 we see the same pattern when it comes to the vertical line as the two latter examples. The area with uncompressed file sizes up to 550000 bytes is all dense and it is hard to predict what kind of compression you will achieve in this area. When we investigate the area above the vertical line it varies to some part from the two latter. We see that there is a higher density of files close to the maximum line. This means that there is a greater number of large files that are hard to compress. Except for the files following the line, the rest spreads out more like the two previous examples, achieving good compression. Also here we notice the two clouds of dots found in the previous image qualities.

Sum of all image qualities

In figure 5.16 we have plotted all the four image qualities with different colors on top of each other in one plot. The plot makes no sense if viewed in black and white. We have placed the image qualities in order according to the area they cover. Image quality one has the smallest files and cover the least area. This one is plotted on top with purple color. Beneath this one is image quality

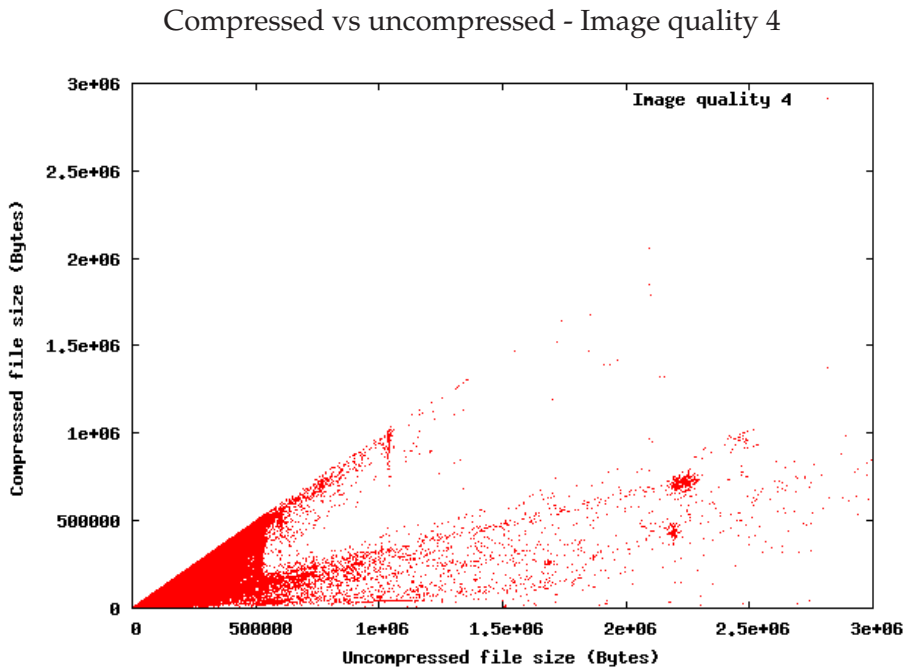

Figure 5.15: Scatter plot of pnguncompressed file sizes vs compressed file sizes fo image quality 4.

two plotted with blue color. Then it is image quality three with green color and at the bottom image quality four plotted with red color. We see that the shapes of the plots are remarkably equal and that they all follow each other to a great extent. The coloring helps us to show the differences. For image quality one we see that it follow the rest for the area below the 30 degree line. We also see that it follows the others a short while one the maximum line. The image quality two, three and four are even more equal. All three have a vertical line at uncompressed file sizes of approximately 550000 bytes. The area below this is all dense. It is hard to predict what effect one could achieve from compression of files in this area since the results divide over the hole scale. They also follow each other remarkably well below the 30 degree line, but we can see that image quality four has a higher density of requests close to the maximum line when we look at uncompressed file sizes above 550000. This means that there is a greater number of large files with image quality four that have less affect by the compression then it is for the other image qualities. We believe that the files that follow the maximum line are mainly pure image downloads.

We can also notice that the two clouds of dots with uncompressed file size of approximately 2.2 million bytes, more than 2MB, appears for all image qualities. It is difficult to explain why we find such concentrations in this area. These are large files to download to a cellular phone, even if the compressed size is reduced by four times. One explanation of the dots could be that the majority of all requests are within the area of uncompressed file sizes up to 550000 bytes. The number of requests above this number is so low that we

can see the requests as single dots. Even if it looks like an area with greater density, it does not mean that there are particularly high number of requests if we compare to the total numbers. The fact that these clouds follow the same pattern for all image qualities can indicate that this is downloads of the same web page, maybe a popular start page.

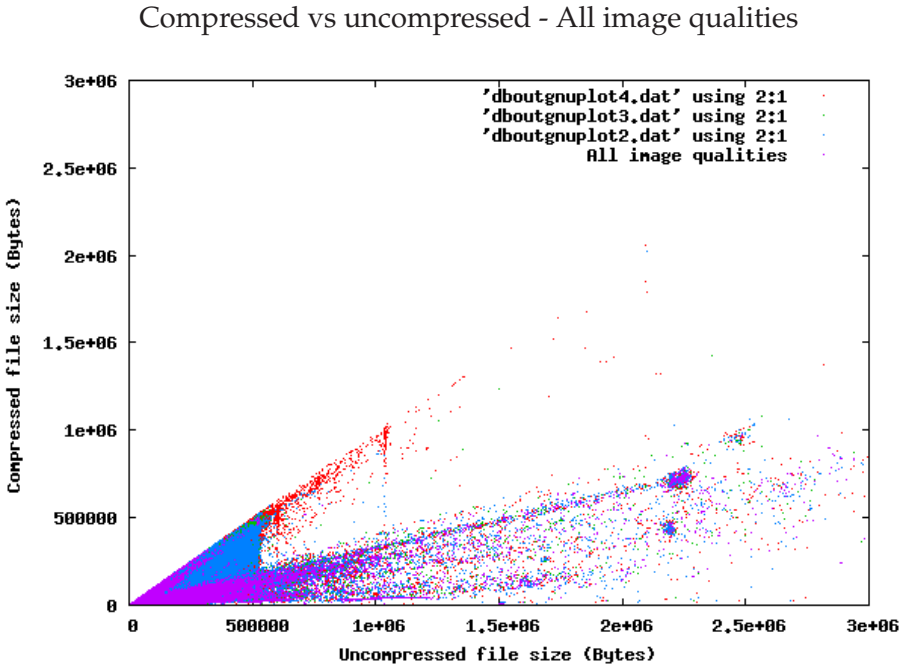

Figure 5.16: Scatter plot of uncompressed file sizes vs compressed file sizes for all image qualities.

5.5.2 Time vs Uncompressed file size

In the next sections we will present some scatter plots of the parameters, 'time' vs 'uncompressed file size'. We have made one scatter plot for each image quality to try to figure out if there is any relationship between size of the downloaded web page and the time it takes to download and compress it at the transcoder. We were expecting to see that the time increased as the files were increasing.

Image quality 1

We see from figure 5.17 that there is a line close above to the x-axis. The line has an angle with a few degrees increase related to the x-axis up to a point where the uncompressed file sizes are about 2.2 million bytes. From this point the line seems to flatten. This line tells us that the for very small files, some are handled in almost 'no time', but when the file sizes increase the values of the minimum times for handling a request increases slightly. We see that many of the small files use large time to handle. This is the opposite of what

we expected. The area below file sizes of 500000 bytes are all dense. We have relatively few requests with large file sizes and they spread over the timescale. It look like large files are faster to download than small files. We see two areas of large files with higher density of dots than its surroundings. One at 1.1 million bytes and one at 2.2 million bytes.

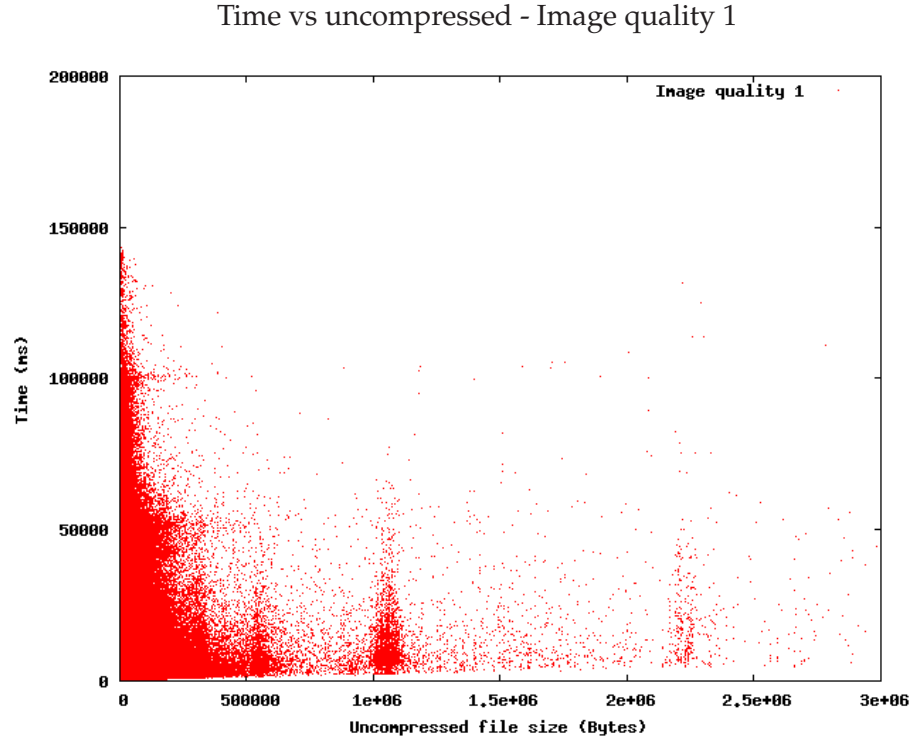

Figure 5.17: Scatter plot of time in transcoder vs compressed file sizes for image quality 1.

Image quality 2

From figure 5.18 we see that the pattern of image quality two has great similarities to the pattern for image quality one. We notice the same line just above the x-axis and we see that there is a high density in the area below 500000 bytes. We also here experience a high number of requests with small file sizes that uses long time to be handled by the transcoder. Above this area we see that the dots are spread out over a relatively large area. Many of these files achieve short handling times even if the files are large. We see the same to areas with high density of dots at 1.1 million bytes and 2.2 million bytes.

Image quality 3

From figure 5.19 we see that the pattern of image quality is much the same as the pattern for image quality two. We see the line following the x-axis and the high density area below 500000 bytes, meaning once more that there are many

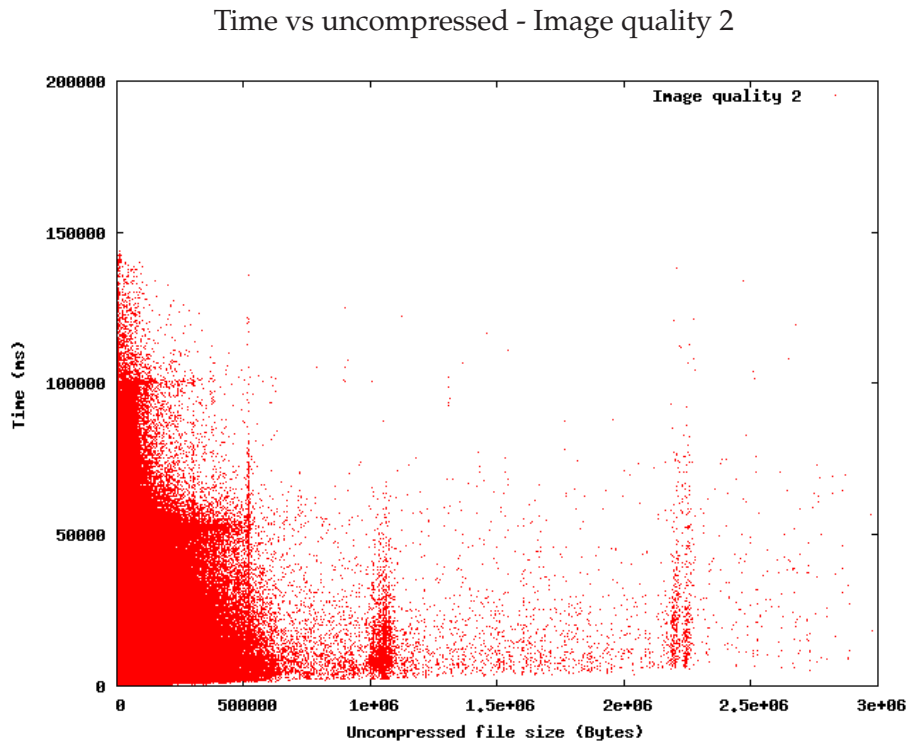

Figure 5.18: Scatter plot of time in transcoder vs compressed file sizes for image quality 2.

small web pages that take long time to handle by the transcoder. Above this area we see that there is a relatively low density of dots. We see the same to areas with high density of dots at 1.1 million bytes and 2.2 million bytes. In addition we also notice a vertical line at approximately 500000 bytes that we also can locate in figure 5.18 when we look closer.

Image quality 4

From figure 5.20 we see that it is astonishingly the same pattern as the others. We see the line along the x-axis, the high density area below 500000 bytes and the low density area above. We see the same to areas with high density of dots at 1.1 million bytes and 2.2 million bytes. We see the vertical line at approximately 500000 bytes and in addition we notice some horizontal lines at 50000 ms and 100000 ms. These have been present in the latter figures and they are maybe most distinct in figure 5.19.

Sum of all image qualities

In figure 5.21 we have plotted all the four image qualities with different colors on top of each other in one plot. The plot makes no sense if viewed in black and white. We have placed the image qualities in order according to the area they cover. Image quality one has the smallest files and cover the least area.

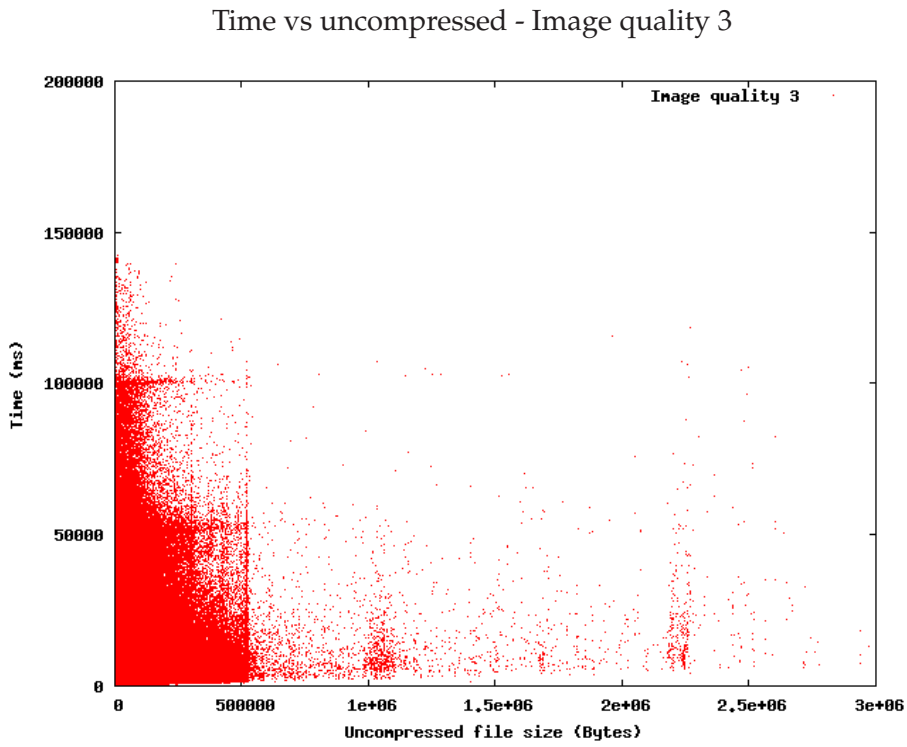

Figure 5.19: Scatter plot of time in transcoder vs compressed file sizes for image quality 3.

This one is plotted on top with purple color. Beneath this one is image quality two plotted with blue color. Then it is image quality three with green color and at the bottom image quality four plotted with red color. The shapes of the plots are far from what we expected, but they are remarkably equal for all image qualities. Image quality one have a lower curve than the rest for the area below 500000 bytes. We truly see this since the blue color is visible in a large part of the figure in this area. We do not see much of the other colors and we interpret this as image quality three and four follow image quality two.

We see that many of the small files use large time to handle. This is not as we expected and we find it hard to explain these results. The line we have seen close above the x-axis tells us that the for very small files, some are handled in almost 'no time', but when the file sizes increase the values of the minimum times for handling a request increases slightly. Anyway we believe the results of the large files are satisfactory. We are more concerned about the results of the small files. A great part of these have very high time values. One explanation could be that there is an overwhelming number of small files. Some of the results with high time values is probably due to some kind of errors, like broken link connections, web server failures or mistyped URL's. Another explanation could be that a lot of files , independent of the size, include commercial banners from heavy loaded servers that delay the download. This will have huge impact for the download time of the small files.

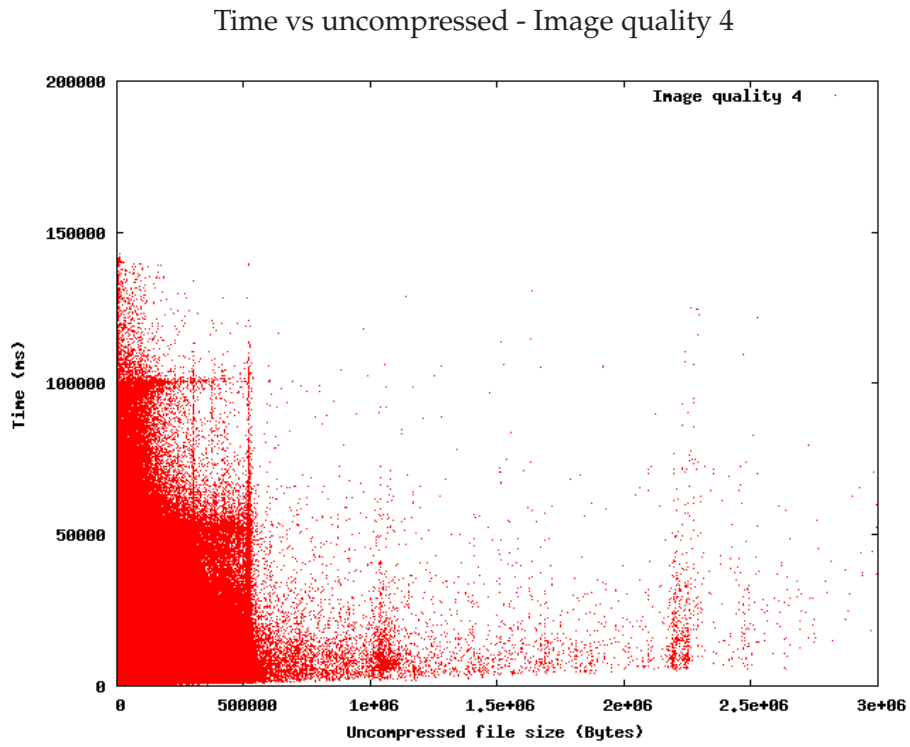

Figure 5.20: Scatter plot of time in transcoder vs compressed file sizes for image quality 4.

An explanation of why the files above 500000 bytes get better results could be that users with poor bandwidth connections will try to avoid downloading these large files, because they know it will be a time consuming process. The time values we see here up to 140000 ms, more than two minutes, will be experienced as 'ages' and most people will get impatience and interrupt the download before it is finished. We have relatively few requests with large file sizes, but it could be that people mainly download these kind of files when they know that they have a good bandwidth connection. The two areas of large files with higher density of dots than its surroundings at 1.1 million bytes and one at 2.2 million bytes, was present for all image qualities. These files are large considered that they are intended to download to a mobile phone. We discussed the appearance of similar areas in section 5.5.1 and we believe that the files with uncompressed file size of 2.2 million bytes are downloads of the same page, perhaps a popular start page.

The vertical line at we experience at approximately 500000 bytes could also be representing a popular start page. It looks like the sizes are all the same, but the time it takes to handle is spread over the hole scale. We also see horizontal lines at 50000 ms and 100000 ms. These have been present more or less in all the figures,but we find it hard to explain why we experience these.

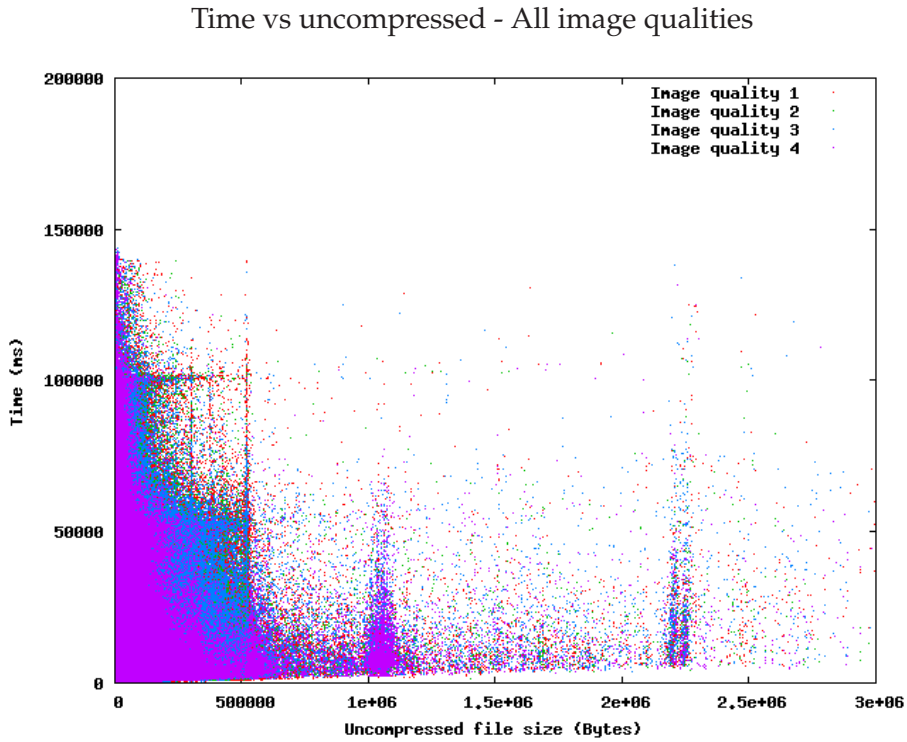

Figure 5.21: Scatter plot of time in transcoder vs compressed file sizes for all image qualities .

Chapter 6

Discussion

6.1 Ensuring the validity of the results

During the hole project the author has done his uppermost to keep the validity of the results as good as possible. No matter how much effort you put in this work, there will always be uncertainties. You can take precautions and create routines to limit the occurrence of errors, but it is impossible to guarantee that no errors will occur. The process of collecting the log files from the Opera Mini transcoders is fully automated and the information is logged continuously. We have seen some cases where the logs contains misinterpret URL's, but this does not affect the results of our analysis. The analysis process is more prone to personal errors. When you analyze huge amounts of data you need to structure each step in the process. When you run similar tests for various parameters you must make sure that you get all correct from the start, or else you will have to redo the steps for all cases. One simple, but important thing to make sure of is to name the different files with understandable names. During a project like this you will generate many files you need to keep track of and a consequent naming will help you to maintain the control.

The discussion and conclusion in this thesis are based on the numbers we found in the result chapter. We have done our best to discuss these results with regards to the information we have got for the various parameters. We are aware that the use of average values can lie. We believe that some of these findings are difficult to explain, to some extend because the results are far from our expectations. One example could be the finding that state that the average compression is the same in number of bytes for all image qualities. The author has just for curiosity tried to download three different web pages to his mobile with all the four different image quality settings in his Opera Mini browser. The results displayed by the mobile phone shows that these settings will have a great impact on the amount of data transferred for this particular web page. Table 6.1 shows the number of bytes downloaded to the authors phone from three various web pages. The result reveal that there are totally different values regarding the differences of image qualities than the results we found based on the numbers from the log files, but we also see the differences between these pages. This might support our findings in the scatter plots saying that it is hard

to predict the effect on the different web pages and that the result distribute all over the scale.

Browser test

Image quality	www.vg.no	www.facebook.com	www.hio.no
1	27kB	3kB	3kB
2	183kB	9kB	28kB
3	287kB	10kB	37kB
4	435kB	12kB	72kB

Table 6.1: Number of bytes downloaded to the authors phone from three various web pages.

6.2 Limitations in replicating the work

This project will be hard to replicate due to restricted access to the Opera Mini log files. This is a problem for further work on the topic. All the four methods we discussed would have been hard to replicate, because they demand collaboration with Opera Software. Data mining of existing data require access to the log files. Traffic measurements require access to a measuring point where all Opera Mini traffic pass through, like a span port on the switch. The controlled experiment also demand for access to Opera equipment to measure or collect the information you have generated. The last method with the testbed would probably require that you get assistance from Opera Mini experts to have the setup and configuration correct. If you collaborate with Opera and get access to the data, the project should be easy to replicate. We have just used a computer with 'normal' specs, freely distributed OS and software.

6.3 Application monitoring in real life

The approach we have used to analyze the Opera Mini log files could easily have been implemented in other cases of data mining. The setup could have been used in a real life working environment, but it might be preferable to run the analysis once per day to keep decent time aspects for the tests. To query the database that covered data from the entire two weeks was a time consuming task. One example of this is that the queries we ran to get the information we used in section 5.3.2, the distributions of file sizes, took approximately two days. The long runtime demands for good planning to make sure that the queries are correct from the beginning. The runtime could have been reduced if we had equipment with better processing capacities. Due to the fact that we were not sure what kind of analysis we were supposed to use and how many times we needed to request the different parameters, we decided to store the information in a database. This makes it easier to keep the overview of all parameters and the access to the data more efficiently. The collaboration with Valeri Chermetiev in the preface of the project has given me great experiences

and the discussions we have had has given me useful reflections on how it is to perform these kind of projects in a working environment.

6.4 Revisiting the problem statements

If we look back at the problem statements defined in section 1.2, we where having some clear assumptions on what contexts that would affect the compression of the data. Our findings do not support all of these assumptions.

6.4.1 PS1

The first question we asked was if the image quality display settings in the Opera Mini browser would affect the size of the downloaded file. We assumed that the size of the web pages would be the same for all cases where images where downloaded to the transcoder. We assumed that the size of the downloads with image quality one, meaning no images, would be less since the transcoder download these pages without the images. From figure 5.1 we see that our assumptions were wrong with regards to average file sizes. We see that the smallest files are the ones without images,represented by pink color. This is as we assumed, but the three other cases differed more than we expected. We see that the the file sizes increase as the image quality display setting increase. It looks like people with high image quality settings download web pages with larger or better quality images and that there is an remarkable difference in the size of the web pages requested by the users with the different settings. It could be possible that users actively change the settings in the browser when they know they are going to visit a web page with high quality images included or when the display quality is of great matter. Examples of this could be when people view images of celebrities or pornographic content. If this is the case it might be a good solution to add an option in the web browser configuration that you can use the selected setting only for 'next session'.

6.4.2 PS2

The next question we asked was if there is a connection between the quality of the image transferred to the client and the achieved level of compression. We assumed that the file size would be less as the image quality was reduced. We see from the same figure 5.1 by looking at the average file sizes that the level of the compression is approximately the same in regards to the reduction in bytes for all image qualities. The reduction seems to be circa 13000 bytes for all cases. The compression of the files with 'no images' gains great compression for the text. The reduction of the files with images is achieved from reducing the images in addition to the compression of the text. That the levels between the different qualities are the same can be due to limit settings of the quality areas decided by the Opera Mini developers. If we look at the achieved level of compression compared to the average file sizes, we see that the small files

experience a much better compression if you compare with their original file sizes. This tells us that the lower image quality display setting we use, the better relative compression we achieve.

If we look at the two upper plots in figure 5.4, we see the total amount of transferred data uncompressed and compressed during the two weeks period. We see that the reduction in bytes transferred is circa two third for the setting with 'no images'. Further on we see that the reduction is more than forty per cent for low image quality, one third for medium image quality and less than thirty per cent for the high image quality. These numbers support our discussion from the last paragraph with average file sizes.

6.4.3 PS3+4

The third question we asked was if there was any relationship between the image quality display setting in the browser ant the time it takes to download and process the request by the transcoder. We assumed that the time would increase as the quality image was reduced due to more extensive compression. From figure 5.4 we see that the distribution of time, lower right plot, has the same distribution as the number of request, lower left plot. We see that the two upper plots with file sizes have different patterns. This tells us that the time it takes to handle a request only depends on the number of requests and not on the type of image quality or size of file.

This is also an answer for our next question of the problem statement where we asked if there was any relation between the size of the downloaded web page and the time it takes to handle the request in the transcoder. These findings can to some degree help us to explain the plots in figure 5.17 to 5.21 where we plot the time it takes to handle the requests in the transcoder compared to the uncompressed file quality. These figures gave us a quite different result than we expected. It looks like it takes longer time to handle requests with small file sizes than requests with large file sizes for all image qualities. The high density of dots makes it hard to interpret the plots, especially for uncompressed file values below 500000 bytes. Anyway we see that a great part of the requests with small file sizes have very high time values. One explanation could be that there is an overwhelming number of small files. Some of the results with high time values is probably due to some kind of errors, like broken link connections, web server failures or mistyped URL's. Another explanation could be that a lot of files , independent of the size, include commercial banners from heavy loaded servers that delay the download. This will have huge impact for the download time of the small files.

An explanation of why many large files get better results could be that users with poor bandwidth connections will try to avoid downloading these large files, because they know it will be a time consuming process. The time values we see here up to 140000 ms, more than two minutes, will be experienced as 'ages' and most people will get impatience and interrupt the download before it is finished. We have relatively few requests with large file sizes,

but it could be that people mainly download these kind of files when they know that they have a good bandwidth connection.

6.4.4 PS5

In the previous question we figured out that there was an overwhelming amount of small files. We decided to plot the distribution for all image qualities with regards to their file sizes to figure out how the files were divided throughout the scale. We see from figure 5.5 to 5.9 that there is a great majority of small files for all image qualities. We have plotted both uncompressed and compressed file sizes in the same figure to see how the compression reduce the file sizes. We see that there is a huge overweight of compressed files in the areas with the smallest file sizes. This tells us that the compression works, because many of the files have been swapped to a lower file size level after compression.

The last question we asked was if there was a relationship between the uncompressed file sizes and the compressed file sizes. We assumed that the compression would follow a linear pattern. We have tried to present this relationship in scatter plots 5.12 to 5.16. We see that the patterns are astonishingly equal for all image qualities. We see that the area with uncompressed file sizes up to 500000 bytes is all dense, except for image quality one where this area cover some smaller parts. The files have all possible values in this area and are hard to interpret. Apart from this we notice that some files follow the 45 degree line, especially for image quality four. This is the line where the compressed files have the same size as the uncompressed files, meaning that the compression had no effect on these files. We believe that these files are pure image downloads. We also notice that it seems to be a gap from the 45 degree line to a line with approximately 30 degrees where there are few requests. Below the 30 degree line there is a relatively high density of dots almost down to the x-axis. These are files that experience great effect from the compression.

Chapter 7

Conclusion

When we started the work with this thesis we had limited knowledge about the Opera Mini technology. During the time of this project this knowledge has increased as well as the respect for people that work with analysis of huge amounts of data. The experience we have gained from this is that this is a time consuming task where you need to be structured to keep control of the work. There has not been performed any similar projects with the Opera Mini log files and that made our method exploratory. We do not know the background for the choices done by the Opera Mini user and we have no knowledge on why they use the configuration they do, or why they download the web pages they do. This makes the interpretation of the results difficult.

A great deal of the focus in our analysis has been on the different image display quality settings in the Opera Mini browser. This is a setting that can be configured by the user and we assumed that this setting was the one with the greatest effect on the amount of data downloaded to the mobile phone.One of our first findings was that the majority of the downloaded web pages are displayed by the user with image quality two, low quality images, in the Opera Mini browser. This is the default setting in the browser and hence an expected result. This is the optimal setting if you want to view images, but still want to download as small amount of data as possible.

Our main problem statement was to figure out what contexts that affected the compression of the data generated by Opera Mini users. We limited the the analysis by defining sub problems that contained the factors we believed would have the greatest impact on the result. We have tried to answer all of our problem statements and add some additional results when we found this beneficial. Some of our findings are according to our assumptions while others are far from.

When we look at PS1 we have shown that our assumptions regarding the size of the uncompressed files. We believed that these would be the same for all image qualities that included images in the downloads to the transcoder. Figure 5.1 shows that this assumption was wrong and that the average uncompressed file sizes increases, when the image quality setting in the Opera Mini browser increases.

For PS2 we have found that the achieved level of average compression is the same for all image qualities with respect to number of bytes. If we calculate the compression compared to the original file sizes (see table 5.2) we see that in general the small files of image quality one achieve the greatest relative compression while the larger files with image quality four achieve the poorest relative compression. The compression of the files with 'no images' gains great compression for the text. The reduction of the files with images is achieved from reducing the images in addition to the compression of the text. That the levels between the different qualities are the same can be due to limit settings of the quality areas decided by the Opera Mini developers.

Another unexpected result we got was related to PS3 and PS4. From the two lower plots in figure 5.4 we see that nor the size of the files or the image quality configuration in the browser had any impact on the total time it took to download and compress a file in by transcoder. We claim this because the plot showing the number of requests and the plot showing the total time in the transcoder have the same shape. These findings are supported by the scatter plots in figure 5.17 to 5.21, where we see that a great part of the small files have high time values. This tells us that the time it takes to handle a request mainly depends on the number of requests and not the type of image quality or size of file.

For PS5 we have used scatter plots to present the compression of the files. To interpret the effect of the compression based on these plots 5.12 to 5.16 is hard due to the high density of requests. Based on the findings in PS2, claiming that the compression has no effect on image files, we believe that the majority of the files following the 45 degree line are image files that achieve no effect from the compression. Due to the pattern of the plot with 'no images', we see that there must be some other formats without images that have no effect of the compression, like already compressed text.

7.0.5 Future work

The analysis we have performed has been limited to log files generated from real traffic. We have focused on a few parameters that we expected would have the greatest impact on the compression of the data. We have shown that the image quality setting in the Opera Mini can tell us something about the user behavior. We have also shown that the compression has relatively small effect on images. For further work based on our problem statement it is possible to use the same methods with other parameters. One example could be to analyze if there are differences in the achieved level of compression for various models of mobile phones. Another way to look at the problem could be to only look at the unique web pages that are downloaded with all image qualities and compare the effect of the compression for each cite. One thing to have in mind is that the achieved reduction in file sizes to some extent can differ based on the specifications of the mobile phones requesting the page.

A different approach we find exiting for a future future project is to perform a controlled experiment. We still believe that the wireless link between the mobile phone and the mobile operator is the weakest link of the connection. This makes it especially interesting to study this part of the connection. A method to do this could be to generate fixed size web pages and request these from a mobile phone. From the log files it is possible to see the level of compression for these files. When you have the possibility to modify the content of the web pages, this will give you an advantage in the sense that you can easily perform modifications and analyze the impact of these. You need to measure both the client and the server side to calculate the transfer time. The main issue to get correct measurements is to make sure that the time on the two sides are synchronized. You can use the information from the log files regarding the time it takes to download and process the web page in the transcoder and compare this with the results you get from your measurements. This will tell you how large part of the transfer that is spent from the Opera server to the mobile phone, with respect to the total download time.

This project could be performed with the existing infrastructure or in a testbed. Both methods have advantages and disadvantages over the other. If you perform it with the existing infrastructure your requests will be handled as real life user traffic and the results will give you information on how the existing service perform. The drawback is that you spend time to extract your requests from a huge amount of data. You will also have less possibilities to modify the setup of the servers. If you build your own testbed you will have the benefit of controlling all parts of the system. This makes it easy to do modifications to your setup. The drawback is that a great deal of time will be used to set up the equipment and you will probably need assistance by Opera Mini experts to get the configuration right.

Bibliography

[1] Bengt Magnor Orstad and Erling Reizer. End-to-end key performance indicators in cellular networks. Master's thesis, Information and Communication Technology, Agder University College, Faculty of Engineering and Science, Grimstad, Norway, 2006.

[2] Data compression. *http://en.wikipedia.org/wiki/Data_compression*.

[3] Opera mini features. *http://www.operamini.com/features/*.

[4] Sqlite. *http://www.sqlite.org/*.

[5] Gnuplot. *http://www.gnuplot.info/*.

[6] World wide web. *http://en.wikipedia.org/wiki/World_Wide_Web*.

[7] Web usage. *http://en.wikipedia.org/wiki/Usage_share_of_web_browsers*.

[8] Web browser usage. *http://marketshare.hitslink.com/report.aspxqprid=0*.

[9] Mobile web browsers. *http://en.wikipedia.org/wiki/Mobile_browser*.

[10] Lara D. Catledge and James E. Pitkow. Characterizing browsing strategies in the world-wide web. Technical report, School of Literature, Communication and Culture and Graphics, Visualization, Usability Center, Georgia Institute of Technology, Atlanta, GA 30332-0280.

[11] Mobilt bredbånd. *http://telenor.no/privat/mobil/abonnement/mobiltbredband/index.jsp*.

[12] Netcom connect. *https://netcom.no/abonnement/connect.html*.

[13] Martin Halvey, Mark T. Keane, and Barry Smyth. Mobile web surfing is the same as web surfing. *Commun. ACM*, 49(3):76–81, 2006.

[14] Wap. *http://en.wikipedia.org/wiki/WAP_2.0*.

[15] Operators-making-money-on-opera-mini. *http://www.opera.com/pressreleases/en/2006/04/06/*.

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

[17] Opera mini. *http://en.wikipedia.org/wiki/Opera_mini*.

[18] Dette er operas hemmelige våpen. *http://www.digi.no/php/art.php?id=504299*.

[19] En-milliard-sidevisninger-gjennom-opera-mini. *http://www.digi.no/php/art.php?id=313791*.

[20] *Developer case study: Managing Java fragmentation, Opera Software's Java ME browser client*.

[21] Internet censorship. *http://en.wikipedia.org/wiki/Internet_censorship*.

[22] Freenet. *http://freenetproject.org/*.

[23] Skweezer. *http://en.wikipedia.org/wiki/Skweezer*.

[24] Personal data act - act of 14 april 2000 no. 31 relating to the processing of personal data. Technical report, Datatilsynet, 2000.

[25] *Analytical Network and System Administration: Managing Human-Computer Systems*. John Wiley Sons, Ltd, 2004.

[26] Antero Kivi. Measuring mobile user behavior and service usage: Methods, measurement points, and future outlook.

[27] Yingjun Zhang James H. Andrews.

[28] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3):9, 2006.

[29] Peggy Wright. Knowledge discovery in databases: tools and techniques. *Crossroads*, 5(2):23–26, 1998.

[30] Matti Siekkinen. Measuring the internet, part 1: The big picture. 2007.

[31] Jeffrey K. Hollingsworth and Barton P. Miller. Dynamic control of performance monitoring on large scale parallel systems. In *ICS '93: Proceedings of the 7th international conference on Supercomputing*, pages 185–194, New York, NY, USA, 1993. ACM.

[32] Haavard Wik Thorkildssen. Passive traffic characterization and analysis in heterogeneous ip networks. Master's thesis, University of Oslo, Department of Informatics, Oslo, Norway, 2005.

[33] Python programming language. *http://www.python.org/*.

[34] Robert W. Jernigan. A photographic view of cumulative distribution functions. *Journal of Statistics, Education Volume 16, Number 1*, 2008.