ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

BY
Habib LOUAFI

A PREDICTION-BASED DYNAMIC CONTENT ADAPTATION FRAMEWORK FOR ENTERPRISE DOCUMENTS APPLIED TO COLLABORATIVE MOBILE WEB CONFERENCING

MONTREAL, OCTOBER 1, 2013

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS:

Mr. Stéphane Coulombe, Thesis Director
Department of Software and IT Engineering at École de technologie supérieure

Mr. Mohamed Cheriet, Committee President
Department of Automation Engineering at École de technologie supérieure

Mr. Roch Glitho, External Examiner
Concordia Institute for Information at Concordia University

Mr. Alain April, Examiner
Department of Software and IT Engineering at École de technologie supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND PUBLIC

ON SEPTEMBER 6, 2013

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

First and foremost, I would like to express my thanks and respectful gratitude to my thesis director, Prof. Stéphane Coulombe, for his confidence in me, guidance, financial and moral support, and constant encouragement during this journey of research. Stéphane, I have learned a lot from you!

I would also like to thank Umesh Chandra, from Nokia Research Center, for his contributions to this research and valuable comments, discussions and suggestions.

My thanks go also to the members of the jury, Professors Mohamed Cheriet, Alain April and Roch Glitho, who have accepted to devote their precious time to evaluating my thesis.

Without the financial contribution of the Nokia Research Center (NRC) and the Natural Sciences and Engineering Research Council of Canada (NSERC), this work would never have been carried out.

I would like to thank Mrs Cynthia Hunt, for her English revisions of my research papers and documents, and her valuable comments which helped me improve my writing skills.

I would like to take this opportunity to thank my colleagues at the Vantrix Industrial Research Chair in Video Optimization Lab, for their sincere and helpful discussions, valuable comments and enjoyable time we spent together in the Lab.

I would like to express my deepest thanks and gratitude to all my family members, my mother and my wife in particular, for their support, patience and understanding throughout this long process.

I would like to thank my closest friends for their personal support and fruitful discussions.

I would like to thank everyone who has contributed to the success of this research.

Lastly, I dedicate this thesis to my former professor of mathematics, Fouad Khalil, for his time, efforts and, above all, his devotion to his honorable profession.

A PREDICTION-BASED DYNAMIC CONTENT ADAPTATION FRAMEWORK FOR ENTERPRISE DOCUMENTS APPLIED TO COLLABORATIVE MOBILE WEB CONFERENCING

Habib LOUAFI

ABSTRACT

Enterprise documents, created in applications such as PowerPoint and Word, can be used and shared using ubiquitous Web-enabled terminals connected to the Internet. In the context of Web conferencing, enterprise documents, particularly presentation slides, are hosted on the server and presented to the meeting participants synchronously. When mobile devices are involved in such meeting conferencing applications, the content (e.g.: presentation slides) should be adapted to meet the target mobile terminal constraints, but more importantly, to provide the end-user with the best experience possible.

Globally, two major trends in content adaptation have been studied: static and dynamic. In static content adaptation, the content is adapted into a set of versions using different transcoding parameter combinations. At runtime, when the content is requested, the optimal of those versions, based on a given quality criterion, is selected for delivery. The performance of these solutions is based on the granularity in use; the number of created versions. In dynamic content adaptation, also called just-in-time adaptation, based on the mobile device context, a customized version is created on-the-fly, while the end-user is still waiting. Dynamically identifying the optimal transcoding parameters, without performing any transcoding operation, is very challenging.

In this thesis, we propose a novel dynamic adaptation framework that estimates, without performing transcoding, near-optimal transcoding parameters (format, scaling parameter and quality factor). The output formats considered in this research are JPEG- and XHTML-based Web pages. Firstly, we define a quality of experience measure to quantify the quality of the adapted content as experienced by the end-user. This measure takes into account the visual aspect of the content as well as its transport quality, which is mostly affected by the network conditions. Secondly, we propose a dynamic adaptation framework capable of selecting dynamically and with very little computational complexity, near-optimal adapted content that meets the best compromise between its visual quality and delivery time based on the proposed quality of experience measure. It uses predictors of file size and visual quality of JPEG images subject to changing their scaling parameter and quality factor proposed in recent researches. Our framework is comprised of five adaptation methods with increased quality and complexity. The first one, requiring one transcoding operation, estimates near-optimal adapted content, whereas the other four methods improve its prediction accuracy by allowing the system to perform more than one transcoding operation.

The performance of the proposed dynamic framework was tested with a static exhaustive system and a typical dynamic system. Globally, the obtained results were very close to optimality

and far better than the typical dynamic system. Besides, we were able to reach optimality on a large number of tested documents. The proposed dynamic framework has been applied to OpenOffice Impress presentations. It is designed to be general, but future work can be carried out to validate its applicability to other enterprise documents types such as Word (text) and Excel (spreadsheet).

Keywords: Web conferencing, dynamic content adaptation, transcoding, quality of experience, JPEG, XHTML

UN CADRE D'ADAPTATION DYNAMIQUE DE DOCUMENTS D'ENTREPRISE BASÉ SUR LA PRÉDICTION APPLIQUÉ AUX CONFÉRENCES WEB MOBILES COLLABORATIVES

Habib LOUAFI

RÉSUMÉ

De nos jours, les documents d'entreprise, créés par des applications telles que PowerPoint et Word, peuvent être utilisés et partagés par des terminaux dotés d'une connexion Web. Dans un contexte de conférence Web, les documents d'entreprise, particulièrement les diapositives, sont hébergés et partagés entre les participants de la conférence de façon synchrone. Quand des appareils mobiles sont impliqués dans de telles applications, le contenu (ex. : diapositives) doit être adapté pour respecter les exigences de l'appareil mobile. Plus important encore, cette adaptation doit fournir la meilleure expérience possible pour l'utilisateur final.

Globalement, dans l'adaptation de contenu, deux tendances ont été étudiées: statique et dynamique. Dans la première, le contenu est adapté en un ensemble de versions en utilisant différentes combinaisons de paramètres de transcodage. Au moment où le contenu est demandé, la meilleure version, évaluée avec un certain critère de qualité, est sélectionnée pour être livrée. La performance de ce genre de solution dépend de la granularité adoptée; le nombre de versions créées. Dans l'adaptation dynamique, appelée aussi juste à temps, en se basant sur le contexte de l'appareil mobile, une version sur mesure est créée, alors que l'utilisateur est toujours en attente. Dans ce cas, identifier les paramètres de transcodage optimaux, sans faire de transcodage préalable, n'est pas une tâche évidente.

Dans cette thèse, nous proposons un nouveau cadre d'adaptation dynamique de contenu qui permet l'identification, sans transcodage, des paramètres quasi optimaux (format, paramètre d'échelle et facteur de qualité). Deux formats ont été considérés dans cette recherche, à savoir des pages Web composées d'images JPEG et des pages Web XHTML composées de texte et d'images. Dans un premier temps, nous avons défini une mesure de la qualité d'expérience telle que perçue par l'utilisateur final. Cette mesure utilise la qualité visuelle du contenu ainsi que le temps requit au contenu pour atteindre sa destination, principalement affecté par les conditions du réseau. En utilisant cette mesure de qualité, nous avons proposé un cadre dynamique d'adaptation de contenu capable d'identifier, dynamiquement et avec une complexité minimale, les contenus adaptés quasi optimaux qui fournissent un meilleur compromis entre la qualité visuelle du contenu et son temps global de livraison. Le cadre proposé utilise des estimateurs de taille de fichier et de qualité visuelle d'images JPEG transcodées, proposés dans la littérature. Notre cadre est composé de cinq méthodes d'adaptation, chacune offrant un compromis différent entre la qualité et la complexité. La première, nécessitant une seule opération de transcodage, estime des paramètres de transcodage quasi optimaux. Les autres quatre méthodes améliorent la précision, en permettant au système de réaliser plus d'une opération de transcodage.

La performance du cadre dynamique proposé a été validée avec un système statique exhaustif et un système dynamique typique. Globalement, les résultats obtenus avec la méthode proposée s'approchent de l'optimal et sont significativement meilleurs que ceux obtenus par le système dynamique typique. De plus, nous avons atteint l'optimalité pour un bon nombre de documents testés. Le cadre dynamique proposé a été appliqué à des diapositives "OpenOffice Impress". Il a été conçu pour être général et des travaux futurs peuvent être réalisés pour valider son applicabilité à d'autres types de documents d'entreprise, tels que Word (texte) et Excel (tableur).

Mot-clés : Conference Web, adaptation dynamique de contenu, transcodage, qualité d'expérience, JPEG, XHTML

CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

AHP	Analytical Hierarchical Process
AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
base-64	binary-to-text encoding schemes
BREW	Binary Runtime Environment for Wireless
CAU	Content Adaptation Unit
CDMA	Code Division Multiple Access
CPU	Central Processing Unit
CSS	Cascading Style Sheet
DCT	Discrete Cosine Transform
DMOS	Differential Mean Opinion Score
DOM	Document Object Model
DSS	Degree of Satisfaction Score
DU	Decision Unit
ELECTRE	Elimination and Choice Translating Reality
GIF	Graphic Interchange Format
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
H.263	Digital video compression and encoding standard

H.264	Digital video compression and encoding standard
gaussmf	Gaussian Membership Function
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IBM	International Business Machines Corporation
IE	Internet Explorer
JIT	Just In Time
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
KB	Kilo Byte
kbps	Kilo Bit Per Second
KHTML	Konqueror Hyper Text Markup Language
LMCKP	Linear Multi-Choice Knapsack Problem
LTE	Long Term Evolution
MADM	Multiple Attributes Decision Making
MB	Mega Byte
MMS	Multimedia Messaging Service
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MS	Microsoft
OMA	Open Mobile Alliance

OOXML	Office Open Extensible Markup Language
OS	Operating System
OTF	Open Type Fonts
OWL	Web Ontology Language
P2P	Peer To Peer
PC	Personal Computer
PDA	Personal Digital Assistant
PNG	Portable Network Graphics
PSNR	Picture Signal-to-Noise Ratio
PVP	Personal View Point
QF	Quality Factor
QoE	Quality Of Experience
QoS	Quality Of Service
RAR	Roshal ARchive
RDF	Resource Description Framework
RGB	Red-Green-Blue
RLE	Run Length Encoding
RTF	Rich Text Format
SAW	Simple Additive Weighting
SAX	Simple API for XML
SMS	Short Message Service

SOA	Service Oriented Architecture
SPARQL	Simple Protocol and RDF (Resource Description Framework) Query Language
SSIM	Structural Similarity
SVG	Scalable Vector Graphics
SVP	Shared View Point
SWRL	Semantic Web Rule Language
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
TTF	True Type Fonts
UA-header	User Agent header
UA-Prof	User Agent Profile
UMTS	Universal Mobile Telecommunications System
UNO	Universal Network Objects
URL	Uniform resource locator
UTF-16	UCS (Universal Character Set) Transformation Format 16 bit
UTF-8	UCS (Universal Character Set) Transformation Format 8 bit
W3C	World Wide Web Consortium
Web 1.0	Web version 1.0
Web 2.0	Web version 2.0
Web 3.0	Web version 3.0
WiFi	Wireless Fidelity

WML	Wireless Markup Language
WML	Wireless Markup Language
WP	Weight Product
WURFL	Wireless Universal Resource FiLe
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
XSLT	eXtended Stylesheet Language Transformations
Zmf	Z-shaped Membership Function

LIST OF SYMBOLS

$\mathcal{C}$	Enterprise document
c_k	Enterprise document page
h_i	Enterprise document page component
$\mathcal{P}$	Set of transcoding parameters
f	Format
z	Scaling parameter (zoom)
QF	Quality factor
$\mathcal{T}$	Transcoding operation
$\mathcal{C}^{f,z,QF}$	Set of adapted contents created form $\mathcal{C}$ using f, z and QF
$c_k^{f,z,QF}$	Adapted content created form c_k using f, z and QF
D	Mobile device
$\mathcal{W}(D)$	Width of mobile device D
$\mathcal{H}(D)$	Height of mobile device D
$\mathcal{S}$	File size
$\mathcal{F}(D)$	Set of formats supported by D
$\mathcal{R}(c_k, D)$	Set of transcoding parameters leading to adapted content renderable by D
$\mathcal{R}^*(c_k, D)$	Set of optimal transcoding parameters leading to adapted content renderable by D
$\mathcal{Q}_V$	Visual quality
$\mathcal{Q}_T$	Transport quality

$\mathcal{Q}_{E}$	Quality of experience
$f^*(c_k, D)$	Optimal format
$z^*(c_k, D)$	Optimal scaling parameter
$QF^*(c_k, D)$	Optimal quality factor
$c_{k,i}^{f,z,QF}$	Adapted component of an adapted content $c_k^{f,z,QF}$
$\mathcal{A}(c_{k,i}^{f,z,QF})$	Area occupied by the component $c_{k,i}$ of the adapted content $c_k^{f,z,QF}$
$\mathcal{Q}_{I}$	Visual quality of image I
T_d	Total delivery time
$N_{B}(D)$	Network bitrate of D
$N_{L}(D)$	Network latency of D
$S_{L}(D)$	Server latency of D
$T_{L}(c_k^{f,z,QF})$	Transcoding latency of $c_k^{f,z,QF}$
α	Waiting time lower boundary
β	Waiting time higher boundary
$\hat{\mathcal{Q}}_{V}$	Estimated visual quality
$\hat{\mathcal{Q}}_{T}$	Estimated transport quality
$\hat{\mathcal{Q}}_{E}$	Estimated quality of experience
z_v	Viewing conditions
QF_{in}	Quality factor befor transcoding
QF_{out}	Quality factor after transcoding (or simply QF)
r	Relative file size

Ψ	Data size of the XHTML wrapper
$\hat{\mathbb{S}}$	Estimated file size
$\hat{r}$	Estimated relative file size
$\hat{r}_I$	Estimated relative file size of image I
I	Set of area (%) occupied by images
T	Set of area (%) occupied by text
$\mathcal{V}$	Set of Impress documents used in the validation
$\mathcal{W}$	Set of trancoded versions of the documents of $\mathcal{V}$
QE	Array that contains computed data
$\widehat{QE}$	Array that contains estimated data
$\hat{f}^*$	Estimated optimal format
$\hat{z}^*$	Estimated optimal scaling parameter
$\widehat{QF}^*$	Estimated optimal quality factor
$\tilde{z}$	Quantized value of the scaling parameter z
$\widetilde{QF}$	Quantized value of the quality factor QF
Δz	Variation in quantized values of the scaling parameter z
ΔQF	Variation in quantized values of the quality factor QF
f_i^*	Optimal format obtained by method i
z_i^*	Optimal scaling parameter obtained by method i
QF_i^*	Optimal quality factor obtained by method i
$\mathcal{R}_i^*$	Set of optimal transcoding parameters obtained by method i

z_I^*	Optimal scaling parameter obtained by interpolation
QF_I^*	Optimal quality factor obtained by interpolation
$\mathcal{N}_e^i$	Set of neighbours of the estimated point, the exponent represents their number
$\mathcal{N}_e^{LRDU}$	Set of neighbours of the estimated point, the exponent represents a search pattern
$W_{E,K}^*$	A vector of the optimal transcoding parameters obtained by the exhaustive static system
$W_{FQF,K}^*$	A vector of the optimal transcoding parameters obtained by the fixed-QF dynamic system
$W_{i,K}^*$	A vector of the optimal transcoding parameters obtained by method i
f_E^*	Optimal format obtained by the exhaustive static system
z_E^*	Optimal scaling parameter obtained by the exhaustive static system
QF_E^*	Optimal quality factor obtained by the exhaustive static system
f_{FQF}^*	Optimal format obtained by the fixed-QF dynamic system
$\mathcal{W}_T(D)$	Visual quality weight
$\mathcal{W}_V(D)$	Transport quality weight

INTRODUCTION

Motivation

Today, enterprise documents, such as Word documents, Excel spreadsheets, and PowerPoint presentations are widely used and shared among peers in many collaborative Web applications. With Google Docs (Google Inc., 2012a) or Zoho Show (Zoho Corp., 2012), for example, users are able to prepare presentation slides, and write, edit, and publish enterprise documents, and then share them using a computer equipped with a conventional Web browser and Web connectivity without the need to install any Office suite, such as MS-Office or OpenOffice. Technically, this has been facilitated on the one hand by the ubiquity of mobile Web browsers (used as standard client machines), and on the other by the emergence of the Web 2.0 technologies, which make it possible to mimic desktop applications on Web browsers. These technologies have also enabled new kinds of collaborative applications, such as hosting a meeting at which PowerPoint slides can be presented to all the participants. In short, meeting attendees no longer need to be located in the same room, or use sophisticated equipment or proprietary software.

A step forward in the use of the Web as a content delivery platform is the integration of mobile devices into the Web ecosystem, where they are replacing or complementing their desktop counterparts. It is now possible to create, edit, share, and publish enterprise documents using mobile devices equipped with Web browsers. This opens up a new usability perspective that liberates users from their traditional dependence on conventional desktop PCs, and enables them to participate in meetings and use documents collaboratively from almost anywhere.

In a collaborative environment, such as hosting a meeting involving PCs and mobile devices, the presentation slides should be shared synchronously to all participants. The obvious solution is to send the presentation (usually PowerPoint) to the participants before the meeting. However, many mobile terminals do not support the PowerPoint format (or any other office document format). Those that do support it still face the problem of downloading a potentially large document, often several MB in size. This operation is not only time-consuming, but drains the battery and is often costly. Finally, the crucial problem of synchronization with

the host remains, as the participants lose track of which slide is being presented. This causes serious usability problems.

Web technologies provide an attractive alternative, since they only require that the terminal be equipped with a (widely supported) browser and Web connectivity (wireless, mobile, or wired Internet). They can customize the content to each participant, and ensure constant synchronization with the presentation by sending a slide only when it is presented (slides are sent one by one). However, as new devices are continually being introduced into the marketplace, their number and diversity are steadily growing (Sudhir and Tao, 2004). These devices differ from one another in many ways: resolution, memory, supported formats, battery life, etc., and there is also great diversity in the communication networks they use. Depending on its location, a single device could use one of a number of different networks (e.g. WiFi, GPRS, UMTS, or LTE). This variety in mobile devices and communication networks has changed the logic behind Web content authoring, with the result that new content representation formats (e.g. XHTML, raster-based, video-based, etc.) have been designed, and new Web content formatting techniques have been introduced (Lum and Lau, 2003; Hwang *et al.*, 2003; Lum and Lau, 2005; Zhang *et al.*, 2006a). Depending on the nature of the target device and the circumstances of the end-user, Web content can be restructured or shrunk, its embedded images can be replaced by text, and text can be converted into audio tracks (if the end-user is driving, for example) (Sudhir and Tao, 2004).

Ultimately, all users, regardless of their terminals and network conditions, should be able to easily and transparently share enterprise documents and collaborate with one another. However, the heterogeneity of the terminals and the diversity of the communication networks make this goal very challenging. Some mobile Web applications, such as Nokia EasyMeet (Li and Chandra, 2008), use OpenOffice to convert PowerPoint documents into JPEG images that can be displayed in Web browsers. However, such formats lack interactivity and scalability. Other formats, such as SVG and Flash, lack support from terminals and cannot be used without installing the appropriate plug-ins. Besides, the communication networks in use have a direct impact on the quality of the content that can be delivered to the end-user within an acceptable

time frame. Before it is sent, the content is analyzed and then adapted to satisfy the constraints of the target mobile device (e.g. supported formats, resolution, memory) and the characteristics of the communication network (e.g. bitrate). When the content is requested, some data are captured from the user's request header, while other data are retrieved from specific databases, such as WURFL (Scientiamobile, 2011) to obtain a full description of the mobile device's characteristics, capabilities, constraints, and environment. Later, these data, which are referred to as *the context*, are used to carefully tailor the content. To achieve this, many context-aware systems have emerged to bridge the gap between technologically limited terminals and rich Web content. A comprehensive review and classification of such context-aware systems can be found in (Hong *et al.*, 2009).

Web content adaptation has been studied extensively and numerous solutions have been proposed. However, in spite of the ubiquitous use of enterprise documents, adapting them has not attracted the same attention as audiovisual adaptation, and their portability to mobile Web browsers and their interoperability are still a challenge. Globally, two major trends have emerged in content adaptation: static adaptation, and dynamic adaptation. A great deal of work has been done in the area of static content adaptation (Noble *et al.*, 1995; Mohan and Smith, 1999; Lum and Lau, 2003; Jan *et al.*, 2006; Zhang *et al.*, 2006a), where different versions of the original content are created and stored on a server. At runtime, when the content is requested, the optimal of these versions, based on a given quality criterion, is selected for delivery. In dynamic content adaptation, also called just-in-time or online content adaptation, a customized version is created on-the-fly, based on the contextual data gathered (Kitayama *et al.*, 1999; Han *et al.*, 1998; Hwang *et al.*, 2003).

There are advantages and disadvantages to using either of these two strategies. The static approach leads to high processing complexity for generating all the versions and a great deal of disk space to store them. In order to avoid huge response times when the content is requested, the processing is often performed offline. In this case, the issue of granularity becomes important, as it determines the compromise between the quality of the delivered content (the more versions are available, the better the quality) and the associated processing complexity, as well

as the storage space available (Lum and Lau, 2002). With the dynamic strategy, the content adaptation is performed on-the-fly, when the terminal's context is known, while the end-user waits. This is not a straightforward task, as the time required to adapt and deliver the content is extremely important. In this case, the server could easily be overwhelmed with a large number of requests, which would negatively affect the end-user's experience (Chandra and Ellis, 1999; Lum and Lau, 2002). In this thesis, the problem of dynamic adaptation of enterprise documents is studied, and particular attention is given to slides, which constitute the most widely used mode of information presentation in Web conferencing environments.

Problem statement

In Web conferencing, Web 2.0 is widely used as a platform for delivering enterprise documents to terminals and ensuring uninterrupted synchronization between the presenter and the participants. In the PC context, the Google and Zoho platforms convert the slides into specific formats (e.g. Google Docs or Zoho Show) and embed them in Web pages, so that they can be visualized on the participants' Web browsers and synchronized with the presenter's delivery of the slides (Google Inc., 2012c). Current mobile Web applications must also adapt the content for specific mobile device types (Google Inc., 2011, 2012b). For instance, EasyMeet (Li and Chandra, 2008) converts PowerPoint slides into images that can be displayed on the Web and on mobile Web browsers. In this research, we propose to use the same approach, which is to adapt enterprise documents and wrap them in Web pages.

In terms of which format to use for adapting the enterprise documents, we focus on JPEG and XHTML. The former is a required format, since it is widely supported by mobile devices and used in so many Web applications (e.g. Nokia EasyMeet and Google Docs). However, this is a raster format and so lacks interactivity and scalability. Consequently, we consider the use of XHTML as well, which is more flexible, in that it allows text editing and keyword searching. Technically, when JPEG is used, an entire document page is converted into a JPEG image and wrapped in a Web page skeleton. In contrast, when XHTML is used, that page's embedded components are converted separately, and individually wrapped in a Web page. Since presentation slides are mostly composed of text and images, the embedded images are converted

into JPEG images and the text elements are resized, which leads to the creation of a conventional Web page, comprising both text and images. As a result, we consider only slide decks composed of text and image components here.

Now, to adapt the document dynamically (on-the-fly), transcoding parameter values must be determined. These comprise the selected output format (JPEG or XHTML), the resolution scaling parameter, and the JPEG quality factor. Most of the existing dynamic solutions use the mobile device's resolution as the target image resolution to determine the scaling parameter and a quality factor value between 75 and 80. Although such solutions provide good visual quality, they don't control the resulting file size, which has a direct impact on the delivery time and usability of the adapted content. Under certain network conditions, the user might even lose interest in that content, owing to an unreasonable wait time. A better user experience can often be achieved by reducing the visual quality slightly, in order to permit a shorter delivery time.

Dynamically determining the optimal transcoding parameter values that maximize the end-user's experience with minimal computation, while at the same time satisfying the target mobile device's constraints is a very challenging problem. To solve it, we first need to define an objective measure that is representative of the user's experience, and then determine the transcoding parameter values that will maximize this measure without requiring too much computational complexity. Although performing an actual transcoding operation for each combination of transcoding parameter values and identifying the combination that maximizes the defined measure is theoretically feasible, it is too complex to implement in a viable communication system. This is because the number of combinations of transcoding parameter values grows exponentially with the number of parameters, and because the adaptation itself is performed online while the end-user waits. Such an exhaustive approach would require far too much computational complexity to be a viable solution. While a certain number of transcoding operations is required for the adaptation, that number should be minimized if at all possible (Mohan and Smith, 1999), (Lum and Lau, 2005), (Jan *et al.*, 2006). With any solution to this problem, we can expect that compromises will need to be made between the computational complexity per-

mitted in searching for the optimal parameter values and how far from the optimal that solution will be.

In this thesis, we propose a novel dynamic adaptation framework that enables the estimation, on-the-fly, of near-optimal transcoding parameters (format, scaling parameter, and quality factor), evaluated under a novel quality of experience (QoE) measure. We first define a QoE measure to quantify the quality of the adapted content as experienced by the end-user. This measure takes into account the visual aspects of the content, as well as its transport quality, which is mostly affected by the network conditions. We also propose an adaptation framework capable of dynamically selecting, with very little computational complexity, the optimal adapted content that achieves the best compromise between visual quality and delivery time (transport quality), based on the proposed QoE measure. The proposed framework includes a novel component, the media characteristics predictor, which is capable of predicting some media attributes as a function of the transcoding parameters. Specifically, in the proposed framework, we reuse predictors of the file size and visual quality of JPEG images, subject to changing their scaling parameter and quality factor, as proposed in recent research (Pigeon and Coulombe, 2008; Coulombe and Pigeon, 2009, 2010; Pigeon and Coulombe, 2011). This novel framework has been applied, but is not limited to, OpenOffice Impress presentations. It is designed to be general enough to be applied to other enterprise document types, such as MS Word (text) and MS Excel (spreadsheet).

Objectives of the research and contributions

The main objective of this research is to design a novel dynamic content adaptation framework to enable mobile devices to use and share enterprise documents dynamically, as is the case in PC Web conferencing environments. The content is adapted and delivered on-the-fly, while the end-user waits online. Mobile devices need only be equipped with Web browsers and Web connectivity (wireless, mobile, or wired Internet), that is, no extra software is required at the client end (mobile device). In this way, we can target a wide variety of mobile devices.
The main objective was broken down into the following specific objectives:

- Create a novel architecture for the adaptation of enterprise documents for mobile devices.
    - Build the architecture around a novel media characteristics prediction unit capable of predicting some media attributes as a function of the transcoding parameters.
    - Build an architecture that is highly efficient computationally (achieve a high level of quality with very few computations).
- Propose a novel quality measure to quantify the quality of the adapted content as experienced by the end-user.

Our scientific contributions include the following:

- A near-optimal dynamic content adaptation framework applied to enterprise documents. The dynamic aspect of the proposed framework is based on the prediction of the adapted content quality prior to transcoding.
- Low-complexity prediction models and methods leading to more nearly optimal adapted content quality.
- A QoE measure that takes into account the visual quality of the adapted content, as well its transport quality, which is affected by the adapted content file size and the conditions of the communication network in use.
- Application of the proposed framework to presentation slide decks, and subsequent validation on these decks, which are mostly used in Web conferencing applications.

Organization of the thesis

The thesis is organized as follows. In chapter 1, we present a brief overview of the Web 2.0 technologies and Web-based applications, as well as their mobile counterparts. In chapter 2, we explain the problem of content adaptation for handheld devices. A literature review of the various techniques, methods, and algorithms developed to adapt content to mobile devices, along with their strengths and weaknesses, are covered in chapter 3. In chapter 4, we present a full description of the architecture supporting the proposed prediction-based dynamic content

adaptation framework. Chapters 5 and 6 detail our methods and algorithms, which are based on the prediction of the optimally adapted content (which is expected to be near-optimal). A set of methods and models for identifying optimally adapted content is proposed, with improved performance and lower computational complexity. Finally, we summarize the scientific contributions that we make with this research in chapter 7.

CHAPTER 1

MOBILE WEB 2.0 TECHNOLOGIES AND ENTERPRISE DOCUMENTS

In this chapter, we present a brief overview of the Web 2.0 technologies and their mobile counterparts (mobile Web 2.0). Special attention is given to the Web and mobile Web applications, their evolution, scope, problems, and features. In addition, the issue of collaboration in heterogeneous contexts involving PCs and mobile devices is outlined, and some known solutions are given. At the end of the chapter, the concept of enterprise documents is introduced, which, in this research, is the content to be adapted to mobile devices.

1.1 Web 2.0 and mobile Web 2.0

Web 2.0 is the second generation of the Web (Lee and Chun, 2007). It's emphasis on interactivity, content sharing, and collaboration using the Web as a platform. The traditional perception of the Web, which is better known as Web 1.0, is that it is limited to providing information that has already been published. In other words, Web browsers have been seen for a long time as a standardized client machine integrated into a huge client-server architecture. Today, this perception is drastically different, and the boundaries between the client side and the server side are now blurred. The user, traditionally a consumer of information, has become a server as well, and is ready to contribute to the richness of the Web. Many applications reflect this reality, such as *wikis, blogs, P2P applications*, etc., and, among the applications based on the concept of Web 2.0, we can cite the following: *Wikipedia*, *LinkedIn*, *Twitter*, *Facebook*, etc. Figure 1.1 gives a graphical illustration of this evolution.

Web 1.0 was initiated in the mid-1990s, and was used only for static Web content at that time. A few years later, dynamic Web content, which returns custom Web pages to users, was introduced. The end of the 20th century was characterized by interactive Web content, referred to as Web 1.5. During this period, the Web allowed users to interact with audio and video content, in the same way as it is possible to interact with real world audio and video stations. Recently, with the emergence of Web-based applications, which mimic PC applications, a new

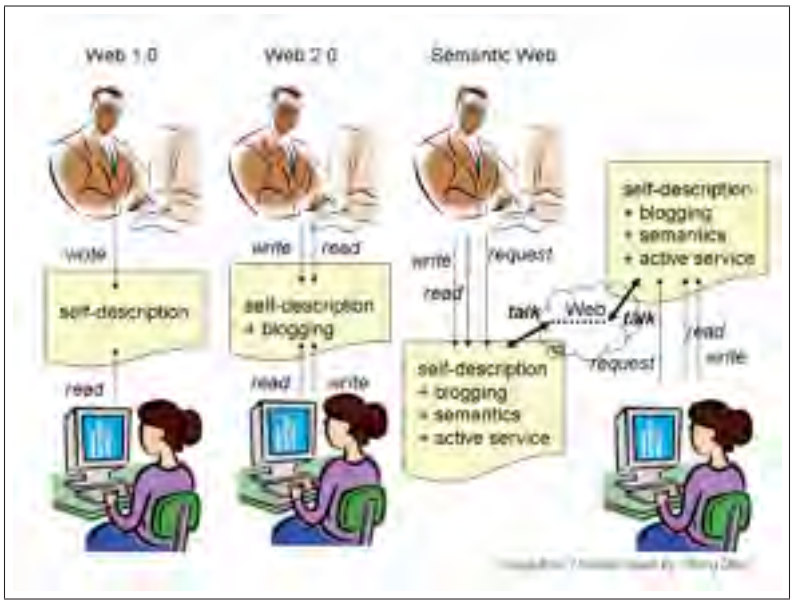

Figure 1.1 Evolution of the Web (Extracted from (Ding, 2007)).

era of the Web emerged, Web 2.0 (Kern, 2008). An interesting example demonstrating the evolution of the Web is the comparison between *Google* and *Twitter*. Google relates to what people have done so far, while Twitter is about what people are doing now. This evolution is so important that researchers have begun theoretical studies designed to formally define Web 2.0, its characteristics, and its trends (Ding and Xu, 2007a,b,c).

According to "The Free Dictionary" (The Free Dictionary, 2012a), the term Web 2.0 was coined by Darcy DiNucci in 1999 in her article *Fragmented Future* (DiNucci, 1999), but it usually more closely associated with Tim O'Reilly, based on the O'Reilly Media Web 2.0 Conference held in 2004 (O'Reilly, 2005). Darcy DiNucci defined the Web 2.0 as follows: *The Web we know now, which loads into a browser window in essentially static screenfuls, is only an embryo of the Web to come. The first glimmerings of Web 2.0 are beginning to appear, and we are just starting to see how that embryo might develop. . . . The Web will be understood not as screenfuls of text and graphics, but as a transport mechanism, the ether through which interactivity happens. It will [. . .] appear on your computer screen, [. . .] on your TV set, [. . .] on your car dashboard, [. . .] on your cell phone, [. . .] on handheld game machines [. . .], and maybe even on your microwave*.

"The free dictionary" defines Web 2.0 as a combination of user-generated content and the technologies currently in use (The Free Dictionary, 2012b): *An umbrella term for the second wave of the World Wide Web, which was coined in a conference on the subject in 2004 by O'Reilly Media and CMP Media (later taking its parent name of "United Business Media"). Sometimes called the "New Internet", Web 2.0 is not a specific technology, but rather an amalgam of two major paradigm shifts. The one most often mentioned is "user-generated content", which relates more to individuals than to businesses. The other, which is equally significant, but more related to businesses, is "cloud computing".*

1.2 Web 2.0 technologies

The minimum set of technologies used in recent Web-based applications is composed of XHTML (W3C, 2002), JavaScript (Flanagan, 2006), and CSS (W3C, 2012). XHTML is a strict, well formed version of HTML inspired by the strictness of XML, which is a tag-based language that can be understood by Web browsers. JavaScript can be added to XHTML to obtain a more interactive application, but this interactivity is limited to local interactions, such as verifying the date format or guaranteeing that all the required fields of a form are filled in. CSS is used to separate the content from formatting styles, such as bold, italic, and the like. A more interesting technology, and the most recent one used in Web-based applications is AJAX (Asynchronous JavaScript and XML) (Garrett, 2005), which is based on JavaScript. AJAX has brought another level of interactivity to Web-based applications, which is not local, but between the Web browser and the server. In fact, AJAX is not a technology as such, but a term that represents a collection of existing Web technologies (O'Donoghue, 2006). When they are used together, they create a new generation of interactive and responsive applications. Google Maps and Gmail, for instance, are pioneers in the utilization of AJAX (O'Donoghue, 2006), which allows interaction with parts of an XHTML document, instead of the traditional page reload technique. When a part of the Web document needs to be updated, it is not necessary to reload (update) the whole page, but only that part. This reduces the amount of information exchanged between the server and the Web browser. In the mobile context, where bandwidth is limited, the use of AJAX could result in a significant gain. Mobile AJAX is the application of

AJAX to mobile devices equipped with Web browsers. Moreover, Mobile AJAX can be used to create mobile widgets, which are, in fact, interactive Web applications. In contrast to traditional widgets, mobile widgets don't need to be installed on the mobile phone, but are used on the mobile Web browser. In order to do so, however, AJAX functions should be supported by the mobile Web browser, which is not always the case. For instance, the following mobile Web browsers support AJAX at some level (O'Donoghue, 2006):

- Opera Mobile ($\geq$ 8.x, not Opera Mini)
- Internet Explorer Mobile (WM 5.0/2003)
- S60 3^{rd} edition (WebKit/KHTML core)
- Minimo (Gecko-based)
- OpenWave ($\geq$ Mercury)
- NetFront ($\geq$ 3.4)
- Safari Mobile (iPhone)

1.3 Beyond Web 2.0

For many, the next generation of the Web is Web 3.0. The definition and characteristics of Web 3.0 have not yet been clearly defined, but it is currently being perceived in various ways. Some of these are based on the evolution of bandwidth, others on how Web applications will be developed (The Free Dictionary, 2012c). This evolution will blur the distinction between simple consumers of Web applications and professional and semi-professional developers (The Free Dictionary, 2012c). Still others perceive Web 3.0 as the realization and extension of the semantic Web, and possibly the convergence of the semantic Web and SOA (*Service Oriented Architecture*) (Spivack, 2006).

According to Nova Spivack (Spivack, 2006), Web 3.0 can be defined as follows: *There are actually several major technology trends that are about to reach a new level of maturity at the same time. The simultaneous maturity of these trends is mutually reinforcing, and collectively*

they will drive the third-generation Web. From this broader perspective, Web 3.0 might be defined as a third-generation of the Web enabled by the convergence of several key emerging technology trends.

Spivak classifies these trends as follows:

- Ubiquitous connectivity
    - Broadband adoption, mobile Internet access, and mobile devices.
- Network computing
    - Software-as-a-Service business models, Web services interoperability, and distributed computing (P2P, grid computing, hosted "cloud computing" server farms, such as Amazon S3).
- Open Technologies
    - Open APIs and protocols, Open data formats, Open source software platforms, and Open data (Creative Commons, Open Data License, etc.).
- Open Identity
    - Open identity (OpenID), open reputation, portable identity and personal data (for example, the ability to port a user account and search history from one service to another).
- The Intelligent Web
    - Semantic Web technologies (RDF, OWL, SWRL, SPARQL, Semantic application platforms, and statement-based datastores, such as triplestores, tuplestores, and the associative databases).
    - Distributed databases, or what he calls "The World Wide Database" (wide-area distributed database interoperability enabled by semantic Web technologies)
    - Intelligent applications (natural language processing, machine learning, machine reasoning, autonomous agents).

1.4 Web-based applications

Web-based applications, also called *Webapps*, are applications that use the Web browser as a client. They are accessed over a network, mostly the Internet or the Intranet. They could also be desktop applications adapted for use on Web browsers. The ubiquitous nature of Web browsers is behind the success and popularity of Web-based applications (Potter and Nieh, 2005). Since the Web browser is widely used as a client application, Web-based applications are preferred to conventional applications (desktop), which are customized to each client and context of use. It is easy to imagine the huge number of users targeted by particular content, and the problems of customizing it and maintaining the client-side applications that may be needed. These problems can be avoided by using Web browsers.

Web-based applications are platform-independent. They are coded once, and deployed to be used on any platform (Window, Linux, Mac OS, etc.). Nevertheless, some details regarding the implementation of Web browsers should be considered. Since they are not implemented by the same vendor, developers may encounter certain differences when they code Web-based applications for different Web browsers. In fact, the well-known standardized technologies, such as XHTML, CSS, JavaScript, and DOM, are implemented differently from one vendor to another (Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, etc.), and so Web-based applications are developed for more than one type of Web browser. When the application is loaded by the Web browser, the application detects and identifies the browser, and uses the corresponding code.

1.5 Mobile Web-based applications

A great deal of research has been conducted to bring Web content to mobile devices (mobile phone, PDA, smartphone, etc.) (Mohan and Smith, 1999; Lum and Lau, 2003, 2005). This process has not been a smooth one, nor has it been automatic, since Web content was originally developed for PCs and laptops with large screens, a keyboard, a mouse, no battery constraints, etc. In addition, technically, Web content data structures were designed only for desktop platforms. Some researchers have tried to adapt Web content to the mobile context

by splitting it into small, logical pieces, and determining the relationship between them (Lum and Lau, 2005). Depending on the screen size of the mobile device, those pieces are visualized either separately or together. Other research has been based on the Web content author's intentions (Kim *et al.*, 2007; Chen and Ma, 2005), which are captured and used to subdivide the content into small, logical blocks that are restructured to be visualized on the mobile device according to the author's intentions and the mobile device's screen size.

The situation can be even more challenging when the Web content is shared and used in a collaborative way by users working on desktops and mobile devices at the same time. The visual and logical organization of the content is not the same on desktops and mobile devices. This is the issue that underlies the problems encountered in content co-browsing and collaborative applications.

1.6 Collaboration in Web-based applications

In Web-based collaboration, two or more users wish to share Web objects to pursue a common goal (Chua *et al.*, 2005). This can be achieved by what is called co-browsing (also called "shared browsing" or "escorted browsing"). In this scenario, users connected using different terminals are able to navigate Web pages and communicate with one another using an audio link or a text chat application (Chua *et al.*, 2005). Several co-browsing solutions, such as MS NetMeeting (Microsoft, 2004) and IBM WebDialogs (IBM, 2007) have been proposed and are reviewed in (Esenther, 2002). The problem with these solutions is that they are designed to be used by conventional terminals with large screens, such as PCs and laptops.

In a heterogeneous environment, where mobile devices and desktop clients participate in a co-browsing session, the Web content should be adapted to be visualized on the mobile devices. This means that the Web content version visualized on the mobile device will typically be different from the original one. Therefore, to ensure a full collaboration between users, the original content, visualized on the desktop client, should be augmented by an extra view of what is visualized on the mobile device. This can be achieved by dividing the screen into two

frames (SVP: *Shared View point* and PVP: *Personal View Point*). In this way, users can refer to the SVP for co-browsing and PVP for browsing other content (Chen *et al.*, 2001).

1.7 Enterprise documents

Since this research focuses on enterprise documents and their adaptation to mobile devices in a collaborative manner, it is important to define them and show how they are used. Enterprise documents are documents produced by office suites, such MS Office and OpenOffice, and widely used in a wide variety of companies and enterprises. At a high level, an enterprise document could be a letter or report (created using Word, for example), a spreadsheet (created using Excel, for example), or a presentation (created using PowerPoint, for example). Formally, documents can be divided into two categories: enterprise documents (.doc, .ppt, and .xls), and social media (audio, video, and interactive graphics). However, an enterprise document could be richer if it contained the so-called social media. For instance, a Word document could be composed of text, images, graphics, audio tracks, and video streams.

At the back end, OpenOffice and MS Office (the 2007 version and later) use an XML-based format (OOXML: office open XML) to encode documents. This kind of encoding makes enterprise documents more interoperable and enhances their portability. In the mobile context, enterprise documents could be created by one of the portable office suites, such as Quickoffice [1] or MS Office mobile 6.1 [2]. These are installable solutions, and so they require a certain degree of capability.

Recently, non installable Web-based solutions, such as Google Docs and Zoho Show, have emerged. They require only a Web browser and Web connectivity (wireless, mobile, or wired Internet). However, these solutions are still immature and not as reliable as the installable ones. Nevertheless, they are a good alternative to the installable solutions, especially in a collaborative meeting scenario, where a document is shared and presented synchronously to the meeting participants. Furthermore, mobile versions of these Web-based applications have been

[1] Quickoffice: http://www.quickoffice.com/

[2] MS-Office mobile: http://www.microsoft.com/windowsmobile/en-us/downloads/microsoft/software-office-mobile.mspx

created to integrate the mobile devices into the Web-based collaborative applications space, although, again, they lack maturity, and considerable research will be needed to address this issue. In fact, the Zoho Show mobile version, *iZoho*, has been developed for the iPhone and the iPod touch only. Besides, in general, the possibility of editing content on a mobile device is still not widely available, even for these targeted mobile devices.

CHAPTER 2

CONTENT ADAPTATION FOR HANDHELD DEVICES

Web content can be accessed by PCs and laptops, but also by mobile devices (smartphones, PDAs, handheld terminals, etc.), which have different characteristics. This diversity is growing ever greater with the introduction of new devices onto the market (Sudhir and Tao, 2004). Mobile devices differ from one another in many ways: display size and resolution, processing power, user interface (navigation buttons are not always available), color capabilities, etc. (Sudhir and Tao, 2004). As a result, Web content typically needs to be adapted to make it usable by the various types of mobile devices and to improve the end-user's experience.

Not only are the device's capabilities key to its usefulness, but the network to which it is connected, characterized by the bitrate, is important as well. The same device could use different networks, depending on its location (WiFi, Bluetooth, GPRS, or UMTS). However, content adapted for use by a mobile device connected to a 54 Mbps WiFi network could not be accessed by the same device when it is connected to a 40 kbps GPRS network. Consider, for example, a service that delivers the weather forecast to mobile devices (Sudhir and Tao, 2004). This service could deliver a simple SMS text to an older GSM phone. Phones that support images could receive an improved version of the service, such as a combination of images, text, and possibly animated images. The latest phones, which support images, video, and audio, could receive the service as a video clip or as streamed video, just like a live weather forecast on television.

Sometimes user preferences are taken into account in the content adaptation process, given that the user's perception usually differs from person to person. For example, the visual quality of the adapted content (e.g. color, sharpness) may be important to one user, and resolution to another (Lum and Lau, 2003). All this information taken together (device capabilities, network characteristics, and user preferences) constitutes the context for which the content adaptation process is performed.

In this chapter, we first review the various media types that Web content can comprise, and how they can be transformed. Then, we describe the landscape of the content representation formats, and the advantages and disadvantages of each. Finally, we discuss the strategies for content adaptation commonly applied, in terms of their features and their problems.

2.1 Multimedia content types

To be adapted, the content first needs to be classified in a category, based on its characteristics, since every content type is encoded and adapted differently. Globally, there are two types of media content: textual, and audiovisual.

2.1.1 Textual content

Text is the most basic type of content found on Web pages. The representation of text can be problematic, however, owing to the diversity of international representation encodings, such as ASCII and Unicode (UTF-8 and UTF-16) (Sudhir and Tao, 2004). The text encoding used for the Chinese language, for example, is different from that used for European languages. Although the latter originated from Latin and share the same basic alphabet, they do not use the same set of characters. For instance, the German language uses ä and the Swedish language uses å, which are unfamiliar to English speakers. That's why it's important to use the text encoding appropriate for the source language. Like any media, text can be transformed into other formats to meet particular constraints. For instance, text can be converted into speech for visually impaired users or for users on the move (e.g. driving) (Sudhir and Tao, 2004; Soo-Chang and Yu-Ying, 2011). Conversely, video streams can be transcoded into descriptor text and added to the video as annotations (Cavallaro *et al.*, 2003). Even the markup language used to encapsulate the content, such as HTML, can be converted into the markup language supported by the target mobile device (e.g. WML).

2.1.2 Audiovisual content

Audiovisual content is composed of both audio and visual components, including images, video, graphics, music, and speech (Sudhir and Tao, 2004). It is deliberately intended to be

rich, in order to improve the user's experience. It is usually compressed, because of its large file size, which could raise storage and transmission issues. The most popular image formats on the Web are JPEG (Joint Photographic Experts Group, 2007) and GIF (W3C, 1987), the former suitable for natural and photographic content, the latter best for graphics, shapes, and icons. Video content can be compressed into the H.263, H.264 (International Telecommunication Union, 2008), and MPEG-4 (The Moving Picture Experts Group (MPEG), 1988) formats. Audio content can be compressed into the MPEG-2 and MPEG-4 formats.

2.2 Formats and adaptation tools used for enterprise documents

The great variety of terminals in use has led to the creation of several formats to represent content (Chebbine *et al.*, 2005), and each has its advantages and drawbacks. In fact, different formats are suitable for different contexts. The decision as to what format to use in our case (adaptation of enterprise documents to be used on Web browsers) depends on an accurate analysis of the landscape of the content representation formats, including their characteristics, strengths, and weaknesses. In addition, we review how enterprise documents can be adapted into these formats, and the tools and techniques that can be used for this purpose.

2.2.1 HTML/XHTML representation

HTML is a widely used language, as it can be understood by the majority of Web browsers. It is one of the basic languages used in Web applications, along with JavaScript and CSS (*Cascading Style Sheets*). The fact that these languages are widely used and supported by Web browsers is a major advantage, as it guarantees a good level of interoperability. Adapting enterprise documents (e.g. PowerPoint) into HTML code can be achieved in two ways:

1. Using specific content editing tools, such as: S3 (Meyer and Meyer, 2012) and HTML Slidy (Raggett, 2010). These tools have their own JavaScript and CSS libraries. The Web version of the presentation (HTML/XHTML code) must be created from scratch using these libraries. The latter can be included in the presentation code or simply referenced by including their HTTP address in the presentation's header. Any HTML editor can be used

to achieve this, such as AMAYA (W3C, 2012b), which is available free of charge. So, the Web version of the document is created like any conventional Web application. The advantages and drawbacks of these tools are the following:

- Advantages:
  - No transformation is needed.
  - They function on any Web browser.
- Drawbacks:
  - Each presentation must be created from scratch.
  - Existing presentations are not reusable.

2. Transformation of the document's content into HTML code using specific filters. Filters for presentations can be found in Office suites, such as OpenOffice and MS Office (using "Save as", choose the desired output format: HTML, XML, RTF, etc.) (Microsoft, 2003). The advantages and drawbacks of these tools are the following:

- Advantages:
  - The transformation can be achieved from existing documents.
  - Reusability is possible.
- Drawbacks:
  - The transformation, using these filters, is not fully reliable, especially when the document contains figures, forms, animations, etc.
  - There is no interactivity with the content.
  - Annotation could be an issue.

2.2.2 Raster-based representation

Raster-based formats are used to represent images as an array of pixels. A widely used adaptation method in Web-based collaborative applications consists of converting a document into a raster-based format, such as JPEG or GIF. Various tools can be used to achieve this, such

as Zamzar (Zamzar, 2012) and ”Free File Converter” (Converter, 2012). Also, as explained in the previous section, Office suite filters can be used to convert enterprise documents into raster formats (using “Save as”, select the target format: JPEG, GIF,. . .). The advantages and drawbacks of using this technique are the following:

- Advantages:
  - The transformation is simple to describe.
  - The format is supported by the majority of Web browsers, if not all of them.
- Drawbacks:
  - All the functionalities related to interactivity, animation, and annotation are not supported.
  - The content cannot be resized dynamically without affecting its perceptual quality.
  - Since raster formats are raw pixels (not raw text), embedded text quality is usually degraded.
  - The output file size grows with increasing resolution.
  - The computational complexity grows with increasing resolution.

2.2.3 Video-based representation

Video-based formats are used to encode sequences of images that represent scenes in motion, such as movies, video-clips, and even presentations. Technically, the document pages (or slides) are converted into raster images, which are in turn converted into video streams with a lower bitrate for the same quality. Actually, with this conversion, spatial and temporal redundancy can be found in the images, and so they are compressed into video formats to reduce their file sizes. The H.264 video compression standard can be used as an output format. Various tools exist to adapt enterprise documents, such as “E.M. PowerPoint Video Converter” (EffectMatrix Ltd., 2011) and “Xilisoft PowerPoint to Video Converter” (Xilisoft Corp., 2012). Using these tools, PowerPoint documents can be converted into various video

formats; H.264, MPEG-4, etc. The advantages and drawbacks of this technique are as the following:

- Advantages:
    - Low bitrate compression is a major benefit for wireless networks.
    - Video formats are increasingly supported by mobile devices.
- Drawbacks:
    - These formats have the same drawbacks as raster formats.
    - The computational cost of video encoding/decoding could be high, and it could require more processing time.
    - Unless each frame is intra coded (in which case, the compression ratio is low), to access and decode a specific inter frame, we need to decode all frames before it starting at its previous intra frame.

2.2.4 Rich media-based representation

This format consists in embedding more than one media component into a single representation. The most popular formats to consider in this representation are: SVG (*Scalable Vector Graphics*) (W3C, 2012a) and Flash (Adobe Systems Incorporated, 2012), which are based on, but not limited to, vector graphics, and can include images, as well as audio and video clips. This representation opens up another level of interactivity, which is based on content scalability. Independently of screen resolution, the content can be visualized using the zoom and pan operations. In addition, advanced graphics features are supported, such as gradients and opacity. Complex animations and synchronization with audio and video formats are supported as well.

1. SVG is an XML-based open format and the only alternative to Flash. Enterprise documents can be adapted to SVG using some free tools, such as the Chinook (SourceForge, 2009) and OpenOffice filters, and some that are not free, such as SVGmaker (Software

Mechanics Pty Ltd., 2011), and docPrint (VeryPDF Knowledge Base, 2012), which is a printer driver. The advantages and disadvantages of SVG are the following:

- Advantages:
  - It is an open format and can be used free of charge.
  - Its code is open-source.
  - The fact that it's based on XML makes it interoperable with other tag-based languages, such as HTML.
  - It is natively supported by the majority of Web browsers, such as Mozilla and Safari.
- Drawbacks:
  - It is not supported by all Web browsers, and requires the installation of appropriate plug-ins.
  - It has a limited number of content creation tools (W3C, 2012a).

2. Flash is a proprietary product belonging to Adobe Inc. A variety of tools can be used to convert enterprise documents into Flash format. For instance, "VeryDOC DOC to Any Converter" (VeryDOC, 2012) and iSpring (iSpring Solutions, Inc., 2012) can be used to achieve this. Note that OpenOffice filters can be used as well. The advantages and drawbacks of using Flash are the following:

   - Advantages:
     - It is a very popular and widely used format.
     - A variety of tools can be used to transform a document into Flash (Adobe Systems Incorporated, 2012).
   - Drawbacks:
     - It is a proprietary format, and so using it requires a licence.
     - Its code is closed-source.
     - It is not extendible.

- It is not natively supported by Web browsers, and so requires the installation of the appropriate plug-in.

2.2.5 Native viewer's representation

The idea behind this representation is the potential of using the content without prior transformation. To make this possible, the Web browser must be able to render the content accurately. Technically, the Web browser must support the appropriate plug-in for any type of content (one for MS Office documents, one for OpenOffice documents, etc.). However, in practice, only a limited number of formats are supported via plug-ins. For instance, a Word plug-in for Safari to render Word documents on the Safari Web browser can be found at (Schubert, 2010). The advantages and drawbacks of this technique are the following:

- Advantages:
  - The content is rendered in its original format.
  - There is no content degradation, so the content quality is preserved.
  - No transformation effort is needed.
- Drawbacks:
  - A limited number of formats are supported by Web browsers.
  - There are very few plug-in APIs to interact with the content.

2.2.6 Web-based representation

The current trend is to use this representation. It enables individuals to create and share documents (e.g. PowerPoint, Word, Spreadsheets) online. The user can connect to the application using the Web browser, and create, edit, modify, save, and share his documents with others, and do so without installing any software or adding any Web browser plug-in. The most popular Web-based solutions are Google Docs (Google Inc., 2012a) and Zoho Show (Zoho Corp., 2012). With Google Docs, for example, it is possible to create, edit, share, and publish office

documents. But, complex graphics are not supported. Also, existing documents (e.g. a .ppt document) cannot be imported. Zoho, by contrast, does support the creation, editing, and sharing of Office documents. Graphics are supported by Zoho, too, but that support is limited to simple shapes. Unlike Google Docs, Zoho Show supports the importing of existing documents without providing any interaction possibilities. Some of the advantages and disadvantages of this category of Web-based solutions are the following:

- Advantages:
  - Widely supported Web browsers can be used.
  - There is no need to install any additional software.
  - There is no need to install any Web browser plug-ins.
- Drawbacks:
  - Some Web technologies must be supported by the Web browser, such as JavaScript and Ajax.
  - Web-based solutions are still young and immature.
  - These solutions don't offer all the functionalities provided, at least currently, by an Office suite.

2.2.7 Summary of content representation formats

A summary of the advantages and drawbacks of the content representation formats and adaptation techniques mentioned above can be found in Appendix A.

2.3 Adaptation of the various media content types

2.3.1 Adaptation of textual content

Nowadays, Web content is mostly XML-based. Even content that is not pure XML, such as XHTML, WML, RDF, etc., can be easily parsed and transformed into XML. The transformation from or to XML can be achieved either by XSLT engines or programmatically using

the DOM or SAX technologies (Sudhir and Tao, 2004). The XSLT engine takes the original document and transforms it into any desired format using the XSTL StyleSheet, which is a set of templates that tells the XSLT engine what to do when it encounters a specific item in the original document. The XSLT StyleSheet includes a set of rules (also called templates) that drives the transformation process. For instance, an OpenOffice document, which is XML-based, can be transformed into a text document by extracting only the textual content using an XSLT engine and the appropriate templates. DOM interfaces, by contrast, are less abstract, since they manipulate the inner structure of the XML document and convert it programmatically. Sometimes, the document is too big and cannot be handled by an XSLT engine, since this technology loads the whole document to be parsed into memory. To solve this problem, the SAX technology can be used. SAX does not load the document, but converts it into streams which are sent to a program to be transformed. However, it is not possible to obtain a view of the whole document using this technique. So, XSLT can be seen as providing a higher level of abstraction and SAX a lower one, while DOM represents a tradeoff between the two (Sudhir and Tao, 2004).

2.3.2 Adaptation of audiovisual content

Generally, audiovisual content is adapted by decoding the content into its original format and re-encoding it into the desired format by selecting the right parameters (Sudhir and Tao, 2004). For instance, GIF images can be decoded into RGB (red, green, blue) format to be re-encoded into JPEG format. The audiovisual content can be adapted to achieve resolution reduction, quality reduction, color depth reduction, audio sampling, etc. It is important to note that audiovisual content adaptation is very computationally intensive, and very often the process is iterative (to reach a given file size, for example). However, numerous techniques have been developed to optimize this process.

In fact, this content can be adapted either in the spatial domain (the content is fully decoded) or in the compressed domain (the content is partially decoded). The first approach consists of decoding the content into the pixel domain, adapting it, and re-encoding it into the desired format and/or using the desired transcoding parameters. In the second approach, the complex-

ity of the uncompressing and recompressing operations is reduced, in order to speed up the adaptation process. This is achieved by partially manipulating the content in the compressed domain, for example: manipulating the DCT or RLE blocks in JPEG images and H.264 video streams (Smith and Rowe, 1993; Chang and Messerschmitt, 1995; Merhav and Bhaskaran, 1996; Shahabuddin *et al.*, 2009). Note that decoding audio content is computationally less intensive than decoding video content. Wang et al. (Wang *et al.*, 2003) present a survey of the various compressed domain features that can be manipulated in audiovisual content.

Furthermore, a great deal of research has been conducted to predict the file size and/or quality of the adapted content prior to transcoding. In the compressed domain, the quality of JPEG images can be predicted (Tsai and Zhang, 2009) as a function of the target quality factor. Han et al. (Han *et al.*, 1998) propose a method to predict the file size of JPEG and GIF images in the spatial domain. Also, the quality and file size of JPEG images, subject to varying their quality factor and scaling, can be predicted (Pigeon and Coulombe, 2008; Coulombe and Pigeon, 2009, 2010; Pigeon and Coulombe, 2011). By predicting such information, the iterative complexity of the content adaptation process is significantly reduced, if not eliminated.

2.4 Adaptation strategies

Depending on the resources available on the server side and the nature of the content to be adapted, two strategies can be used: dynamic, and static. Each of these strategies has its advantages and drawbacks, and each is suitable in particular contexts of use.

2.4.1 Dynamic content adaptation

The dynamic content adaptation strategy, also called just-in-time adaptation or on-the-fly adaptation, consists of creating the adapted content upon receiving the request from the mobile device. Based on the information extracted, mainly from the context of the user's terminal and the nature of the original content, the system decides what adapted content to generate. The system then generates the optimal version that meets the device constraints. However, when the number of users requesting the content becomes significant, the waiting time for the content increases. As a result, some users won't receive their content within a reasonable time. This

technique could be useful in a case where delivery time is not a big issue, or when there are enough computing resources available on the server side to satisfy every request in a timely fashion.

2.4.1.1 Typical dynamic content adaptation architecture

Generally, a dynamic content adaptation system is composed of four fundamental components: capability negotiation, a capability database, adaptation policies, and a content adaptation engine (Sudhir and Tao, 2004). Figure 2.1 shows a graphical representation of such a system. A

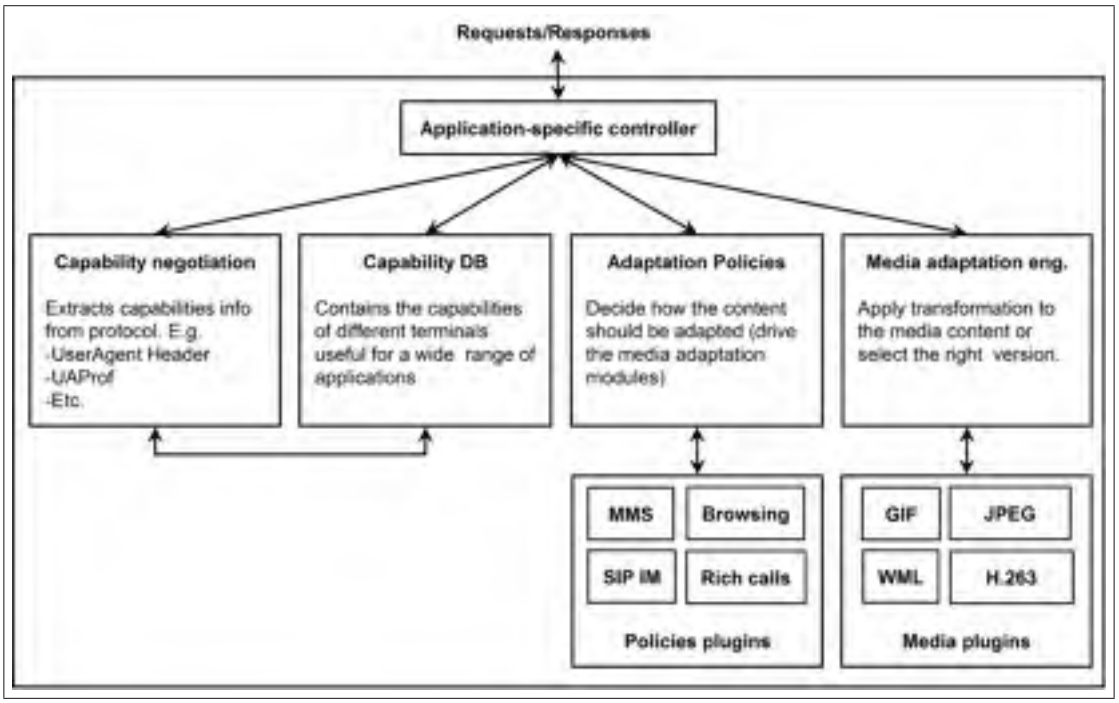

Figure 2.1 Typical architecture of dynamic content adaptation.
(Adapted from (Sudhir and Tao, 2004))

capability database might not be necessary if the device capabilities extracted from the request are complete. The capability negotiation module is responsible for extracting the device capabilities or characteristics from the protocol used by the request. If the protocol used is HTTP, the device capability can be extracted from the user-agent header (UA-header) (W3C, 1999). Sometimes, the information provided in the request is incomplete. In this case, to resolve the

device capability issue, this module uses the capability database module, which contains device capabilities (also called the user-agent profile: UAProf) (Open Mobile Alliance, 2006). Actually, UAProf is an XML file that describes the capabilities of mobile devices (e.g. vendor, model, screen size, etc.). A URL link to the mobile device's UAProf is included in the header within the HTTP request. The UAProf data are not always available, since their production depends on the nature of the mobile device. For example, the UAProfs of GSM devices are provided by the device's vendor (e.g. Nokia, Samsung, etc.), whereas those that use CDMA/BREW networks are provided by the telecommunications company (e.g. Verizon, Sprint, etc.). Some of the drawbacks to using UAProf databases (Glover and Davies, 2005; Chao, 2011) are the following:

- Profile information is not always provided by manufacturers.
- Published UAProfs are not always available (or are dead).
- There is no widely used standard to follow in describing mobile device profiles. Even if there were such a standard, the introduction of new devices might require new vocabularies that the adopted standard does not contain.
- Parsing UAProf is not always successful, owing to the absence of a unified standard.
- Online retrieval and parsing of UAProfs introduces a delay into the response. That's why they are cached to be used locally, which means that they need to be refreshed and maintained regularly to keep them up to date.

The adaptation policies module contains a set of rules that drives the content adaptation process. It takes into account the mobile device capabilities and the modalities it supports. Other information is taken into consideration, such as the bitrate of the link connecting the mobile device to the Internet. Sometimes, in order to shorten the delay in sending it, content is adapted to reduce its file size, for example, by reducing the quality factor in a JPEG image. However, it is important that the time taken by the adaptation process not increase the global time required to send the content.

The adaptation engine module uses the information received from the adaptation policies module to transform the original content into the desired version. The transformation depends on the nature of the content itself. Textual content is adapted differently from audiovisual content, and even the techniques and tools used are not the same.

2.4.1.2 Advantages and drawbacks of dynamic content adaptation

Table 2.1 summarizes the advantages and disadvantages of using the dynamic approach to adapt content (Sudhir and Tao, 2004).

Table 2.1 Advantages and drawbacks of dynamic content adaptation.

Advantages	**Drawbacks**
• It delivers content to users who were not able to get it without adaptation. • The transformation is automatic, so there is no need for human supervision.	• Its process is very often computationally complex. • There is no guarantee that the adapted content will be usable. • The original content may be protected by law, so copyright issues may arise.

2.4.2 Static content adaptation

The static content adaptation strategy (also called content selection) consists of selecting the best adapted content from a set of pre-created versions. Often these versions have been validated by a human to ensure quality. This strategy could be useful in certain circumstances, where the adaptation time is significant or there are not enough resources on the server side to perform a live transformation (processing time, busy servers,...). This approach resolves the first two drawbacks of the dynamic approach (see Table 2.1). In this case, the alternative versions could be created during the server's idle time. The adapted versions should be tailored to a wide variety of mobile devices, the terminal receiving the best quality content based on its capabilities and context. Chung-Sheng et al. (Chung-Sheng *et al.*, 1998) and Mohan et al. (Mohan

and Smith, 1999) have proposed an architecture composed mainly of two components: *infopyramid*, and *customizer*. The former is a database that contains all the adapted versions of the content created in advance, and the latter is a module responsible for selecting the right content for the right user (mobile device). Figure 2.2 illustrates the various components that comprise this architecture.

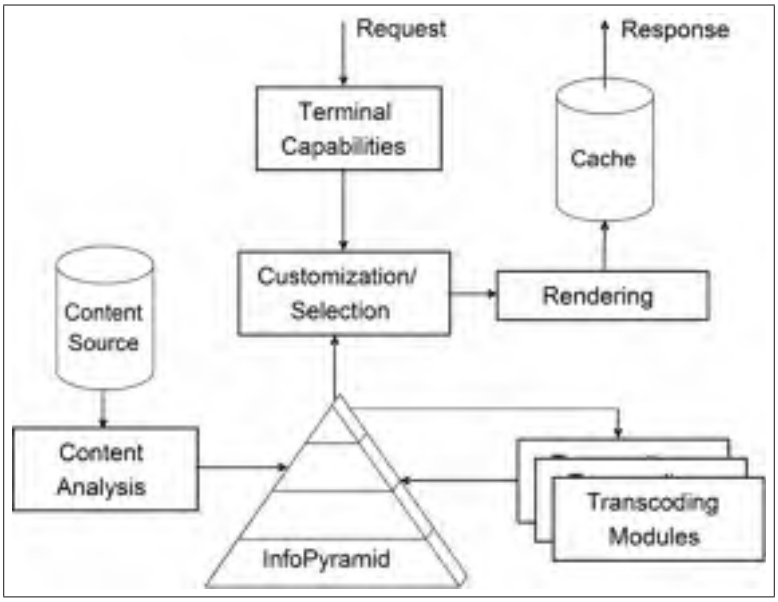

Figure 2.2 Architecture of the content selection system.
(Adapted from (Mohan and Smith, 1999))

2.4.2.1 The infopyramid

The infopyramid is a data representation scheme in which content items on a Web page are adapted into multiple resolution and modality versions, to enable them to be rendered by different devices (Mohan and Smith, 1999). For instance, a video can be adapted into a set of images to be rendered by mobile devices that don't support video. This scheme is composed of two axes: modality, and resolution (Sudhir and Tao, 2004). Figure 2.3 shows a graphical representation of the infopyramid. The modality axis provides the same information in different types of media (audio, video, image, text, etc.), to enable the infopyramid to select the media type that suits the targeted mobile device. For instance, audio modality is appropriate for users

on the move, whereas text, images, and video are preferable in meeting contexts (Sudhir and Tao, 2004). Once the modality is selected, the resolution axis provides various content version options. The resolution selected determines the quality of the content (Sudhir and Tao, 2004). This resolution information is very useful for meeting the target mobile device resolution if the original resolution is greater than that of the mobile device. It can also affect the delivery time.

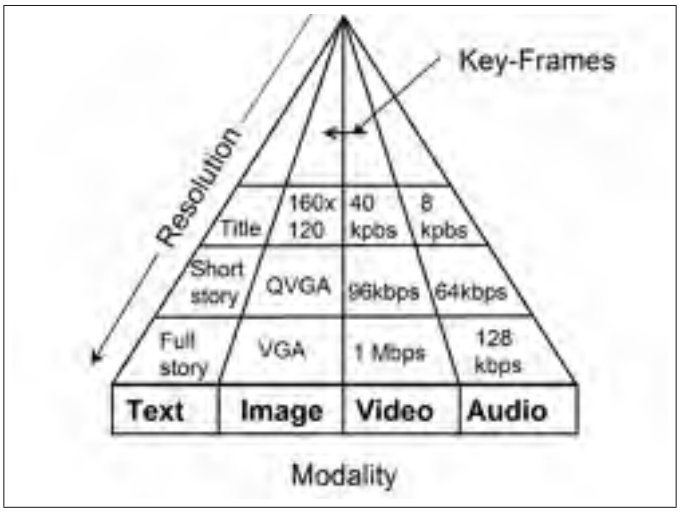

Figure 2.3 The infopyramid.
(Adapted from (Mohan and Smith, 1999))

2.4.2.2 The customizer

The customizer selects the best version of the content items from the infopyramid to meet the target mobile device's resources (Mohan and Smith, 1999). The selected content version can be cached before delivery, so that it can be used for another user with the same capabilities or for the same user in another session.

2.4.2.3 Creation of the infopyramid

The process for creating the infopyramid is as follows. Multiple versions of the content are created by varying values of the modality and resolution axes (Sudhir and Tao, 2004), and

then those versions are stored in the infopyramid. Even though this process is automated, it is often supervised by the content's author to guarantee that all the versions created are acceptable. Since certain combinations of modality and resolution, and their corresponding content versions, are not usable by all devices, those combinations should be filtered out and discarded. The granularity of the content is determined by the number of versions created. It is clear that higher granularity makes it possible to deliver the best adapted content version for specific terminal capabilities. If, however, the resolution of the device requesting the content falls between two resolutions, the system will deliver the content with the lower resolution. In this case, it may not deliver the right version, simply because it is not available. Therefore, higher granularity ensures better customized content for more terminals. At the same time, higher granularity implies a larger number of versions, and consequently longer processing time and more storage space are required.

2.4.2.4 Advantages and drawbacks of the static content adaptation strategy

Table 2.2 summarizes the advantages and disadvantages of the static content adaptation approach.

2.4.3 Mixture of dynamic and static adaptation

A combination of the dynamic and static approaches could be an option to consider in content adaptation. An adapted content set could be created in advance during the server's idle time, and a dynamic adaptation performed to complement it, as needed. When the system receives the request, if the right content is available, it is delivered; otherwise, dynamic adaptation is performed, if possible (availability of server's resources, reasonable adaptation time). If the right content is not available and just-in-time adaptation is not possible, the system selects the most appropriate content from the content set created in advance. Furthermore, the created content set could be enriched by adding any content created by the dynamic adaptation. In this case, the granularity of the content selection strategy is refined and the system becomes extendible.

Table 2.2 Advantages and drawbacks of the static content adaptation strategy.

Advantages	**Drawbacks**
• Issues related to dynamic adaptation are solved. • Content is delivered to users who were unable to get it without adaptation. • Content is selected and delivered automatically with little computational complexity, once the content versions have been created.	• Server resources must be available to perform the content adaptation in advance. • Human supervision is required (ideally) to filter the adapted content versions. • Storage space is required for saving the adapted content versions, and the amount increases with the granularity, which might be very high. • The user may not receive the best version, owing to gross granularity issues. • Some quality parameters may not be taken into account when versions are created, such as varying the quality factor of JPEG images and the color depth of GIF images.

2.5 Location of content adaptation

It is clear from the above sections that the server's resources have a major influence on the decision as to what strategy to follow: dynamic, or static. Regarding the location where the content adaptation can be performed, three options can be considered, and their usability should be evaluated. These options are the following: adapt the content on the server side or on the client side (mobile devices), or balance the load between the server and the mobile device. There are pros and cons to each of these techniques. Another option is to use an intermediary server (e.g. a proxy server), which could be dedicated to content adaptation, and be viewed as such. This would lighten the server's load. It is clear that this intermediary server would be subject to all the problems that any server may have, such as engorgement and lack of resources. Therefore, in the following, we consider only the three cited options.

2.5.1 Adaptation on the server side

The most natural strategy consists of adapting the content on the server side to suit a wide variety of mobile devices, from the highly capable to the less capable. Since, not all mobile devices are equipped with the necessary resources (e.g. memory, processing power) to be able to locally process the original content, the latter should be processed and formatted on the server side, according to the mobile device's constraints. The server, however, should be able to deliver the content within a reasonable time. If there are not sufficient resources on the server to perform the adaptation, the waiting time for the content could create a bottleneck, as discussed earlier (dynamic vs. static adaptation). In this case, some users would simply quit the application. The other problem that may render this approach challenging is access to the device's capabilities (Sudhir and Tao, 2004). We have mentioned that the capabilities extracted from the request header are sometimes incomplete, and must be complemented with capabilities databases. Unfortunately, those databases are not always reliable. Also, some devices are not included, and the ones that are included may not be up to date.

2.5.2 Adaptation on the client side

Why are we considering the features of mobile devices at the server level, when we can send the content to the mobile device itself for processing? If the content is processed locally, there would be no need to consider the mobile device's capabilities, nor its completeness problems, at the server. Also, the mobile device knows its own capabilities best, and so should be able to adapt and visualize content downloaded to it.

The answer is not a simple one. First, in addition to network issues that may render the delivery time unacceptable to the user, the size of the content may exceed the device's memory capacity, which would prevent content reception. Second, the terminal might not support the received format (e.g. H.264 video), and, even if the mobile device is able to perform the transformation using XML and XSLT, the processing time and resource consumption required may make this option unreasonable.

For instance, we have tested the possibility of sending original content to a mobile device to be processed locally. The mobile device was a Nokia N810 [1], which was a very recently developed technology when this research began, and supported XML and XSLT. The data sent consisted of an XML file containing the content itself and an XSLT StyleSheet to be used in the transformation, as well as an XHTML file used as an output to the transformation. We came to the conclusion that, when the file becomes sufficiently large, the mobile device cannot complete the adaptation task within a reasonable time. A complete description of the experiment can be found in section 4.3.2.2. On a positive note, the transformation was still possible for some smaller file sizes. Therefore, a tradeoff between adaptation on the client side and adaptation on the server side may be an option worth considering.

2.5.3 Hybrid adaptation

Based on the experiment described in the previous subsection, we know that when the content's file size exceeds a certain threshold, adaptation is not possible on a mobile device. To solve this problem, the content can be split into small pieces, depending on the target mobile device's capacity. For instance, an XML-based document, such as an OpenOffice Impress presentation, can be split into multiple small presentations, each of which contains only a limited number of slides. The number of slides can be determined by the mobile device's resources. In this way, the mobile device can be exploited without reaching its limits and contribute to reducing the server's computational burden. However, the impact of this technique (dividing the document) on the server resources should be evaluated, since this process would need to be performed on the server side. In other words, the process shouldn't increase the content delivery time. Finally, what adaptation can be performed on the device should be known beforehand, in order to ensure that sending unadapted content to the device will not affect the user's experience as a result of impacting the delivery time and rendering time.

[1] Nokia N810 specifications: http://europe.nokia.com/find-products/devices/nokia-n810/specifications

CHAPTER 3

CONTENT OPTIMIZATION TECHNIQUES

In this chapter, we present a review of the state-of-the-art research, methods, and algorithms proposed for the optimization of content for mobile devices. The objective is to review the various methods and techniques used in quantifying the adapted content quality, which enables the computation of the optimal adapted content version that satisfies the constraints of the target mobile terminal and improves the end-user experience. More specifically, we are looking for an automatic transcoding system with the best compromise between the user's experience, the system's computational complexity, and storage space. This system must take into account the end user's terminal capabilities. The user's experience should consider the perceived visual quality of the transcoded content as well as the time required for its delivery. Some of this research adapts the content to satisfy the target mobile device's resolution without considering the communication network conditions. It focuses only on the structure and order of the elements of the content to be adapted, and how they should be reorganized to better fit the mobile device's screen size. Other optimization algorithms adapt the content by changing its embedded element characteristics (e.g. text, images, etc.) to reduce the content's file size or to obtain good visual quality. In general, the overall content quality is evaluated by considering the quality of the individual content elements using a quality criterion, and combining them to obtain a single quality score. These approaches, techniques, and algorithms, as well as their strengths and weaknesses, are detailed in this chapter. They can be categorized into two groups, according to the strategy followed in adapting the content: static, or dynamic.

3.1 Static content adaptation techniques

In the context of static content adaptation, also known as content selection, the adapted content is sometimes considered as a service. The problem consists of selecting the optimal content from a set of adapted content versions which have already been created using various adaptation parameters. The optimal adapted content selected is content of the highest quality, evaluated using a quality metric, that can be supported by the terminal and its environment.

3.1.1 Content quality evaluation

Research involving mobile devices has led the notion of context (memory, bitrate, screen resolution, etc.), and a new paradigm has been introduced, that is, context-aware quality of service (also called QoS-awareness or quality of experience (QoE))(Lum and Lau, 2002, 2003; Zhang *et al.*, 2006a,b,c,d). This paradigm has led to the development of metrics to evaluate the quality of the adapted content targeted by mobile devices. Here, the term *quality of service* (QoS) can be used to quantify the quality of each adapted content version, but it has a broader meaning than the term "quality of service" that is used in networking to measure the quality of a network service subject to various parameters and variations (bandwidth, packet loss, throughput, jitter, and delay). We prefer to reserve the term QoS for networking issues and instead use the term QoE (quality of experience) to represent the quality of the delivered content, from the perspective of the end-user.

The QoE is affected by three factors (Kuipers *et al.*, 2010):

1. The quality of the content at the source, that is, the quality of the adapted content before delivery.

2. The quality of service (QoS), which is affected by the delivery of the adapted content over the network.

3. The humain perception of the adapted content (audiovisual quality, usability, the time required to obtain it, etc).

At a high level of abstraction, the QoE of adapted content is affected by its audiovisual quality and the transport quality (quality associated with the total delivery time). The first expresses how the content is appreciated audiovisually, and the second expresses the impact of the total delivery time on the appreciation of the content. However, this is not the only way of evaluating the QoE, and, as explained in (Kuipers *et al.*, 2010), it constitutes, as does the QoS evaluation, a completely separate research topic. We should point out that this way of computing the QoE is neither shared nor adopted by all researchers, as will be explained in the following sections.

3.1.2 Context parameter classification

Some researchers classify the context parameters according to their role in the evaluation of the quality of the adapted content. Certain parameters are used as constraints which should be respected by the target mobile device, so that the adapted content can be received, whereas others are used to measure the quality of the adapted content (Zhang *et al.*, 2006a,b,c,d). Their classification is as follows:

- Constraint parameters: These are parameters that should be respected by the adapted content, to ensure that it is accepted by the mobile device. For example, the file size of the adapted content is considered a constraint, and shouldn't exceed the mobile device's memory that is reserved for this purpose.
- Quality parameters: These are parameters that affect the quality of the adapted content. For instance, the higher the adapted content's color depth, the greater its quality, up to a point of saturation, after which there is no improvement.

Consider the following context parameter list:

1. The user's profile: name, role, and location.
2. The mobile device's capability: memory, resolution, color depth, and battery life.
3. The network status: bandwidth and latency

In general, the elements of the first group are considered as constraint parameters, and those of the third group as quality parameters. The use of some services may be restricted to specific people designated by name or role. For instance, in a meeting, the content presentation service, comprising the possibility of updating the slide content, adding annotations, etc, should be reserved for the presenter only (and not to meeting attendees), in order to preserve the flow of the presentation. From the second group, the memory parameter is always considered as a constraint, whereas the others (resolution, color depth, and battery life) are perceived differently in the research, as we explain in the subsection 3.1.3.

The user preferences, which represent the user's perception of the various parameters, are usually considered in the content adaptation process. However, they are not considered as context parameters, since they don't add new contextual information. They tell the system what the importance is of each parameter in the user's perception, such as preferring to receive the content faster even though the quality is lower, or asking for the best quality no matter how long it takes to receive it. These preferences will affect the quality measure, and thus the optimal adapted content.

This classification into *constraint parameters* and *quality parameters* is very useful, as it allows the set of adapted content versions that are supported by the target mobile device to be identified, and then the quality of each version to be computed, leading to the final selection of the optimal version.

3.1.3 Context parameter evaluation

3.1.3.1 Constraint parameter evaluation

Research (Lum and Lau, 2003; Zhang *et al.*, 2006a) has shown that first order logic inference can be used to identify the adapted content set supported by a mobile device. With this method, the characteristics of that content are compared with the features of the target mobile device. For instance, at the very least, for adapted content to be accepted by a mobile device, its file size shouldn't exceed the mobile device's memory. This constraint can be formulated as follows:

$$f_{\leq}(\text{file size of the adapted content}, \text{memory size of the mobile device}) \rightarrow \{0, 1\} \tag{3.1}$$

where $f_{\leq}$ represents a first order inference, which is evaluated as follows:

$$\begin{aligned} &if\ (\text{file size of the adapted content} \leq \text{memory size of the mobile device}) \\ &\quad return\ 1 \\ &else\ return\ 0 \end{aligned} \tag{3.2}$$

Some researchers consider bandwidth and network latency to be quality parameters (not constraints). To the authors of (Lum and Lau, 2003), however, they are constraints, and they can be combined and compared with a threshold that expresses the transmission time tolerated by the user:

$$2D_{RTT} + \frac{\text{file size of the adapted content}}{\text{bandwidth of the channel}} \leq t_{threshold} \tag{3.3}$$

where $2D_{RTT}$ is the network round trip time, and $t_{threshold}$ is the transmission time tolerated.

In (Zhang *et al.*, 2006a), Zang et al. consider the color depth and screen size as quality parameters, and evaluate them using fuzzy logic functions, which produce values between 0 and 1. In our opinion, this is not really an accurate assumption, as the adapted content could exceed the screen size and still be accepted by the mobile device, as long as it doesn't exceed the screen's maximum resolution (not its size). This is quite reasonable, since the current trend is to deliver content that fits the device's resolution, and allows the user to adapt his view by zooming or panning. This means that the device's maximum resolution could be evaluated as a constraint and a quality parameter at the same time, that is, the greater the resolution of the content, the better its quality, up to the screen's maximum resolution. The same is true for color depth. For instance, why send 16-bit adapted content when the targeted mobile device is limited to 2-bit color depth? The extra bits are not used by the mobile device, and only increase the content's file size and the end-user's waiting time. A thorough review of this method is presented in section 3.1.3.2.3.

3.1.3.2 Quality parameter evaluation

As discussed, the quality of the adapted content is affected by various quality parameters. We can express quality, Q, as:

$$Q = f(p_1, p_2, ..., p_n) \tag{3.4}$$

where $p_1, p_2, ..., p_n$ are n quality parameters. For simplicity, the quality function is often expressed as the sum of the various quality functions (Lum and Lau, 2003; Zhang *et al.*, 2006a):

$$Q = \sum_{i=1}^{n} f_i(p_i) \tag{3.5}$$

where $f_i(p_i)$ is the quality function associated with the i^{th} context parameter, p_i.

Since the quality parameter values are not necessarily within the same range and do not use the same system of units, most of the research proposes to normalize them before evaluation. The normalization process for each parameter is achieved by plotting a curve representing its behavior, the values on the curve being confined to between 0 and 1. The most important normalization and evaluation methods of the quality parameters are presented in the following subsections.

3.1.3.2.1 Logarithm-based quality parameter evaluation

Richards et al. (Richards *et al.*, 1998) have proposed to model the behavior of the quality parameters by means of a family of logarithmic curves. They believe that all the quality parameters can be modeled by logarithmic curves by changing their sensitivity, as shown in Figure 3.1. In these curves, a minimum point (M) and an ideal point (I) are defined as boundaries. The values that are lower than M are the unsatisfactory ones, and those that are higher than I provide no improvement in user satisfaction. The values between M and I represent increased improvement in perceived satisfaction. After modeling the behavior of each quality parameter, its value is mapped to its corresponding logarithmic curve, and a normalized value between 0 and 1 is generated. For a quality parameter x, the logarithmic function they propose for normalization $s(x)$ is defined as follows:

$$s(x) = a.ln(bx + c) \tag{3.6}$$

where:

$$\begin{aligned} a &= \frac{1}{p - 10} \\ b &= \frac{e^{\frac{1}{a}} - 1}{I - M} \\ c &= \frac{I - Me^{\frac{1}{a}}}{I - M} \end{aligned} \tag{3.7}$$

So, in equation 3.6, $s(x)$ is a function of the quality parameter x, and the parameter p represents the sensitivity of the logarithmic function that determines the actual normalization curve.

After normalizing all the quality parameter values, the quality of the service or the adapted content is calculated by the following formula:

$$S_{tot} = \frac{n}{\sum_{i=1}^{n} \frac{1}{S_i}} \tag{3.8}$$

where S_i is a normalized value of the i^{th} quality parameter, n is the total number of these parameters, and S_{tot} is the total service quality. According to the authors of this method, this formula is not unique, and there is no theoretical or empirical support for the use of either this formula or parameter behavior modeling.

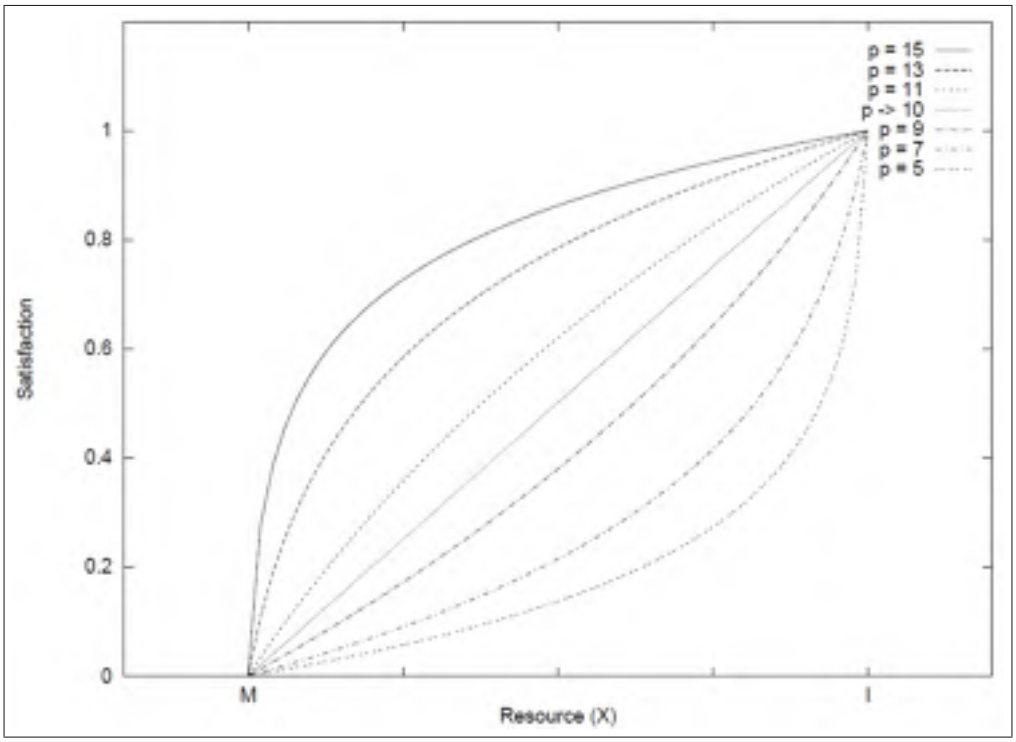

Figure 3.1 Quality parameter normalization using logarithmic curves. (Extracted from (Richards *et al.*, 1998))

3.1.3.2.2 SAW-based quality parameter evaluation

Lum et al. (Lum and Lau, 2003) propose to model the behavior of each quality parameter using linear or second order curves, as shown in Figure 3.2. The current value of the quality parameter is mapped to its corresponding curve to produce a normalized value between 0 and 1. For instance, to normalize the color depth behavior, the following second order function can be used:

$$qv = a.qs^2 + b.qs + c \tag{3.9}$$

where qs is the quality parameter value and qv is its normalized value. The values of a, b, and c are determined by observing the quality parameter behavior, that is, from Figure 3.2, three points can be used, from which we obtain the following system of equations:

$$\begin{aligned} 0 &= a.qs_{min}^2 + b.qs_{min} + c \\ 1 &= a.qs_{max}^2 + b.qs_{max} + c \\ 0 &= 2a.qs_{max} + b \text{ (saturation point } \Rightarrow \text{ derivative is equal to 0.)} \end{aligned} \tag{3.10}$$

where qs_{min} and qs_{max} are the minimum and maximum values of the quality parameter interval. For the color depth, these values could be 1 bit (black and white) or 16 bits (65,536 colors) respectively.

The authors of this method believe that it is possible to model the majority of the quality parameters, if not all, using linear or second order curves. Once again, there is no theoretical or empirical work supporting their claim.

To evaluate the quality parameters, they use a user-centric methodology to select the service best suited to an end-user's needs via a negotiation process (Lum and Lau, 2003). After modeling the quality parameters, the generated values are then weighted by values supplied by the end-user (user preferences). The sum of the weighted values represents the score of that service, according to the following formula:

$$score = \sum_{i=1}^{n} qv_i.w_i = \sum_{i=1}^{n} f_i(qs_i).w_i \tag{3.11}$$

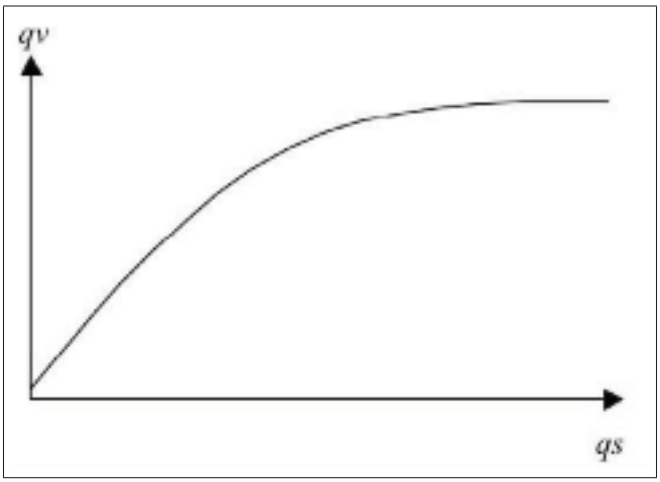

Figure 3.2 The color depth modeled by a second order curve.
(Extracted from (Lum and Lau, 2003))

where qs_i is the i^{th} quality parameter value, and f_i, qv_i, and w_i are its behavior function, normalized value, and associated weight respectively.

These scores are then organized into a binary search tree to reduce the search complexity. For each service, a score is calculated, which depends on the user's perception of the various contextual parameters (represented here by the weights). However, for the parameters that cannot be evaluated by order relations, the authors propose to look at each node in the binary tree. For instance, to check if a modality used by a service is supported by a mobile device, the system should explore all the score nodes of the tree. This methodology is computationally expensive, even though the authors have demonstrated that the search complexity is bounded. Besides, the idea of directly weighting the quality parameters and summing the weights to establish a final score that represents the quality of the adapted content is not really an accurate strategy, for at least two reasons:

1. First, not all the end-users are able to know the context parameters, especially those of the network in use (e.g. bandwidth and network latency). Even if this information is known, users may not understand the relationship between these parameters and the quality of the service they requested. An interesting approach would be to give the end-user a mechanism for expressing his preferences with respect to information that is meaningful

to him. For example, the system should ask the user what kind of service he prefers: one providing good visual quality, a faster service, or a service that doesn't rapidly drain the battery. We would describe such a mechanism as being at a high level of abstraction, in that it deals with information that can be handled by the end-user.

2. Second, the scores are weighted and the weights are summed. Some parameters should not be combined in this way, because they are not compensatory. For instance, if the bandwidth is close to zero, the service should simply not be delivered. But, according to equation 3.11, if the other context parameters are highly weighted, the quality score will be higher, which is misleading. In fact, this method is one of a set of scoring methods used in the resolution of MADM problems (*Multiple Attribute Decision Making*), which is called SAW (*Simple Additive Weighting*) (Hwang and Yoon, 1981). It can be used in a context where the parameters (attributes) satisfy the so-called compensatory property (Lee and Anderson, 2009; Dieckmann *et al.*, 2009), that is, the loss or gain in one attribute can compensate for a loss or gain in the others. However, this property is not always shared. For instance, it is not shared by color depth and bandwidth, and so compensation is not possible in this case. For MADM problems, such attributes are called not comparable (Hwang and Yoon, 1981). A non exhaustive list of MADM methods is as follows:

   - SAW (*Simple Additive Weighting*)
   - WP (*Weight Product*)
   - AHP (*Analytical Hierarchical Process*)
   - TOPSIS (*Technique for Order Preference by Similarity to Ideal Solution*)
   - ELECTRE (*Elimination and Choice Translating Reality*)

We maintain that the weight product method (WP) is more appropriate for solving this kind of problem, as we explain in our proposed dynamic content adaptation framework (see chapter 5).

3.1.3.2.3 Fuzzy logic-based quality parameter evaluation

Zhang et al. (Zhang *et al.*, 2006a) propose to organize the context parameters into a hierarchical QoS model tree to facilitate their evaluation. As shown in Figure 3.3, the context parameters fall into three groups: usability, device capability, and network status.

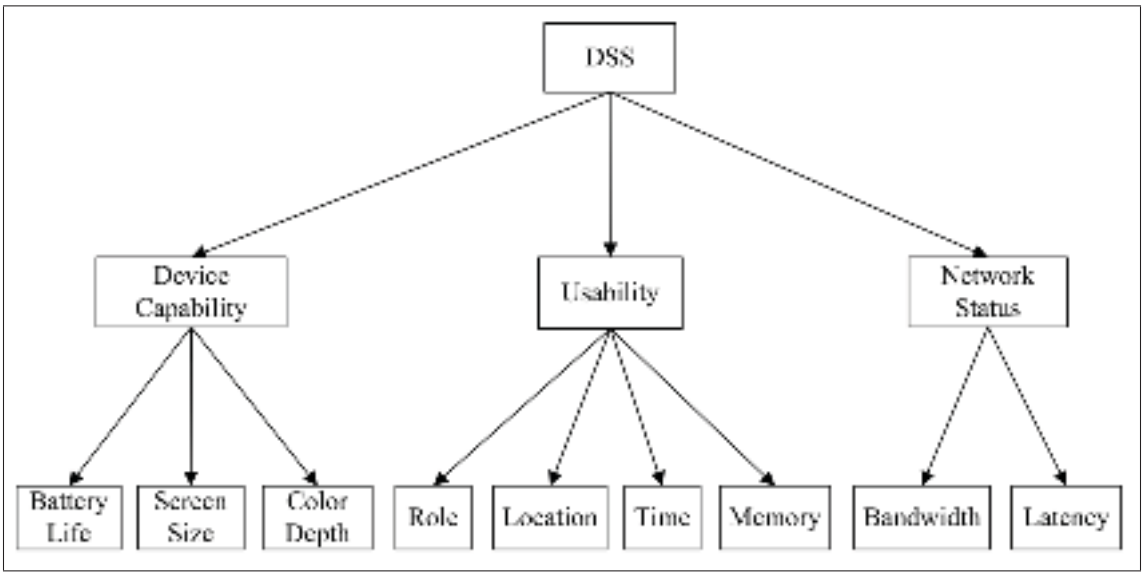

Figure 3.3 Hierarchical QoS Model.
(Extracted from (Zhang *et al.*, 2006a))

The parameters of the usability group represent the constraints that should be respected by the service to enable acceptance by the target mobile device, and so they are evaluated by first order logic, as explained in section 3.1.3.1. Those of the device capability and network status groups determine the quality of the service, and so they are evaluated by fuzzy logic. The first order logic evaluation returns a scale in $\{0, 1\}$, whereas the fuzzy evaluation returns a scale in $[0, 1]$. The product of the Boolean result and the fuzzy result represents the final score that determines the quality of the service. If one of the constraints is not respected (e.g. the service's file size exceeds the mobile device's memory), the Boolean evaluation returns 0, and so the final score is equal to 0. If all the constraints are respected, the final score will be determined by the

fuzzy evaluation. In addition, a threshold is added to determine what services are acceptable for delivery. This threshold can be learned from practical experience.

This solution enables us to associate a range of values with each quality parameter of a service. For instance, a suggested bandwidth value for a service could be defined by a predicate, such as: “bandwidth > 50 kbps". In other words, for this service to be usable, the bandwidth should be greater than 50 kbps. This information is encoded in an XML file, called a “service profile”, as shown in Figure 3.4. The predicates associated with these parameters are then normalized

```
<service name–”multimedia introduction”>
  <btContext name=”Usability”>
    <Context> <name> Location </name>
      <Relation operator=”Equal”>room 324</Relation></Context>
    <Relation operator=”And”>
    <Context> <name> Role </name>
      <Relation operator=”Equal”>everyone</Relation></Context>
  </ btContext >
   ……
  <ftContext name=”Device Capability”>
    <Context> <name> Battery Life </name>
      <Relation operator=”Great-Than”>75</Relation></Context>
      ……
  </ftContext>
</service >
```

Figure 3.4 Service profile.
(Extracted from (Zhang *et al.*, 2006a)

using fuzzy membership functions, such as the sigmf [1], and gaussmf [2]. The quality parameters that are represented by fixed values can be modeled by a curve representing the behavior of that quality parameter, such as that of color depth, described in (Lum and Lau, 2003).

[1] Sigmoidally shaped built-in membership function, http://www.mathworks.com/help/toolbox/fuzzy/sigmf.html
[2] Gaussian built-in membership function, http://www.mathworks.com/help/toolbox/fuzzy/gaussmf.html

At the same time, mobile device features are mapped to the behavioral curves associated with the service parameters, and a scale between 0 and 1 is generated. That scale represents the degree of satisfaction (DSS) of the mobile device as provided by the service regarding that quality parameter. Finally, these computed DSS values, representing the degree of satisfaction of each quality parameter, are weighted (using the user's preferences) and the weights summed to compute the final service's degree of satisfaction.

This QoS model (Zhang *et al.*, 2006a) is very interesting, since it organizes the context parameters based on how they are evaluated. Also, it associates a curve with each service's quality parameter, which varies from one service to another. However, there are drawbacks to both this model and the DSS evaluation method, if we want to apply them to enterprise document sharing using the Web:

1. **Context parameter classification**

   The proposed context parameter classification is confusing, since the memory parameter, which represents a device capability, is in the "usability" group and not in the "device capability" one. Since there is already a group called "device capability", it is more appropriate to include the memory parameter in this group, otherwise it should be given a different name.

2. **Color depth and screen size issues**

   In this QoS model, color depth and screen size are not considered as constraints, and their values only contribute to the quality of the service. In fact, mobile devices are usually able to render the services that are encoded with a color depth greater than that supported by the mobile device's color screen. Also, a service could exceed the screen size and be accepted by the mobile device, as long as it doesn't exceed the maximum supported resolution (for which it is capable of decoding and scaling down the content). For this reason, the mobile device resolution should be considered both a constraint and a quality parameter. That is, the service shouldn't exceed the mobile device resolution; and, the better the service resolution, the better the service.

Regarding the color depth parameter, when only the visual aspect is considered, a second order curve is sufficient to represent color depth behavior as presented in (Lum and Lau, 2003; Zhang *et al.*, 2006a). However, to represent the quality of the experience (QoE), which is also affected by the delivery time (file size,...), color depth should be modeled differently. A Gaussian membership function [3], for example, in which the peak represents the maximum color depth supported by the mobile device, is more appropriate. In such a curve, the service quality increases until the peak (maximum mobile device color depth) is reached, and starts decreasing when the color depth is greater than that supported by the mobile device. Of course, the additional bits do not improve visual quality, but only increase the service's file size, which negatively affects delivery time. The idea is to introduce a penalty for the extra color depth bits.

3. **Supported formats**

   The Web browser features are not considered in this model. For example, suppose that the requested service is an AJAX-based Web page and the targeted mobile device's Web browser doesn't support AJAX. In this case, the proposed QoS model fails. This can be generalized to all the content representation formats, such as JPEG, SVG, and XHTML.

4. **Server issues**

   When the number of users requesting the service is high, the server can easily be overwhelmed. Depending on the server performance, the period of time spent by the request in the server waiting to be processed could affect the QoS to be delivered. However, the proposed model doesn't take this aspect into consideration. Instead, it supposes that the request is processed upon reception, which is not always the case.

3.1.3.2.4 Fidelity-based quality parameter evaluation

In Jan et al. (Jan *et al.*, 2006), the authors introduce a new measure to quantify the quality of adapted Web content, which they call the measure of fidelity. For a Web document P composed of a set of components d_i, P is represented by $P = \{d_1, d_2, \ldots, d_n\}$. A component d_i can be

[3] Gaussian built-in membership function: http://www.mathworks.com/help/toolbox/fuzzy/gaussmf.html

transcoded into different versions by varying the component's resolution and modalities, that is, $d_{i1}, d_{i2}, \ldots d_{iJ_i}$. To each version d_{ij}, the measure of fidelity v_{ij} is computed as follows:

$$v_{ij} = \frac{\text{perceived value of transcoded version } d_{ij}}{\text{perceived value of original } d_{i1}} \tag{3.12}$$

where $0 \leq v_{ij} \leq 1$. The perceived value can be assigned by the author to each version computed using a function that represents the general trend of the fidelity measure. In their paper, the authors propose to use the data size of the versions as perceived values, as follows:

$$f(w_{ij}) = \sqrt{\frac{w_{ij}}{w_{i1}}} \tag{3.13}$$

where w_{ij} and w_{i1} are the data sizes of the transcoded version $d_{ij,}$ and that of its original version d_i respectively. After computing the quality of each transcoded content version, the problem consists of selecting the optimal version that meets the maximum file size that can be transmitted over the network, based on a reasonable waiting time, which is affected mainly by the network bandwidth. They formulate the problem as an LMCKP (*linear multi-choice knapsack problem*) problem, as follows:

$$\begin{aligned} \text{Maximize } & \sum_{i=1}^{n} \sum_{j=1}^{J_i} v_{ij} x_{ij} \\ \text{Subject to } & \sum_{i=1}^{n} \sum_{j=1}^{J_i} w_{ij} x_{ij} \leq W \\ & \sum_{j=1}^{J_i} x_{ij} = 1, 1 \leq i \leq n \\ & x_{ij} = 0 \text{ or } 1, \text{ for all } i, j. \end{aligned} \tag{3.14}$$

where v_{ij} and w_{ij} are the measures of fidelity and data size of the component version d_{ij} respectively. W represents the maximum data size permitted (payload).

In this solution, the visual quality of the adapted content is not taken into account because the authors use the data size of the versions instead of perceived values. Moreover, the adapted content can satisfy the network file size constraint and not accepted by the target mobile device,

if , for example, its file size exceeds the mobile device memory size or if it is simply encoded in a format that is not supported by the mobile device. Finally, the method doesn't consider the user experience with respect to the delivery time.

3.1.4 User preferences representation

The best service is may be perceived differently by different users. While some users prefer to receive the service rapidly, others could be less concerned about the waiting time and prefer the best visual quality possible. This had led to a great deal of research into taking account the end-user's preferences in the process of creating or selecting services. Most of the research weights these context parameters directly, such as bandwidth and network latency. For instance, linguistic terms, such as "unimportant", "important", and "very important", have been used to express the user's preferences regarding the context parameters, and these are modeled using fuzzy logic functions (Zhang *et al.*, 2006a). Others use ranking techniques to quantify the user's preferences (Lum and Lau, 2003).

However, not all users understand the context parameters or are able to weight them directly, especially those related to the network. In fact, because it is applied to the context parameters directly, this kind of evaluation can be seen as low level evaluation. Therefore, it is more appropriate to allow the user to express his preferences by means of terms that have real meaning for him, such as: *best visual quality*, *fastest service*, and *less energy consuming service*. A similar, but limited solution was proposed by Han et al. (Han *et al.*, 1998) to model the user's preferences. This solution uses a slide bar that can be adjusted by the end-user to express his preferences regarding the download speed of an adapted image (see Figure 3.5). The boundaries of this slide bar are the following: slower download (less distillation), and fast download (more distillation). This solution is limited to the download speed, but could be generalized to express the QoE by taking into account visual quality and the actual battery charge.

In this thesis, and as detailed in chapter 5, we propose to use such terms in modeling the end-user's preferences. We refer to them as high-level terms, in contrast to those proposed in the state-of-the-art research.

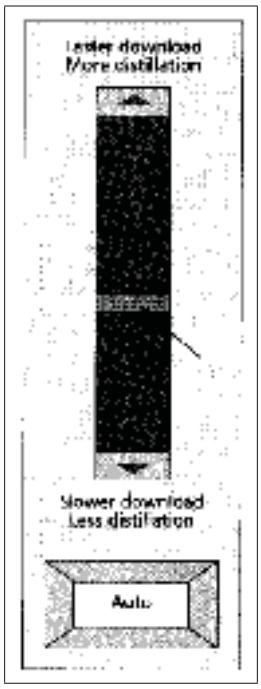

Figure 3.5 User's preferences as modeled by a slide bar.
(Extracted from (Han *et al.*, 1998)

3.2 Dynamic content adaptation techniques

In static content adaptation, because of the large number of adapted content versions created by varying the transcoding parameters (scaling, modalities, etc.), the content is adapted offline, and sophisticated algorithms are used to select the optimal version when a user requests that content. In dynamic content adaptation, also called on-the-fly content adaptation, the problem is more complex. Since no transcoding is performed prior to the user's request for content, every adaptation and quality evaluation has to be performed while the end-user is waiting. Consequently, it is highly desirable to perform as few transcoding operations as possible (create a single version, if possible), yielding the best QoE achievable (create the optimal version, if possible). The goal is to find a good set of transcoding parameters for content that not only result in adapted content supported by the terminal, but also having the best possible quality.

In the context of enterprise document sharing, the straightforward solution is to use the target mobile device's resolution and fix certain quality parameters. For instance, in dynamic Web

content adaptation (Xiao *et al.*, 2008), the Web page is usually transcoded based on the target mobile device's resolution without changing the quality parameters of the embedded images. JPEG images are also adapted using the mobile device's resolution and a quality factor value between 75 and 85, which provides good visual quality. However, such a solution doesn't take into account the resulting file size, which may, depending on the network conditions, create transmission issues. This can be especially problematic when a high-resolution image is delivered over a low-bitrate network. In a meeting context, for instance, the user may find himself waiting for slides while the presenter is talking about them, which creates a serious usability problem. Furthermore, scientifically speaking, these solutions are neither grounded nor validated from a usability standpoint.

Certainly, the static adaptation techniques presented above could be applied in this context. However, they are computationally very expensive, since they require adaptation of the content into several versions before it is requested by the end-user.

In general, dynamic content adaptation solutions can be classified in two classes: device capability-based methods, and prediction-based methods.

3.2.1 Device capability-based dynamic content adaptation

This class of dynamic content adaptation methods comprises solutions that only consider the target mobile device capabilities (e.g. resolution, supported formats, etc.) and ignore its environment, such as the network bitrate and latency. Although the adapted content could be rendered by the target mobile device, it cannot improve the end-user's experience, as this requires consideration of the QoS aspect of the content as well, which is mainly influenced by the communication network conditions (Kuipers *et al.*, 2010).

In dynamic Web content adaptation, Xiao et al. (Xiao *et al.*, 2008) propose breaking down the Web page into small, logical blocks and storing them in a repository. When the Web page is requested, a thumbnail image of the whole Web page is created and formatted into small, logical areas that correspond to the blocks already created. The user can dynamically visualize the desired section (or information) by moving the mouse pointer to that section. Technically,

when a block is selected for visualization, an Ajax request is fired to the server to retrieve the actual block and adapt it to fit screen of the target mobile device. In this way, the content is adapted dynamically based on the user's interaction with the content. However, the adaptation focuses only on the size of the screen of the target mobile device. As a result, there is no guarantee that the adapted blocks will be rendered on the mobile device. This could be because the block data size exceeds the mobile device memory size, or the media types used are not supported by the mobile device, for example.

The solution proposed in (Mérida *et al.*, 2008) is to adapt the content taking into account only the user preferences and the target mobile device capabilities, including screen size, maximum resolution, and image and color capabilities. The components of the content are first extracted and stored in a repository, and, when the content is requested, they are adapted using a set of predetermined transcoding parameters, such as color depth and media type. Then, these adapted components are assembled into Web pages (XHTML, JPEG, GIF, etc.). Figure 3.6 shows the architecture they propose. This solution doesn't take into account the mobile device environment (network conditions), resulting file size, or the visual aspect of the content.

Another kind of dynamic solution has been proposed by Liang et al. (Liang *et al.*, 2006), in which a set of adaptation strategies is defined in advance. These strategies consist of the adaptation methods to be followed when a given mobile device requests the content. They are based on frequently encountered contexts. This solution can be enriched by adding new contexts; however, it is not device-independent, as it is based on predefined contexts. Furthermore, no details are given on how these strategies are constructed and what impact they have on the end-user's experience.

3.2.2 Prediction-based dynamic content adaptation

To resolve the difficult problems encountered with the solutions described in the previous subsection, which involve creating, on-the-fly, the best adapted content representing the best QoE possible, research is turning to prediction techniques. These are intended to predict certain characteristics of the adapted content (e.g. quality, file size, transcoding time, etc.) without

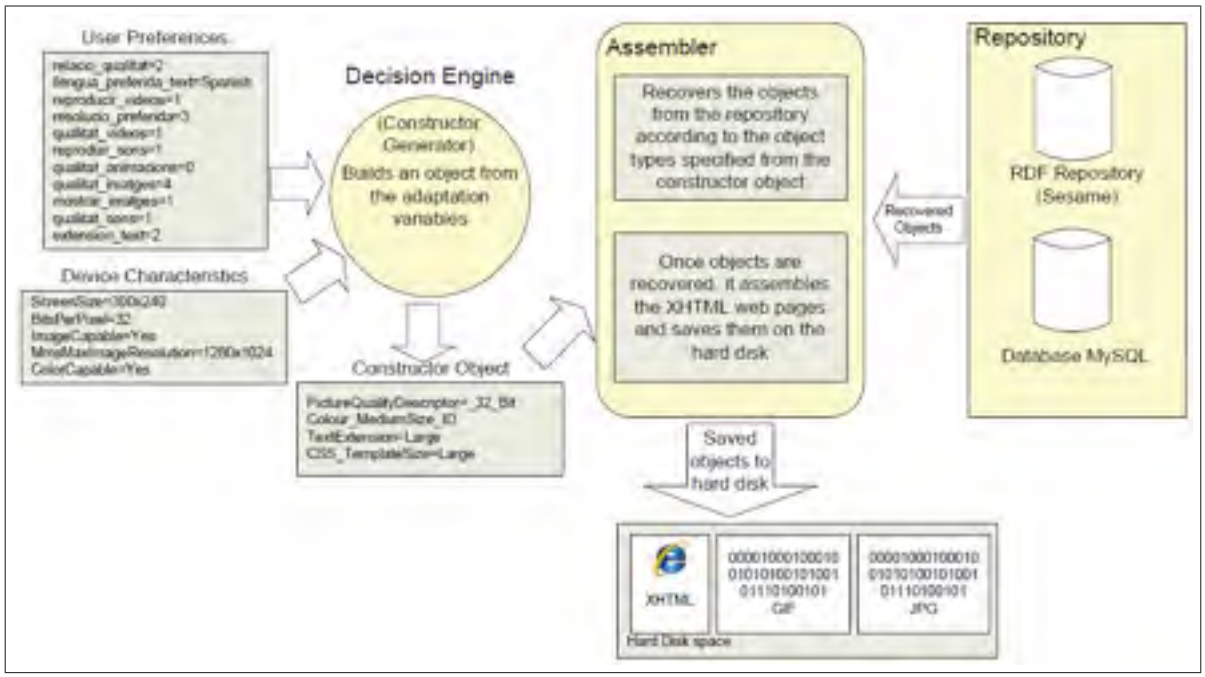

Figure 3.6 Dynamic content generator proposed by Mérida et al.
(Extracted from (Mérida *et al.*, 2008))

having to adapt it, which allows the best adapted content to be computed with fewer transcoding operations. This is an important concept, since it can lead to significant reductions in transcoding computational complexity.

For instance, to provide an efficient dynamic adaptation framework for images, the authors of (Han *et al.*, 1998) propose to predict the resulting file size of JPEG and GIF images, as well as the transcoding time they may require. Using this solution, it is possible to perform one transcoding operation and ensure that the adapted content will be received by the target mobile device, taking into account the network conditions (bitrate and network latency). This is an important step in performing dynamic content adaptation requiring fewer transcoding operations, since earlier methods transcoded the content several times until the content size was close enough to a desired value. However, this method doesn't consider the visual aspect of the adapted images and focuses only on the transmission issues, which are affected by the file size and transcoding time. From the QoE point of view, as described in section 3.1.1, it

is not enough to consider the transport issues alone in attempting to optimize the end-user's experience.

The authors of (Pigeon and Coulombe, 2008, 2011; Coulombe and Pigeon, 2009, 2010) estimate the resulting file size and visual quality, measured by the well-known SSIM (Wang *et al.*, 2004) image quality metric. These two aspects are of great interest in practical applications, since they represent the QoE as explained in (Kuipers *et al.*, 2010), that is, the visual aspect of the adapted content and its QoS. However, these two pieces of predicted information need to be combined (or a least considered together) to be used in practice, such as the delivery of Web or enterprise documents to mobile devices.

3.3 Discussion

In this chapter, we have reviewed various methods and techniques used in the literature to adapt content to mobile devices. Most of these methods were dedicated to Web content and image adaptation. We first presented the static techniques, which require that a set of adapted content versions be created from which the optimal one is selected, also referred to as "content selection". Then, we showed how little work has proposed dynamic adaptation of content to mobile devices, owing to the challenging nature of the problem.

In spite of their widespread use, enterprise document adaptation has not attracted the same attention as its Web counterpart. As detailed in the next chapter, enterprise documents are encoded using XML, especially those created with the latest Office suites, such as OpenOffice and MS Office (version 2007 and later). This makes them interoperable with tag-based languages and parsing engines, such as XSLT. Furthermore, as they are created with a specific Office suite, they necessarily follow a particular grammar. This is quite different from Web content generated by different authors and tools, which do not follow any particular grammar. As a result, enterprise document parsing should be easier and document adaptation more reliable.

In the next chapter, we present the general context of the dynamic framework we propose, specifically the architecture and its components, in particular the decision unit (responsible

for identifying the optimal adapted content) and the transcoding engine unit (to be used in the adaptation of enterprise documents).

CHAPTER 4

PROPOSED PREDICTION-BASED CONTENT ADAPTATION SYSTEM

In this chapter, we describe the architecture of the novel dynamic content adaptation framework that we propose, and the modules comprising the adaptation process. In particular, we present the *decision unit*, which determines the best transcoding parameters, and the *content adaptation unit*, which performs the content adaptation based on these parameters. We show the content adaptation unit in detail, and explain how it can be implemented using OpenOffice filters. The decision unit is described in more detail in the subsequent chapters.

Although high-level, this architecture is not new. What is new is modifying it to support prediction-based dynamic content adaptation for enterprise documents and Web applications. The modification consists, mainly, in the creation and integration of a transcoding engine capable of predicting near-optimal transcoding parameters computed for (but not limited to) a novel quality of experience measure and using them to adapt dynamically enterprise documents into Web-based ones. Actually, this is the second research project carried out on prediction-based dynamic content adaptation. The first addresses the problem of JPEG image adaptation in the context of Multimedia Messaging Service (MMS) (Pigeon and Coulombe, 2008, 2011; Coulombe and Pigeon, 2009, 2010). Applying prediction-based dynamic content adaptation to collaborative Web conferencing presents new challenges.

4.1 Proposed dynamic content adaptation architecture

Based on the analysis performed in the previous chapter on the state-of-the-art research related to content adaptation, and because Web 2.0 technologies are very appealing in the context of enterprise documents, we have decided to take a Web-based approach to solving the problem of content adaptation in collaborative Web conferencing applications. Our approach involves using existing Web 2.0 technologies and existing content representation formats (XHTML and JPEG) to adapt content for mobile collaborative Web applications dedicated to enterprise documents. Since the mobile device capabilities and their Web browser features differ, the proposed

solution consists of tailoring (customizing) the content to the characteristics of individual mobile devices. What we are proposing is a context-aware system, the context being that of the target mobile device. The clients we consider in this research are mobile devices equipped with Web browsers and having Web connectivity. The content we consider for adaptation are enterprise documents (e.g. PowerPoint slides). In a meeting where a PowerPoint presentation is shared, some mobile devices may receive rich XHTML content (with editable text and images), whereas others will receive a JPEG image (image snapshot of the whole slide wrapped into an XHTML skeleton) with specific resolution and quality values.

The proposed dynamic content adaptation architecture will comprise four tiers, as depicted by Figure 4.1. These tiers are the following:

1. The client tier, represented by the mobile Web browser.

2. The server tier, designated by the Web server and consisting mainly of a servlet and a bean. The servlet plays the role of middleware between the client tier (the Web browser) and the business tier (the transcoding engine), but via a bean, which can be a java class helper. The bean parses the information received from the servlet, extracts the information needed (the user profile, the mobile phone characteristics, the requested content, etc.), converts it into a processable format (e.g. JSON or XML), and sends it to the transcoding engine.

3. The business tier, representing the transcoding engine and comprising two units: a decision unit (DU), and a content adaptation unit (CAU). The DU computes the best, ideally optimal, transcoding parameters using the context information received from the bean and the composition of the original content. The CAU converts the content into an adapted version, using the transcoding parameters, to be returned to the terminal.

4. The persistence tier, consisting of a repository database that is used to store the content, such as presentations loaded by users. If desired, both the adapted content and the transcoding parameters used in its creation can be stored for other users, depending

on the policy adopted, that is, on-the-fly transcoding or, when possible, selection of a version that has already been created.

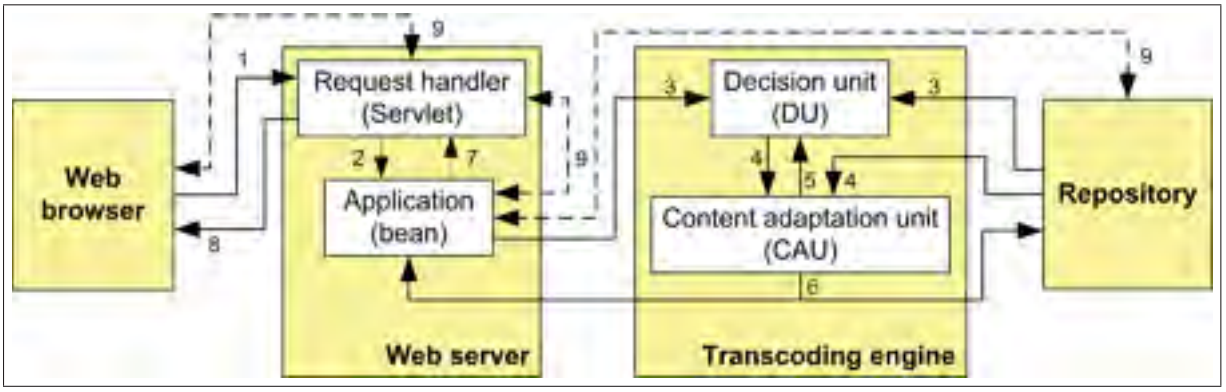

Figure 4.1 Global view of the proposed dynamic content adaptation architecture.

For processing, the user's request goes through several steps, as shown in Figure 4.1. Typically, these steps are the following:

1. An HTTP request is sent to the Web server to download the main page (XHTML) of the Web application (e.g. the main page of a meeting presentation).

2. In the Web server, the servlet receives the request, extracts the important data, such as the user profile and the file name of the document requested, among other things. Then, these data are sent to the bean, which is responsible for communicating with the third tier.

3. The DU receives the data and requests certain meta data of the enterprise document to be adapted (its composition), which have already been extracted from the document and stored in the repository. Using these data, it computes the best, ideally optimal, set of transcoding parameters and sends them to the CAU.

4. The CAU requests the enterprise document from the repository, and uses the transcoding parameters received from the DU to generate the best adapted content version.

5. Once the adapted content has been created, its characteristics are compared to the terminal's capabilities and context to ensure that it satisfies those requirements. If it does not, the DU must calculate new transcoding parameters until the terminal's constraints are met.

6. Once compliant adapted content has been created, it is stored in the repository, along with the parameters used in the adaptation process. At the same time, a persistent version of the adapted content is sent back to the bean in the Web server, in response to the user's request.

7. The bean receives the adapted content and sends it back to the servlet.

8. The servlet, which is responsible for communicating with the client side, forwards the adapted content to the user.

9. The dashed line in the figure represents the information being sent back and forth between the user and the persistent version of the adapted content. This sequence can occur if the user has received an editable version of the content. Each time the content is updated (e.g. insertion, modification, deletion, etc.), a request (XMLHttpRequest) is fired to update the content in the repository. This is very important when the content is shared with other users, and ensures that a correct and up-to-date version is sent back when content is requested, and not an old one. The content sent from the servlet to the Web browser is received first by an AJAX engine, which is normally supported by the Web browser. The AJAX engine is responsible for how the data are to be represented on the Web browser, as well as interpreting a user's actions performed on the content. In fact, the AJAX engine gives us the ability to mimic desktop applications by offering interactivity and responsiveness. This makes it possible to run Web applications that are similar to Office applications (a PowerPoint presentation, for example) on mobile devices.

4.2 Prediction-based content adaptation system

The proposed architecture is illustrated in greater detail in Figure 4.2. In this framework, we focus on the business tier, which comprises the transcoding engine and its components. For clarity and to highlight what is novel in this architecture, we present the transcoding engine and the elements with which it interacts, namely the Web server and the preprocessing subsystem.

4.2.1 The preprocessing subsystem

After the enterprise document has been uploaded, it can be processed offline to extract its characteristics. These are used later by the transcoding engine when the document is requested. In fact, the document characteristics can be extracted before the document is requested, in order to save the precious processing time of the transcoding engine. This feature is called the *Preprocessing subsystem*. In reality, in most collaborative mobile Web conferencing services, a PowerPoint presentation will be uploaded by the presenter prior to the meeting.

4.2.2 The Web server

At the time of the meeting, when the enterprise document is requested, the users' requests are captured on the Web server by a *request handler* module, which keeps track of these requests and computes the time they spent on the server before they were processed. The device's capabilities and the user preferences are extracted and resolved on the Web server, with the option of using a cached UAProf database. These data, comprising the device capabilities, the user preferences, and the waiting time are sent to the decision unit in the transcoding engine, where they are needed to compute the best transcoding parameters.

4.2.3 The transcoding engine

The transcoding engine, which consists of the DU and the CAU, represent the core of the proposed architecture.

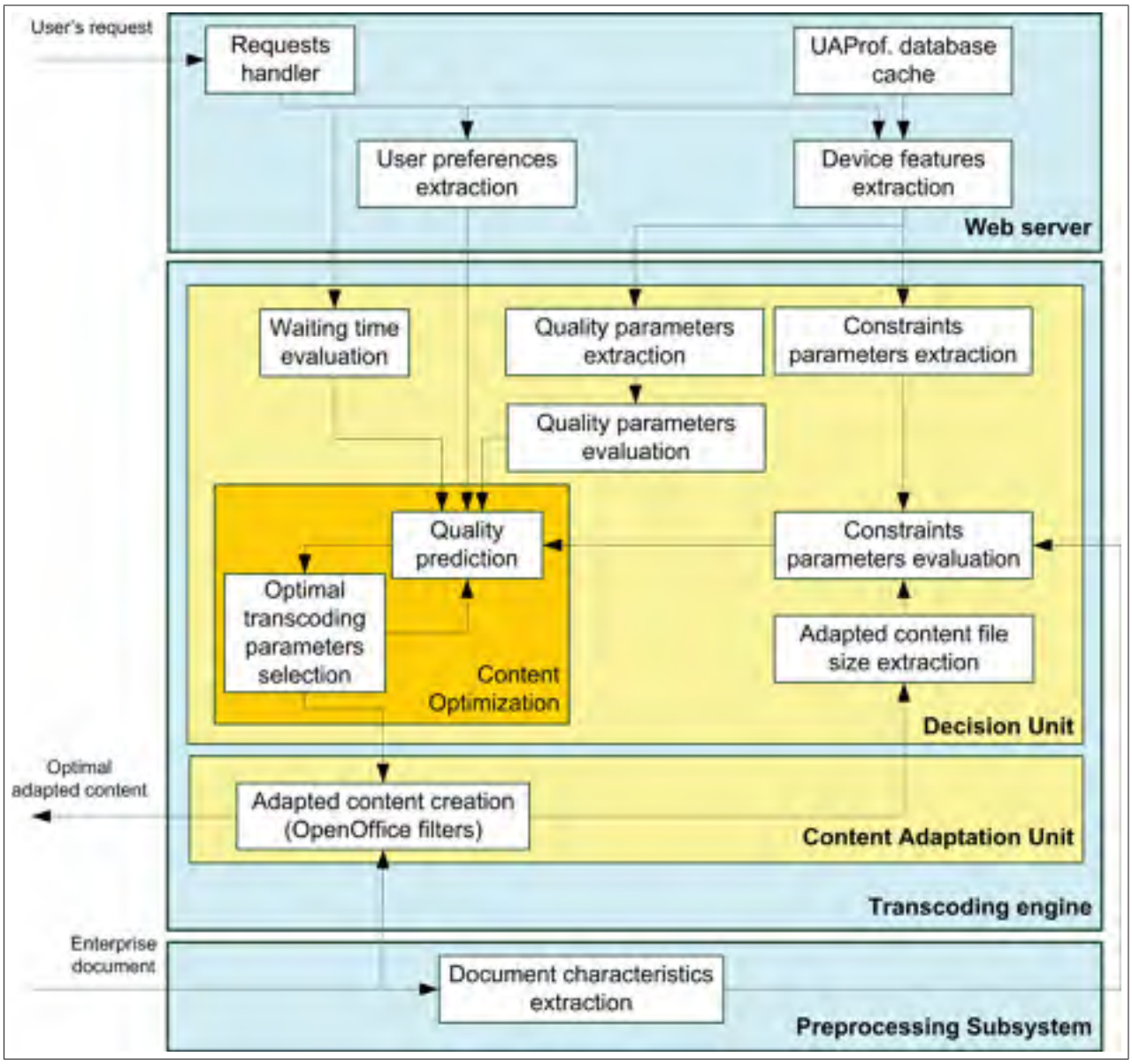

Figure 4.2 Proposed prediction-based dynamic content adaptation system.

In the DU, the constraints and quality parameters are extracted from the device capabilities by the *Constraints parameters extraction* and the *Quality parameters extraction* modules respectively. Then, they are evaluated by the *Constraints parameters evaluation* and *Quality parameters evaluation* modules respectively. The time spent by the user's request in the server before it is processed is evaluated by a *Waiting time evaluation* module. After these data have been evaluated, they are sent to the *Content optimization* module, which comprises a *Qual-*

ity prediction and a *Optimal transcoding parameters selection* module. The first module uses the constraints and quality parameters evaluated, the user preferences, and the waiting time to predict the quality of the adapted content as a function of a set of transcoding parameter combinations. The second module selects the optimal transcoding parameter combination that corresponds to the best predicted quality.

The content adaptation unit uses the predicted optimal transcoding parameters to adapt the enterprise document into the best adapted content version (it is expected to be near optimal, as it is based on predicted transcoding parameters). Before sending it back to the end-user, the adapted content is sent to the DU, where its characteristics are extracted and evaluated against the context parameter characteristics to ensure that it meets the target mobile device constraints. If the constraint parameter evaluation fails, another set of transcoding parameters should be predicted and sent to the CAU. Finally, when an acceptable version of the content is created, it is sent back to the target mobile device and cached in the repository.

In this section, we have described the two units that make up the transcoding engine, their modules, and how they interact with each other in order to adapt enterprise documents to create the best possible adapted content version. The methods and algorithms used in the DU to evaluate the constraints and quality parameters, as well as the content optimization (quality prediction and optimal transcoding parameter selection), are presented in the next two chapters, as these constitute the core of our contribution. In the meantime, in the next section, we provide a full description of the content adaptation unit that we propose.

4.3 The content adaptation unit

In our search for an adaptation engine for enterprise documents, in particular for presentation slides, we found that several approaches are possible, as detailed in chapter 2, section 2.2. There are also several software packages which could be used to implement some of these approaches. We have opted for the OpenOffice suite for several reasons. It is available free of charge and its code is open-source, which gives us the ability to modify it and adapt it to our needs. Furthermore, it can be launched in server mode, and it accepts local and remote

connections. To interact with its filters and manipulate its documents, the OpenOffice suite has some APIs that can be used by tiered applications (OpenOffice, 2010). These APIs are known as UNO (*Universal Notation Objects*). Its filters allow enterprise documents to be adapted into various formats, such as JPEG, GIF, HTML, etc. Furthermore, the OpenOffice file format is based on XML, which makes its content interoperable with other languages. In the following subsections, we show how OpenOffice files are encoded and how they can be adapted into Web pages.

4.3.1 OpenOffice file format

OpenOffice documents are, like MS Office documents (version 2007 or later), archive files mainly comprising XML files. They can be opened by the majority of file compression tools, such as WinZip and RAR. As OpenOffice documents can be converted into MS Office documents, and vice versa, we focus on OpenOffice documents in this section. We show how they are encoded and how they can be adapted into Web pages that can be rendered on mobile Web browsers. Figure 4.3 provides an example of an OpenOffice document's content that has been opened by WinZip.

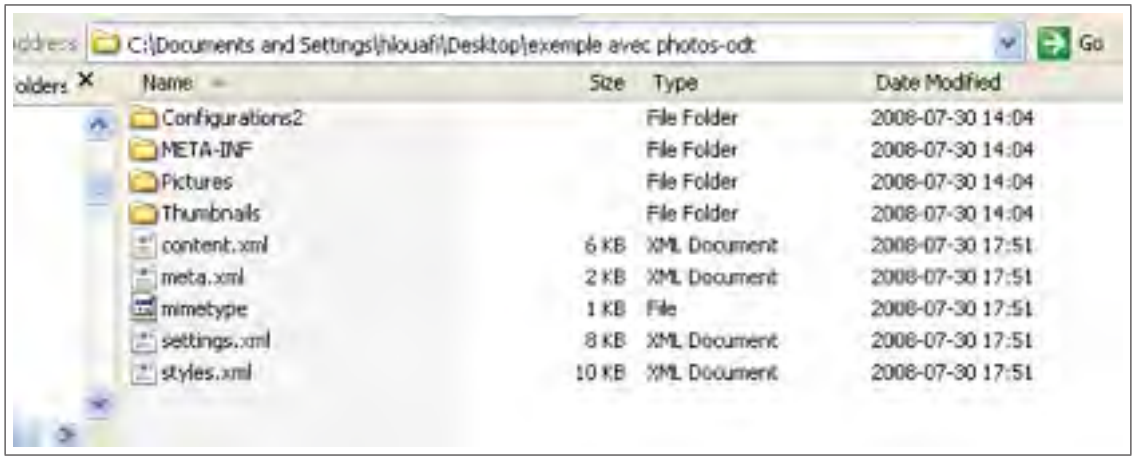

Figure 4.3 Example of an OpenOffice document's content.

The description and functions of the important files that make up an OpenOffice document are as follows:

- The file *content.xml* represents the core of the OpenOffice document. This is an XML file that contains the content (e.g. the text in a Word document), as well as the style information, such as bold, italic, and so on. The file comprises two sections. The first is reserved for the style information, and the second for the content itself.
- The file *meta.xml* is an XML file containing the meta data of the OpenOffice file. The information that can be found in this file is: author creator, creation date, number of pages, word count, and so on.
- The file *styles.xml* is an XML file containing all the style information available during file editing. It is similar to the CSS file that accompanies an XHTML file.
- The folder *META-INF* contains the archive file (manifest.xml), which lists all the files included in the archive file – see Figure 4.4 for an example of a *manifest.xml* file.
- The folder *Pictures* contains all the images that are embedded in an OpenOffice file. References to these images are already included in the file *content.xml*.

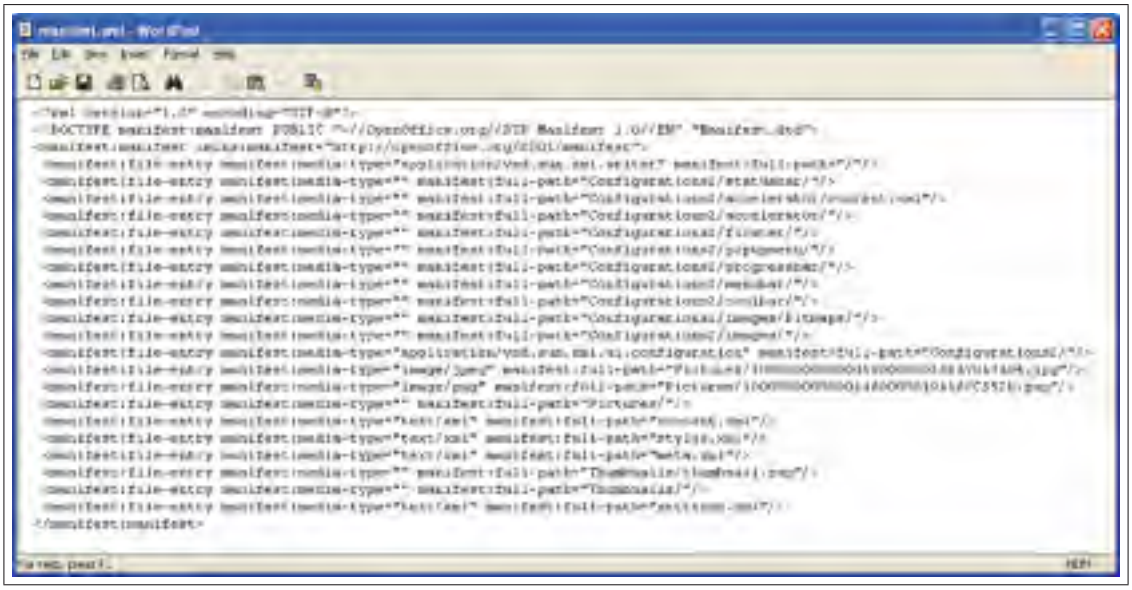

Figure 4.4 Content of the file *manifest.xml*.

4.3.2 OpenOffice document adaptation

Theoretically, since OpenOffice files are XML-based, they can be adapted to any desired format. For instance, using an XSLT engine, OpenOffice documents can converted to XHTML files and visualized on mobile Web browsers. This can be achieved using one of the following three methods:

1. Understand the logic used by OpenOffice to encode documents (the grammar), in order to write appropriate templates for the XSLT engine.

2. Perform the adaptation at a higher level (from a program perspective) using OpenOffice APIs. This method, a more interesting option, can be used to extract both the content and the formatting styles. The extracted information can then be recombined to produce an XHTML file.

3. Use OpenOffice filters, which are mainly composed of a set of XSLT templates.

Each method has its advantages and disadvantages. The first two methods are very time-consuming and require effort, since the templates (in method 1) must be created from scratch, and the APIs (in method 2) must be programmed from scratch. The use of the OpenOffice filters (method 3), as detailed in chapter 2, section 2.2, is not totally reliable, especially when the document includes images or graphics.

Adaptation with methods 2 and 3 should be performed on the server side. Conversely, with method 1 (in which we write our own templates), we can decide where the adaptation is performed, that is, on the server, on the Web browser, or on both. This leads to another question: Is it possible to perform the adaptation on the client side (mobile phone)? First, let us determine whether or not the XML adaptation is possible, and, if it is, at what level.

4.3.2.1 Server-side vs. client-side XML adaptation

There are three approaches to using XML adaptation in Web applications (Jacobs, 2006):

1. The XML content is adapted on the server, and preformatted (XHTML) data are sent to the Web browser. In this approach, the content is processed by the server's scripts (e.g. JSP) and sent to the Web browser wrapped in XHTML format at the same time as the styling information (CSS). Then, the Web browser, using the XHTML code and the styles, will be able to render the content correctly.

2. The XML content is extracted on the server, and the adaptation is performed either on the server or on the Web browser. If the latter supports XSLT, the adaptation is performed on the Web browser. Otherwise, we have no choice but to perform the adaptation on the server. In both cases, the styling information should be sent to the Web browser to be used to render the content.

3. The XML content is sent to the Web browser, where the adaptation should be performed. In fact, adaptation on the Web browser reduces the number of round-trips to the server. Without this technique, the Web browser should send a request to the server every time new data are needed. In this approach, the user downloads the application's interface (e.g. meeting presentation) and the styling information once, followed by the XML content. This way, all the modifications performed by the user are processed locally. If the Web browser is not able to perform the adaptation (no support for XSLT), the content is received as preformatted (XHTML) data. In this case, any modification to the document triggers a request to the server to update the document.

Table 4.1 summarizes the advantages and drawbacks of each approach.

The problem with the third approach is that the Web browser must have an appropriate level of XSLT support. Increasingly, recent mobile devices support XSLT. However, it is not clear when this support becomes profitable in terms of processing time.

4.3.2.2 Experimentation: XML adaptation on mobile devices

To answer the question raised in the previous subsection, we have conducted several experiments. Using an XHTML file that calls up a set of XML files (one at a time) and an XSLT

Table 4.1 XML adaptation (server-side vs. client-side).

Approach	Advantages	Drawbacks
Server-side adaptation	• Satisfies Web browsers that don't support XSLT. • Preformatted data are sent. • Mobile device resources are preserved (memory and CPU). • Battery life is saved.	• Server is overloaded.
Client-side adaptation	• Number of round-trips to the server are reduced. • Server load is reduced. • More autonomy for the Web browser.	• Use of mobile device resources, which are already reduced. • Accelerated drain on the battery.
Hybrid-side adaptation (client and server)	• The processing between the server and the Web browser is balanced. • Mobile device resources are exploited without reaching their limits.	• Content is subdivided according the mobile device's capability, which increases the burden on the server.

template, which parses the XML files, extracts some data, and formats them in a certain manner. The XML files created, which contain lists of CDs, were of various sizes. These files were extracted from (Jacobs, 2006) and can be found in Appendix B. The experiment involved opening the XHTML file with a conventional Web browser (MS-IE7) on a typical PC and with a mobile Web browser (Nokia N810). Every time the file was opened, the time taken by the XSLT adaptation process was measured.

Table 4.2 summarizes the time taken by each file to be rendered. The table shows that the XML adaptation on the mobile device becomes less effective as the file size increases. For example, we didn't get any response when we tried to convert (cdcatalog4.xml), a file of 6 KB. This

Table 4.2 Experimentation: XML adaptation on mobile devices.

File name	**Size (KB)**	**MS-IE7 Web browser Time (seconds)**	**Nokia N810 Time (seconds)**
cdcatalog1.xml	1.472	1	52
cdcatalog2.xml	2.002	1.5	69
cdcatalog3.xml	3.039	3	102
cdcatalog4.xml	6.072	5	No response
cdcatalog5.xml	9.105	9	No response
cdcatalog6.xml	12.139	15	No response

led us to discard the idea of sending the content to the mobile device to be processed locally. Current technologies don't permit this kind of processing (too CPU-intensive), and so only server-side processing is currently realistic.

However, other ideas could be investigated, such as subdividing the XML document into small parts and only sending one part at a time to the mobile device. In the presentation document case, this can be achieved by sending a limited number of slides at a time. The number of slides can be determined by the processing capability of the mobile device. At the same time, the impact of this technique on the server's capability should be taken into account, such as keeping track of which slides were sent to which user.

Rather than going further in that direction, we instead examined another technique that is more effective and requires less effort. This technique consists of using OpenOffice filters as a CAU. This solution can be effective, since OpenOffice has some filters that can be used to convert documents into different formats (e.g. HTML, PDF, JPEG, etc.).

4.3.3 OpenOffice filters

Using OpenOffice HTML filters [1], it is possible to adapt enterprise documents to two forms of Web pages. The first is a text-based Web page that contains only the document's text. The second is a raster-based Web page, which is, in fact, a raster image (GIF, JPEG, or PNG)

[1] In this thesis, we used OpenOffice version 3.1, which was the latest version available when this research began. Even now, the various issues raised and corrected in this version have not yet been resolved, as we explain later.

wrapped in an XHTML page skeleton. That is, each slide is converted into an image and wrapped in an XHTML page. As stated in the objectives of this research, the two formats under consideration are JPEG and XHTML. Therefore, our research focuses on the JPEG-based and text-based filters. The graphical interfaces of the OpenOffice filters require that the parameters that drive the adaptation be set as follows:

- If the desired format is a JPEG-based XHTML page: Quality factor (25, 50, 75, and 100) and resolution (640x480, 800x600, and 1024x768).
- If the desired format is a text-based XHTML page: There are no parameters to choose from.

Though a few values are offered via the graphical interfaces, precise values can be submitted to the filters programmatically, using OpenOffice APIs.

The Web pages produced by the JPEG-based filter are of good quality, that is, all the details of the slide are preserved in the Web page version. However, the text-based filter is very rudimentary and we found numerous bugs in it. For example, only the embedded text boxes are outputted to the Web page, and all the embedded images and graphics are discarded. Even the formatting styles are replaced with a single style (one font for the whole document). Moreover, the original layout is not preserved, that is, all the components of the presentation slide are serialized on the Web page.

Therefore, to produce acceptable XHTML Web pages comprising text and images, like any conventional Web page, we have extended the text-based filter. First, we studied its behavior and identified the templates it uses to adapt documents. Second, we identified numerous bugs and various functionalities that had not yet been implemented. Appendix C summarizes the details on the behavior of the text-based filter, and lists the bugs identified and how they were fixed, as well as the filter extensions. Here, this filter is called an XHTML-based filter. As a result, we have two filters that produce Web pages: JPEG-based and XHTML-based filters.

We now examine how texts are actually represented by OpenOffice filters. We look at which fonts are used, especially when it is a question of generating content for different platforms (different mobile devices and different operating systems).

4.3.4 Font issues in OpenOffice

The actual fonts of the document are used in the rendering process, if they are known to OpenOffice. Otherwise, they are replaced by other fonts. OpenOffice uses the OS system fonts (e.g. C:\Windows\Fonts). There is no specific (additional) directory of fonts in OpenOffice. Also, OpenOffice only partially supports TrueType fonts, and only some types of OpenType fonts. The support of OpenType fonts is limited to those resembling TrueType fonts (those with the extension .TTF and not those with the extension .OTF). Nevertheless, conversion from .OTF to .TTF is always possible (OpenOffice, 2008).

Furthermore, OpenOffice doesn't support embedded fonts[2] and ignores them when present. Instead, it uses a substitution method that can be set automatically (OpenOffice uses a default font whenever it finds an unknown font in a document) or manually (the user can set a mapping between fonts). In summary, OpenOffice will perfectly render the fonts present in the OS system font directory, and will replace all other fonts (even if they are embedded). If a presentation is created with MS Office and uses exotic (unknown) fonts and opens later with OpenOffice, the latter will use a default font, but will keep the font name as a reference. If we copy a piece of text that uses the exotic font from OpenOffice and paste it into an MS Office document, the latter will be rendered perfectly, because it still has the original font name. So, OpenOffice can always add new fonts to the OS system font directory.

Therefore, on the server side, we assume that all the fonts used by the enterprise document are known to OpenOffice (and stored in the OS system font directory), whatever the office suite used to create it (e.g. PowerPoint documents created by MS Office). This way, we can ensure that any enterprise document loaded by OpenOffice will be faithfully rendered prior to its adaptation.

[2]Even OpenOffice version 3.1 doesn't support embedded fonts.

On the client side (mobile device), rendering the adapted content depends on the OpenOffice filter selected. These filters function as follows:

- With the JPEG-based filter, after the transcoding parameters (e.g. resolution and quality factor) have been selected, the adapted document is rendered perfectly (as expected) on both standard Web browsers (Internet Explorer, FireFox, etc.) and mobile phones that support JPEG. The fonts are preserved, since the whole document page is converted into a JPEG image and wrapped in an XHTML skeleton.
- With the extended XHTML-based filter, all the fonts are preserved during the adaptation process. An adapted version of the enterprise document is generated, assuming that all the embedded fonts are available to the mobile device (the device has already downloaded the fonts). If the device doesn't support the fonts, then they will be substituted locally on the device. For instance, at the beginning of a meeting presentation, the Web application responsible for supporting mobile devices with slides could send a Web page containing a link where the fonts required for this presentation can be downloaded. The system could even track what fonts have been downloaded/installed by specific mobile devices, so that it would know when to send instructions to download new fonts. We could even provide such fonts to the mobile devices by creating font packages from the fonts offered by the OpenOffice OS system (in the right format). Alternatively, there could be a set of fonts available for download for every new user joining the Web application, in which case we could assume that users have installed the required fonts.

4.4 Summary

In this chapter, we have presented the architecture of the dynamic content adaptation framework that we propose. The transcoding engine, which comprises a decision unit and a content adaptation unit, constitute the core of this architecture. The content adaptation unit is composed of OpenOffice filters, namely the JPEG- and XHTML-based filters. The decision unit, particularly the content optimization module, is the most important component of the transcod-

ing engine. It contains the algorithms and methods to be used to select the best transcoding parameters. This topic is covered in the next two chapters.

CHAPTER 5

PREDICTION-BASED DYNAMIC CONTENT ADAPTATION FRAMEWORK FOR ENTERPRISE DOCUMENTS

To complete the description of the proposed prediction-based dynamic content adaptation system (see Figure 4.2) and of the way it functions, we now present the decision unit (DU), a key component of the system, in greater detail. As mentioned earlier, the device features and user preferences are extracted on the Web server and sent to the DU to be processed in order to identify the optimal transcoding parameters. From the device features, both the constraints and the quality parameters are extracted and processed separately. The constraint parameters are evaluated to ensure that the adapted content meets the target mobile device constraints. The quality parameters quantify the quality of the adapted content, making it possible to identify the optimal set of parameters.

In this chapter, we first mathematically formulate the problem of identifying the optimal transcoding parameters. Secondly, we introduce a quality of experience (QoE) measure that resolves the major drawbacks of the various techniques presented in the literature and described in chapter 3. Then, we show how the proposed QoE is evaluated, and how it can be predicted to allow dynamic computation of the optimal transcoding parameters. Finally, we present the experimental setup and results, and conclude with a discussion. The results presented in this chapter were published in (Louafi *et al.*, 2012).

5.1 Problem statement

Let $\mathcal{C}$ be an enterprise document, referred to here as "the original document" or "the content", composed of a set of pages (or slides) c_k made up of various components $c_{k,i}$ (e.g. text or images). We can write this formally as follows:

$$\begin{aligned} \mathcal{C} &= \{c_k\}_{k=1}^{n} \\ c_k &= \{c_{k,i}\}_{i=1}^{m(k)} \end{aligned} \tag{5.1}$$

where n is the total number of pages in $\mathcal{C}$, and $m(k)$ is the total number of components of the k^{th} page. For instance, $\mathcal{C}$ could be a PowerPoint presentation and c_k the k^{th} slide composed of various components $c_{k,i}$. Theoretically, a component can be any object. For instance, in a slide c_k composed of a text box and a JPEG image, $c_{k,1}$ represents the text box and $c_{k,2}$ represents the JPEG image.

For a page c_k, let $h_{k,1}$, $h_{k,2}$,..., and $h_{k,m}$ be sets of characteristics that can be adjusted to adapt that page's components, $c_{k,1}$, $c_{k,2}$,..., and $c_{k,m}$ respectively. For example, for a JPEG image (represented by $c_{k,2}$) embedded in a presentation, we may have the set $h_{k,2}$={resolution, quality factor}.

To be rendered by the target mobile device, the original document must often be adapted. Various adaptation operations can be used to achieve this, and, in principle, different transcoding parameter combinations can be used by the adaptation operations for each page. Let $\mathcal{P}$ be the possible transcoding parameters that can be used to adapt the original document's pages and their components. For simplicity, we concentrate on the adaptation of enterprise documents comprising text and JPEG images, bearing in mind that the concepts can be extended to other media types and formats. We have:

$$\mathcal{P} = \{f, z, QF\} \tag{5.2}$$

where:

- $f \in \{\text{JPEG}, \text{XHTML}\}$ is the output format into which the original page is to be transcoded,
- $z \in [0, 1]$ is a scaling factor that defines the output resolution of the adapted page, and
- $QF \in [0, 100]$ represents the quality factor of the outputted JPEG images on the adapted page.

These parameters are applied as follows:

- If, for a given page c_k, the selected output format is JPEG, the whole page is rasterized into a JPEG image and wrapped in an XHTML skeleton. As a result, the whole page is converted into a Web page that contains only one JPEG image. In this case, the parameters z and QF are used to create that JPEG image.
- If the selected output format is XHTML, the whole page is transformed into an XHTML file, which may include both text and images. As a result, the output XHTML file will contain the same number of components (text and images) as the original document. In this case, to preserve the initial intentions of the author of the original document, the same z is used for all the components of the original pages and the same QF for all the embedded JPEG images. By preserving the author's intentions, the adapted page will have the same layout (relative sizes and positions of embedded components) as the original one.

We define $\mathcal{T}$ as the transcoding operation that adapts the document page (or slide) c_k into a Web page using the transcoding parameters f, z, and QF, as follows:

$$\begin{aligned} \mathcal{T} : \mathcal{C} \times \mathcal{P} &\rightarrow \mathcal{C}^{f,z,QF} \\ c_k \times (f, z, QF) &\mapsto c_k^{f,z,QF} \end{aligned} \tag{5.3}$$

where $\mathcal{C}^{f,z,QF}$ is the set of all the possible adapted content versions that can be created by $\mathcal{T}$ from $\mathcal{C}$, using all the parameters from $\mathcal{P}$. In other words, $c_k^{f,z,QF}$ represents the adapted content version of c_k created by $\mathcal{T}$ using f, z, and QF.

Given a page c_k, let $\mathcal{W}(c_k)$ and $\mathcal{H}(c_k)$ be its width and height, in pixels, respectively.

Let D be the target mobile device and $\mathcal{W}(D)$, $\mathcal{H}(D)$, $\mathcal{S}(D)$, and $\mathcal{F}(D)$ be its maximum permissible image width, image height, file size (in bits), and supported formats respectively.

From the set of adapted content versions that can be created from c_k using $\mathcal{T}$, only a subset can be rendered by D. Let $\mathcal{R}(c_k, D)$ be the set of transcoding parameter combinations that can be

used to create these renderable versions:

$$\begin{aligned}\mathcal{R}(c_k, D) = \Big\{ (f, z, QF) \mid \mathcal{W}(c_k^{f,z,QF}) = z\mathcal{W}(c_k) \leq \mathcal{W}(D) \text{ and} \\ \mathcal{H}(c_k^{f,z,QF}) = z\mathcal{H}(c_k) \leq \mathcal{H}(D) \text{ and} \\ \mathcal{S}(c_k^{f,z,QF}) \leq \mathcal{S}(D) \text{ and} \\ f \in \mathcal{F}(D) \Big\}\end{aligned} \tag{5.4}$$

where $\mathcal{S}(c_k^{f,z,QF})$, $\mathcal{W}(c_k^{f,z,QF})$, and $\mathcal{H}(c_k^{f,z,QF})$ are the file size, and the width and height of the adapted content $c_k^{f,z,QF}$ respectively.

Since there could be multiple transcoding parameter combinations leading to adapted content versions renderable by D, the objective is to compute the ones that maximize the user's QoE, which we denote here by $\mathcal{Q}_{_E}(c_k^{f,z,QF}, D)$, and which will be defined in the next section. Let $\mathcal{R}^*(c_k, D) \subseteq \mathcal{R}(c_k, D)$ be the subset of optimal transcoding parameter combinations that maximize $\mathcal{Q}_{_E}(c_k^{f,z,QF}, D)$, which is given by:

$$\begin{aligned}\mathcal{R}^*(c_k, D) &= \Big\{ \big(f^*(c_k, D), z^*(c_k, D), QF^*(c_k, D)\big) \Big\} \\ &= \underset{(f,z,QF)\in\mathcal{R}(c_k,D)}{\arg\max} \mathcal{Q}_{_E}(c_k^{f,z,QF}, D)\end{aligned} \tag{5.5}$$

Note that there may be several solutions to (5.5). In this case, the parameters leading to the best visual quality are arbitrarily selected.

5.2 Proposed quality of experience measure

The quality of the delivered content, as experienced by the end-user, or *quality of experience* ($\mathcal{Q}_{_E}$), is affected by three factors (Kuipers *et al.*, 2010):

1. The quality of the content at the source, that is, the quality of the adapted content before delivery.

2. The quality of service QoS, which is affected by the delivery of the adapted content over the network.

3. The human perception of the adapted content.

In other words, $\mathcal{Q}_E$ is affected by the visual quality and transport quality (quality associated with the total delivery time). The first expresses how the content is appreciated visually, and the second expresses the impact of the total delivery time on the appreciation of the content. Based on these qualities, for a target mobile device D, we propose to evaluate the $\mathcal{Q}_E$ of the adapted content $c_k^{f,z,QF}$ as follows:

$$\mathcal{Q}_E(c_k^{f,z,QF}, D) = \mathcal{Q}_V(c_k^{f,z,QF}, D)\mathcal{Q}_T(c_k^{f,z,QF}, D) \tag{5.6}$$

where $0 \leq \mathcal{Q}_V \leq 1$ and $0 \leq \mathcal{Q}_T \leq 1$ represent the visual quality and the transport quality respectively. This is not the only way of evaluating the QoE, and, as explained in (Kuipers *et al.*, 2010), the $\mathcal{Q}_E$ and the QoS evaluation are completely separate research topics. In our framework, we propose the product of $\mathcal{Q}_V$ and $\mathcal{Q}_T$ rather than their sum, to prevent large disparities in $\mathcal{Q}_V$ and $\mathcal{Q}_T$ from being able to produce a high $\mathcal{Q}_E$. Indeed, the product is more appropriate than the sum, since $\mathcal{Q}_V$ and $\mathcal{Q}_T$ are not compensatory attributes. In our problem here, when a JPEG image is aggressively transcoded, its $\mathcal{Q}_T$ will be close to 1 (a very lightweight image) and its $\mathcal{Q}_V$ close to 0 (a very distorted image). If $\mathcal{Q}_V$ and $\mathcal{Q}_T$ are summed, the resulting $\mathcal{Q}_E$ will be close to 1, which is misleading. Unlike the sum, the product will be close to 0, which is more reasonable. In fact, before combining two or more attributes to obtain a single measure that reflects the nature of the problem in context, these attributes should first be classified into compensatory and non compensatory attributes. The former can be summed, whereas the latter cannot. This is the fruit of research performed elsewhere, particularly in the marketing and decision making fields (Lee and Anderson, 2009; Dieckmann *et al.*, 2009).

Although further research and validation are required to establish a metric that accurately matches the user's experience, the proposed metric is adopted here to illustrate the benefits over existing methods of performing prediction-based dynamic content adaptation. Similar

benefits are expected with other metrics which consider a compromise between visual quality and delivery time.

5.2.1 Visual quality evaluation

Let $c_k = \{c_{k,i}\}_{i=1}^{m(k)}$ be a page composed of a set of components, and $c_k^{f,z,QF}$ its adapted version created by $\mathcal{T}$, which comprises a set of adapted components and can be formulated as follows:

$$c_k^{f,z,QF} = \left\{ c_{k,1}^{f,z,QF}, c_{k,2}^{f,z,QF}, \ldots, c_{k,m(k,f)}^{f,z,QF} \right\} \tag{5.7}$$

where $c_{k,i}^{f,z,QF}$ is the i^{th} transcoded component and $m(k,f)$ is the total number of components. Ignoring the XHTML wrapper, which has no impact on quality in either case, it is given by:

$$m(k,f) = \begin{cases} m(k) & \text{if } f = \text{XHTML} \\ 1 & \text{if } f = \text{JPEG} \end{cases} \tag{5.8}$$

Of course, the visual quality of the adapted content depends on the visual quality of its components, but it also depends on the area occupied by each component (the larger the area, the larger the weight it should have in terms of quality). Therefore, we propose to compute the visual quality as a weighted sum of the visual quality of each of its components, each weight being the area that they occupy. So, we have:

$$\mathcal{Q}_V(c_k^{f,z,QF}, D) = \frac{\sum_{i=1}^{m(k,f)} \mathcal{A}(c_{k,i}^{f,z,QF}) \mathcal{Q}_V(c_{k,i}^{f,z,QF}, D)}{\sum_{i=1}^{m(k,f)} \mathcal{A}(c_{k,i}^{f,z,QF})} \tag{5.9}$$

$$\mathcal{Q}_V(c_{k,i}^{f,z,QF}, D) = \begin{cases} \mathcal{Q}_I(c_{k,i}^{f,z,QF}, D) & \text{if } c_{k,i}^{f,z,QF} \text{ is an image} \\ 1 & \text{if } c_{k,i}^{f,z,QF} \text{ is text} \end{cases} \tag{5.10}$$

where:

- $\mathcal{Q}_I$ measures image quality, such as PSNR or SSIM (Wang *et al.*, 2004) (or any other reliable full reference objective metric).
- We have assumed that the text is rendered perfectly, and, without loss of generality, the visual quality of the text components is set to 1. However, more sophisticated metrics could be used to take into account the resizing of the text components. The other textual characteristics, such as color, font, and the like, should not be affected, in order to preserve the original intentions of the document's author.
- $\mathcal{A}(c_{k,i}^{f,z,QF})$ is the visible (not hidden) area occupied by $c_{k,i}^{f,z,QF}$. We always have $\mathcal{A}(c_{k,i}^{f,z,QF}) \leq \mathcal{H}(c_{k,i}^{f,z,QF})\mathcal{W}(c_{k,i}^{f,z,QF})$, since two components are allowed to partially overlap one another. For instance, a text region can completely or partially overlap an image region. However, if they do, the hidden regions of an image should not be considered for computing either its region $\mathcal{A}$ or its visual quality $\mathcal{Q}_I$. This is particularly important when the page contains a background.
- When the adapted content $c_k^{f,z,QF}$ comprises one image (e.g. JPEG), its visual quality is reduced to the visual quality of that image, as is the case when the output format to be used is $f =$ JPEG.

5.2.2 Transport quality evaluation

The second factor that affects the user's QoE is transport quality. This factor is itself affected by the total delivery time, which is made up of the time required to perform the adaptation operation, plus the time taken by the adapted content to reach the target mobile device. For adapted content $c_k^{f,z,QF}$ and a target mobile device D, the total delivery time T_d can be defined as:

$$T_d\big(c_k^{f,z,QF}, D\big) = \frac{\mathcal{S}\big(c_k^{f,z,QF}\big)}{N_B(D)} + N_L(D) + S_L(D) + T_L(c_k^{f,z,QF}) \tag{5.11}$$

where:

- $\mathcal{S}(c_k^{f,z,QF})$ is the file size in bits of $c_k^{f,z,QF}$.

- $N_{B}(D)$ and $N_{L}(D)$ are the bitrate and latency of the network to which D is connected respectively.
- $S_{L}(D)$ is the server latency. For a device D, it represents the time spent by the request on the server (i.e. in the queue) waiting to be processed. This period of time depends on the performance of the server, but also on the number of requests waiting for the service. So, for a given server, this value may be different for each device.
- $T_{L}(c_{k}^{f,z,QF})$ is the transcoding latency. It represents how long the adaptation operation takes to complete, and depends on the original content, c_k, and the transcoding parameters f, z, and QF in use. It can be estimated based on past transcoding operations. On high-end computers, this value should be small.

There is no doubt that the longer it takes to deliver the adapted content, the less it is appreciated by the end-user. As the total delivery time increases, its perceived quality is reduced accordingly. That is, transport quality is inversely proportional to total delivery time. We therefore propose to evaluate transport quality using a normalization *Z-shaped built-in membership function* (Zmf) (The MathWorks, 2012). This was inspired by the work of (Lum and Lau, 2003; Zhang *et al.*, 2006a), in which the authors used the *sigmf* and *gaussmf* membership functions to normalize various parameters, such as the network bandwidth and latency (see chapter 3). This function, (Zmf), expresses the end-user's appreciation of (or frustration with) the adapted content, as a function of the wait time, in terms of a behavior. An example of such a behavior is depicted in Figure 5.1. In fact, the appreciation or frustration varies from one individual to another, which is why the values of α and β (see Figure 5.1) are used. These values can be determined by experience or defined by the end-user. The value α expresses the period of time in which the end-user is fully satisfied with the response time. The value $(\alpha + \beta)/2$ expresses the period of time in which that appreciation is reduced to 50%. When the total delivery time reaches the value β, the user's appreciation falls to 0. According to research performed to estimate the wait time that users will tolerate when accessing Web content (Nah, 2004; Ryan and Valverde, 2006), the values α and β can be set to model the user's actual behavior regarding

wait time. Thus, transport quality can be formulated as follows:

$$\begin{aligned} \mathcal{Q}_{_T}(c_k^{f,z,QF}, D) &= \mathrm{Zmf}(x, [\alpha, \beta]) \\ &= \begin{cases} 1 & \text{if } x \leq \alpha \\ 1 - 2\left(\frac{x-\alpha}{\beta-\alpha}\right)^2 & \text{if } \alpha \leq x \leq \frac{\alpha+\beta}{2} \\ 2\left(\frac{x-\beta}{\beta-\alpha}\right)^2 & \text{if } \frac{\alpha+\beta}{2} \leq x \leq \beta \\ 0 & \text{if } x \geq \beta \end{cases} \end{aligned} \tag{5.12}$$

where $x = T_d\big(c_k^{f,z,QF}, D\big)$.

Figure 5.1 Transport quality behavior for $\alpha = 5$ and $\beta = 10$.

5.3 Quality of experience estimation

If adapted content were available for every possible set of parameters, it would be straightforward to compute its QoE, using (5.6), and identify the optimal parameter set. The challenge with the dynamic content adaptation system that we propose is to be able to estimate the QoE of adapted content without having to perform any transcoding operation. This estimation process

is the key to the proposed system's reduced computational complexity. As $\mathcal{Q}_{_E}$ is a function of $\mathcal{Q}_{_V}$ and $\mathcal{Q}_{_T}$, the objective is to estimate $\mathcal{Q}_{_V}$ and $\mathcal{Q}_{_T}$.

For adapted content $c_k^{f,z,QF}$ and a target mobile device D, let $\hat{\mathcal{Q}}_{_V}(c_k^{f,z,QF}, D)$, $\hat{\mathcal{Q}}_{_T}(c_k^{f,z,QF}, D)$, and $\hat{\mathcal{Q}}_{_E}(c_k^{f,z,QF}, D)$ be its estimated visual quality, transport quality, and QoE respectively. Using equation 5.6, the estimated QoE becomes:

$$\hat{\mathcal{Q}}_{_E}(c_k^{f,z,QF}, D) = \hat{\mathcal{Q}}_{_V}(c_k^{f,z,QF}, D)\hat{\mathcal{Q}}_{_T}(c_k^{f,z,QF}, D) \tag{5.13}$$

We now examine how the visual and transport qualities can be estimated.

5.3.1 Visual quality estimation

As formulated in equation 5.9, the visual quality of adapted content is a function of the visual quality of its components and the areas they occupy.

The adapted content component areas can be known at runtime (when the content is requested). That is, if f = XHTML, these areas can be computed by scaling the areas of the original content's components using the scaling parameter z. When f = JPEG, the area of the whole of the original content is scaled using z. From equation 5.10, the visual quality of the adapted components is set to 1 for text, and for images it is defined as the adapted image's quality using a given quality metric. The hope is that the image quality can be predicted using a solution proposed in (Coulombe and Pigeon, 2009), in which the author shows that it is possible to estimate the SSIM of JPEG images (characterized by QF_{in}, their actual QF), subject to changing their scaling parameter (z) and quality factor (QF_{out}), and for viewing conditions (z_v). For an original content component $c_{k,i}$, the value of z_v controls the resolution, $z_v\mathcal{W}(c_{k,i}) \times z_v\mathcal{H}(c_{k,i})$, to which the original and the transcoded images should be scaled for comparison, in order to compute their SSIM. For instance:

- When $z_v = 1$, the two images are compared at the resolution of the original one.
- When $z_v = z$, the two images are compared at the resolution of the transcoded one.

- When $z_v = \min\left(\frac{\mathcal{W}(D)}{\mathcal{W}(c_{k,i})}, \frac{\mathcal{H}(D)}{\mathcal{H}(c_{k,i})}, 1\right)$, the two images are compared at the maximum resolution supported by the terminal or the original size of the image, whichever is smaller.

In practice, this value can be set by the maximum resolution of the target mobile device.

Specifically, when a JPEG image is transcoded using a scaling parameter z and a quality factor QF, the SSIM of the transcoded image can be estimated using the predicted data that are tabulated in (Coulombe and Pigeon, 2009), and which are indexed by QF_{in}, z_v, z, and QF. Table 5.1, which is extracted from that paper, shows a sub-array of predicted SSIM values of transcoded JPEG images characterized by their actual $QF_{in} = 80$, transcoded using z and QF, and evaluated under viewing conditions $z_v = 40\%$. As commercial products generally use a QF value between 75 and 85 to encode documents (or re-encode images) into JPEG images to preserve their visual quality, we present a sub-array for $QF_{in} = 80$ in Table 5.1. OpenOffice, for instance, proposes a default value of $QF = 75$. According to this table, we predict that SSIM=0.90 when an image encoded with $QF_{in} = 80$ is transcoded using $z = 50\%$ and $QF_{out} = 70$, and viewed at $z_v = 40\%$.

Note that these predicted SSIM values were computed by training and clustering, in which only the QF (QF_{in}) of the original image, and the transcoding parameters z, QF, and z_v were considered. A more sophisticated clustering method, taking into consideration two additional features (the number of bits per pixel of the original image and $QF_{out} - QF_{in}$) was proposed in (Pigeon and Coulombe, 2011) to improve prediction accuracy.

Using tables such as those in (Coulombe and Pigeon, 2009), the visual quality of adapted content can be estimated as follows:

- When the format to be used is $f =$ XHTML, the adapted content will comprise the same number of components as the original one. In this case, using the SSIM index, the visual

Table 5.1 Sub-array of predicted SSIM values computed for $QF_{in} = 80$ and $z_v = 40\%$. (Extracted from (Coulombe and Pigeon, 2009).

	Scaling, z,%									
QF_{out}	10	20	30	40	50	60	70	80	90	100
10	0.25	0.43	0.55	0.62	0.69	0.73	0.76	0.79	0.80	0.82
20	0.30	0.52	0.65	0.73	0.79	0.82	0.85	0.87	0.88	0.89
30	0.33	0.56	0.69	0.77	0.83	0.86	0.89	0.90	0.91	0.92
40	0.35	0.58	0.72	0.80	0.85	0.88	0.90	0.92	0.92	0.94
50	0.36	0.61	0.74	0.82	0.87	0.90	0.92	0.93	0.94	0.95
60	0.38	0.63	0.76	0.84	0.89	0.92	0.93	0.94	0.95	0.96
70	0.39	0.65	0.78	0.86	0.90	0.93	0.94	0.95	0.95	0.97
80	0.42	0.68	0.81	0.89	0.93	0.95	0.96	0.96	0.97	1.00
90	0.45	0.72	0.85	0.92	0.95	0.96	0.97	0.97	0.98	0.99
100	0.49	0.78	0.91	0.97	0.98	0.98	0.99	0.99	0.99	1.00

quality of the adapted content's components (5.10) becomes:

$$\mathcal{Q}_{_V}(c_{k,i}^{f,z,QF}, D) = \begin{cases} SSIM(c_{k,i}^{XHTML,z,QF}, c_{k,i}, z_v) & \text{if } c_{k,i} \text{ is an image} \\ 1 & \text{if } c_{k,i} \text{ is text} \end{cases} \tag{5.14}$$

where $c_{k,i}^{XHTML,z,QF}$ is the XHTML transcoded version of $c_{k,i}$.

The SSIM of the embedded images of the adapted content can be estimated using the predicted SSIM values (Coulombe and Pigeon, 2009), and so the estimated visual quality of the adapted components becomes:

$$\hat{\mathcal{Q}}_{_V}(c_{k,i}^{f,z,QF}, D) = \begin{cases} \widehat{SSIM}(z_v, QF_{in}(c_{k,i}), z, QF) & \text{if } c_{k,i} \text{ is an image} \\ 1 & \text{if } c_{k,i} \text{ is text} \end{cases} \tag{5.15}$$

where $QF_{in}(c_{k,i})$ represents the quality factor of $c_{k,i}$, and $\widehat{SSIM}(z_v, QF_{in}(c_{k,i}), z, QF)$ is the estimated SSIM value that can be extracted from the predicted SSIM arrays using z_v, $QF_{in}(c_{k,i})$, z, and QF. In this way, the visual quality of the adapted content can be estimated using (5.9). This process is illustrated in Figure 5.2.

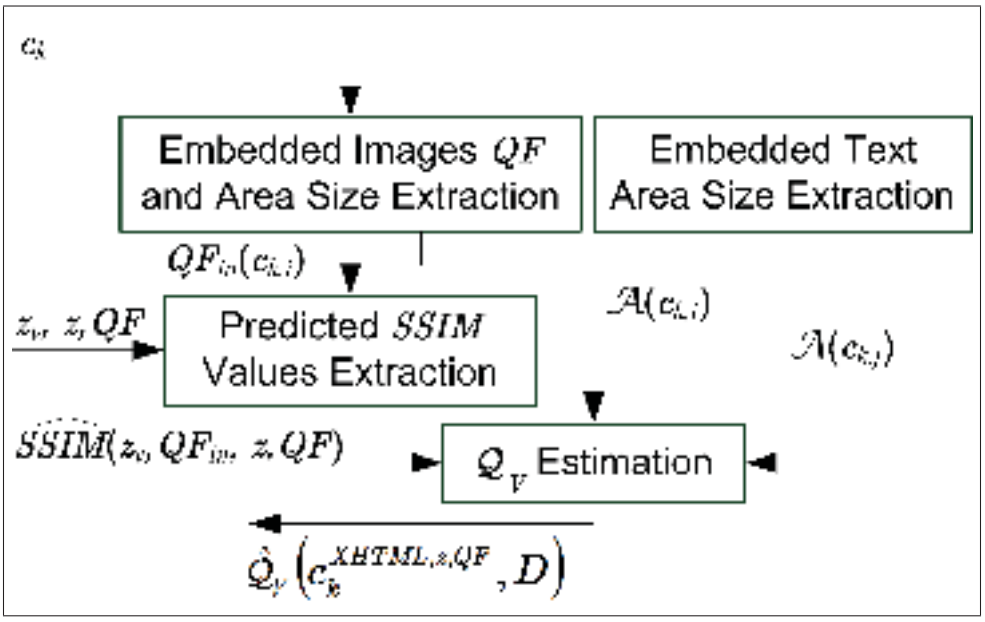

Figure 5.2 XHTML $\mathcal{Q}_V$ estimation.

- When the format to be used is f = JPEG, the adapted content will comprise only one JPEG image. Let $c_{k,1}^{JPEG,z,QF}$ be this image, transcoded at z and QF, and $c_{k,1}^{JPEG,100\%,80}$ be the image created using $z = 100\%$ and $QF = 80$, which will be used as a reference image. Note that in estimating the visual quality, the image $c_{k,1}^{JPEG,100\%,80}$ is not actually created. It is mentioned here only to illustrate the visual quality estimation process. However, from (Pigeon and Coulombe, 2008), this reference image ($c_{k,1}^{JPEG,100\%,80}$) is needed to estimate the file size of the adapted content, as described in section 5.3.2. Now, using the SSIM index, the visual quality of the adapted content (5.9) becomes:

$$\begin{aligned} \mathcal{Q}_V(c_k^{f,z,QF}, D) &= \mathcal{Q}_V(c_{k,1}^{JPEG,z,QF}, D) \\ &= SSIM(c_{k,1}^{JPEG,z,QF}, c_{k,1}^{JPEG,100\%,80}, z_v) \end{aligned} \tag{5.16}$$

Similarly, this visual quality can be estimated using the predicted SSIM values computed for various z_v and QF_{in}, as illustrated in Figure 5.3. For example, Table 5.1 shows such values for $z_v = 40\%$ and $QF_{in} = 80$. In this case, the estimated visual quality of the

adapted content becomes:

$$\hat{\mathcal{Q}}_V(c_k^{f,z,QF}, D) = \widehat{SSIM}(z_v, 80, z, QF) \tag{5.17}$$

where $\widehat{SSIM}(z_v, 80, z, QF)$ is the estimated SSIM value that can be extracted from the predicted SSIM arrays using z_v, $QF_{in}(c_{k,1}^{JPEG,100\%,80}) = 80$, z, and QF. For example, using Table 5.1, we obtain: $\widehat{SSIM}(40\%, 80, 50\%, 80) = 0.93$.

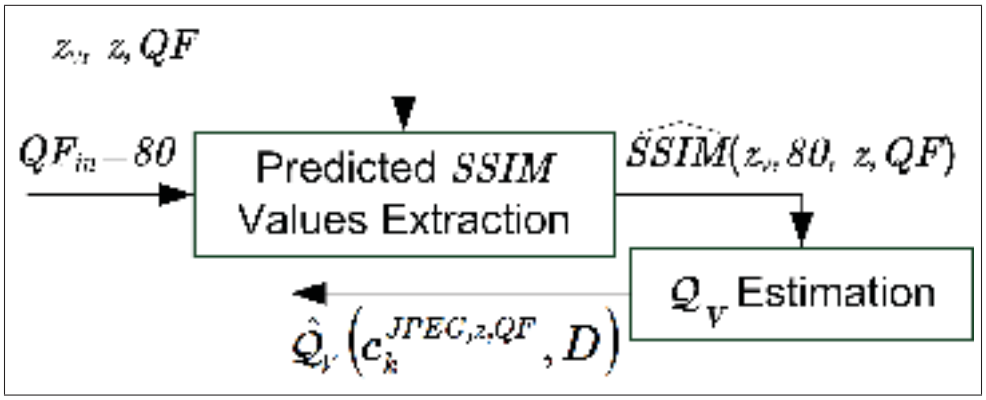

Figure 5.3 JPEG $\mathcal{Q}_V$ estimation.

5.3.2 Transport quality estimation

To compute the total delivery time (equation 5.11) and its associated transport quality (equation 5.12), all their variables must be estimated at runtime if they are not available (unknown). To estimate the network bitrate at runtime, various algorithms have been proposed (Ningning and Steenkiste, 2003). The network latency is generally estimated by "pinging" the target mobile device at runtime, or by taking a mean value of previous probings that could have been performed when the user registered (Svoboda *et al.*, 2007).

The challenge is to compute the adapted content file size, which can be estimated using the method proposed in (Pigeon and Coulombe, 2008). In this method, when a JPEG image $c_{k,i}$ (characterized by its QF, denoted QF_{in}) is transcoded into another JPEG image ($c_{k,i}^{JPEG,z,QF}$) using a scaling parameter z and quality factor QF (QF_{out}), the relative file size between them,

denoted r, can be predicted, as follows:

$$r(c_{k,i}^{JPEG,z,QF}, c_{k,i}) = \frac{\mathbb{S}(c_{k,i}^{JPEG,z,QF})}{\mathbb{S}(c_{k,i})} \tag{5.18}$$

For instance, Table 5.2 shows predicted relative file sizes for $QF_{in} = 80$ and various values of z and QF (or QF_{out}).

As explained in section 5.3.1, these predictors were computed by training and clustering, where only QF_{in}, z, and QF were considered. In (Pigeon and Coulombe, 2011), the authors proposed two new features (the original image's number of bits per pixel and $QF_{out} - QF_{in}$) to increase prediction accuracy.

Table 5.2 Sub-array of predicted relative file sizes computed for $QF_{in} = 80$. (Extracted from (Pigeon and Coulombe, 2008)).

	Scaling, z,%									
QF_{out}	10	20	30	40	50	60	70	80	90	100
10	0.03	0.04	0.05	0.07	0.08	0.10	0.12	0.15	0.17	0.20
20	0.03	0.05	0.07	0.09	0.12	0.15	0.19	0.22	0.26	0.32
30	0.04	0.05	0.08	0.11	0.15	0.19	0.24	0.21	0.34	0.41
40	0.04	0.06	0.09	0.13	0.17	0.22	0.28	0.34	0.40	0.50
50	0.04	0.06	0.10	0.14	0.19	0.25	0.32	0.39	0.46	0.54
60	0.04	0.07	0.11	0.16	0.22	0.28	0.36	0.44	0.53	0.71
70	0.04	0.08	0.13	0.18	0.25	0.33	0.42	0.52	0.63	0.85
80	0.05	0.09	0.15	0.22	0.31	0.41	0.52	0.65	0.78	0.95
90	0.06	0.12	0.21	0.31	0.44	0.59	0.75	0.93	1.12	1.12
100	0.10	0.24	0.47	0.75	1.05	1.46	1.89	2.34	2.86	2.22

The predicted relative file size can be used to compute the total delivery time (T_d) and its associated quality ($\mathbb{Q}_T$), as follows:

- When the format to be used is $f =$ XHTML, the file size of the adapted content can be computed by summing the file sizes of its embedded images and text boxes, and then

adding an additional data size related to the XHTML wrapper, as follows:

$$\mathcal{S}\left(c_k^{f,z,QF}\right) = \sum_{i=1}^{m(k)} r(c_{k,i}^{XHTML,z,QF}, c_{k,i})\mathcal{S}(c_{k,i}) + \psi \tag{5.19}$$

where $\mathcal{S}(c_{k,i})$ represents the file size of the components $c_{k,i}$ and $r(c_{k,i}^{XHTML,z,QF}, c_{k,i})$ the relative file size between $c_{k,i}$ and its XHTML transcoded version $c_{k,i}^{XHTML,z,QF}$. ψ represents the XHTML wrapper data size.

Using the predicted relative file sizes (Pigeon and Coulombe, 2008) (e.g. Table 5.2), the adapted content's file size can be estimated as illustrated in Figure 5.4. For instance, Table 5.2 shows that an image transcoded using $QF_{out} = 80$ and $z = 80\%$ will occupy 65% of its original file size. Formally, the estimated file size is given by:

$$\hat{\mathcal{S}}\left(c_k^{f,z,QF}\right) = \sum_{i=1}^{m(k)} \hat{r}(c_{k,i}^{XHTML,z,QF}, c_{k,i})S(c_{k,i}) + \psi \tag{5.20}$$

$$\hat{r}(c_{k,i}^{XHTML,z,QF}, c_{k,i}) = \begin{cases} \hat{r}_{_I}(QF_{in}(c_{k,i}), z, QF) & \text{if } c_{k,i} \text{ is an image} \\ 1 & \text{if } c_{k,i} \text{ is text} \end{cases} \tag{5.21}$$

where:

- $QF_{in}(c_{k,i})$ is the QF of the original image $c_{k,i}$.
- $\hat{r}_{_I}(QF_{in}(c_{k,i}), z, QF)$ is the estimated relative file size between the image $c_{k,i}$ and its transcoded version $c_{k,i}^{XHTML,z,QF}$, which can be extracted from the predicted relative file sizes arrays tabulated in (Pigeon and Coulombe, 2008) (Table 5.2 shows such an array for $QF_{in} = 80$ and various values of z and $QF = QF_{out}$).
- ψ represents the added size of the XHTML wrapper. Typically, the file size of the XHTML wrapper for one slide is equal to 1 KB. Therefore, we set $\psi = 1$KB.

It is interesting to note that although the file size prediction model does not use explicit statistics related to the compressed form of the input image (such as the number of zeroed DCT coefficients), it implicitly takes into account the compressibility of the original image through its file size $\mathcal{S}(c_{k,i})$.

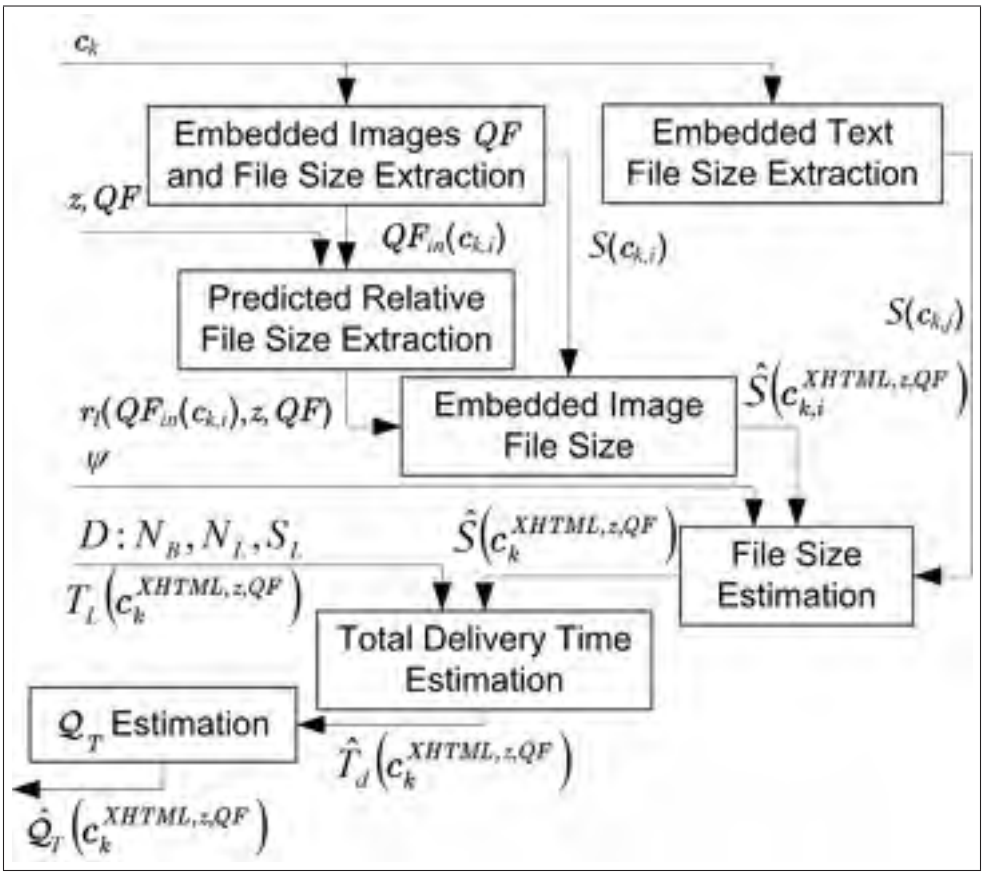

Figure 5.4 XHTML $\mathcal{Q}_T$ estimation.

- When the format to be used is $f =$ JPEG, using the reference JPEG image created before ($c_{k,1}^{JPEG,100\%,80}$), the file size of the adapted content becomes:

$$\mathcal{S}\left(c_k^{f,z,QF}\right) = r(c_{k,1}^{JPEG,z,QF}, t_{k,1}^{JPEG,100\%,80})\mathcal{S}(c_{k,1}^{JPEG,100\%,80}) + \psi \tag{5.22}$$

The file size of the adapted content can be estimated using the predicted relative file size (Pigeon and Coulombe, 2008), as illustrated in Figure 5.5. Formally, the estimated file size of the adapted content is given by:

$$\hat{S}\left(c_k^{f,z,QF}\right) = \hat{r}(c_{k,1}^{JPEG,z,QF}, c_{k,1}^{JPEG,100\%,80})\mathcal{S}(c_{k,1}^{JPEG,100\%,80}) + \psi \tag{5.23}$$

$$\hat{r}(c_{k,1}^{JPEG,z,QF}, c_{k,1}^{JPEG,100\%,80}) = \hat{r}_I(QF_{in}(c_{k,1}^{JPEG,100\%,80}), z, QF) \\ = \hat{r}_I(80, z, QF) \tag{5.24}$$

where:

- $QF_{in}(c_{k,1}^{JPEG,100\%,80}) = 80$ is the quality factor of $c_{k,1}^{JPEG,100\%,80}$
- $\hat{r}_I(80, z, QF)$ is the estimated relative file size between the two images $c_{k,1}^{JPEG,z,QF}$ and $c_{k,1}^{JPEG,100\%,80}$, which can be extracted from Table 5.2.
- ψ is, as before, the XHTML wrapper size.

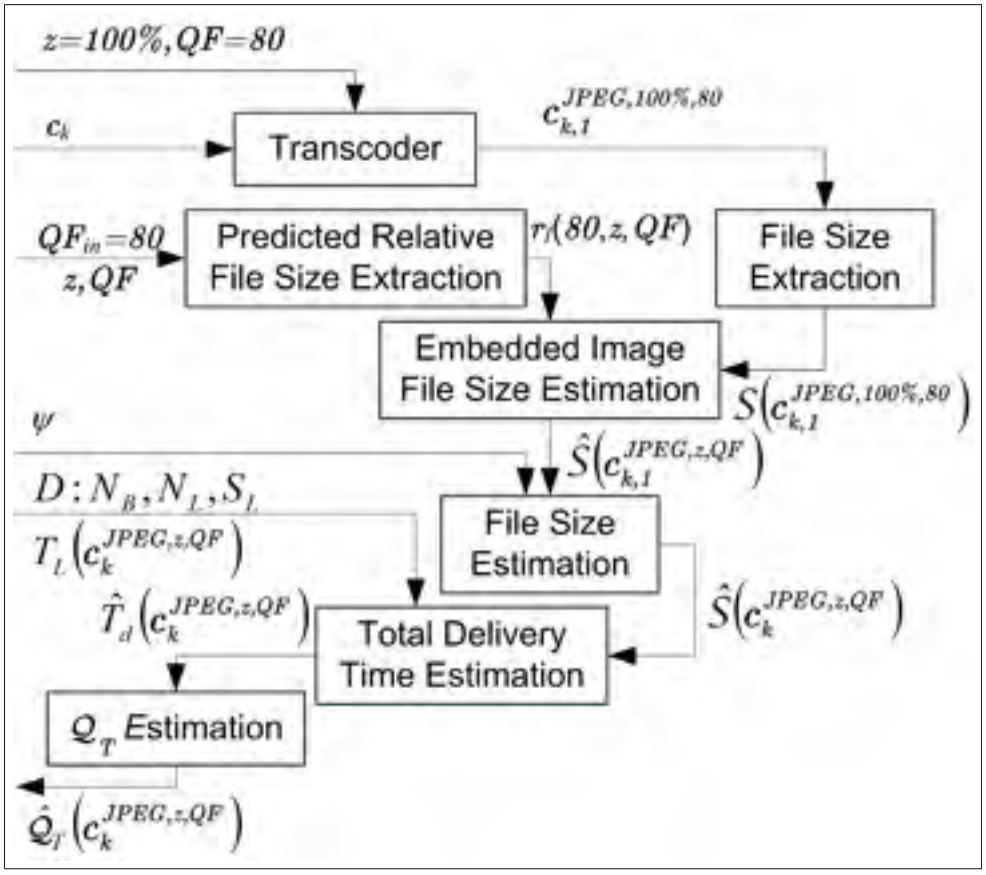

Figure 5.5 JPEG $\mathcal{Q}_T$ estimation.

Finally, after estimating the adapted content's visual and transport qualities, its $\mathcal{Q}_E$ can be estimated as shown in equation 5.13.

5.4 Proposed user's preferences model

As detailed in chapter 3, the user's preferences are usually taken into account in the process of adapted content selection (or creation) to deliver optimal content which improves the user's experience. According to the literature review in chapter 3, the end-user can express his preferences by weighting the context quality parameters directly, such as network bandwidth and color depth (Lum and Lau, 2003; Zhang *et al.*, 2006a). We believe that this is a low-level technique that is not always well understood. Indeed, not all users are able to understand the context parameters, and even less so those related to the communication network (e.g. network latency).

Therefore, we propose to use a high-level technique in which terms are used that are easily understandable by non experts, and that hold real meaning for them. In this technique, the user is asked what kind of adapted content he prefers: content with good visual quality, or content that is delivered quickly. This can be done by weighting the visual quality ($\mathcal{Q}_V$) and the transport quality ($\mathcal{Q}_T$) using linguistic expressions (e.g. less important, important, and very important), or using values ranked between 0 and 1. Even when linguistic expressions are used, the system should be able to convert these into ranked values, using fuzzy functions, for example, as proposed in (Zhang *et al.*, 2006a). Note that ranked values are multiplied by the attribute values and then summed when the function to optimize is formulated as a simple additive weighting (SAW). However, when the problem is formulated as a weight product (WP), the weights are applied differently.

Generally, an optimization function comprises of a set of attributes that maximizes (or minimizes) the function's final score and a set of attributes minimizing (or maximizing) that score. The attributes in the first set are called *benefit* attributes, and those in the second set are called *cost* attributes. When the problem is formulated as a product, positive exponents are used with the benefit attributes, and negative exponents are used with the cost attributes (Yoon and Hwang, 1995).

Since $\mathcal{Q}_{_E}$ is formulated as a product of two attributes ($\mathcal{Q}_{_V}$ and $\mathcal{Q}_{_T}$), the weights become exponents associated with these attributes. Besides, since both $\mathcal{Q}_{_V}$ and $\mathcal{Q}_{_T}$ are benefit attributes, they should be associated with positive exponents. In fact, the total delivery time could be considered as a cost attribute if it is used instead of $\mathcal{Q}_{_T}$; however, $\mathcal{Q}_{_T}$ is computed in such a way as to reverse this behavior using a Zmf function (see Figure 5.1).

Using the user preference model that we are proposing, the adapted content $\mathcal{Q}_{_E}$, equation 5.6, and its estimated value $\hat{\mathcal{Q}}_{_E}$, equation 5.13, becomes:

$$\mathcal{Q}_{_E}(c_k^{f,z,QF}, D) = \Big(\mathcal{Q}_{_V}(c_k^{f,z,QF}, D)\Big)^{\mathcal{W}_{_V}(D)} \Big(\mathcal{Q}_{_T}(c_k^{f,z,QF}, D)\Big)^{\mathcal{W}_{_T}(D)} \tag{5.25}$$

$$\hat{\mathcal{Q}}_{_E}(c_k^{f,z,QF}, D) = \Big(\hat{\mathcal{Q}}_{_V}(c_k^{f,z,QF}, D)\Big)^{\mathcal{W}_{_V}(D)} \Big(\hat{\mathcal{Q}}_{_T}(c_k^{f,z,QF}, D)\Big)^{\mathcal{W}_{_T}(D)} \tag{5.26}$$

where, for a user, represented by his mobile device D, $\mathcal{W}_{_V}(D)$ and $\mathcal{W}_{_T}(D)$ are the weights associated with visual quality and transport quality respectively. For convenience and to prevent these weights (exponents) from being able to reduce $\mathcal{Q}_{_V}$ and $\mathcal{Q}_{_T}$ to very small quantities, we propose to confine them within the $]0, 1]$ range. The value of 0 is trivial, and, when associated with an attribute, becomes useless. For this reason, it is excluded, and we have:

$$\begin{cases} \mathcal{W}_{_V}(D) \in]0, 1] \\ \mathcal{W}_{_T}(D) \in]0, 1] \end{cases} \tag{5.27}$$

Since the attributes $\mathcal{Q}_{_V}$ and $\mathcal{Q}_{_T}$ are normalized values between 0 and 1, the weights are interpreted as penalties (increasing the weight value will decrease the value of the weighted attribute). In fact, when the base of the exponential function is between 0 and 1 (which is the case here), the function decreases monotonically, as depicted in Figure 5.6.

5.5 Use of the proposed dynamic content adaptation framework

To estimate the optimal combination of transcoding parameters that should be used to adapt original content c_k, we compute, two arrays $\hat{\mathcal{Q}}_{_V}(c_k^{f,z,QF}, D)$ and $\hat{r}(c_k^{f,z,QF})$, using all the com-

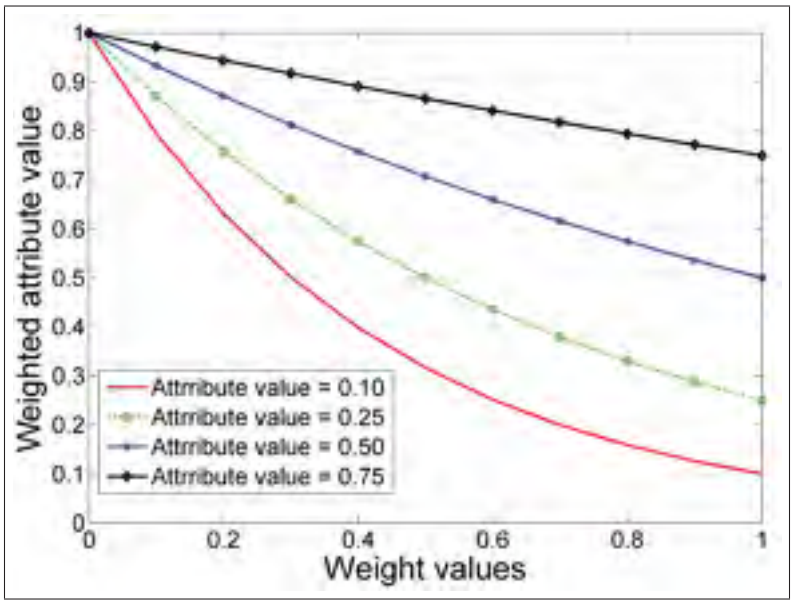

Figure 5.6 Weighted attribute behavior for some attribute values within the $[0, 1]$ range using an exponential function, the base of which is within that range.

binations of f, z, and QF. Based on these arrays, we compute the estimated QoE array $\hat{\mathcal{Q}}_E(c_k^{f,z,QF}, D)$, on which we solve (5.5) to determine the best solution (i.e. the best combination of transcoding parameters). We expect the solution to be near-optimal.

5.6 Experimental setup

5.6.1 Slides corpus

To test and validate the proposed framework, a large corpus of enterprise documents is required. In this thesis, we validate our framework by means of presentation slides, as they constitute the most widely used content in Web conferencing environments. Such slides could have been collected from the Web. However, to test and analyze the proposed framework on a wide range of slide types, we preferred to create slides composed of components of various sizes and occupying different positions on the slide. To be representative of existing content, both the images and the text were collected from Web sites such as (The USC-SIPI, 2012). To create our slide corpus, we developed a Java-based application that uses OpenOffice APIs (UNO) to create a set of Impress slides (OpenOffice, 2010). The size of our slides varied between 12 KB

and 122 KB. Note that a slide can contain text and images that share the same area (overlap). The background was set to None (no master style); however, an inserted image could cover 100% of the slide, and therefore be considered as background. The positions of the text boxes and images on the slides were set randomly by a random number generator which is part of the same application. Since the dimensions of the images and text boxes could be continuous, and to avoid context dilution, quantized values representing the percentage of areas occupied by images (I) and text boxes (T) were used, as follows:

$$\begin{aligned} I &\in \{0\%, 10\%, 20\%, \ldots, 100\%\} \\ T &\in \{0\%, 10\%, 20\%, \ldots, 100\%\} \end{aligned} \tag{5.28}$$

For instance, $I = 40\%$ and $T = 30\%$ mean that the area occupied by the images represents 40% of the slide, and that occupied by the text boxes 30%. An example of a slide composed of $I = 40\%$ and $T = 25\%$ is shown in Figure 5.7(a).

To facilitate validation, each document constitutes one slide. This restriction, which can be removed later, does not affect the credibility of the validation, since each slide can be seen as separate content, and so is converted and sent separately. Let $\mathcal{V}$ be this validation set.

5.6.2 Transcoding methodology

To compare the quality of the transcoded content, each slide from $\mathcal{V}$ is transcoded using OpenOffice JPEG and XHTML filters, which produce JPEG- and XHTML-based Web pages respectively. The first filter converts the whole slide into an image and wraps it in a skeleton Web page. In the proposed dynamic framework, to be able to estimate the $\hat{\mathrm{Q}}_V$ and $\hat{r}$ of any adapted content when the format used is JPEG, the JPEG image created, $c_{k,1}^{JPEG,100\%,80}$, is used as a reference image from which the other images ($c_{k,i}^{JPEG,z,QF}$) are created using ImageMagick command line tools (ImageMagick, 1999). These images replaced those created by the JPEG-based filter. Consequently, we can use the predicted SSIM and relative file sizes tabulated in (Coulombe and Pigeon, 2009; Pigeon and Coulombe, 2008) to estimate the $\hat{\mathrm{Q}}_V$ and $\hat{r}$ of the adapted content.

As mentioned in the description of the DU (see section 4.3), the native OpenOffice XHTML filter capability was very limited, and was found to have numerous bugs. Ultimately, we improved the filter by fixing these important bugs and limitations, and adding the possibility of manipulating images and their characteristics, such as the ability to scale and change the quality factor of embedded JPEG images. After these extensions were added, the modified OpenOffice XHTML filter was able to convert the slide into a standard XHTML file (which could include both text and images) using the transcoding parameters z and QF. An example of a slide as exported by the extended OpenOffice XHTML filter is shown in Figure 5.7. Figure 5.7(a) shows the original slide as rendered by OpenOffice, and Figures 5.7(b), 5.7(c), and 5.7(d) show its exported versions with the extended XHTML filter using $z = 30\%$ and $QF = 80$, $z = 80\%$ and $QF = 80$, and $z = 50\%$ and $QF = 60$ respectively.

A full description of the native OpenOffice XHTML filter bugs and limitations that have been identified and corrected can be found in Appendix C.

The sets of transcoding parameters used by these filters, and used as explained in section 5.1, are the following:

$$f \in \{\text{JPEG}, \text{XHTML}\}$$
$$z \in \{10\%, 20\%, 30\%, \ldots, 100\%\}$$
$$QF \in \{10, 20, 30, \ldots, 100\}$$

We finally define $\mathcal{W}$, the set of adapted content created from the original content of $\mathcal{V}$ using the transcoding parameters f, z, and QF.

5.6.3 Validation methodology

The XHTML filters of OpenOffice (or MS Office suite), which produce JPEG-based XHTML pages, offer the option of selecting the target resolution and JPEG quality factor. Though few parameters are offered via their graphical interfaces (high, low, and medium quality), more precise parameters can be programmed using their APIs. This is what is done in commercial

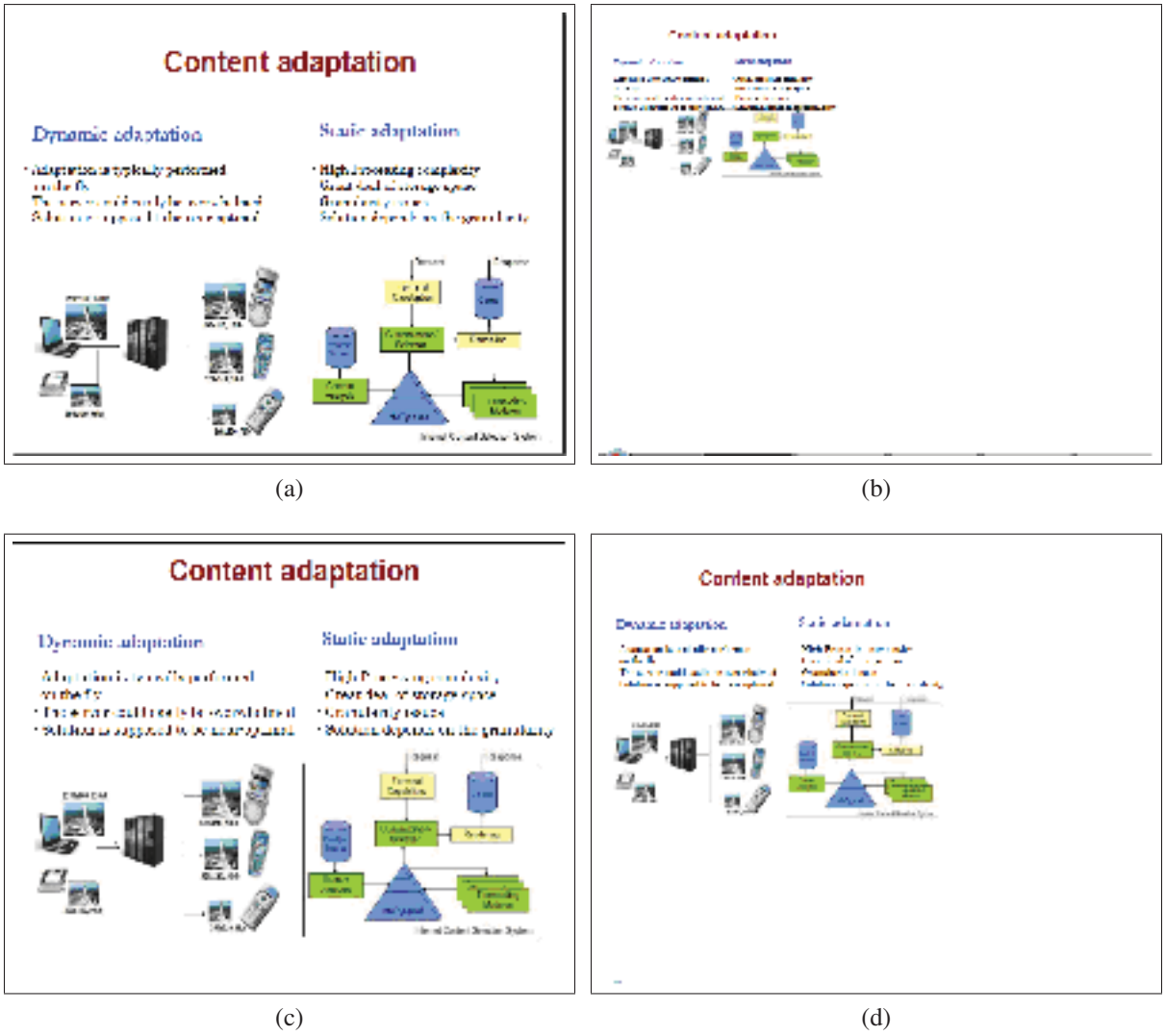

Figure 5.7 A slide as exported by our extended OpenOffice XHTML filter: (a) the original slide, (b) transcoded using $z = 30\%$ and $QF = 80$, (c) transcoded using $z = 80\%$ and $QF = 80$, (d) transcoded using $z = 50\%$, $QF = 60$

dynamic solutions (Li and Chandra, 2008). However, these solutions do not use any QoE criterion in tailoring the content. They typically adjust the JPEG resolution to the maximum resolution of the target mobile device and use a fixed JPEG quality factor (e.g. 80) to provide good visual quality, regardless of the resulting file size (which negatively affects transport quality). This kind of dynamic system is denoted below as a *fixed-QF system*. By contrast, with static solutions, different versions of the content are created, and so the QoE of each can be taken into account to select the best version.

We now compare our method with a typical dynamic system (fixed-QF system) and various static systems based on different granularity levels. Most static transcoding systems create different versions to suit a wide variety of target mobile devices (Lum and Lau, 2003; Zhang *et al.*, 2006a; Mohan and Smith, 1999). When the content is requested by the mobile device, the best adapted version among those created in advance is selected for delivery. The granularity of the created versions should be adequate to deliver the best user experience possible. However, in practice, it is not always possible to reach that level of granularity, owing to numerous constraints, such as lack of storage space, CPU processing time limitations, etc. Consequently, we propose to use the following hypothetical static transcoding systems, which have been inspired by realistic needs:

- **Exhaustive static system:** This system creates the maximum possible number of adapted content versions. We can say that it uses all combinations of these quantized values: $z \in \{10\%, 20\%, \ldots, 100\%\}$ and $QF \in \{10, 20, \ldots, 100\}$ for both the JPEG and XHTML formats. As a result, it creates 200 versions for each slide. We assume that this system provides content of high enough granularity, and so constitutes a good benchmark for comparing the best adapted content provided by each system.
- **Granularity-based static systems:** In practice, it is not always possible, nor desirable, to transcode content into 200 or more versions. An Impress presentation composed of 30 slides, for example, would require the creation of 6,000 versions, which would mean that a server dedicated to organizing meetings would have to handle a very large number of versions, involving a very long processing time and a great deal of storage space. As an alternative, we might consider using only a limited number of values for z and QF. Since the most widely used quality factor is 80, we propose to compare our solution with ten systems based on that quality factor and various quantized values of z. From the first system to the tenth, the granularity is enriched gradually, always building onto the previous system (i.e. system $i + 1$ adds one more version on top of system i, the characteristics of which are selected to cover the parameter set). For instance, the first system creates only one version (using $z = 100\%$) for each format, while the next system creates two versions (using $z = 100\%$ and 50%), and so on, as shown in Table 5.3. This

is not the only possible schema, and other sets of systems could be used, depending on the available resources, such as varying QF above or below 80 (e.g. 70 or 60).

To review, our dynamic framework is compared below with a fixed-QF dynamic system, an exhaustive static system, and ten granularity-based static systems. These systems are summarized in Table 5.3.

Table 5.3 Transcoding systems used in the validation.

N	Systems	Scaling, z,%									
1	Granularity-based static systems ($QF = 80$)	100									
2		100	50								
3		100	50	70							
4		100	50	70	30						
5		100	50	70	30	80					
6		100	50	70	30	80	60				
7		100	50	70	30	80	60	40			
8		100	50	70	30	80	60	40	90		
9		100	50	70	30	80	60	40	90	20	
10		100	50	70	30	80	60	40	90	20	10
11	Exhaustive static system	$z \in \{10\%, 20\%, \ldots, 100\%\}$ $QF \in \{10, 20, \ldots, 100\}$									
12	Fixed-QF dynamic system	z is based on the target mobile device's resolution $QF = 80$									

Note that the proposed static systems are actually more sophisticated than those that are commonly used. Typical static systems select the highest resolution of the content supported by a device, regardless of the delivery time. But the proposed static systems, using the same $\mathcal{Q}_{_E}$ metric, will lead to a fairer comparison of the static and dynamic approaches, in terms of reaching their full potential.

The comparison focuses on two aspects. The first aspect is the performance of the proposed dynamic system, from a QoE perspective, relative to that of the dynamic fixed-QF system and the static systems (i.e. how many versions must a static system generate to match our system). We also compare the quality of the proposed dynamic system to that of the exhaustive static

system to measure how far we are from optimality. The second aspect is the storage space required by each system.

Each adapted content $c_k^{f,z,QF}$ in $\mathcal{W}$ is, in fact, a Web page. It is parsed, and its actual $\mathcal{Q}_{_V}(c_k^{f,z,QF}, D)$ and $r(c_k^{f,z,QF})$ values are computed. Next, we compute $\mathcal{Q}_{_T}(c_k^{f,z,QF}, D)$ and $\mathcal{Q}_{_E}(c_k^{f,z,QF}, D)$ for the target mobile device D. By contrast, the $\hat{\mathcal{Q}}_{_V}(c_k^{f,z,QF}, D)$ and $\hat{r}(c_k^{f,z,QF})$ of each slide c_k in $\mathcal{V}$ are computed using the proposed dynamic framework. We then compute $\hat{\mathcal{Q}}_{_T}(c_k^{f,z,QF}, D)$ and $\hat{\mathcal{Q}}_{_E}(c_k^{f,z,QF}, D)$ for this mobile device.

Note that the SSIM index exhibits a highly nonlinear relationship with the DMOS (Differential Mean Opinion Score), and therefore cannot be used directly as a measure of the human perception of quality. Therefore, to address the third requirement regarding the $\mathcal{Q}_{_E}$ design (Kuipers *et al.*, 2010) (see section 5.2), we map the SSIM values to their corresponding subjective MOS (Mean Opinion Score) values using a logistical function and regression (Sheikh *et al.*, 2012). In other words, we compute or estimate the SSIM, but then map it to its corresponding MOS value.

As a result, two arrays were created; one computed (QE), and the other estimated ($\widehat{QE}$). Their schemes are as follows:

$$\begin{aligned} QE &: \left[c_k, f, z, QF, \mathcal{Q}_{_V}(c_k^{f,z,QF}, D), \mathcal{Q}_{_T}(c_k^{f,z,QF}, D), \mathcal{Q}_{_E}(c_k^{f,z,QF}, D)\right] \\ \widehat{QE} &: \left[c_k, f, z, QF, \hat{\mathcal{Q}}_{_V}(c_k^{f,z,QF}, D), \hat{\mathcal{Q}}_{_T}(c_k^{f,z,QF}, D), \hat{\mathcal{Q}}_{_E}(c_k^{f,z,QF}, D)\right] \end{aligned} \tag{5.29}$$

The best adapted content obtained by the proposed validation process and transcoding systems are computed as follows:

- **Exhaustive static system:** The best adapted content is identified by solving (5.5) on the QE array for each slide c_k.
- **Granularity-based static systems:** First, a sub-array is obtained from QE by selecting the rows corresponding to the values of z and QF that define each system (see Table 5.3).

Then, the best adapted content for each slide c_k is identified by solving (5.5) on that sub-array.

- **Fixed-QF dynamic system:** The best adapted content for each slide c_k is obtained from QE by setting the value of z based on the maximum resolution of the target mobile device and $QF = 80$.
- **Proposed dynamic system:** The best transcoding parameters $\widehat{f}^*(c_k, D)$, $\widehat{z}^*(c_k, D)$, and $\widehat{QF}^*(c_k, D)$ are estimated by solving (5.5) on $\widehat{QE}$ for each slide c_k (i.e. solved using predicted values, not actual transcoding values). Using these optimal parameter estimates, the actual $\mathcal{Q}_E$ is retrieved from the QE array (i.e. from an actual transcoding operation), which corresponds to $\mathcal{Q}_E(c_k^{\widehat{f}^*(c_k,D),\widehat{z}^*(c_k,D),\widehat{QF}^*(c_k,D)}, D)$. The latter represents the actual QoE obtained by the proposed dynamic system, which is compared to the QoEs obtained by the other transcoding systems.

5.7 Experimental results

To compare the performance and precision of the proposed dynamic framework with the validation transcoding systems previously described (see Table 5.3), various aspects have been considered, such as $\mathcal{Q}_E$, the average deviation between the best adapted content obtained by each system and that of the exhaustive static system (i.e. optimality), and storage space issues. Since the computed data were too numerous to be presented here, we arbitrarily selected one scenario in which a mobile device available in the marketplace and a commercial communication network.

In this scenario, the proposed mobile device D is a Nokia N8 with a resolution of 640×360 and it is connected to a GPRS network with a bitrate of $N_B(D) = 50$ kbps and network latency of $N_L(D) = 488$ ms (Svoboda *et al.*, 2007). Since the default resolution of the slide, as rendered on a PC by the OpenOffice JPEG filter, is 1058×794, the maximum viewing conditions are computed by $\min\left(\frac{640}{1058}, \frac{360}{794}\right) \approx 45\%$, which suggests, from (Coulombe and Pigeon, 2009), a comparison of images at $z_v = 40\%$.

To define actual transport quality behavior (see section 5.3.1), α and β, which express the end-user's behavior regarding wait time, were set as follows: $\alpha = 5$s and $\beta = 10$s. These values are based on research conducted to estimate the wait time that users will tolerate when accessing Web content (Nah, 2004; Ryan and Valverde, 2006).

Let us say that the end-user of D has registered his preferences regarding the quality of the content he is requesting as follows: the best possible visual quality that can be received, and as quickly as possible. In other words, the end-user wants to maximize the visual quality and the transport quality at the same time, which can be interpreted as no preference for either of these qualities. This means that the same weight should be used for both qualities. Using the proposed user preference model (see section 5.4), the weights to be associated with $\mathcal{Q}_{_V}$ and $\mathcal{Q}_{_T}$ can be set as follows:

$$\begin{cases} \mathcal{W}_{_V}(D) = 1 \\ \mathcal{W}_{_T}(D) = 1 \end{cases} \tag{5.30}$$

These mobile device and network characteristics are tested using the validation set $\mathcal{V}$ and various facets of the experiments, as presented in the following subsections. It should be pointed out that similar conclusions were reached with other scenarios (mobile devices, network conditions, and user's preferences), which can be found in Appendix D.

5.7.1 Average optimal $\mathcal{Q}_{_E}$ versus network bitrates

First, the average $\mathcal{Q}_{_E}$ is computed for various bitrate values for the proposed system and the fixed-QF dynamic system, the exhaustive static system, and the static systems with one and five versions. The results are presented in Figure 5.8 for JPEG, and in Figure 5.9 for XHTML.

As expected, $\mathcal{Q}_{_E}$ increases with the bitrate up to a point of saturation (quite visible for XHTML and occurring at higher bitrates for JPEG). The average $\mathcal{Q}_{_E}$ values obtained for the proposed dynamic solution are close to those of the exhaustive system for JPEG and very close for

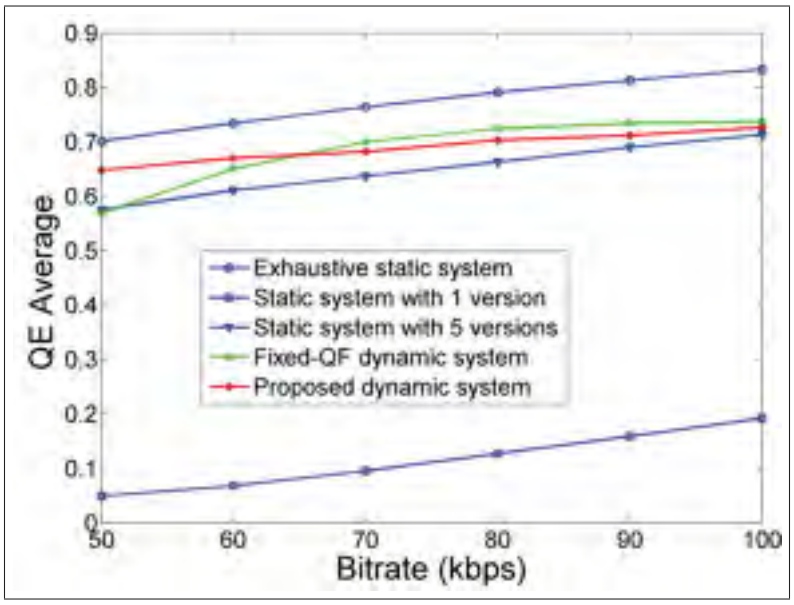

Figure 5.8 $\mathcal{Q}_E$ as a function of bitrate for $f =$ JPEG and $N_L(D) = 488$ ms.

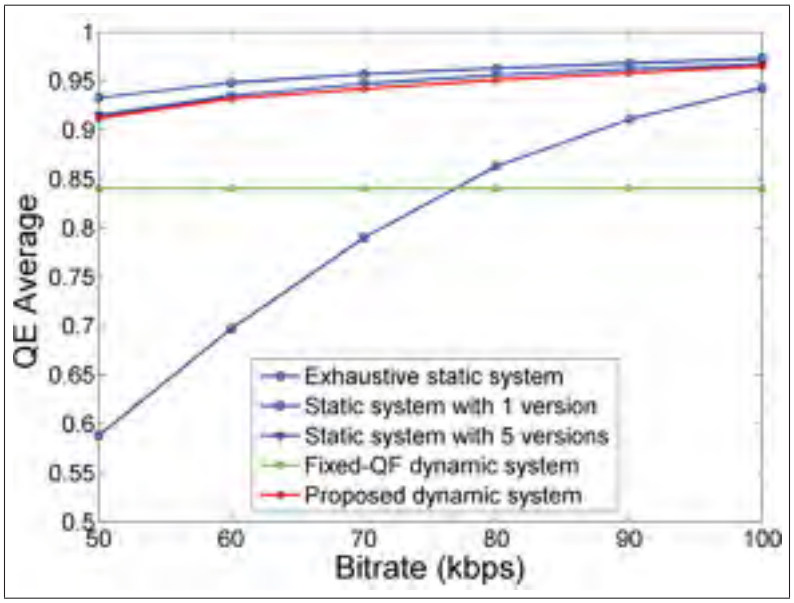

Figure 5.9 $\mathcal{Q}_E$ as a function of bitrate for $f =$ XHTML and $N_L(D) = 488$ ms.

XHTML. When $f =$ JPEG, the proposed dynamic framework performs better than the static system with up to five versions and very close to its performance when $f =$ XHTML.

5.7.2 $\mathcal{Q}_E$ average deviation from optimality

The average deviation of $\mathcal{Q}_E$ from optimality, as computed for this example, is plotted in Figure 5.10 for $N_B(D) = 50$ kbps and $N_L(D) = 488$ ms. This figure shows the difference between the proposed dynamic system, a fixed-QF dynamic system, and the static systems (those with 2 to 20 versions and the exhaustive one with 200 versions). Further, it shows the precision of the adapted content achieved for each system by computing the average deviation of $\mathcal{Q}_E$ for each system from that of the exhaustive static system.

For this mobile device, when the format used is XHTML, the proposed dynamic system provides better quality than the granularity-based static system with fewer than 3 versions and slightly lower quality, compared to the granularity-based static systems with more than 3 versions, all of them very close to the optimum when the number of versions is greater than 3. When the format used is JPEG, seven versions are needed for the granularity-based static systems to reach the quality obtained by the proposed dynamic system. Note that the number of versions required for the granularity-based static systems to achieve the performance of the proposed dynamic system depends on the bitrate (the number increases for lower bitrates).

5.7.3 Optimal $\mathcal{Q}_E$ achieved by each system

According to Figures 5.8, 5.9, and 5.10, although the $\mathcal{Q}_E$ of our dynamic solution is, on average, significantly better than that of the fixed-QF system for XHTML, the fixed-QF system can perform better under some conditions for JPEG. However, these results hide a defect of the fixed-QF system. To show this defect, we go deeper, and present the optimal $\mathcal{Q}_E$ reached by the proposed and fixed-QF dynamic systems and compare them with those reached by the exhaustive static system (optimality). As shown in Figures 5.11 and 5.12, overall, the proposed dynamic system behaves in the same way as the exhaustive static one. By contrast, the $\mathcal{Q}_E$ results provided by the fixed-QF system are highly variable. The curves are somewhat periodic, due to the nature and order of the documents submitted to the test. As mentioned earlier, the documents are created by varying the areas taken up by images and text boxes. In the first ten documents, the area taken up by images represents 10% of the slide, and the area taken up by

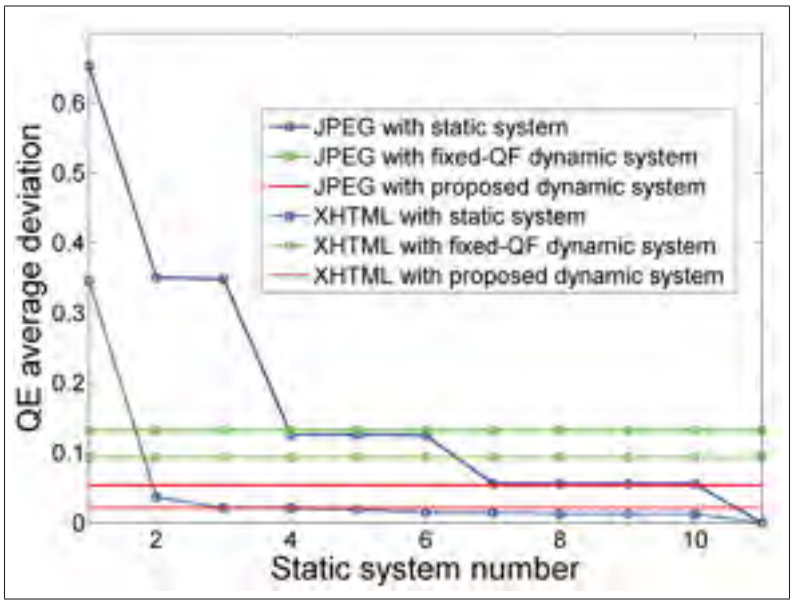

Figure 5.10 Average $\mathcal{Q}_E$ deviation from optimality for $N_B(D) = 50$ kbps, $N_L(D) = 488$ ms.

text boxes varies from 10% to 100%. In the second group of ten documents, the image area is increased to 20% and the text box area varies from 10% to 100%, and so on. We can see that the fixed-QF method performs poorly when the area taken up by the text boxes is large because of the impact it has on the transport quality, as will be shown in the next subsection.

5.7.4 Total delivery time aspect

Although the fixed-QF dynamic system usually provides good visual quality by setting $QF = 80$, it has no control over file size, which affects total delivery time, and therefore $\mathcal{Q}_E$. This aspect has been tested on the same set of slides by computing the total delivery time for each system. Figures 5.13 and 5.14 show the total time required by each slide to be delivered when the format used is JPEG and XHTML respectively. In these two figures, the proposed dynamic system provides a total delivery time very close to that of the exhaustive static system, whereas the fixed-QF system exhibits a highly variable delivery time (more than 10 s in some instances). This aspect (total delivery time) confirms the unreliable behavior of the fixed-QF dynamic system presented in the previous subsection, and supports that of the proposed dynamic system.

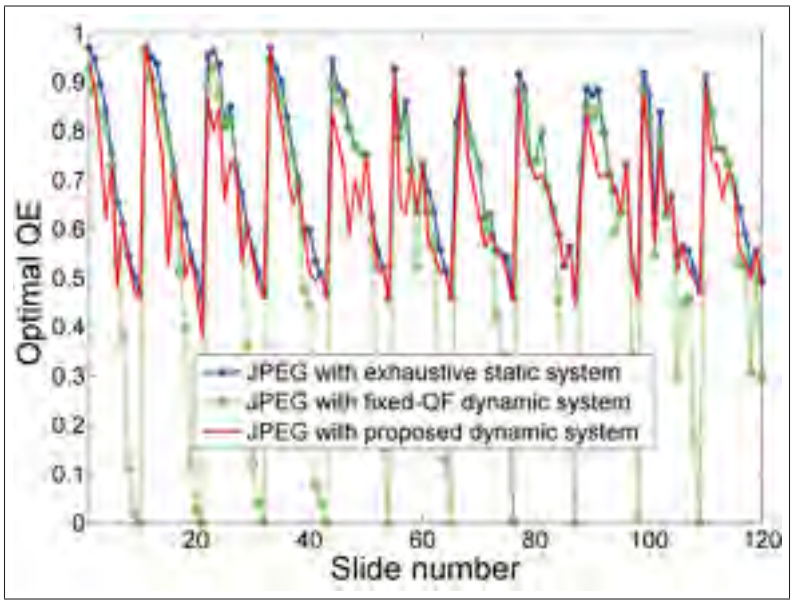

Figure 5.11 Optimal $\mathcal{Q}_E$ computed for f = JPEG, $N_B(D) = 50$ kbps and $N_L(D) = 488$ ms.

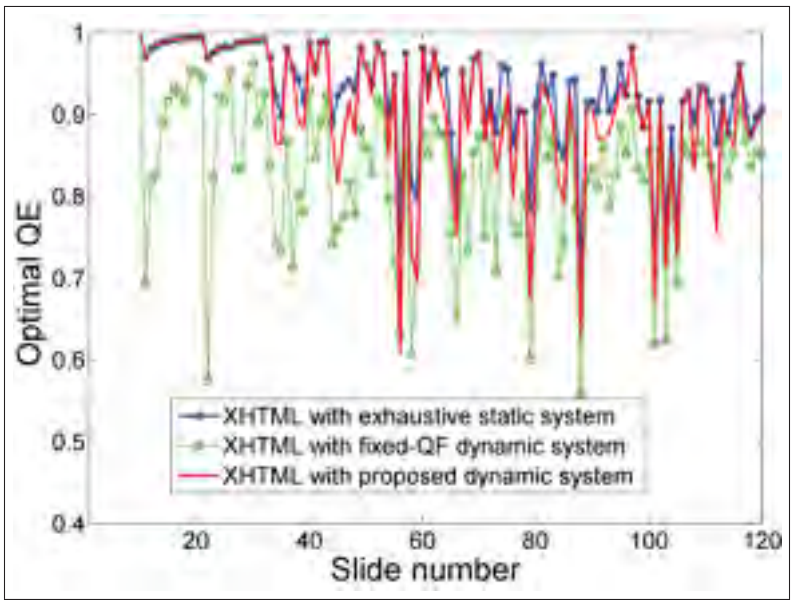

Figure 5.12 Optimal $\mathcal{Q}_E$ computed for f = XHTML, $N_B(D) = 50$ kbps and $N_L(D) = 488$ ms.

5.7.5 Storage space aspect

We now examine the behavior of the storage space needed for the versions created for each transcoding system. To do so, the average file size of the versions created by each system is

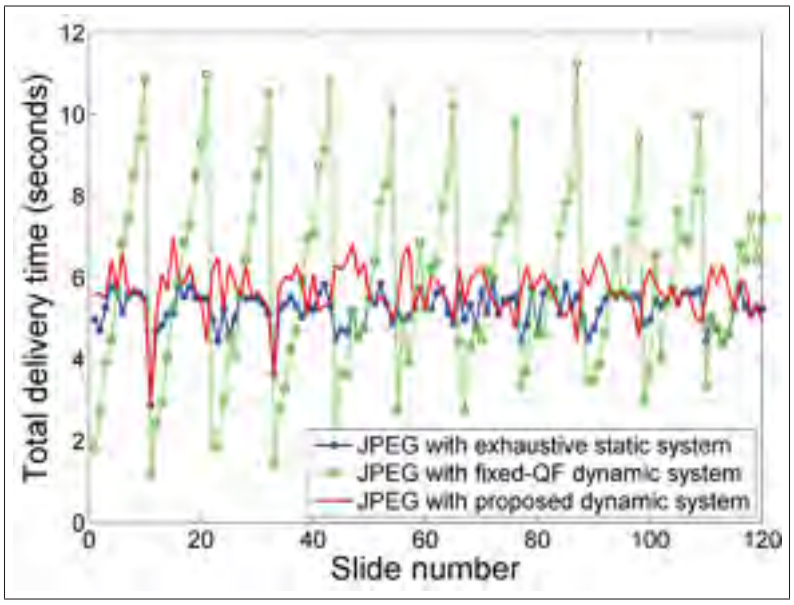

Figure 5.13 Total delivery time computed for f = JPEG, $N_B(D) = 50$ kbps, and $N_L(D) = 488$ ms.

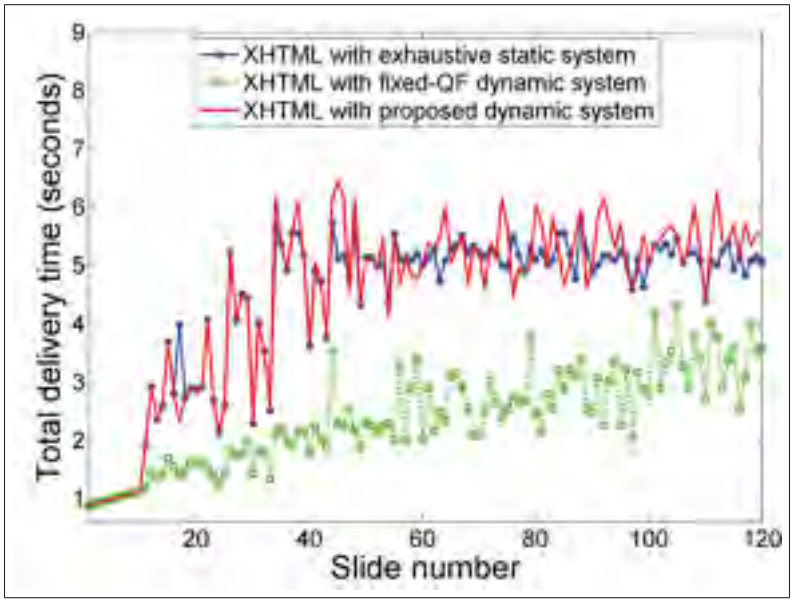

Figure 5.14 Total delivery time computed for f = XHTML, $N_B(D) = 50$ kbps, and $N_L(D) = 488$ ms.

computed and plotted in Figure 5.15. As shown by the curves, our solution becomes increasingly competitive as the granularity of the static transcoding systems increases. As previously

stated, in this example, seven versions are needed to achieve the quality of our estimated optimal adapted content when JPEG is used. If the seventh system is used, the total storage space that should be required is 487 KB on average, whereas only one version is created, and only 55.5 KB is needed on average with the proposed dynamic solution. This means that nearly 9 times more space should be available to accommodate the seventh transcoding system. In the XHTML solution, three versions are needed to achieve our estimated adapted content. In this case, the total space needed is 75.8 KB. The latter is reduced relative to that needed by the JPEG solution, but is still far more than what is needed by our solution (16.9 KB on average for only one version). Finally, for this example, ten versions (seven JPEG versions and three XHTML versions) should be created by the static transcoding systems, whereas only one version (JPEG or XHTML) is created using our solution. Again, these conclusions apply to this example and may vary depending on terminal capability and available bitrate. This is problematic for static systems, as they can't select a number of versions beforehand that will ensure a good QoE, regardless of terminal and network characteristics, without a great deal of computation. For instance, seven versions were sufficient in this example at 50 kbps to match the quality of our dynamic system, but this will not be sufficient if the bitrate is reduced (we can see how the gap becomes larger between the 5-version system and the system proposed in Figure 5.8). Therefore, the number of versions required to match the proposed dynamic system increases as the bitrate decreases. Therefore, unless numerous versions are generated, the granularity-based static systems won't be able to match the performance of the proposed dynamic method for all bitrates.

5.7.6 JPEG versus XHTML

From the $\mathcal{Q}_E$ accuracy point of view, the estimated XHTML data are more precise than the JPEG data. The average deviation from optimality is about 1% for XHTML compared to 6% for JPEG. This is explained by the fact that only the SSIM and relative file size of the embedded images are estimated in the XHTML solution (not the textual parts of the slide, for which quality and size are known rather than estimated), whereas the estimated data are computed for the whole slide in the JPEG solution.

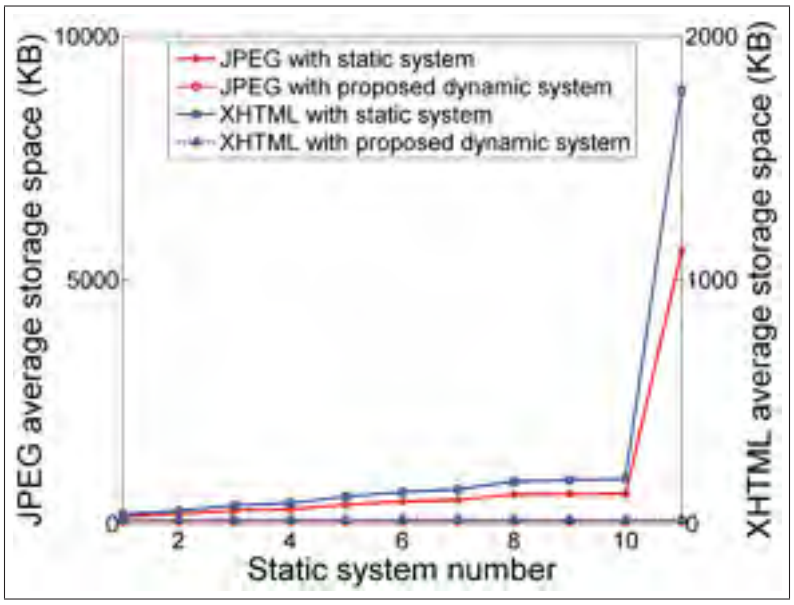

Figure 5.15 Average storage space computed for the entire slide corpus.

Regarding the storage space needed by the proposed dynamic system, we can see that the XHTML solution is very lightweight compared to the JPEG solution. In fact, in this scenario, 55.5 KB is needed, on average, if JPEG has been estimated to be the optimal format, and only 16.9 KB if XHTML was selected. This is quite reasonable, since only the embedded images are rasterized in the XHTML solution, and not the whole slide, as is the case with the JPEG solution.

Overall, when the bitrate is very high, there is no significant advantage in terms of QoE to selecting XHTML over JPEG for any of the systems presented. However, XHTML is increasingly attractive as the bitrate becomes smaller, since it always ensures crisp and readable text. For static systems, XHTML requires significantly fewer versions, and less storage space than JPEG, for a given $\mathcal{Q}_E$ average deviation. It also offers other advantages, such as text editing and keyword search. Therefore, it is clear that XHTML is, on average, a better format for sharing presentations than JPEG. Even for the proposed dynamic framework, XHTML leads to more accurate $\mathcal{Q}_E$ prediction and overall system performance.

5.7.7 Recapitulation

It is important to note that we have been very conservative in our evaluation of the performance of static systems by assuming that the terminal could render every image received. For example, in this scenario, it was assumed that the static system with one version would send a 1058×794 JPEG image which, upon reception, would be scaled by the terminal to fit the target screen resolution. However, in reality, it is possible that none of the versions generated by a static system will be supported by the terminal (especially when the number of versions is small), providing another significant advantage to the proposed dynamic framework, as it only sends content that the terminal can support.

Certainly, the performance of the proposed dynamic system is highly dependent on the accuracy of the estimated SSIM and file size of the adapted content. Therefore, using more accurate estimates with higher granularity (which are limited to 10×10 in tables in (Coulombe and Pigeon, 2009; Pigeon and Coulombe, 2008)), the proposed dynamic system could perform even better.

At the same time, transcoding many versions is CPU-intensive, and should be performed offline, which is not always possible. In its favor, the proposed dynamic solution provides near-optimal adapted content on-the-fly, while the end-user is still online. This makes our solution very attractive compared to the static transcoding systems previously presented. It reaches a good compromise between performance (little storage space and less processing time) and good quality (close to that of the exhaustive static system).

5.8 Complexity of the proposed framework

Compared to the exhaustive static system, which requires 200 transcoding operations, the proposed dynamic framework requires a single transcoding operation, which is performed following estimation of the transcoding parameters. We must add to that the computation of the $\hat{\mathcal{Q}}_V$ array (that can be computed offline), the $\hat{\mathcal{Q}}_T$ and $\hat{\mathcal{Q}}_E$ arrays (performed at runtime), and a look-up search in the $\hat{\mathcal{Q}}_E$ array. These added steps are very computationally light, compared to a single transcoding operation, and can be negligible on high-end servers.

The fixed-QF system also requires a single transcoding operation, but, as shown, it exhibits a highly variable quality and is very unreliable (it always yields good quality, but at the cost of potentially long delivery times). Granularity-based systems require more than one transcoding operation, and so are more complex. As a result, the proposed dynamic system offers exceptional quality with minimal computational complexity.

5.9 Conclusion

In this chapter, we have presented a prediction-based dynamic content adaptation framework and showed its applicability to adapting presentation slides for Web-enabled mobile devices. The framework is designed to be quite general, and future research can be conducted to validate its applicability to other enterprise documents, such as Word and Excel.

The exceptional results obtained by the proposed dynamic framework have motivated us to improve the accuracy of adapted content, in terms of $\mathcal{Q}_E$, still more, even at the cost of increased complexity. We want to explore the effect of the prediction results on quality, if a small number of transcoding operations can be tolerated. This is the subject of the next chapter.

CHAPTER 6

MODELS AND METHODS TO IMPROVE THE PROPOSED DYNAMIC FRAMEWORK

We showed in the previous chapter that the proposed prediction-based dynamic content adaptation framework can produce near-optimal adapted content dynamically in a single trancoding operation. In most cases, the results are very reliable and close to those obtained by the exhaustive static transcoding system, which is considered to be optimal. However, for certain documents, the estimated results were not, in fact, very close to optimality (especially in JPEG format), for various reasons: SSIM and relative file size prediction errors, which are amplified by SSIM linearization using a logistic function and combining the visual and transport qualities to compute the quality of experience. All these computations, which were performed on the predicted SSIM and relative file size, introduced noise, which negatively affected the accuracy of the results obtained.

In this chapter, we aim to improve the accuracy of the results obtained by the proposed dynamic framework by allowing the system to perform more than one transcoding operation. The idea is that, because the estimated adapted content is near-optimal, the optimal result should be in the neighborhood of that result. So, finding the optimal solution will involve performing a small number of transcoding operations in the neighborhood of the estimated near-optimal adapted content. The challenge is how to explore that neighborhood with lower complexity, but with the greatest impact. The results presented in this chapter have been published in (Louafi *et al.*, 2013a) and (Louafi *et al.*, 2013b).

6.1 Proposed methods and models

In the previous chapter, we presented a prediction-based dynamic content adaptation framework to solve equation 5.5, with which near-optimal transcoding parameters are computed. We propose a set of models and methods in this section designed to improve the accuracy of these parameters, while keeping the number of transcoding operations very low. Let us call the

solution presented in the previous chapter, and published in (Louafi *et al.*, 2012), *Method 1 - Estimation*. There are some inaccuracies in these estimated parameters, but they nevertheless represent a good starting point from which the other proposed methods improve. These methods are, in fact, variants of our first method. They improve accuracy, but at the expense of increased complexity (number of transcoding operations).

6.1.1 Method 1 - Estimation

In (Coulombe and Pigeon, 2009; Pigeon and Coulombe, 2008; Coulombe and Pigeon, 2010), and also in (Louafi *et al.*, 2012, 2013a), quantized values of z and QF are used instead of continuous ones, in order to limit the parameter space; that is, using a granularity of $\Delta z = 0.1$ and $\Delta QF = 10$, the quantized values of z and QF used are as follows:

$$\begin{aligned} \widetilde{z} &\in \{0.1, 0.2, 0.3, \ldots, 1\} \\ \widetilde{QF} &\in \{10, 20, 30, \ldots, 100\} \end{aligned} \tag{6.1}$$

So, the solution space consists of 200 distinct combinations of parameters (100 combinations for each format: JPEG and XHTML). With this solution space, an exhaustive method will perform 200 transcoding operations and select the best one.

With method 1, for the original content c_k and a target mobile device D, $\mathcal{Q}_{_E}(c_k^{z,QF}, D)$ can be estimated for the two quantized values z and QF. By solving (5.5) in the estimated solution space, we can identify the near-optimal solution, which consists of the transcoding parameter combinations that maximize the user's QoE. For instance, Figure 6.1 shows the estimated $\mathcal{Q}_{_E}$ values for the various combinations of $\widetilde{z}$ and $\widetilde{QF}$ computed for a document. The shaded cell represents the estimated near-optimal $\mathcal{Q}_{_E}$ obtained by method 1.

Let $\mathcal{R}_1^*(c_k, D)$ be the subset of near-optimal transcoding parameter combinations obtained by this method. It is a modified formulation of the one presented in the previous chapter (see equation 5.5). We added the index 1 to clearly indicate that the near-optimal transcoding parameters

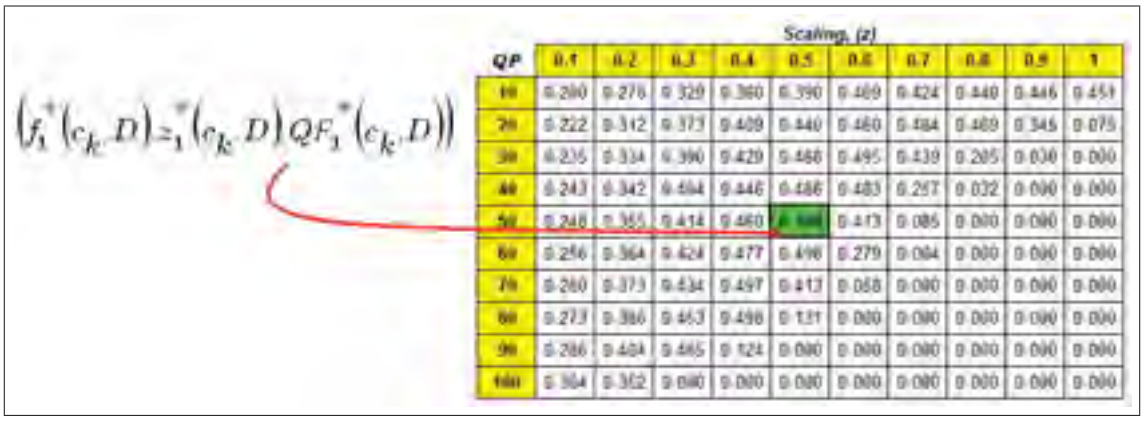

Figure 6.1 Example of estimated transcoding parameters computed for a given document using a $\mathcal{Q}_E$ table. The shaded cell contains the estimated optimal $\mathcal{Q}_E$ obtained by method 1.

were obtained using method 1. So, we have:

$$\begin{aligned}\mathcal{R}_1^*(c_k, D) &= \left\{ \left(f_1^*(c_k, D), z_1^*(c_k, D), QF_1^*(c_k, D) \right) \right\} \\ &= \underset{(f,z,QF)\in\mathcal{R}(c_k,D)}{\arg\max} \mathcal{Q}_E(c_k^{f,z,QF}, D)\end{aligned} \tag{6.2}$$

In terms of complexity, as shown in section 5.8, method 1 requires only a single transcoding operation.

6.1.2 Method 2 - Estimation and interpolation

In this method, instead of using quantized values of z and QF, we let the solution space be continuous. Since the estimated solution obtained by method 1 is near-optimal, the optimal solution should be in the same neighborhood. Consequently, using the estimated near-optimal solution and its four nearest neighbors, we suppose that the optimal solution is within the region covered by these five points. We fix the near-optimal format $f_1^*(c_k, D)$ (which is known from method 1) and model the QoE in this region using a bivariate quadratic function defined as follows:

$$f(x, y) = ax^2 + bx + cy^2 + dy + e \tag{6.3}$$

where x and y represent z and QF in a continuous space respectively.

The optimal point in this region is where the gradient is null:

$$\begin{cases} \dfrac{\partial f}{\partial x} & = 2ax + b = 0 \\ \dfrac{\partial f}{\partial y} & = 2cy + d = 0 \end{cases} \tag{6.4}$$

Using the estimated near-optimal point and its four estimated nearest neighbors, we compute the coefficients a, b, c, d, and e. Then, using (6.4), we compute the estimated interpolated near-optimal transcoding parameters $z_I^*(c_k, D)$ and $QF_I^*(c_k, D)$. We expect the interpolated near-optimal point to be close to the actual optimal. Two adapted contents are created using the estimated and interpolated transcoding parameters, and the best of the two is selected as the near-optimal adapted content obtained by method 2. Figure 6.2 shows, for a given document, the estimated near-optimal $\mathcal{Q}_E$ obtained by method 1 and its four nearest neighbors, and the bivariate quadratic function obtained by modeling this region. On the surface of this function, we have two points: the one obtained by method 1, and the one obtained by interpolation (where the gradient is null).

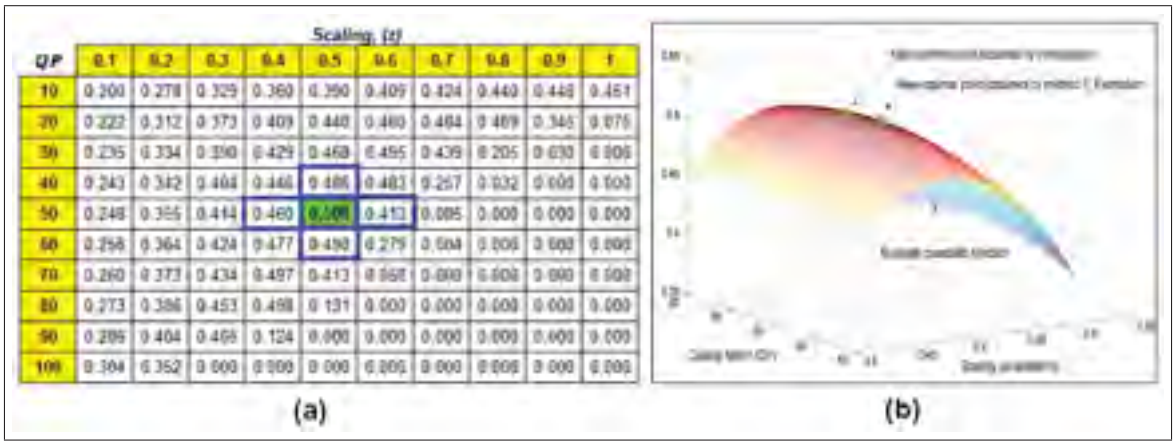

Figure 6.2 (a) Example of the estimated transcoding parameters computed for a given document. The shaded cell contains the estimated optimal $\mathcal{Q}_E$ obtained by method 1. (b) The bivariate quadratic function used to model the region covered by this optimal point and its four nearest neighbors.

Formally, the near-optimal transcoding parameters obtained by method 2 are given by:

$$\begin{aligned}\mathcal{R}_2^*(c_k, D) &= \Big\{\big(f_2^*(c_k, D), z_2^*(c_k, D), QF_2^*(c_k, D)\big)\Big\} \\ &= \underset{(f,z,QF)\in\mathcal{N}_e^2}{\arg\max}\ \mathcal{Q}_{_E}(c_k^{f,z,QF}, D)\end{aligned} \tag{6.5}$$

where $\mathcal{N}_e^2$ is a set comprising two elements: the estimated and the interpolated (denoted I below) near-optimal parameter combinations. Formally, we have:

$$f_2^*(c_k, D) = f_1^*(c_k, D) \tag{6.6}$$

$$\begin{aligned}\mathcal{N}_e^2 = \Big\{&\big(f_1^*(c_k, D), z_1^*(c_k, D), QF_1^*(c_k, D)\big), \\ &\big(f_1^*(c_k, D), z_I^*(c_k, D), QF_I^*(c_k, D)\big)\Big\}\end{aligned} \tag{6.7}$$

In terms of complexity, this method requires two transcoding operations.

6.1.3 Method 3 - Estimation and one-step Diamond search

In this method, we use the estimated near-optimal adapted content (computed from method 1) and its four nearest neighbors. Unlike method 2, here, these five points are transcoded versions, rather than merely estimated content, and the best of them is selected. For instance, Figure 6.3 shows the computed $\mathcal{Q}_{_E}$ array for the same document presented in the previous methods. The shaded cell contains the computed $\mathcal{Q}_{_E}$ obtained using the estimated near-optimal transcoding parameters of method 1, and diamond search points formed by its four nearest neighbors. In this example, the left neighbor represents the near-optimal point obtained by method 3.

Formally, the optimal transcoding parameters obtained by this third method are as follows:

$$\begin{aligned}\mathcal{R}_3^*(c_k, D) &= \Big\{\big(f_3^*(c_k, D), z_3^*(c_k, D), QF_3^*(c_k, D)\big)\Big\} \\ &= \underset{(f,z,QF)\in\mathcal{N}_e^5}{\arg\max}\ \mathcal{Q}_{_E}(c_k^{f,z,QF}, D)\end{aligned} \tag{6.8}$$

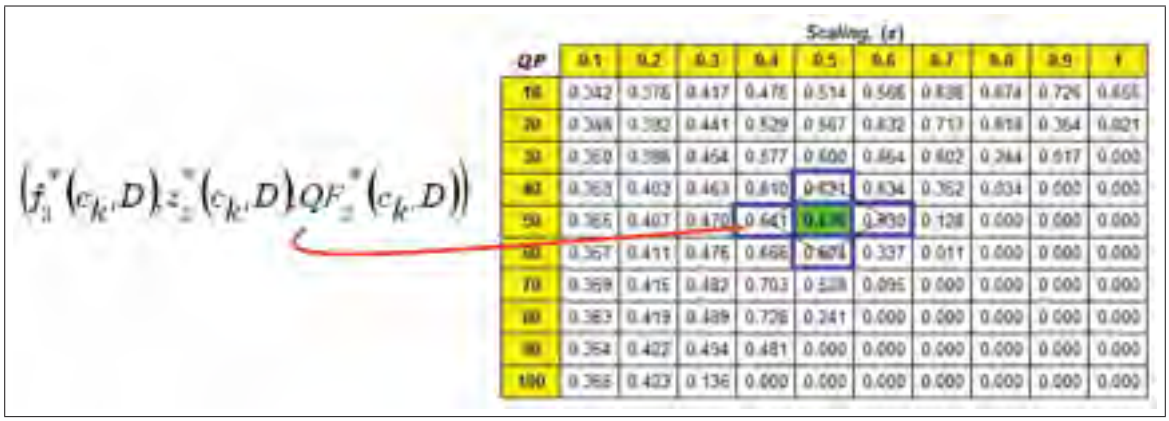

Figure 6.3 Example of estimated transcoding parameters and their computed $\mathcal{Q}_E$ for a given document. The shaded cell contains the near-optimal computed $\mathcal{Q}_E$ obtained by method 1. The diamond shows its four computed nearest neighbors.

where $\mathcal{N}_e^5$ is a set containing five elements: the estimated near-optimal parameters and their four nearest neighbors. So, we have:

$$f_3^*(c_k, D) = f_1^*(c_k, D) \tag{6.9}$$

$$\begin{aligned} \mathcal{N}_e^5 = \Big\{ & \big(f_1^*(c_k, D), z_1^*(c_k, D), QF_1^*(c_k, D)\big), \\ & \big(f_1^*(c_k, D), z_1^*(c_k, D) \pm \Delta z, QF_1^*(c_k, D)\big), \\ & \big(f_1^*(c_k, D), z_1^*(c_k, D), QF_1^*(c_k, D) \pm \Delta QF\big) \Big\} \end{aligned} \tag{6.10}$$

Regarding complexity, this method requires five transcoding operations.

6.1.4 Method 4 - Estimation and two-steps diamond search

Like method 3, in this method, we identify and create the estimated optimal adapted content and its four nearest neighbors. From these five points, we identify the best one (equal to that obtained by method 3) and use it as a starting point to explore its four nearest neighbors. This is the origin of the term *two-steps diamond search*. If the best point is equal to that obtained by method 1, there is no need to perform the second step in the search, and so the near-optimal point returned by this method is the one obtained by method 3 $\big(f_3^*(c_k, D), z_3^*(c_k, D), QF_3^*(c_k, D)\big)$. Otherwise, we identify and create its four nearest neighbors, one of which had already been

created by method 3. The best of these points becomes the near-optimal point computed by method 4. Figure 6.4 shows the same example presented in method 3, in which the two diamond search points are outlined. In this example, the neighbor below the optimal solution obtained by method 3 represents the optimal point of this method ($z = 0.4$ and $QF = 60$).

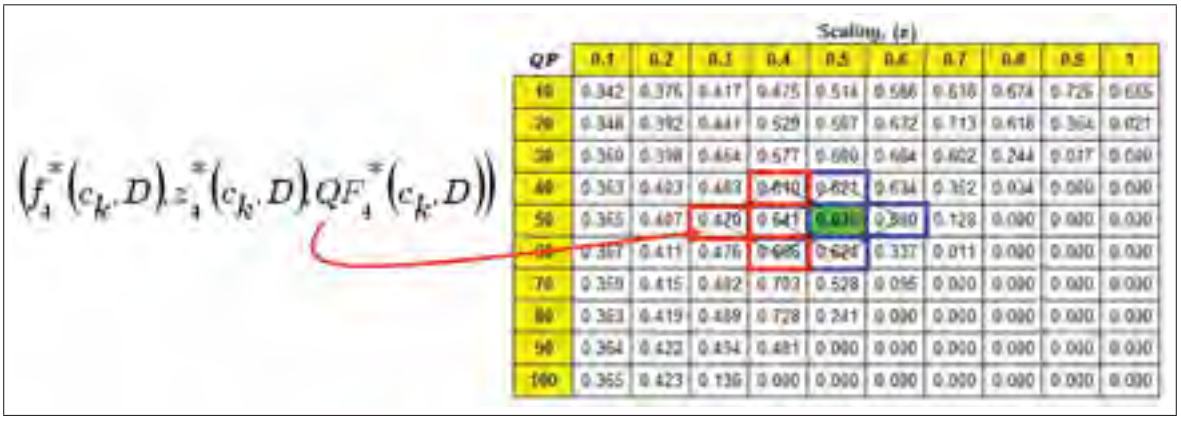

Figure 6.4 Example of estimated transcoding parameters and their computed $\mathcal{Q}_E$ for a given document. The shaded cell contains the near-optimal computed $\mathcal{Q}_E$ obtained by method 1. The two diamonds show its evaluated neighbors.

This can be formulated as follows:

$$\begin{aligned}\mathcal{R}_4^*(c_k, D) &= \left\{\left(f_4^*(c_k, D), z_4^*(c_k, D), QF_4^*(c_k, D)\right)\right\} \\ &= \underset{(f,z,QF)\in\mathcal{N}_e^{5,8}}{\arg\max}\ \mathcal{Q}_E(c_k^{f,z,QF}, D)\end{aligned} \tag{6.11}$$

where $\mathcal{N}_e^{5,8}$ is the set of transcoding parameter combinations used in this method. Thus, we have:

$$f_4^*(c_k, D) = f_1^*(c_k, D) \tag{6.12}$$

$$\begin{aligned}\mathcal{N}_e^{5,8} = \mathcal{N}_e^5 \cup \Big\{&\left(f_3^*(c_k, D), z_3^*(c_k, D) \pm \Delta z, QF_3^*(c_k, D)\right), \\ &\left(f_3^*(c_k, D), z_3^*(c_k, D), QF_3^*(c_k, D) \pm \Delta QF\right)\Big\}\end{aligned} \tag{6.13}$$

The complexity of this method is either five or eight transcoding operations:

- If $\big(f_3^*(c_k, D), z_3^*(c_k, D), QF_3^*(c_k, D)\big)$=$\big(f_1^*(c_k, D), z_1^*(c_k, D), QF_1^*(c_k, D)\big)$, then we have $\mathcal{N}_e^{5,8} = \mathcal{N}_e^5$, and so only five transcoding operations are required.
- Otherwise, we have $\mathcal{N}_e^{5,8} \supset \mathcal{N}_e^5$, and so eight transcoding operations are performed.

6.1.5 Method 5 - Estimation and greedy search

In this method, we use the estimated near-optimal point (from method 1) as a starting point and explore its neighborhood, seeking to improve the $\mathcal{Q}_E$ obtained until convergence is reached; that is, until this $\mathcal{Q}_E$ cannot be improved any further. We tested various patterns and found that, for the problem at hand, following one of these patterns: LRUD, LRDU, RLUD, RLDU, UDLR, UDRL, DULR, or DURL, improved this $\mathcal{Q}_E$ significantly with the fewest transcodings, compared to other patterns. Note that, using these selected patterns, the performance of this method (optimal $\mathcal{Q}_E$ versus complexity) varies slightly from one pattern to another. But the difference is so small that it can be neglected, and so these patterns can be used interchangeably. Before detailing this method, we explain what the letters L, R, U, and D stand for. For a given point, they constitute its nearest left-hand, right-hand, upward, and downward neighbors respectively. For instance, the four nearest neighbors of the estimated point of method 1 are given by:

$$\begin{aligned}
\text{Left neighbor}: &\ \big(f_1^*(c_k, D), z_1^*(c_k, D) - \Delta z, QF_1^*(c_k, D)\big) \\
\text{Right neighbor}: &\ \big(f_1^*(c_k, D), z_1^*(c_k, D) + \Delta z, QF_1^*(c_k, D)\big) \\
\text{Up neighbor}: &\ \big(f_1^*(c_k, D), z_1^*(c_k, D), QF_1^*(c_k, D) - \Delta QF\big) \\
\text{Down neighbor}: &\ \big(f_1^*(c_k, D), z_1^*(c_k, D), QF_1^*(c_k, D) + \Delta QF\big)
\end{aligned} \tag{6.14}$$

Using the LRDU pattern, for example, this method proceeds as follows: We start from the point $\big(f_1^*(c_k, D), z_1^*(c_k, D), QF_1^*(c_k, D)\big)$, then, we verify whether or not the neighbor to the left provides a better solution. If it does, we move towards the left until there is no further improvement. Otherwise, we verify whether or not the neighbor to the right provides a better solution, and, if so, we move towards the right until there is no further improvement. The same process is then performed in the downward and upward directions. Each time a new point

is evaluated, a new transcoding is performed. For instance, Figure 6.5 shows, for the same example, the set of points visited when an LRDU greedy search is performed on the computed $\mathcal{Q}_E$ array. The optimal point in this example corresponds to $z = 0.4$ and $QF = 80$. The pseudo-code of the proposed greedy search algorithm (LRDU pattern) is presented in Algorithm 1.

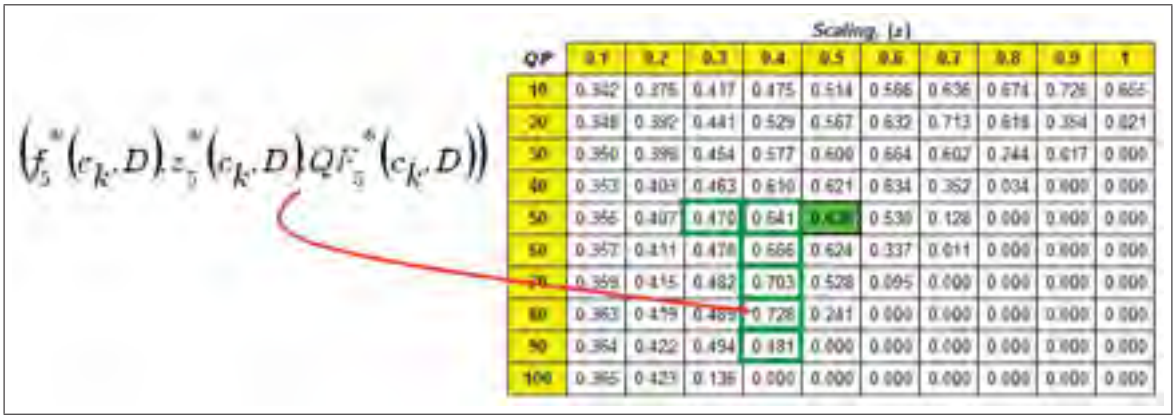

Figure 6.5 Example of estimated transcoding parameters and their computed $\mathcal{Q}_E$ for a given document. The shaded cell contains the near-optimal computed $\mathcal{Q}_E$ obtained by method 1. The points evaluated by this method are outlined.

Formally, the near-optimal transcoding parameters combinations obtained by this fifth method are given by:

$$\begin{aligned}\mathcal{R}_5^*(c_k, D) &= \left\{\left(f_5^*(c_k, D), z_5^*(c_k, D), QF_5^*(c_k, D)\right)\right\} \\ &= \underset{(f,z,QF)\in\mathcal{N}_e^{LRDU}}{\arg\max}\ \mathcal{Q}_E(c_k^{f,z,QF}, D)\end{aligned} \tag{6.15}$$

where $\mathcal{N}_e^{LRDU}$ is the set of points evaluated in this method, which can vary greatly, depending on c_k and D. Similarly, $f_5^*(c_k, D) = f_1^*(c_k, D)$.

The number of transcoding operations can be very high if either the starting point is chosen randomly or an exhaustive search is performed. However, in this method, we take advantage of the prediction of the estimated transcoding parameters, which are very reliable. Unlike an exhaustive search, we start here from an estimated point that is relatively close to the optimal. Indeed, experimental results (see section 6.3) show that the number of transcoding operations is

```
1  function LRDU_Search(c_k, D)
2  begin
3      z ← z_1^*(c_k, D), QF ← QF_1^*(c_k, D)
4      Q_E ← Q_E(c_k^{z,QF}, D))
5      (z, QF, Q_E) ← LR_Search(c_k, D, z, QF, Q_E)
6      (z, QF, Q_E) ← DU_Search(c_k, D, z, QF, Q_E)
7      return (z, QF, Q_E)
   end

8  function LR_Search(c_k, D, z, QF, Q_E)
9  begin
10     if Q_E(c_k^{z−Δz,QF}, D)) > Q_E then
11         (z, QF, Q_E) ← search(c_k, D, z, QF, Q_E, −1, 0)
       else
12         if Q_E(c_k^{z+Δz,QF}, D)) > Q_E then
13             (z, QF, Q_E) ← search(c_k, D, z, QF, Q_E, +1, 0)
           end
       end
14     return (z, QF, Q_E)
   end

15 function DU_Search(c_k, D, z, QF, Q_E)
16 begin
17     if Q_E(c_k^{z,QF+ΔQF}, D)) > Q_E then
18         (z, QF, Q_E) ← search(c_k, D, z, QF, Q_E, 0, +1)
       else
19         if Q_E(c_k^{z,QF−ΔQF}, D)) > Q_E then
20             (z, QF, Q_E) ← search(c_k, D, z, QF, Q_E, 0, −1)
           end
       end
21     return (z, QF, Q_E)
   end

22 function search(c_k, D, z_o, QF_o, Q_E, λ_z, λ_QF)
23 begin
24     z ← z_o + λ_z Δz, QF ← QF_o + λ_QF ΔQF
25     while Q_E(c_k^{z,QF}, D)) > Q_E do
26         Q_E ← Q_E(c_k^{z,QF}, D)
27         z ← z + λ_z Δz, QF ← QF + λ_QF ΔQF
       end
28     return (z, QF, Q_E)
   end
```

Algorithm 1: LRDU greedy search pseudo-code.

between 4 and 7 - 5.2 on average, which means that we are still in the same range of transcoding operations as with previous methods.

6.2 Experimental setup

To demonstrate the effectiveness of the proposed set of methods, and particularly the $\mathcal{Q}_{_E}$ improvements made by methods 2 to 5 over method 1, we used the same experimental setup as presented in the previous chapter. That is, for the same corpus of slides, the same target mobile device D ($z_v = 0.4$), and the same communication network (GPRS: $N_{_B}(D) = 50$ kbps and $N_{_L}(D) = 488$ ms), we used the optimal adapted content obtained by the exhaustive static system (which plays the role of optimality) and the best content obtained by the fixed-QF dynamic system (a typical dynamic system with $QF = 80$ and $z = 0.4$).

Regarding method 1, as shown in the previous chapter, the estimated transcoding parameters were very reliable overall. However, in the case of JPEG, there were some outlier combinations. That is, for certain documents, the $\mathcal{Q}_{_E}$ of the estimated near-optimal transcoding parameters was not as close to optimality as the rest of the documents. To visualize this, the $\mathcal{Q}_{_E}$ obtained by the proposed dynamic framework and the exhaustive static system were sorted according to the $\mathcal{Q}_{_E}$ obtained by the latter and plotted (see Figure 6.6). We tested the framework with various bitrate values, and reached the conclusion that these outlier points vary with the communication network conditions (e.g. bitrate). After analyzing the curve behavior of $\mathcal{Q}_{_E}$, we found that these outliers were caused by the nature of the Zmf curve (see Figure 5.1) used to model the $\mathcal{Q}_{_T}$, which was set using $\alpha = 5$ and $\beta = 10$. Indeed, the Zmf curve for this scenario decreases aggressively between the two values α and β, which makes the $\mathcal{Q}_{_E}$ curve highly sensitive to delivery time (file size and network conditions). Therefore, in method 1, since the file size prediction error in (Pigeon and Coulombe, 2008) can reach 15%, we have increased the predicted file size by this amount to ensure that the transcoded file size will not lead to a drastically lower $\mathcal{Q}_{_T}$ than predicted. We sacrifice the quality slightly to ensure a good $\mathcal{Q}_{_T}$, as it is much more sensitive to file size. In this scenario, it was not necessary to increase the estimated file size in the case of XHTML, as the estimated XHTML parameters were very precise (see chapter 5). However, for lower bitrate values, we observed the same behavior as that observed

in the JPEG case, which can be corrected by adjusting the predicted file size (i.e. adding a safety factor). Some experiments with lower bitrate values exhibiting this phenomenon can be found in Appendix D.

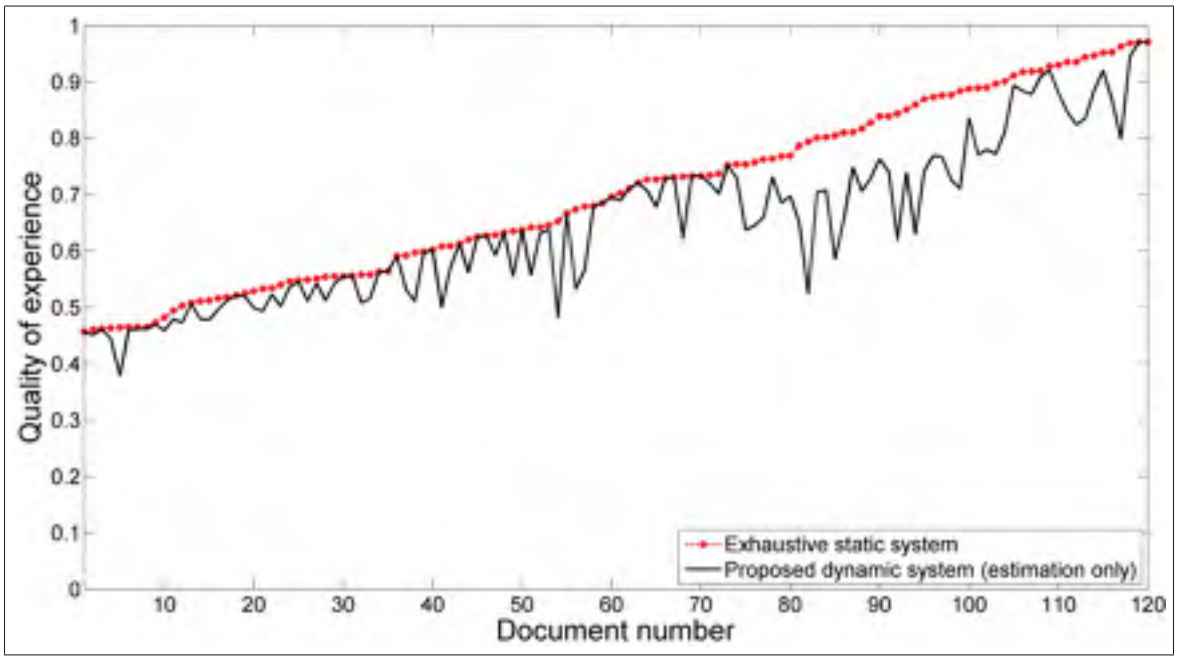

Figure 6.6 Optimal $\mathcal{Q}_E$ obtained by the proposed dynamic framework vs. that of the exhaustive static system when f = JPEG.
The slides are sorted according to the $\mathcal{Q}_E$ of the exhaustive static system.

Based on the estimated near-optimal transcoding parameters obtained by method 1, the near-optimal $\mathcal{Q}_E$ obtained by method 2 to method 5 were computed for each slide by solving equations 6.5, 6.8, 6.11, and 6.15 respectively. As a result, for each slide c_k, its computed and estimated near-optimal transcoding parameters and their $\mathcal{Q}_E$ were stored in arrays, as follows:

$$\begin{aligned} W^*_{E,k} &= \left[c_k, f^*_E(c_k, D), z^*_E(c_k, D), QF^*_E(c_k, D), \mathcal{Q}_E(c_k^{f^*_E(c_k,D), z^*_E(c_k,D), QF^*_E(c_k,D)}, D)\right] \\ W^*_{FQF,k} &= \left[c_k, f^*_{FQF}(c_k, D), 0.4, 80, \mathcal{Q}_E(c_k^{f^*_{FQF}, 0.4, 80}, D)\right] \\ W^*_{i,k} &= \left[c_k, f^*_i(c_k, D), z^*_i(c_k, D), QF^*_i(c_k, D), \mathcal{Q}_E(c_k^{f^*_i(c_k,D), z^*_i(c_k,D), QF^*_i(c_k,D)}, D)\right] \end{aligned} \tag{6.16}$$

where:

- $W^*_{E,k}$ is an array that contains the optimal transcoding parameters $(f^*_{E}(c_k, D), z^*_{E}(c_k, D), QF^*_{E}(c_k, D))$ that were computed by the exhaustive static system, and their corresponding $\mathcal{Q}_{E}$.
- $W^*_{FQF,k}$ is an array containing the best transcoding parameters ($f^*_{FQF}(c_k, D), 0.4, 80$), as computed by the fixed-QF dynamic system, and their $\mathcal{Q}_{E}$.
- $W^*_{i,k}$ is an array that contains the near-optimal transcoding parameters $(f^*_{i}(c_k, D), z^*_{i}(c_k, D), QF^*_{i}(c_k, D))$ attained by method $i \in \{1, 2, 3, 4, 5\}$, and its $\mathcal{Q}_{E}$.

Lastly, for each of the five methods, the near-optimal transcoding parameters obtained and their corresponding $\mathcal{Q}_{E}$ are compared to that of the exhaustive static and fixed-QF dynamic systems, the results of which are presented in the next section.

6.3 Experimental results and discussion

6.3.1 Optimal $\mathcal{Q}_{E}$ attained by each method

For each slide c_k, the near-optimal $\mathcal{Q}_{E}$ obtained by methods 1 to 5, as well as those computed by the exhaustive static method (from $W^*_{E,k}$) and the fixed-QF dynamic method (from $W^*_{FQF,k}$), were plotted. To visualize this, all the $\mathcal{Q}_{E}$ obtained were sorted according to those of the exhaustive static system, and presented in Figures 6.7 and 6.8 for JPEG and XHTML respectively. Overall, the proposed methods have a $\mathcal{Q}_{E}$ close to that of the exhaustive static system. However, the $\mathcal{Q}_{E}$ obtained by the fixed-QF dynamic system is highly variable, and this is very obvious for lower $\mathcal{Q}_{E}$ values. The fixed-QF dynamic system is especially problematic for large documents and low network bitrate values. Note that the outlier points shown in the previous section (see Figure 6.6) were corrected in method 1, and, of course, in methods 2 to 5.

6.3.2 $\mathcal{Q}_{E}$ improvement made by method 2 to method 5 over method 1

To show the improvements obtained by methods 2 to 5 over method 1, their relative gains in $\mathcal{Q}_{E}$ were computed. For instance, given a slide c_k, the relative gain obtained using method i is

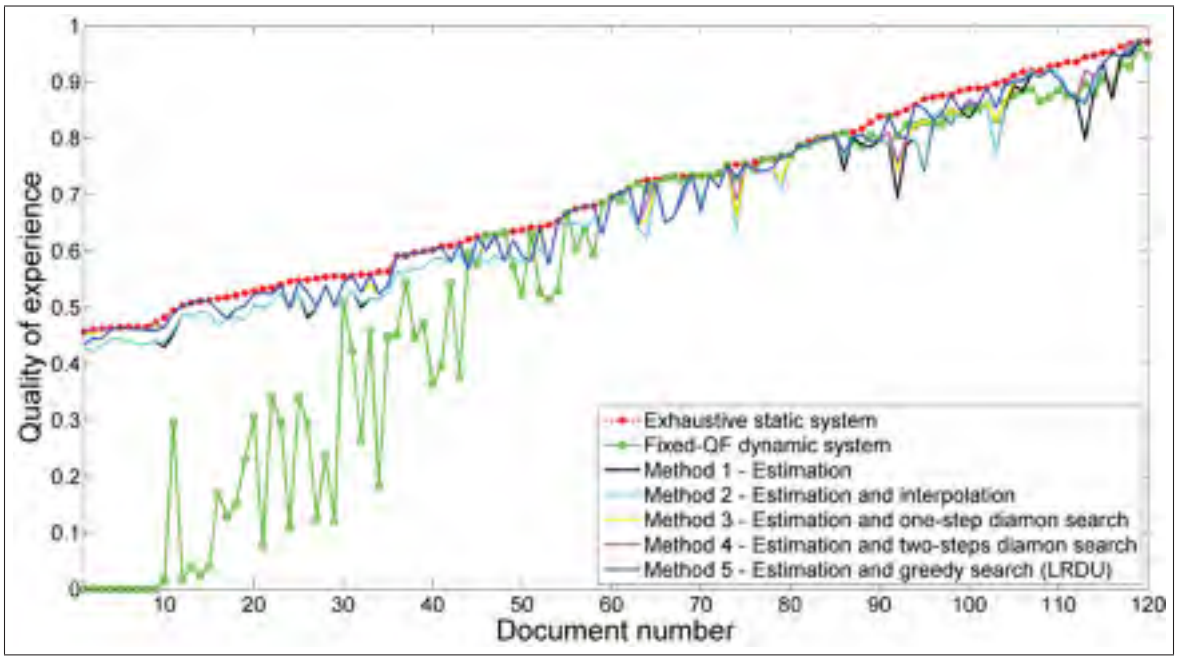

Figure 6.7 Optimal $\mathcal{Q}_E$ obtained by methods 1 to 5 vs. that of the exhaustive static and fixed-QF dynamic systems when $f_1^*(c_k, D) =$ JPEG.
The slides are sorted according to the $\mathcal{Q}_E$ of the exhaustive static system.

computed as follows:

$$\frac{\mathcal{Q}_E(c_k^{f_i^*(c_k,D),z_i^*(c_k,D),QF_i^*(c_k,D)}, D) - \mathcal{Q}_E(c_k^{f_1^*(c_k,D),z_1^*(c_k,D),QF_1^*(c_k,D)}, D)}{\mathcal{Q}_E(c_k^{f_1^*(c_k,D),z_1^*(c_k,D),QF_1^*(c_k,D)}, D)} \times 100\% \qquad (6.17)$$

These computed $\mathcal{Q}_E$ relative gains were plotted as scattered points, as depicted in Figures 6.9 and 6.10 for JPEG and XHTML respectively. For instance, for JPEG, sub-figures 6.9(a), 6.9(b), 6.9(c), and 6.9(d) show the $\mathcal{Q}_E$ relative gain obtained by methods 2, 3, 4, and 5 respectively. In the case of XHTML, the relative gain obtained by methods 2 to 5 are presented in sub-figures 6.10(a), 6.10(b), 6.10(c), and 6.10(d) respectively. The diagonal line represents the target relative gains, which were computed from $W_{E,k}^*$. The scattered points represent the different slides, and their positions indicate the relative gain obtained versus the target relative gain.

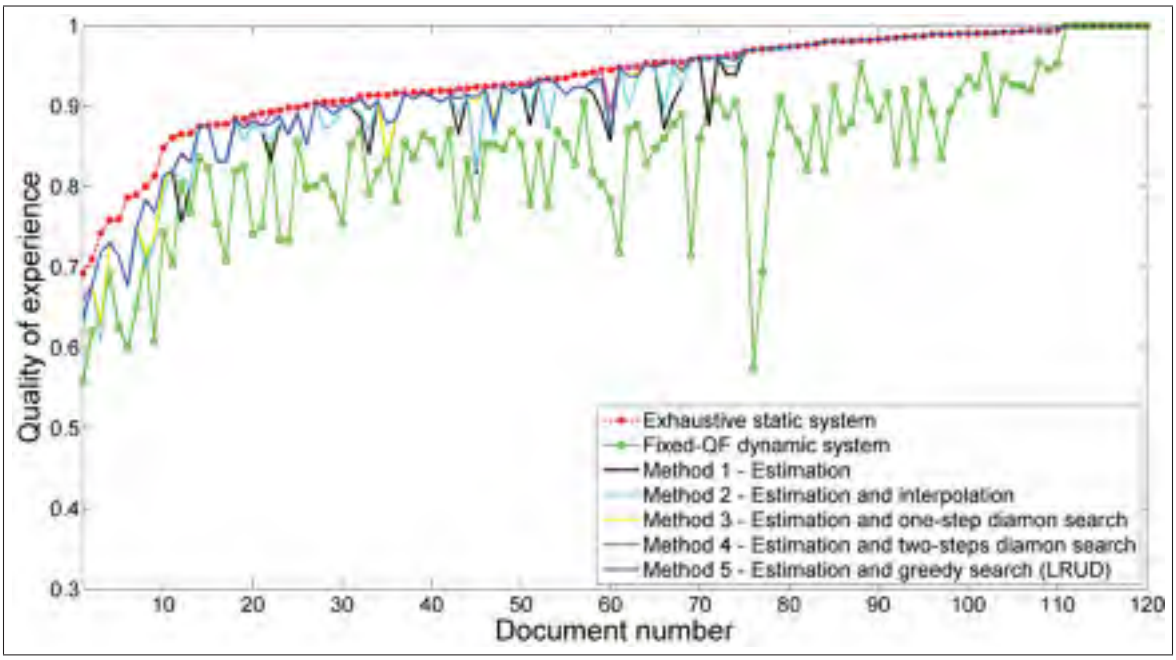

Figure 6.8 $\mathcal{Q}_E$ obtained by methods 1 to 5 vs. that of the exhaustive static and fixed-QF dynamic systems when $f_1^*(c_k, D) =$ XHTML.
The slides are sorted according to the $\mathcal{Q}_E$ of the exhaustive static system.

6.3.3 Number of documents with an improved $\mathcal{Q}_E$

Another view, showing the number of documents with an improved computed $\mathcal{Q}_E$ as a result of applying the proposed methods is depicted in Figures 6.11 and 6.12 for JPEG and XHTML respectively. To show this aspect graphically, the $\mathcal{Q}_E$ range has been split into 10 bins ([0,0.1],]0.1,02],...,]0.9,1]), and the documents that are in the same $\mathcal{Q}_E$ bin were counted and their numbers plotted as a histogram.

In the case of JPEG, using the fixed-QF dynamic system, the first four bins (for poor quality documents) contain almost 30% of the documents, while these bins are empty for the other methods (methods 1 to 5, in addition to the exhaustive static one). Also, unlike the other methods, for the fixed-QF dynamic system, the last bin (for the best quality documents) contains very few documents. This can also be seen in Figure 6.7 for very low or very high $\mathcal{Q}_E$ values.

In the case of XHTML, the first 6 bins are empty for all the methods, except for the fixed-QF dynamic system, for which there are some documents in bin]0.5-06]. By contrast, the number

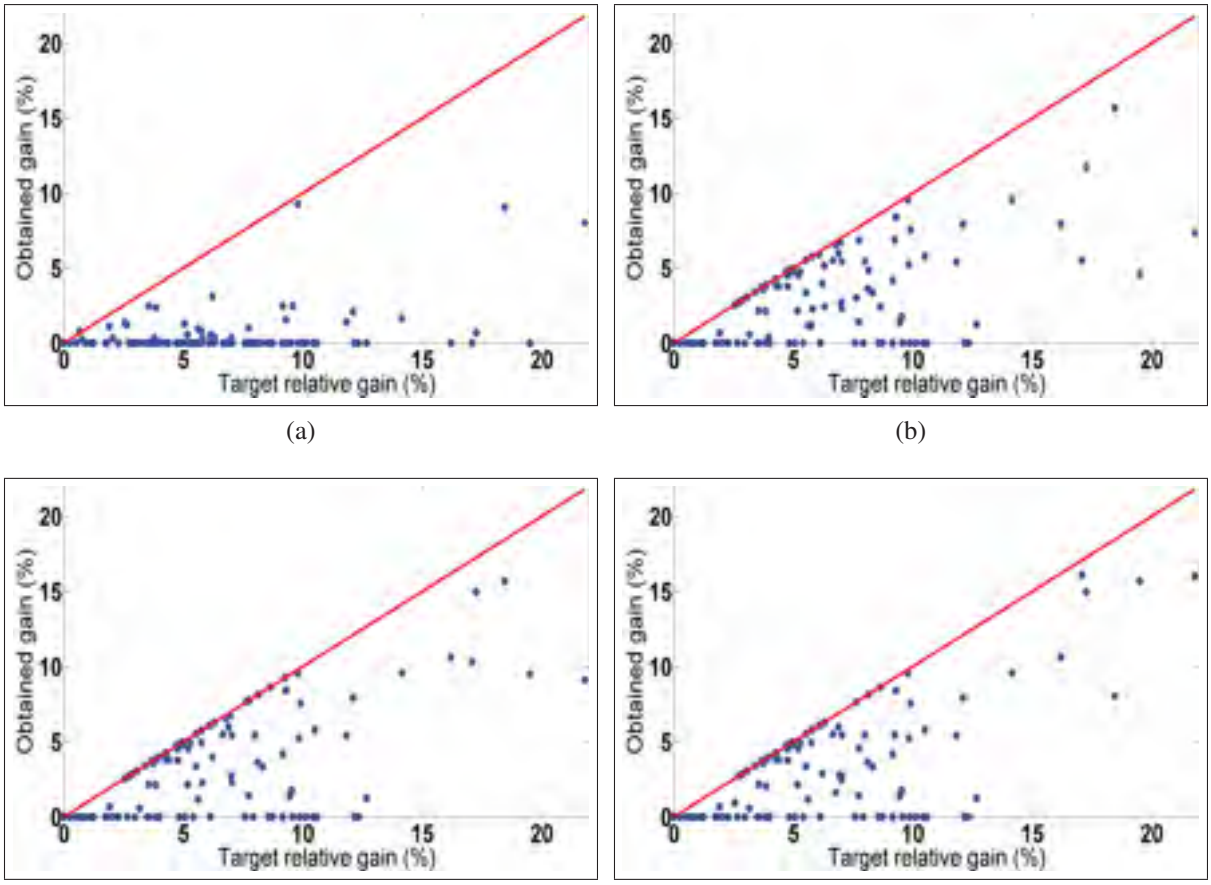

Figure 6.9 $\mathcal{Q}_E$ relative gains for methods 2 to 5 with respect to method 1, when $f_1^*(c_k, D) =$ JPEG.
(a) Method 2 - Estimation and interpolation, (b) Method 3 - Estimation and one-step diamond search, (c) Method 4 - Estimation and two-steps diamond search, (d) Method 5 - Estimation and greedy search.

of documents corresponding to the fixed-QF dynamic system in bins]0.6-07] to]0.8-09] is far from that of the exhaustive static system, although this number is closer to that of the exhaustive static system when methods 1 to 5 are used.

These two figures show the improvements achieved by the proposed methods in terms of the number of documents with an increased $\mathcal{Q}_E$. They also serve as a comparison, in terms of accuracy, between the proposed methods. Except for method 4, the greater the complexity of

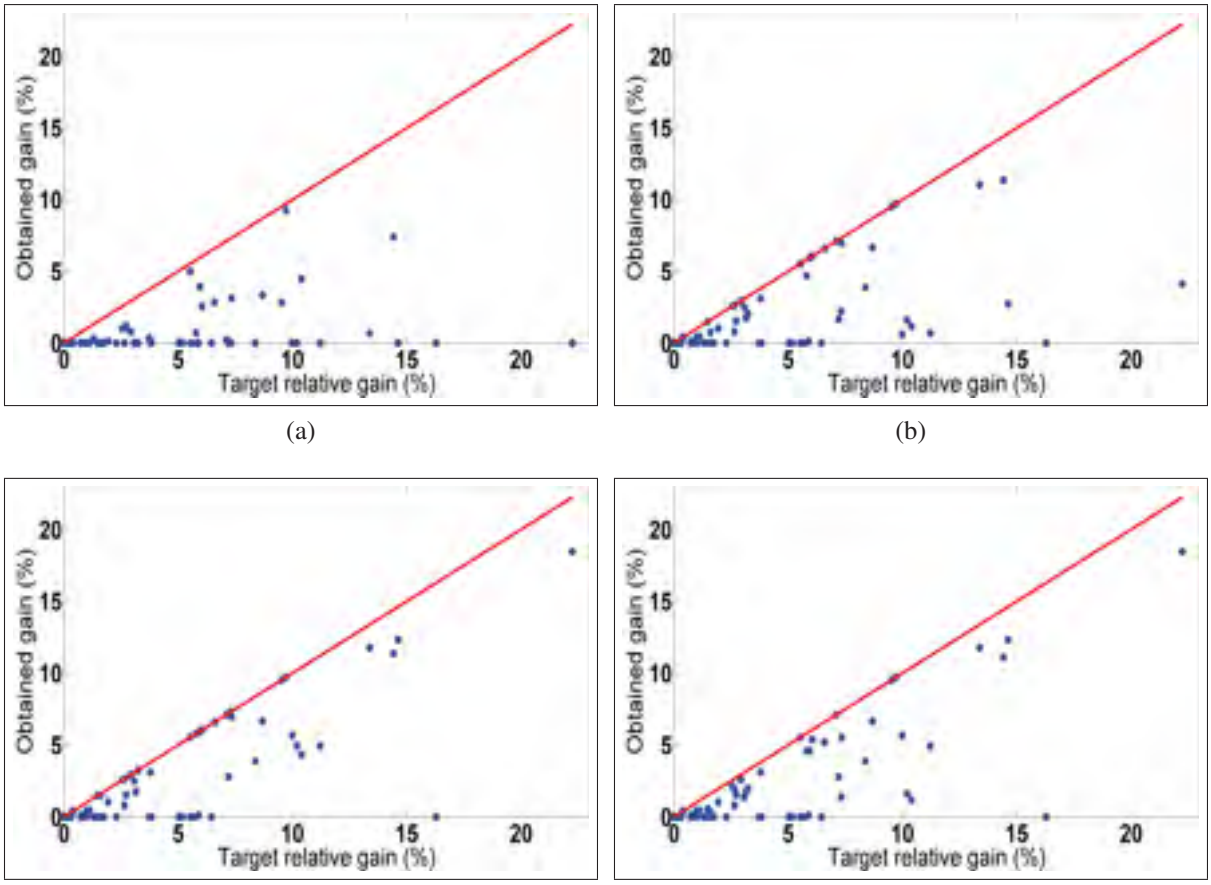

Figure 6.10 $\mathcal{Q}_E$ relative gains for methods 2 to 5 with respect to method 1, when $f_1^*(c_k, D) =$ XHTML.
(a) Method 2 - Estimation and interpolation, (b) Method 3 - Estimation and one-step diamond search, (c) Method 4 - Estimation and two-steps diamond search, (d) Method 5 - Estimation and greedy search.

the method used, the greater the accuracy (the number of documents in each bin is closer to that of the exhaustive static system).

6.3.4 Complexity of the proposed methods

The percentage of average $\mathcal{Q}_E$ obtained by methods 1 to 5 compared to that obtained by the exhaustive static system, versus the average complexity of these methods, is plotted in Figure 6.13 for JPEG and in Figure 6.14 for XHTML.

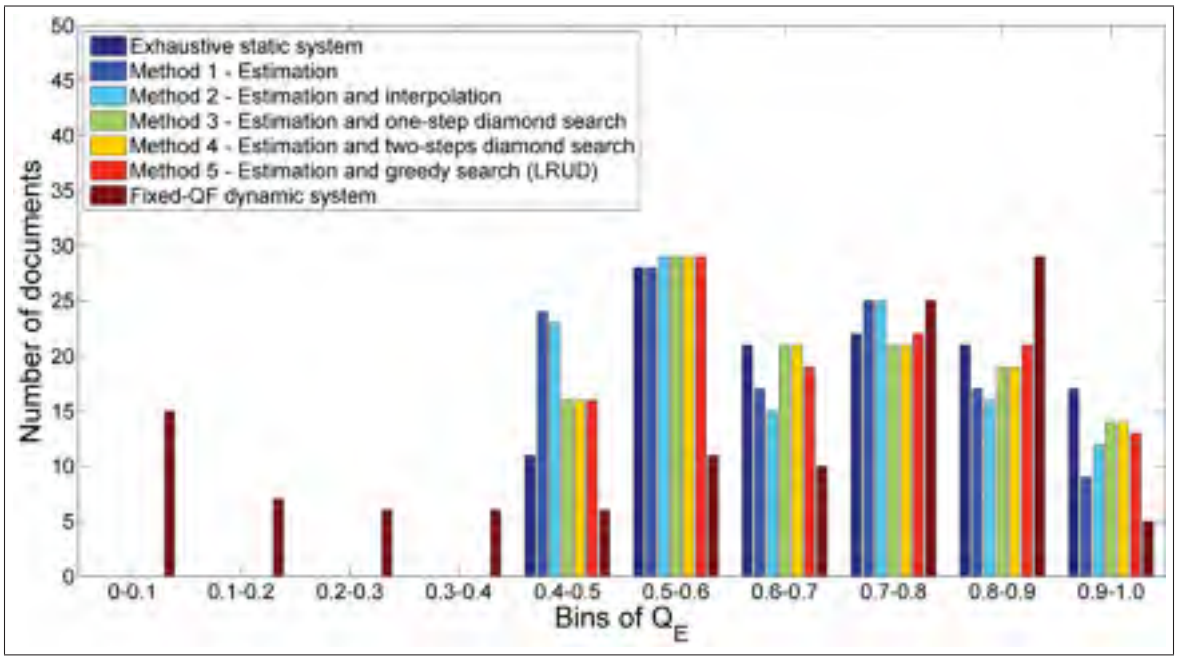

Figure 6.11 Performance of the proposed methods by $\mathcal{Q}_E$ slices of 10% when $f_1^*(c_k, D) =$ JPEG.

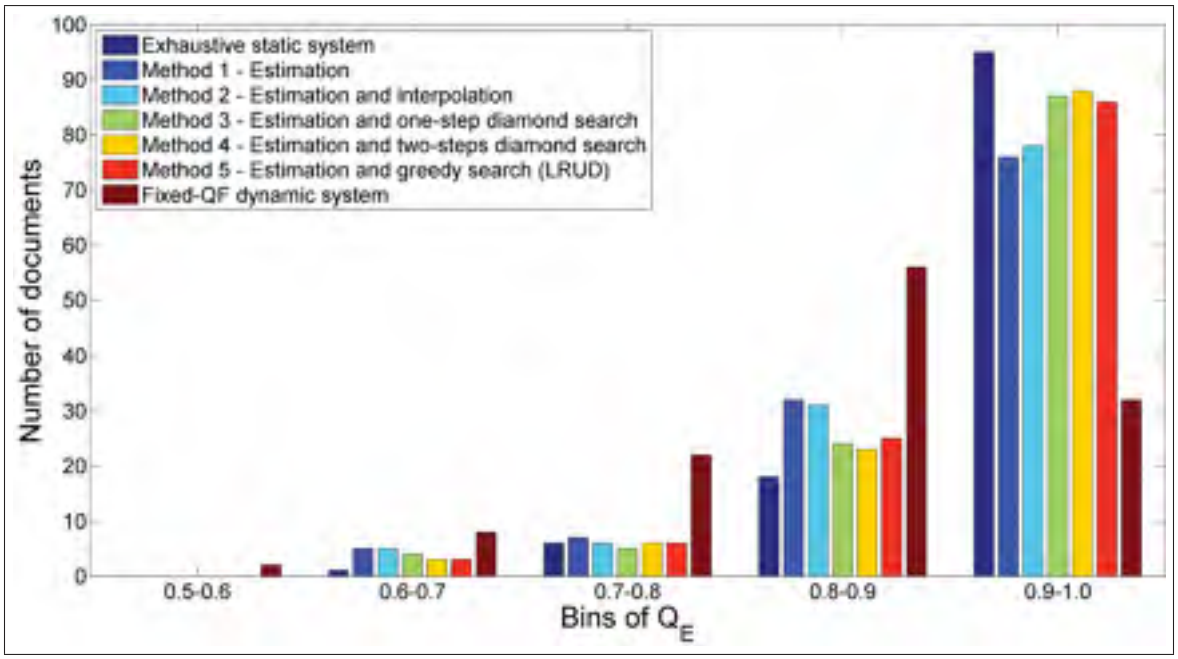

Figure 6.12 Performance of the proposed methods by $\mathcal{Q}_E$ slices of 10% when $f_1^*(c_k, D) =$ XHTML.

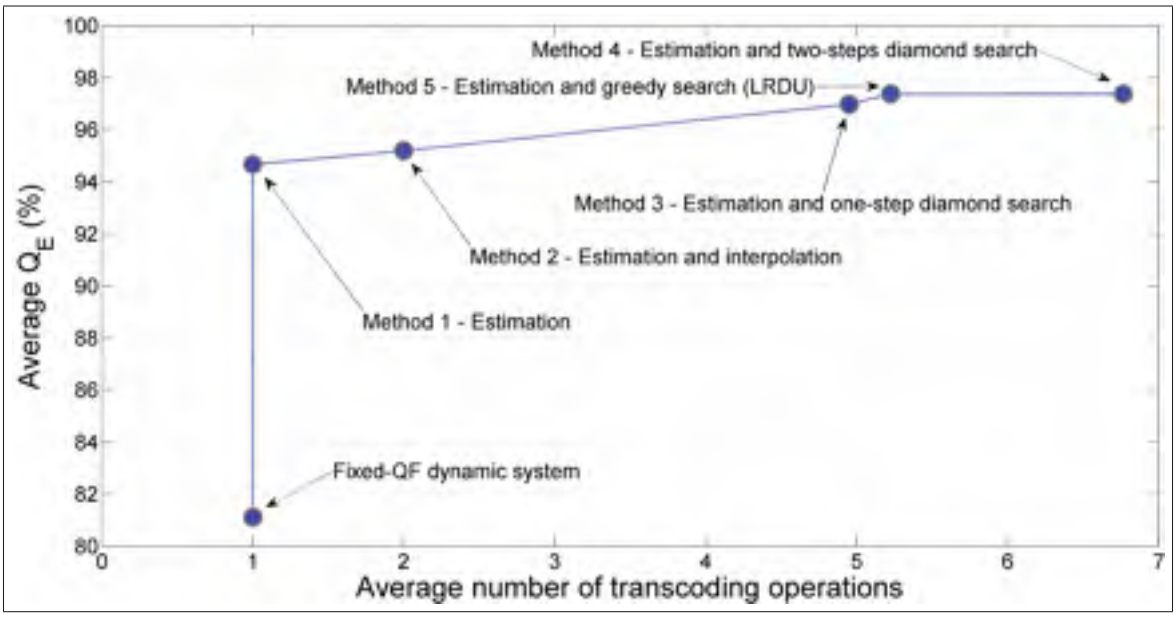

Figure 6.13 Average $\mathcal{Q}_E$ vs. average complexity $f_1^*(c_k, D)$ = JPEG.

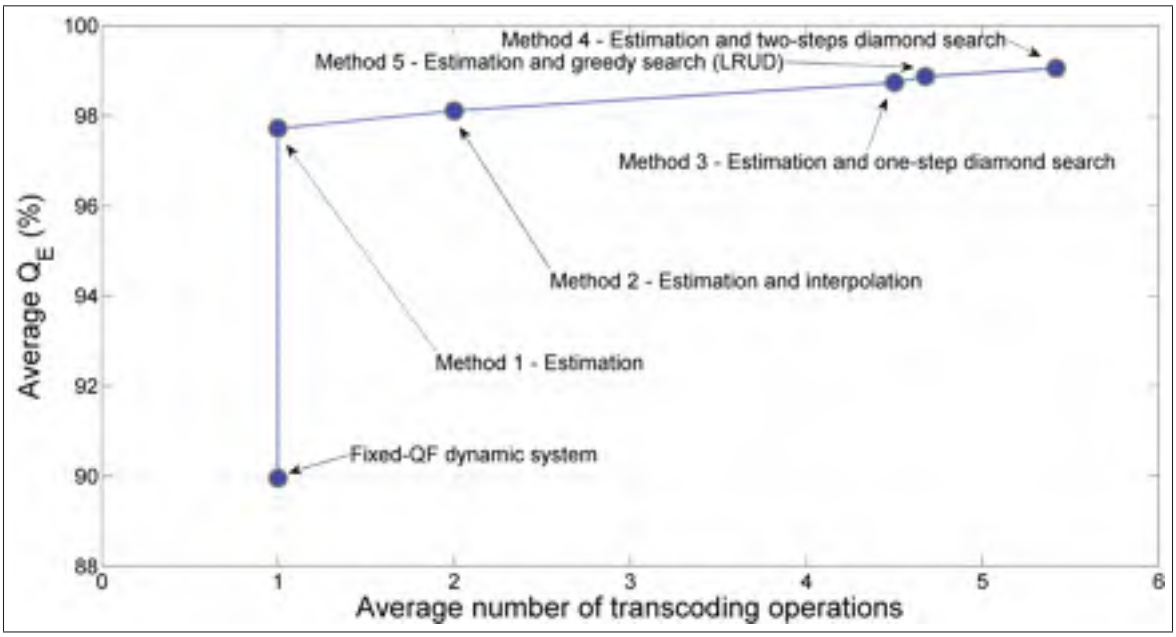

Figure 6.14 Average $\mathcal{Q}_E$ vs. average complexity when $f_1^*(c_k, D)$ = XHTML.

In the case of JPEG, the $\mathcal{Q}_E$ obtained by method 1 (estimation only) is, on average, close to that obtained by the exhaustive static system, that is, 94% for JPEG and 97% for XHTML. They are even closer when the other methods are used: from 94% to 97% for JPEG, and from 97%

to 99% for XHTML. For instance, for JPEG, method 5 reached 97%, which is only 3% away from optimality with a complexity of close to 5 operations, and for XHTML, it is less than 1% from optimality with a complexity near 5 operations.

The average improvement in $\mathcal{Q}_E$ obtained by methods 2 to 5 is relatively small. However, these figures hide the fact that the improvement in $\mathcal{Q}_E$ obtained follows that needed to reach optimality. This is clearly visible in Figures 6.9 and 6.10, where we see that, when the target's relative gain is larger, the improvement obtained is also larger; conversely, the average relative gain is small because, for some slides, the target gain is small as well. This conclusion is also justified by the fact that the $\mathcal{Q}_E$ obtained by method 1 are around 94% for JPEG and 97% for XHTML, which is already an improvement.

Figures 6.9 and 6.10 are very interesting, as they show that we can reach the optimal $\mathcal{Q}_E$ for a large number of slides using the proposed methods, especially using method 5 for JPEG and method 4 for XHTML (the scattered points that are on the diagonal line). Statistically speaking, for JPEG, 10% and 30% of the documents of $\mathcal{V}$ reached optimality using methods 1 and 5 respectively. For XHTML, 45% and 59% of the documents of $\mathcal{V}$ reached optimality using methods 1 and 4 respectively. Furthermore, overall, the results obtained were very reliable and close to optimality, as illustrated in Table 6.1, where the average deviation from optimality is computed, as well as the variance of these deviations, for each method.

Table 6.1 Average deviation from optimality, and its variance.

	Average deviation from optimality		Variance ($\times 10^{-3}$)	
Methods	JPEG	XHTML	JPEG	XHTML
Method 1-Estimation	0.037	0.021	0.872	0.951
Method 2-Estimation and interpolation	0.034	0.018	0.676	0.729
Method 3-Estimation and one-step diamond search	0.021	0.012	0.553	0.508
Method 4-Estimation and two-steps diamond search	0.018	0.009	0.485	0.276
Method 5-Estimation and greedy search	0.018	0.011	0.453	0.320

Unlike JPEG, in the case of XHTML, method 4 is, on average, better than method 5 in terms of performance (average deviation versus complexity). As explained in section 5.7.6, XHTML is very precise, and therefore, we don't need to search far from the second diamond area (as detailed in method 4). Indeed, we reached 1% of average deviation using method 4, which is very close to optimality.

On a final note, from Figure 6.7, we can see that the delivery time is problematic for only about 45% of the slides, where we see the fixed-QF dynamic system performing very poorly. Obviously, transport is an issue, since the fixed-QF dynamic system always yields good visual quality using $QF = 80$. If we use a scenario with a lower bitrate, the number of problematic slides would increase for the fixed-QF dynamic system, and could easily reach 100% with a low enough bitrate. This would make the fixed-QF dynamic system totally unusable. Of course, the opposite is also true, if the bitrate is high enough, the fixed-QF dynamic system would provide an excellent QoE. One advantage of the proposed methods is that they perform as well as possible under any circumstances. In fact, we could even exceed a QF of 80 if the bitrate were very high (while the fixed-QF dynamic system has constant quality). This is very important, as the bitrate can vary significantly during a Web conferencing session.

6.4 Conclusion and future work

Dynamically identifying the optimal transcoding parameters in the context of enterprise document adaptation to handheld devices is not a straightforward task, as the number of parameter combinations could be very high. To tackle this problem, we presented a dynamic content adaptation framework in the previous chapter, the results of which were very reliable and close to optimality. However, the results for some documents were not as close to optimality as the others. To correct this and to improve the results even more, we presented a set of methods in this chapter based on the proposed dynamic framework. These methods use the estimated near-optimal transcoding parameters obtained by this framework and explore their neighborhood to improve their accuracy. Each of the methods has its own specific performance and complexity. The first method is based on the prediction of the quality and file size of JPEG images, subject to changes in their resolution and quality factor. It serves as a good starting point, even

though it exhibits some inaccuracies. These were improved by the other four methods, which involve different levels of accuracy and complexity (number of transcoding operations). For some instances, the optimal transcoding parameters were reached, and for others, the accuracy was improved significantly.

It is important to note that the communication network (bitrate and latency) used has a direct impact on the performance of the first method and on that of the others as well. This was observed from the Zmf curve, which was used to model transport quality. In this chapter, and for the scenario presented, we needed to adjust the adapted content file size by 15% for JPEG. For lower bitrate values, the resulting file size also has to be adjusted by 15% for XHTML (see Appendix D). Consequently, we believe that work is needed to establish the right file size ratio as a function of the network conditions.

The whole framework has been tested on OpenOffice Impress slides, and future work can be carried out to validate its applicability to other enterprise document types, such as MS Word documents and MS Excel spreadsheets.

CHAPTER 7

CONTRIBUTIONS

The scientific contributions of this research are as follows:

1. Innovation in the field of dynamic content adaptation.

2. A prediction-based dynamic content adaptation architecture for enterprise document content adaptation, mainly comprising two important modules: the content adaptation unit, and the decision unit. The first can be used in adapting enterprise documents into Web pages that can be rendered on Web-enabled mobile terminals. The second, which represents the core of the architecture, can be used to estimate, on-the-fly, the $\mathcal{Q}_E$ of adapted content before transcoding, and to decide on the optimal transcoding parameters.

3. Extension of the OpenOffice XHTML filter to be used as a content adaptation unit in the proposed architecture. The native version is very limited and contains various bugs. With the extended version, we give the community a powerful tool that can be used in adapting enterprise documents into conventional Web pages comprising text and images.

4. A quality of experience measure that respects the three requirements of the QoE framework design (Kuipers *et al.*, 2010). It is based on the visual quality of the adapted content, as well as the time it takes for the content to reach the recipient.

5. A dynamic content adaptation framework based on the predicted SSIM and the relative file size of transcoded JPEG images that are tabulated in (Pigeon and Coulombe, 2008; Coulombe and Pigeon, 2009, 2010). Using these predicted values and the proposed QoE measure, the proposed dynamic framework estimates, on-the-fly, near-optimal transcoding parameters to generate adapted content. The proposed solution confers great benefits, compared to current state-of-the-art solutions:

- No need to store multiple transcoded versions.
- Very low computational cost, compared to systems performing several versions.
- Very high quality, compared to systems performing a single transcoding.
- The high quality of systems performing several versions with the complexity of systems performing a single transcoding.

6. A set of methods and models to improve the results obtained by the proposed dynamic framework. These methods are variants of the proposed dynamic framework, with increased performance and lower complexity. Using this set of methods, quality has been increased significantly, and we were able to reach optimality for almost 30% of the documents tested. This figure will vary, of course, depending on the context: target mobile device, and network conditions.

7. Application and validation of the proposed framework on OpenOffice Impress presentation slides.

   - We compared the performance of the proposed dynamic system with a typical dynamic system (fixed-QF) and various static systems (including the exhaustive static system, which had been considered to be optimal). Our results show that the proposed dynamic system is very appealing, as it provides a good compromise between visual quality and delivery time under any circumstances.
   - We rigorously compared JPEG- and XHTML-based Web pages generated by the various validation transcoding systems in terms of $\mathcal{Q}_E$ in the context of enterprise document adaptation to handheld terminals.

8. Proposal of a new perspective on the dynamic adaptation of enterprise documents (which are widely used and shared among peers), based on accurate predictors.

These contributions have been the subject of two journal papers (Louafi *et al.*, 2012, 2013b) and an international conference paper (Louafi *et al.*, 2013a).

CONCLUSION

In this research, we studied the adaptation of enterprise documents, considering two target formats: JPEG-based and XHTML-based Web pages. In the latter, the components of each enterprise document page are converted separately and wrapped in a Web page, which can comprise both text and images. Unlike the JPEG-based format, the XHTML-based format provides more flexibility, allowing text editing and keyword searching.

To dynamically identify the optimal transcoding parameter combinations that provide the end-user with the best user experience possible while satisfying the target mobile device's constraints, we proposed a prediction-based dynamic content adaptation framework. First, we defined an objective quality of experience measure that resolved the major issues arising in the measures presented in the literature. It takes into account the visual quality of the adapted content and the time it takes that content to reach the recipient (transport quality), and so addresses the three requirements needed in any QoE design (Kuipers *et al.*, 2010). Unlike state-of-the-art methods that sum the quality parameters to quantify the quality of the adapted content, we have proposed a method that multiplies two high level quality terms: visual quality, and transport quality. We have shown that quality parameters cannot be summed, as they cannot compensate for one another. Moreover, a product is more appropriate in our context.

Using the proposed QoE measure as a quality criterion, the proposed framework estimates, on-the-fly, near-optimal transcoding parameters (format, scaling, and quality factor), that maximize this measure with less computational complexity. It exploits the predicted SSIM and the relative file sizes of transcoded JPEG images, subject to changing their scaling parameter and quality factor (Pigeon and Coulombe, 2008; Coulombe and Pigeon, 2009, 2010). The framework comprises five methods with increased performance. One of these, which requires only one transcoding operation, estimates near-optimal transcoding parameters dynamically, and the other four improve the accuracy of the results obtained with more (2 to 5 operations) transcoding operations.

The experimental results show that, overall, the estimated transcoding parameters are close to optimality (exhaustive static system) when method 1 was used, and very close to it using the other methods. For instance, with JPEG, the results obtained were 6% and 3% far from optimality respectively, using methods 2 and 5. In the case of XHTML, they were 3% and 1% far from optimality, using methods 2 and 5 respectively. More importantly, for the validation scenarios, we were able to reach optimality in 30% and 59% of the documents for JPEG and XHTML respectively. By contrast, the results of the fixed-QF dynamic system were highly variable, as this system always seeks the best visual quality without any control of the resulting file size, which negatively affects the delivery time and the quality of the experience, making it unusable for low bitrate situations. The framework we propose provides near-optimal adapted content under any circumstances, based on the best compromise between the visual quality of the adapted content and its delivery time.

Future work

The following are some of our observations regarding the proposed dynamic content adaptation framework, and a few suggestions for extending it.

Regarding the QoE measure, we have considered the visual and transport qualities. The research can be extended by also taking into account the battery life of the target mobile device. We believe that it is possible to find the curve that models the behavior of the battery appropriately, in terms of both its life and its impact on the quality of the adapted content to be delivered. This is important, as the battery of the mobile device is very limited, and the content doesn't drain the battery in the usual way. Also, depending on the end-user's circumstances, he may be interested in content that doesn't drain his battery as rapidly. After modeling the behavior of the battery's life as a function of the actual charge, it can be included in the proposed user's preferences model. In this case, the end-user can be asked to express his preferences regarding the adapted content's visual quality, transport quality, and battery quality.

In the proposed QoE measure, we proposed to combine the visual and transport qualities using the well-known PW (*product weighting*) method. It will be very interesting to investigate other

combination methods, especially those used in decision theory, and study their strengths and weaknesses taking into account the problem at hand.

It will be important to investigate the relationship between the proposed $\mathcal{Q}_E$ measure and human perception. Alternatively, a mapping function could be found to map the $\mathcal{Q}_E$ values to human perception.

Finally, the proposed framework has been applied to OpenOffice Impress presentations, which are mostly used in Web conferencing applications. Though our framework is designed to be general, future research could be conducted to validate its applicability to other enterprise document types, such as MS Word and MS Excel.

APPENDIX A

SUMMARY OF THE ADVANTAGES AND DRAWBACKS OF CONTENT REPRESENTATION FORMATS

Table A.1 summarizes the advantages and drawbacks of the content representation formats presented in chapter 2.

Table A.1 Summary of the advantages and drawbacks of each content representation format

The format		**Advantages**	**Drawbacks**
HTML	S5 and HTML-Slidy	- No transformation is needed. - They function on any Web browser.	- Each presentation should be created from scratch. - Existing presentations are not reusable.
	Office filters	- The transformation can be achieved from existing documents. - Reusability is possible.	- The transformation is not fully reliable especially when the document contains figures, forms and animations. - No interactivity with the content. - The annotations could be are problematic.

Continued on next page

The format	Advantages	Drawbacks
Raster	- The transformation is simple to describe. - The format is supported by the majority, if not all, of the Web browsers.	- No support of interactivity, animation and annotations. - Resizing, dynamically, the content cannot be done without affecting its perceptual quality. - Since raster formats are raw pixels (not raw text), embedded text quality is usually degraded. - Increasingly large output file size with increasing resolution. - Increasingly large computational complexity with increasing resolution.

Continued on next page

The format		Advantages	Drawbacks
Video		- Video formats are, more and more, supported by mobile devices. - Low bitrate compression which represents a major advantage for wireless networks.	- These formats present the same drawbacks as the raster ones. - Video encoding/decoding could require high computational resources and more processing time. - Unless each frame is intra coded (in which case compression ratio is low), to access and decode a specific inter frame, we need to decode all frames before it starting at its previous intra frame.
Enriched media	SVG	- It is an open format and is free of utilization. - It is open source code. - The fact that it's based on XML makes it interoperable with other tag-based languages such as HTML. - It is natively supported by the majority of Web browsers, such as Mozilla and Safari.	- It is not supported by all the Web browsers and requires installing appropriate plug-ins. - Limited content creation tools for SVG.

Continued on next page

The format		Advantages	Drawbacks
	Flash	- Very popular format. - Variety of tools can be used to transform a document into Flash.	- Flash is a proprietary format and therefore its utilization requires a license. - It is closed-source code. - It is not extensible. - It is not natively supported byWeb browsers, and so it needs the installation of the appropriate plug-in.
Native Viewer		- The content is rendered in its original format. - No content degradation and thus the content quality is preserved. - No content degradation. - No transformation effort is needed.	- Limited formats are supported by Web browsers. - Very limited plug-ins' APIs to interact with the content.

Continued on next page

The format	Advantages	Drawbacks
Web-based	- Use of Web browsers that are widely supported. - No need to install any additional software. - No need to install any Web browser's plug-ins.	- Some Web technologies must be supported by the Web browser such as JavaScript and Ajax. - Web-based solutions are still young and immature. - These solutions don't offer all the functionalities that an Office suite can offer (at least, currently).

APPENDIX B

CONTENT OF THE FILES USED IN THE EXPERIMENTATION OF SECTION 4.3.2.2

1 Content of the file: *cdcatalog.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
  </cd>
. . .
</catalog>
```

2 Content of the file: *cdcatalog.xsl*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html> <body>
```

```
6      <h2>My CD Collection</h2>
7      <table border="1">
8        <tr bgcolor="#9acd32">
9          <th align="left">Title</th>
10         <th align="left">Artist</th>
11       </tr>
12       <xsl:for-each select="catalog/cd">
13        <tr>
14         <td><xsl:value-of select="title" /></td>
15         <td><xsl:value-of select="artist" /></td>
16        </tr>
17       </xsl:for-each>
18     </table>
19  </body> </html>
20 </xsl:template>
21 </xsl:stylesheet>
```

3 Content of the file: *cdcatalog.html*

```
1  <html>
2  <head>
3  <script>
4  function loadXMLDoc(fname){
5    var xmlDoc;
6    // code for IE
7    if (window.ActiveXObject){
8      xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
9    }
10   // code for Mozilla, Firefox, Opera, etc.
11   else if (document.implementation
12   && document.implementation.createDocument){
13     xmlDoc=
14     document.implementation.createDocument("","",null);
15   }
16   else{
17     alert('Your browser cannot handle this script');
```

```
18    }
19    xmlDoc.async=false;
20    xmlDoc.load(fname);
21    return(xmlDoc);
22  }
23  function displayResult(){
24    xml=loadXMLDoc("cdcatalog.xml");
25    xsl=loadXMLDoc("cdcatalog.xsl");
26    // code for IE
27    if (window.ActiveXObject){
28      ex=xml.transformNode(xsl);
29      document.getElementById("example").innerHTML=ex;
30    }
31    // code for Mozilla, Firefox, Opera, etc.
32    else if (document.implementation
33    && document.implementation.createDocument){
34      xsltProcessor=new XSLTProcessor();
35      xsltProcessor.importStylesheet(xsl);
36      resultDocument=
37      xsltProcessor.transformToFragment(xml,document);
38      document.getElementById("example").
39      appendChild(resultDocument);
40    }
41  }
42  </script>
43  </head>
44  <body id="example" onLoad="displayResult()">
45  </body>
46  </html>
```

APPENDIX C

EXTENDING OPENOFFICE XHTML FILTER

As explained, in this research two kinds of formats were considered; JPEG and XHTML. In other words, the OpenOffice enterprise documents can be converted into JPEG-based and XHTML-based Web pages. The JPEG-based Web page is in fact an XHTML page skeleton wrapping a JPEG image; each slide is converted into an JPEG image and wrapped in an XHTML Web page. The OpenOffice JPEG-based filter already exists and is adequate in the context of desktops. Its portability on mobile devices requires a certain attention, such as deciding on the optimal transcoding parameters to be used in the conversion process, and the presence, on the server, of same set of fonts used in the enterprise document.

However, the XHTML-based Web page, which outputs only text, is very rudimentary and not reliable at all. It doesn't take into account the font issues; all the fonts used in the presentation are replaced by only one font (the Web browser's default font). The images, graphics and even the background are completely absent in the outputted Web pages, though, in the XTML source code, images' binary codes are still included. Besides, the layout aspect is not taken into account; all the objects (text boxes, images, etc) are overlaid on top of each other.

Therefore, we have extended the native OpenOffice XHTML filter by fixing the aforementioned drawbacks and adding some features that were not considered at all; such as the graphics. Note that, only the features which are important to our research are fixed. Figures C-1, C-2 and C-3 depict the impact of the extensions we have achieved. The first figure shows a presentation slide containing text boxes, images and some graphics. The second figure shows the Web page as produced by the native OpenOffice XHTML filter, whereas the third one shows the Web page as produced by the extended version of the OpenOffice XHTML filter.

Note that, the integration of the source code of the extended OpenOffice XHTML filter is under process.

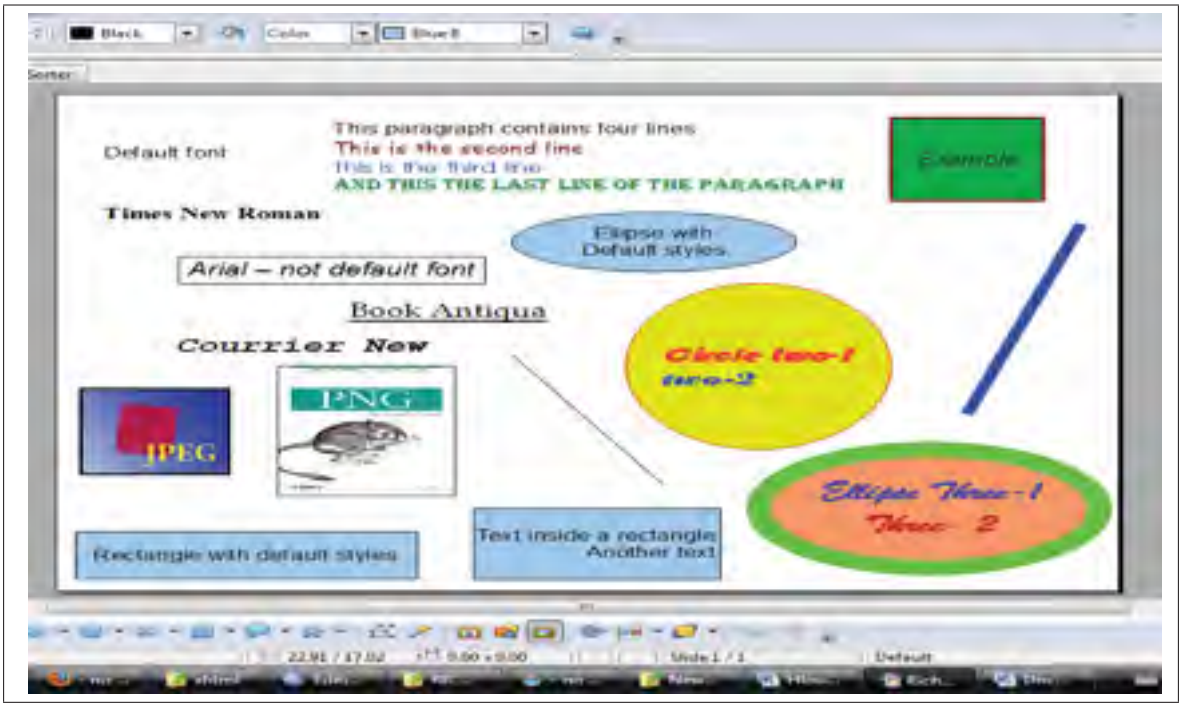

Figure C-1 Example of a slide containing different objects.

The following sections will be about the various identified bugs and how they were fixed as well as the added features.

1 Styles issues in the native XHTML filter

The following is tested using OpenOffice 3.1 version. The XHTML filters of the 3.1 and 2.4 versions have the same behavior, as described in the previous section.

1.1 How the styles are represented in the presentation document

After analyzing how the styling information, used in the presentation document, is represented in the back-end (XML content), we have realized that not all this information is included in the XML content. In fact, the styles used in the presentation are of two types:

1. *The default styles*: When the user writes a text in the presentation without modifying any styles, the default ones are used. In other words, the styling information is not specified

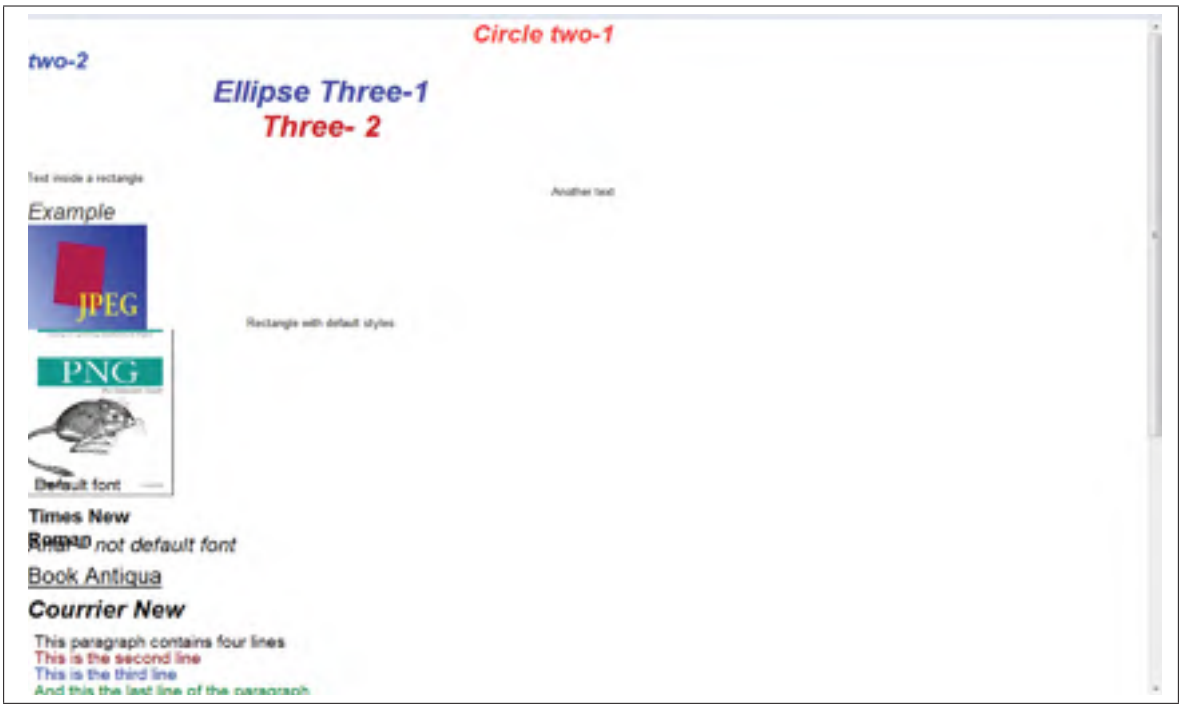

Figure C-2 The Web page version of the previous slide as exported by the native OpenOffice XHTML filter.

by the user, but by the OpenOffice engine. On the back-end side, the styles' information is not included at all in the XML content "content.xml". In fact, they are included in the "styles.xml" file and referenced in the XML content of the presentation.

2. *The user-specified styles*: In this case, the typed text is re-styled by the user, such as modifying the font-name, font-size, font-color, etc. In other words, the styles' information is explicitly specified by the user. In this case, this information is included in the XML content of the presentation

1.2 How the styles are converted by the native XHTML filter

The styles used in the presentation are first collected by the "style_collector.xsl" template. Then, they are converted to CSS styles using the "style_mapping_css.xsl" template. After analyzing how the fonts are collected and mapped to their corresponding CSS styles, we have realized the following:

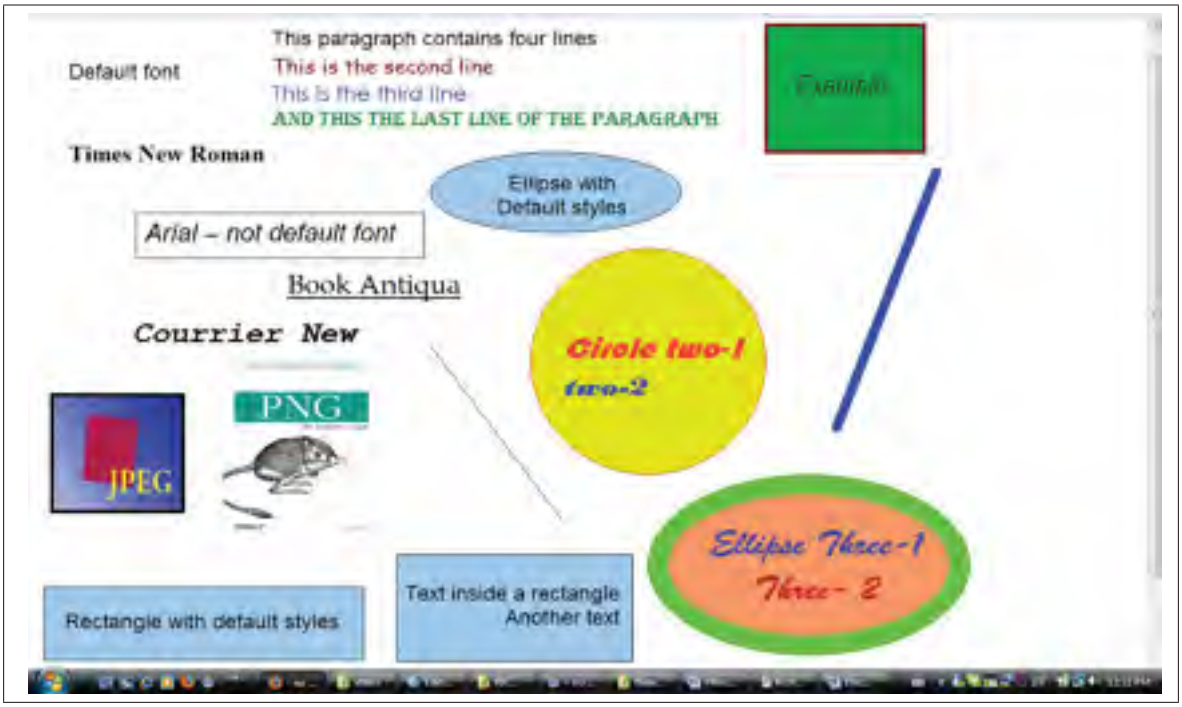

Figure C-3 The Web page version of the previous slide as exported by the extended OpenOffice XHTML filter.

1. *The default styles*: We said that the default styles used in the presentation are not explicitly written in the XML content of the presentation, but in the styles XML file. The filter is able to convert all the styles information, except the font name. In the version of OpenOffice we have installed, the default font name was “Arial”. This information is lost during the export operation; it is replaced by “Times New Roman”. The latter is no more than the default font name of the Web browser. Since no font name is specified by the filter in the XHTML output, the Web browser uses its default font name. But, the good thing is that all the other styles (color, italic, underline, etc) are taken into account by the filter.

2. *The user-defined styles*: When the styles are specified explicitly by the user, we have noticed the same phenomena; the font’s names used in the presentation are not reproduced by the filter.

The problem is in the “style_mapping_css.xsl” template that doesn’t perform the mapping operation correctly. It uses a syntax that is different from that used by OpenOffice in the “content.xml” and “styles.xml” files. After fixing this bug, all the font’s names used in the presentation (the default one and those specified by the user) are reproduced with fidelity as illustrated by Figures C-4, C-5, and C-6.

Figure C-4 A slide that contains different fonts.

2 Layout issues

Figure C-7 shows a slide comprised of seven text boxes. One of them is situated opposite to the others on the right side. The native filter is not able to preserve this layout as shown by Figure C-8; all is aligned to the left. The problem is that the filter doesn’t convert the coordinates of each text box. That’s why they are serialized in the XHTML output. In OpenOffice, these coordinates are referred by x and y; which represent the distances of the text box from the left and top of the slide respectively. They should be converted to their corresponding CSS left and

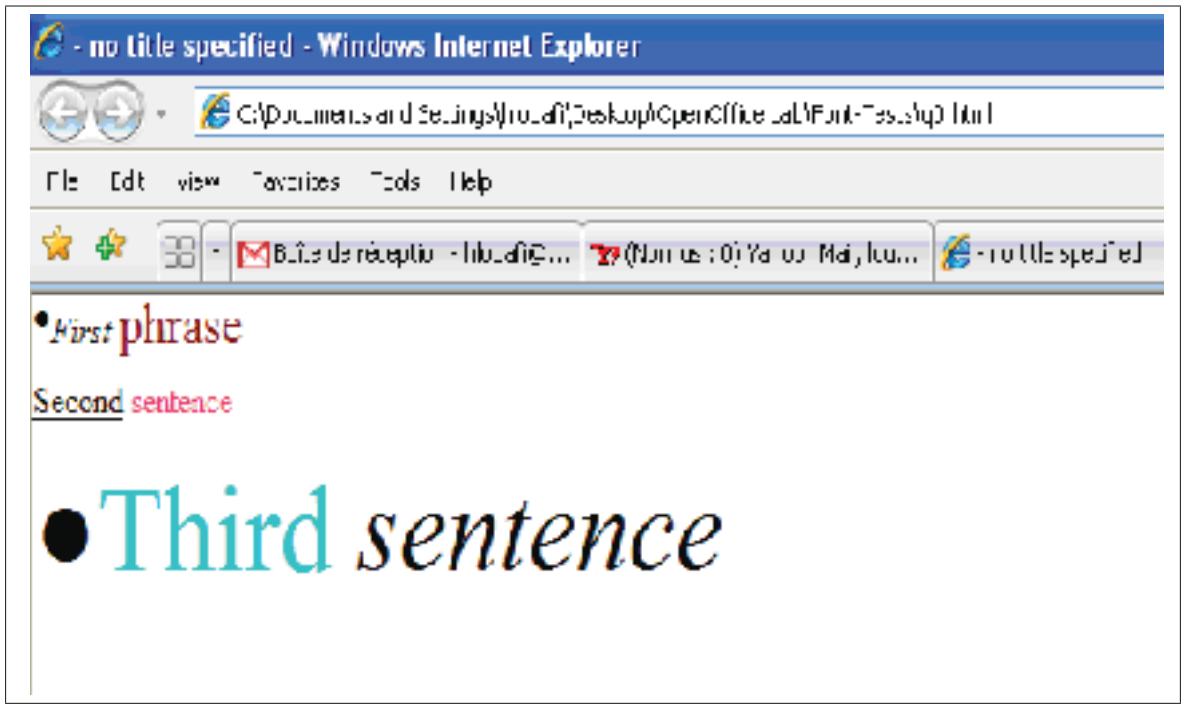

Figure C-5 The XHTML Web page version of the previous slide as exported by the native OpenOffice XHTML filter.

top positions. Figure C-9 shows the XHTML output after fixing this issue. The modifications have been done in the template: "..\OpenOffice.org 3\Basis\share\xslt\export\xhtml\body.xsl"

3 Images issues

The OpenOffice native XHTML filter converts images into binary codes and incorporates them in the XHTML code. It doesn't use the traditional technique that consists in putting all the images in a folder and including URL references to them in the XHTML code. Figure C-11 shows the XHTML source code of the example shown by Figure C-10. Albeit, the embedded images are binary coded, they can be rendered by the majority of the Web browsers, such as Firefox, Safari, etc. The problem is with the MS-IE, which is not able to render binary coded images.

In fact, the images are coded using the base-64 encoding (Kolich, 2009). This way of coding images has a negative impact on the file size of the presentation. On average, an image encoded

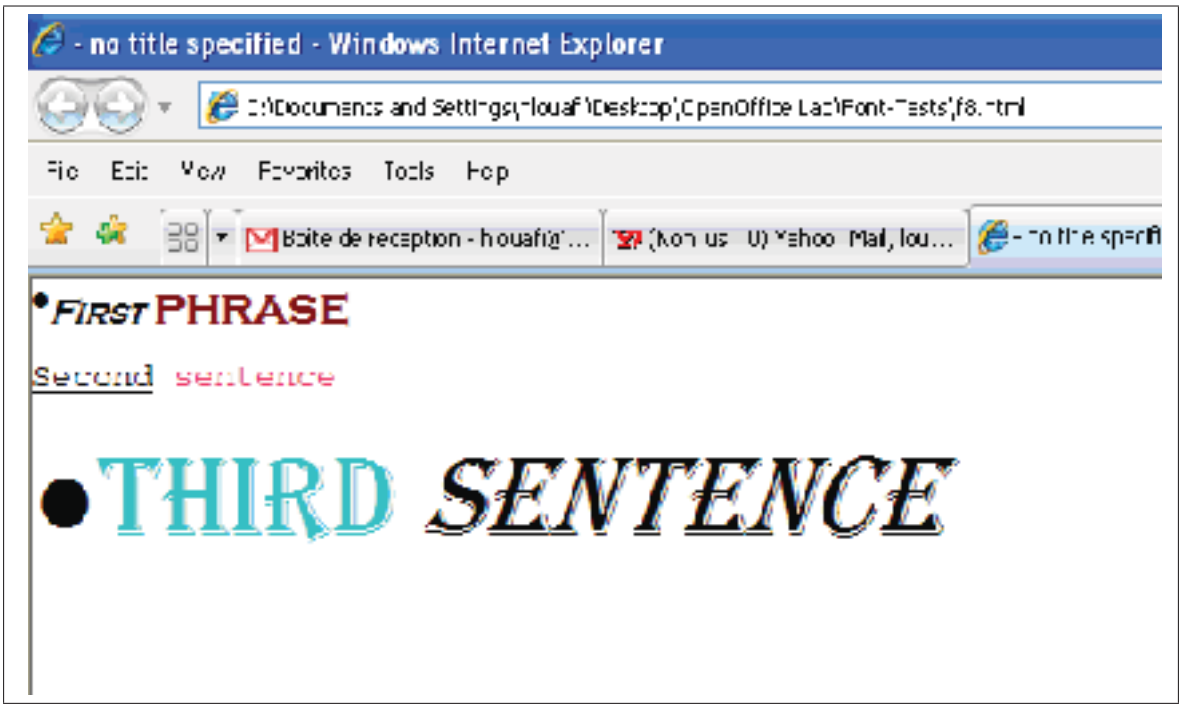

Figure C-6 The XHTML Web page version of the previous slide as exported by the extended OpenOffice XHTML filter.

using base-64 is 33% larger than a binary image (Wikipedia, 2013). Besides, there are pros and cons to using this kind of representation; file size and bandwidth versus network latency.

Two options are offered when the image is to be inserted in the presentation. When the user is asked to enter or select the file name of the image to insert, a check box, named *Link*, can be checked to indicate whether the physical URL of the image should be inserted or not.

1. If *Link* is not checked, the image is converted into base-64 and included in the content XML file. In the latter, a relative link to PNG files is used as a URL to the image.

2. If *Link* is checked, the image is not converted and an absolute URL link is inserted in the XML content file. That URL represents the physical location, in the disk, of the inserted image. This way, the URL and type of the image (e.g., JPEG, GIF,...) are preserved.

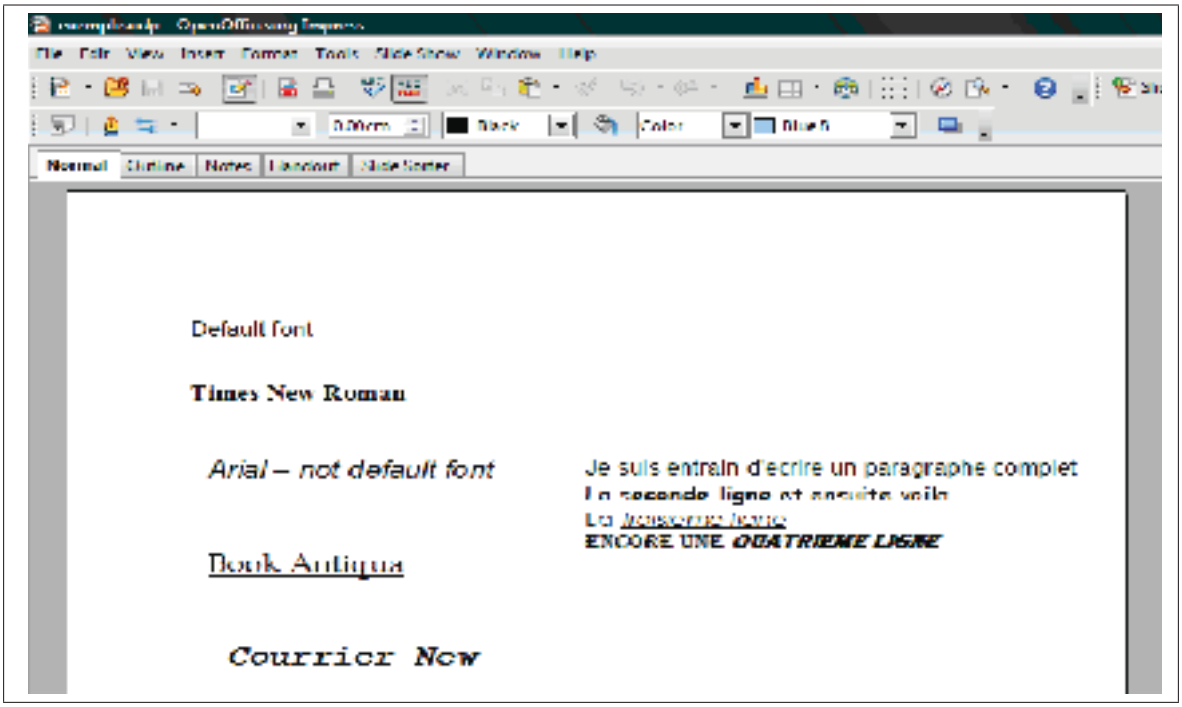

Figure C-7 A slide that contains different textboxes dispersed.

Using the second option, it is possible to use the traditional way of representing images in Web pages. This way, it is possible to insert the URL of the image in the XHTML code and adding an extra folder; which includes the embedded images.

4 Graphics issues

In order to enhance the existing OpenOffice XHTML filter, we have added the possibility to export the following graphics: rectangles, circles, ellipse and lines. To this end, we have added a Javascript library that enables us to draw different shapes on the Web browser. This library is free of use and open source code (Zorn, 2008). Figure C-12 shows an example of a slide containing rectangles, an ellipse, a circle and two lines. The existing filter doesn't output any graphics. As illustrated by Figure C-13, the modifications added to the filter are able to render those shapes on Web pages. The modifications have been achieved as follows:

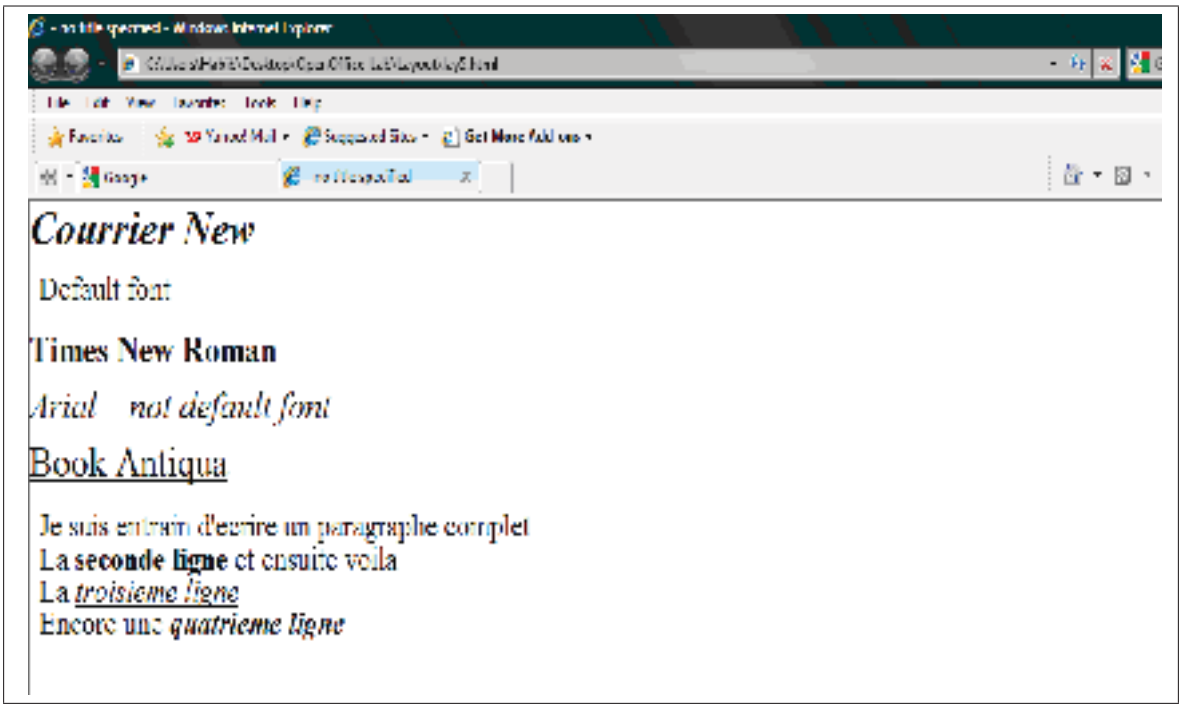

Figure C-8 The XHTML Web page version of the previous slide as exported by the native OpenOffice XHTML filter.

1. The "..\OpenOffice.org 3\Basis\share\xslt\common\measure_conversion.xsl" is included in the template:
   "..\OpenOffice.org 3\Basis\share\xslt\export\common\styles\style_mapping_css.xsl".

2. The templates used to capture the styles used by the shapes; such as the color of the line and the background color are added to the template:
   "..\OpenOffice.org 3\Basis\share\xslt\export\common\styles\style_mapping_css.xsl".

3. The templates used to draw the rectangle, ellipse, circle and line shapes are added to the template:
   "..\OpenOffice.org 3\Basis\share\xslt\export\xhtml\body.xsl".

4. In the XSLT source code, these templates can be located by searching their corresponding keyword. For example, to locate the template that draw the ellipse, you can use the "draw:ellipse" keyword.

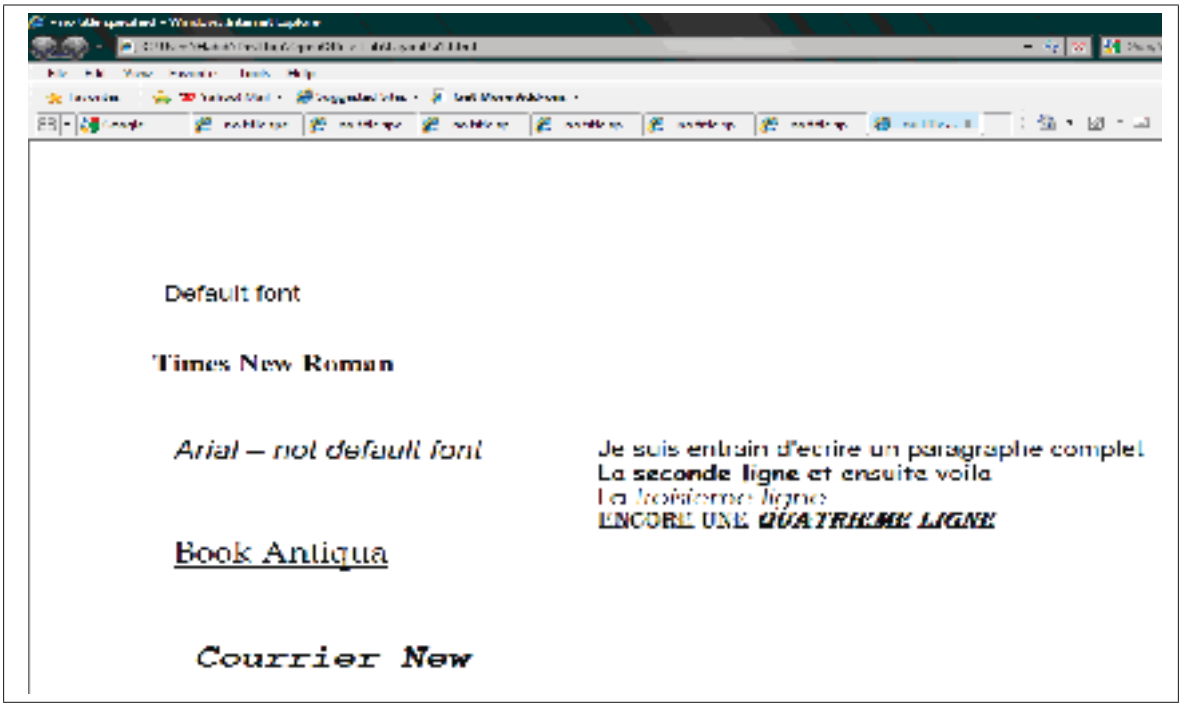

Figure C-9 The XHTML Web page version of the previous slide as exported by the extended OpenOffice XHTML filter.

5 Text-boxes and images adaptation

Now, the extended version of the XHTML filter allows us to adapt presentation slides into XHTML web pages comprised of text-boxes and images. The next step is to give this filter the possibility to adapt the presentation document components using a scaling parameter (which can be applied to both text boxes and images) and a quality factor (applied to embedded JPEG images).

To allow the XHTML filter resizing text boxes using a scaling parameter z, we modified the template: "..\OpenOffice.org 3\Basis\share\xslt\export\xhtml\body.xsl". For embedded JPEG images, we developed a Java-based application that uses imageMagick tools to convert images using z and QF. Then we registered this application with OpenOffice by providing the class-path in which the application is stored on the disk. As a result, using OpenOffice APIs, it is possible now to adapt presentation document, using the desired scaling parameter and quality factor, into Web pages renderable on Web-enabled mobile devices. Figure 5.7 form Chap-

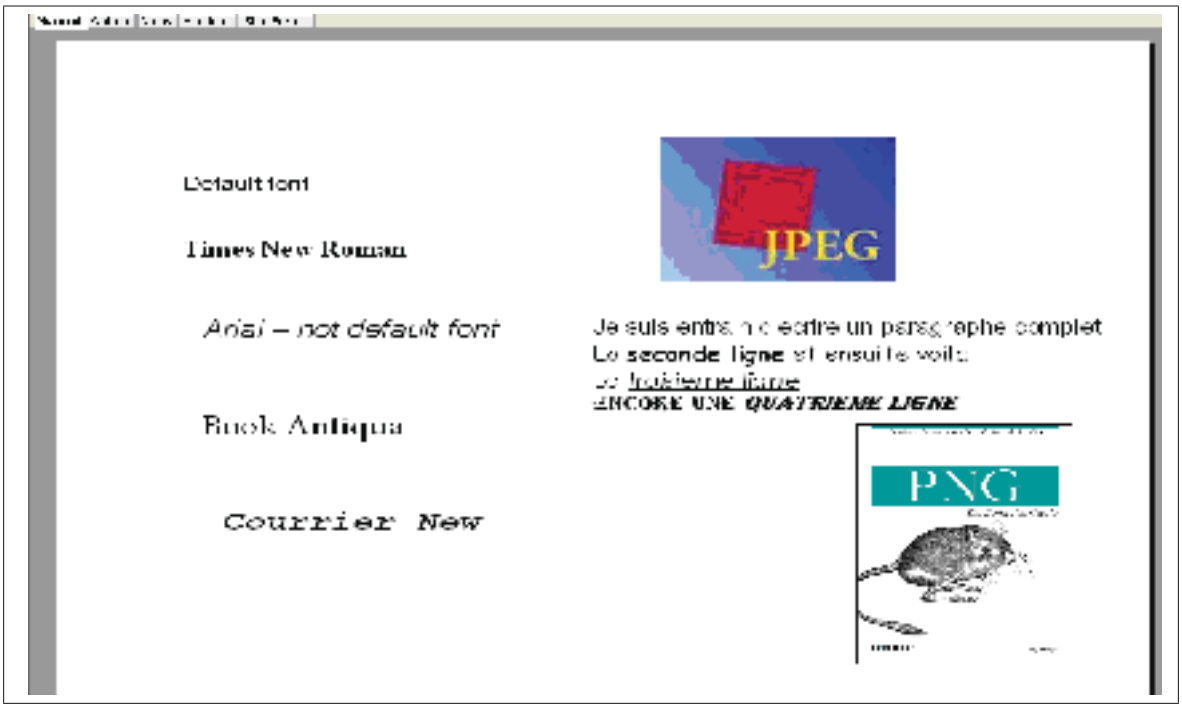

Figure C-10 A slide that contains different text boxes and images.

ter 5 shows a slide as exported by the OpenOffice XHTML extended version using different combinations of z and QF.

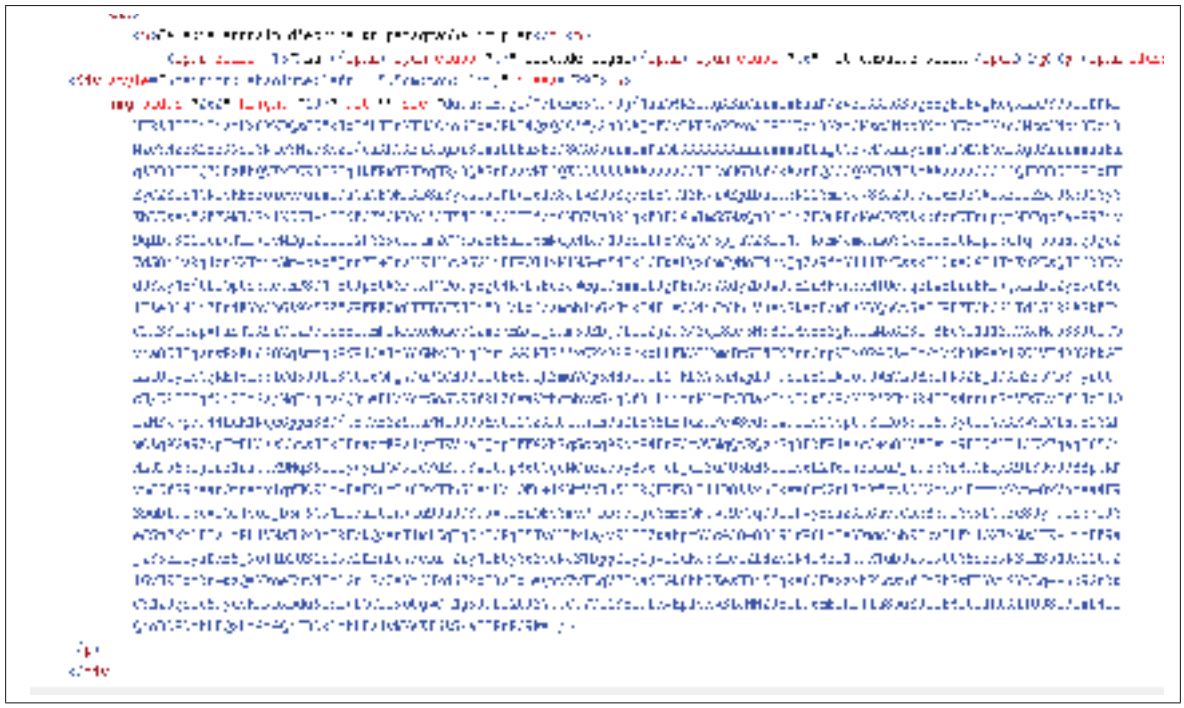

Figure C-11 The source code of the XHTML Web page version of the previous slide as exported by the native OpenOffice XHTML filter (The images are binary coded).

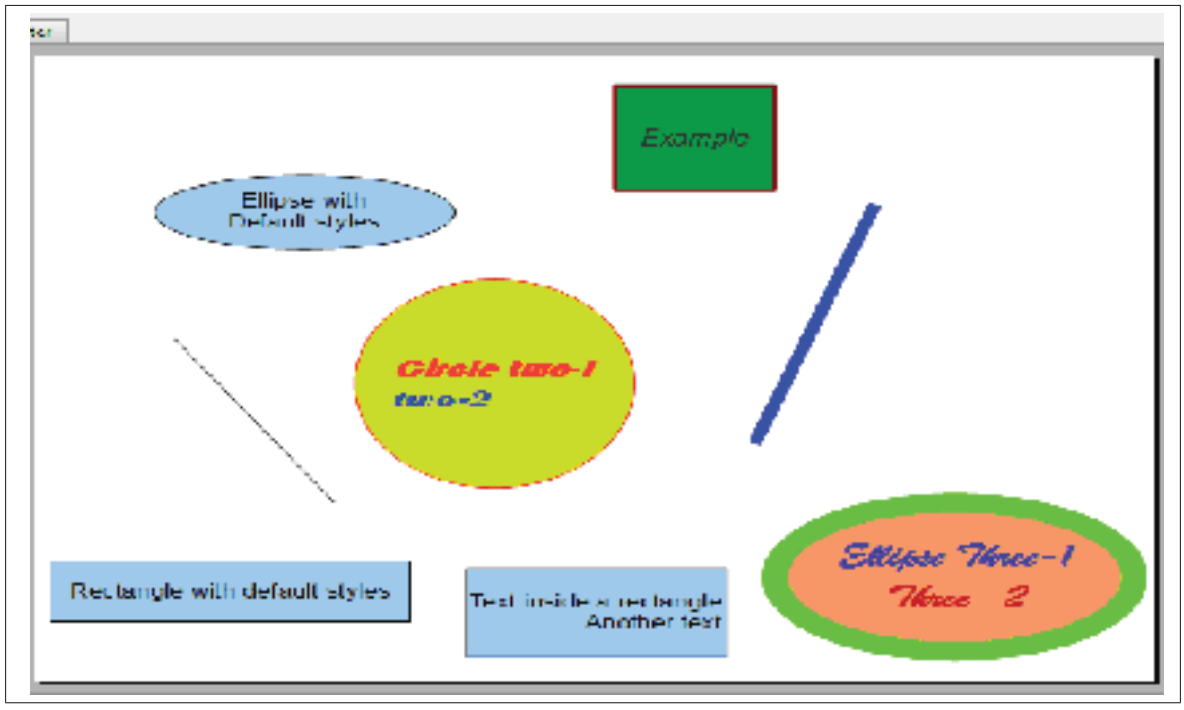

Figure C-12 A slide that contains different shapes.

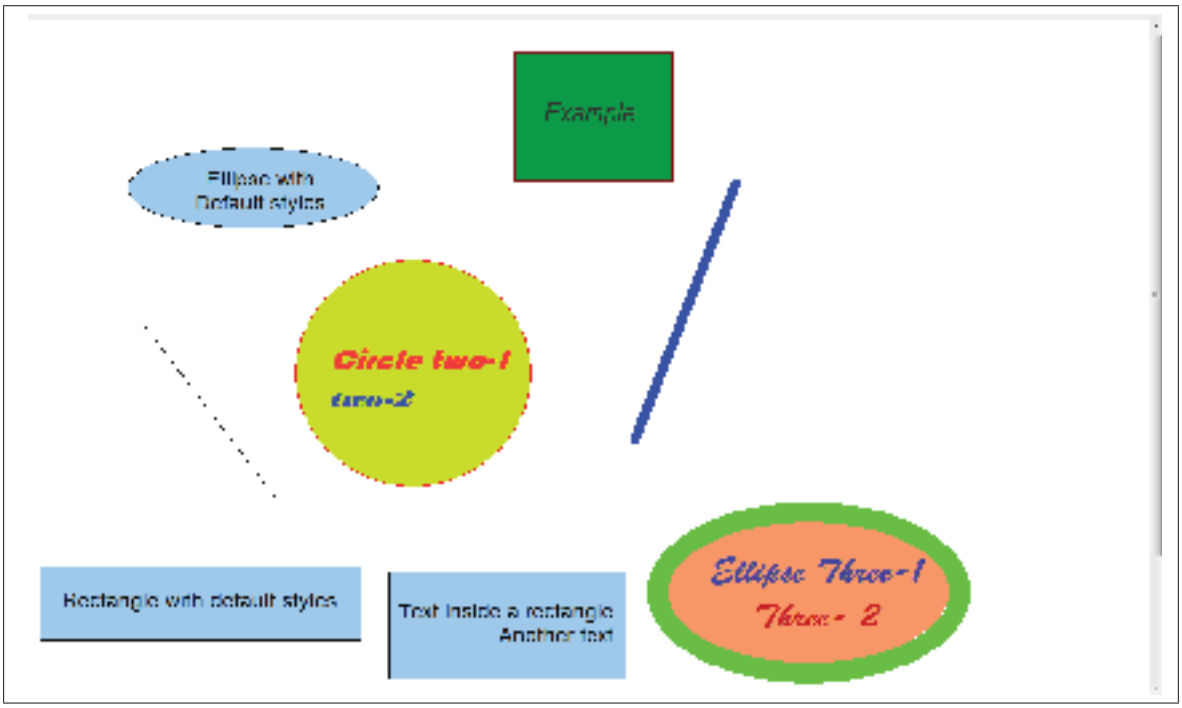

Figure C-13 The XHTML Web page version of the previous slide as exported by the actual OpenOffice XHTML filter.

APPENDIX D

ADDITIONAL EXPERIMENTAL RESULTS

In this appendix, we present additional results obtained with other context information, such as low and high network bitrate values and different combinations of user's preferences. These results show the performance of the proposed dynamic framework in different environments. They complement those presented in chapters 5 and 6. Therefore, before reading this appendix, the reader is invited to first read both chapters 5 and 6.

1 Network conditions

In this section, we show the impact of the network conditions, particularly the variation of the bitrate, on the performance of the proposed dynamic content adaptation framework.

1.1 Low network bitrate

In this experiment we keep the same context parameters as in chapters 5 and 6, and suppose that the actual network bitrate has decreased to $N_{_B}(D) = 20$ kbps.

Figures D-1 and D-2 show the optimal $\mathcal{Q}_{_E}$ obtained by the proposed dynamic framework versus that obtained by the exhaustive static system for JPEG and XHTML, respectively. These results were computed without adjusting the estimated file size. Globally, the obtained results are close to optimality and as expected they present some outliers points for both JPEG and XHTML. Therefore, we multiplied the estimated file size by a ratio of 1.15, and the results are presented in Figures D-3 and D-4. These figures show also the performance of the whole framework (methods 1 to 5) versus the exhaustive static and fixed-QF dynamic systems. As expected, the fixed-QF dynamic system performs very poorly under lower bitrate values whereas the proposed dynamic system is very close to optimality. Statistical details, showing the average deviation from optimality and its variance, are presented in Table D-1.

2 High network bitrate

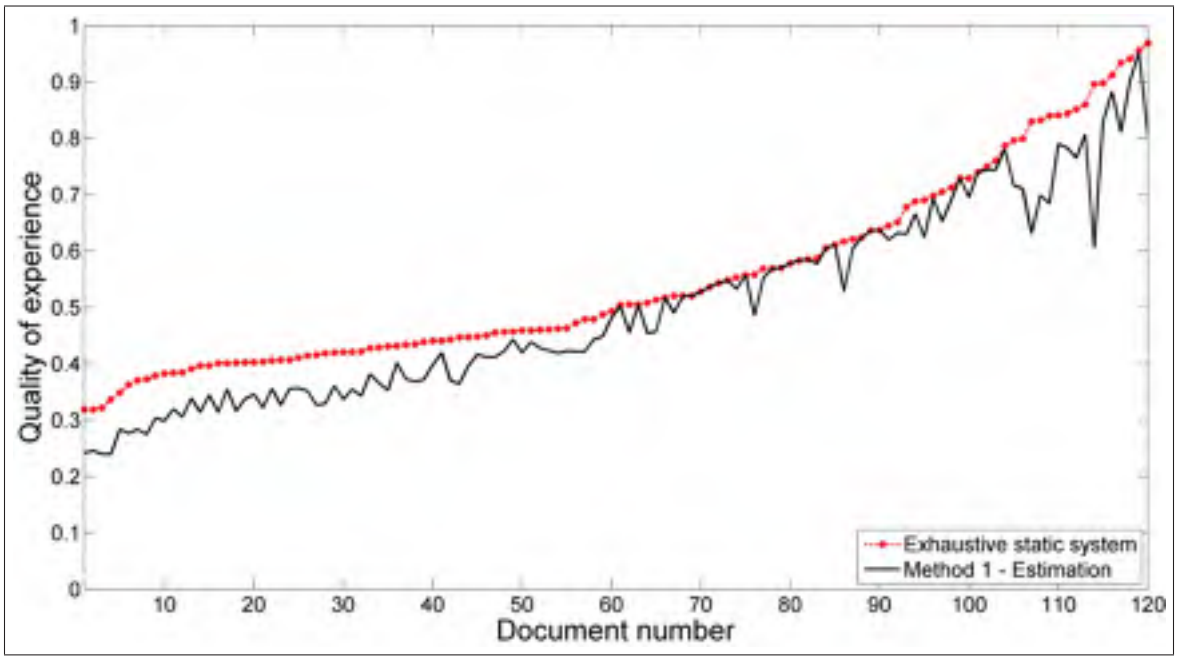

Figure D-1 JPEG optimal $\mathcal{Q}_E$ obtained by our framework (method 1) vs. that of the exhaustive static system computed with a file size ratio of 1, $N_B(D) = 20$ kbps and $N_L(D) = 488$ ms.
The slides are sorted according to the $\mathcal{Q}_E$ of the exhaustive static system.

Table D-1 Average deviation form optimality and its variance as computed with a file size ratio of 1.15, $N_B(D) = 20$ kbps and $N_L(D) = 488$ ms.

Methods	Average deviation from optimality		Variance ($\times 10^{-3}$)	
	JPEG	XHTML	JPEG	XHTML
Method 1-Estimation	0.058	0.029	0.985	1.248
Method 2-Estimation and interpolation	0.055	0.027	0.997	0.959
Method 3-Estimation and one step diamond search	0.027	0.014	0.785	0.530
Method 4-Estimation and two steps diamond search	0.017	0.011	0.529	0.324
Method 5-Estimation and greedy search	0.016	0.012	0.445	0.315

In this experiment, we suppose that the mobile device D is connected to an EDGE network that has the following characteristics: $N_B(D) = 240$ kbps and $N_L(D) = 504$ ms (Svoboda *et al.*, 2007). The optimal $\mathcal{Q}_E$ obtained by the proposed dynamic system is compared to that obtained by the exhaustive static and fixed-QF dynamic systems, and the results are presented in Figures D-5 and D-6 for JPEG and XHTML, respectively.

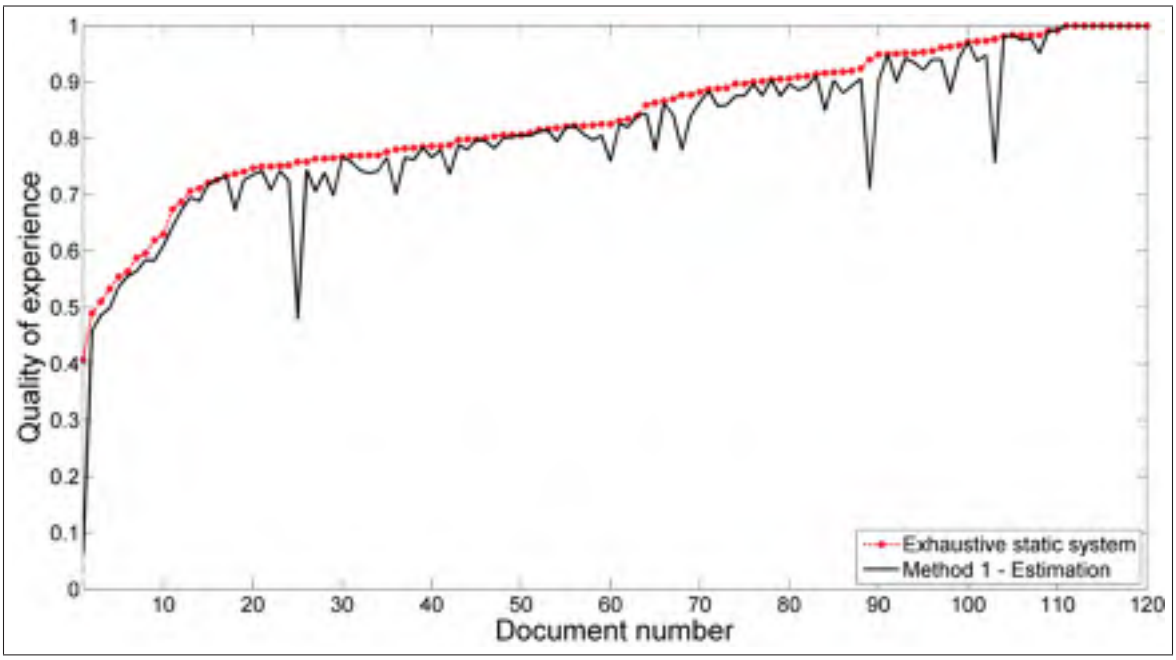

Figure D-2 XHTML optimal $\mathcal{Q}_E$ obtained by our framework (method 1) vs. that of the exhaustive static system computed with a file size ratio of 1, $N_B(D) = 20$ kbps and $N_L(D) = 488$ ms.
The slides are sorted according to the $\mathcal{Q}_E$ of the exhaustive static system.

For JPEG, the optimal $\mathcal{Q}_E$ obtained by the proposed dynamic framework is close to optimality, especially using methods 4 and 5. Besides, almost 50% of the documents reached optimality, as shown in Figure D-5. Note that, these results were computed without adjusting the estimated file size. To improve even more the obtained results accuracy, we tested various file size ratio values (5%, 10%, 15% and 20%) and no noticable improvement was obtained. As discused in chapter 6, we believe, that future research should be conducted to establish the right file size ratio in fucntion of the network conditions. The fixed-QF-dynamic system is still far from optimality compared to the proposed dynamic framwork.

For XHTML, the obtained results are exceptional as we reached optimality in 99% of the documents. This confirms again the fact that XHTML is more precise than JPEG, wich makes it a very attractive format to consider in enterprise documents adapation. On the other hand, the fixed-QF dynamic system is very variable and gets far from optimality for a large number of documents.

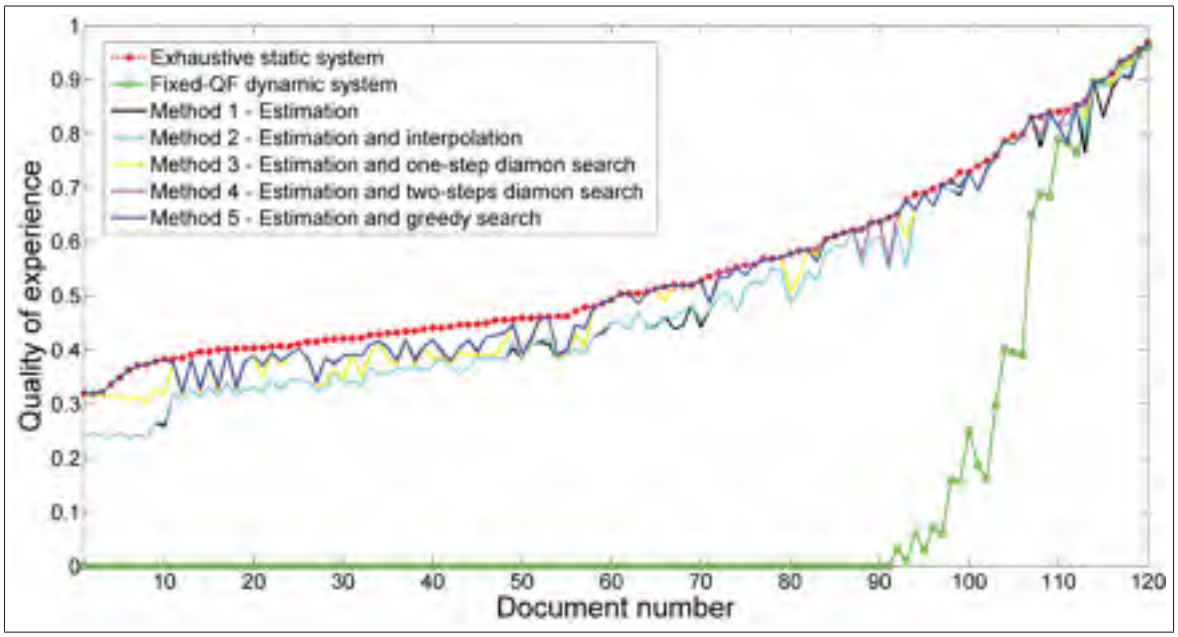

Figure D-3 JPEG optimal $\mathcal{Q}_E$ obtained by our framework vs. that of the exhaustive static and fixed-QF dynamic systems computed with a file size ratio of 1.15, $N_B(D) = 20$ kbps and $N_L(D) = 488$ ms.
The slides are sorted according to the $\mathcal{Q}_E$ of the exhaustive static system.

The accuracy of the obtained results can be read also from Table D-2, which shows the average deviation form optimality and its variance for both JPEG and XHTML.

Table D-2 Average deviation form optimality and its variance as computed with a file size ratio of 1, $N_B(D) = 240$ kbps and $N_L(D) = 504$ ms.

Methods	Average deviation from optimality		Variance ($\times 10^{-3}$)	
	JPEG	XHTML	JPEG	XHTML
Method 1-Estimation	0.070	0	7.443	0
Method 2-Estimation and interpolation	0.054	0	5.329	0
Method 3-Estimation and one step diamond search	0.047	0	5.526	0
Method 4-Estimation and two steps diamond search	0.043	0	5.449	0
Method 5-Estimation and greedy search	0.043	0	5.710	0

3 User's preferences

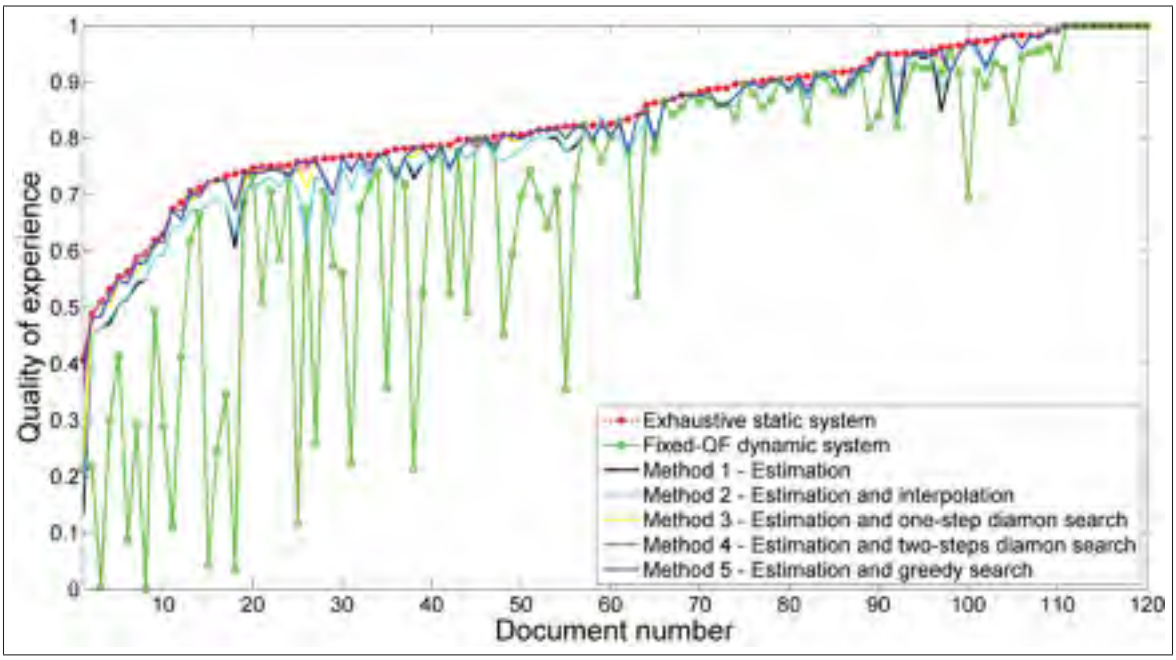

Figure D-4 XHTML optimal $\mathcal{Q}_E$ obtained by our framework vs. that of the exhaustive static and fixed-QF dynamic systems computed with a file size ratio of 1.15, $N_B(D) = 20$ kbps and $N_L(D) = 488$ ms.
The slides are sorted according to the $\mathcal{Q}_E$ of the exhaustive static system.

In chapters 5, we proposed a user's preferences model, in which the user's preferences are applied to the visual and transport qualities as positive exponents, the value of which are confined between 0 and 1 (see equation 5.25). We used neutral weight values ($\mathcal{W}_V(D) = 1$, $\mathcal{W}_T(D) = 1$) in the experiments of chapters 5 and 6. In this section we want to show the behavior of the three qualities ($\mathcal{Q}_V$, $\mathcal{Q}_T$ and $\mathcal{Q}_E$) when different weight values are used. Therefore, we proposed to test three weight combinations:

$$\begin{aligned} \mathcal{W}_V(D) = 1, \mathcal{W}_T(D) = 1 \\ \mathcal{W}_V(D) = 0.5, \mathcal{W}_T(D) = 1 \\ \mathcal{W}_V(D) = 1, \mathcal{W}_T(D) = 0.5 \end{aligned} \quad \text{(A D-1)}$$

Using these weight combinations, we computed the optimal $\mathcal{Q}_E$ and its corresponding $\mathcal{Q}_V$ and $\mathcal{Q}_T$ using method 1 (estimation only) and the same mobile device used in chapter 5 ($N_B(D) = 50$ kbps and $NLD = 488$ ms). Figures D-7, D-8 and D-9 show the obtained optimal $\mathcal{Q}_V$,

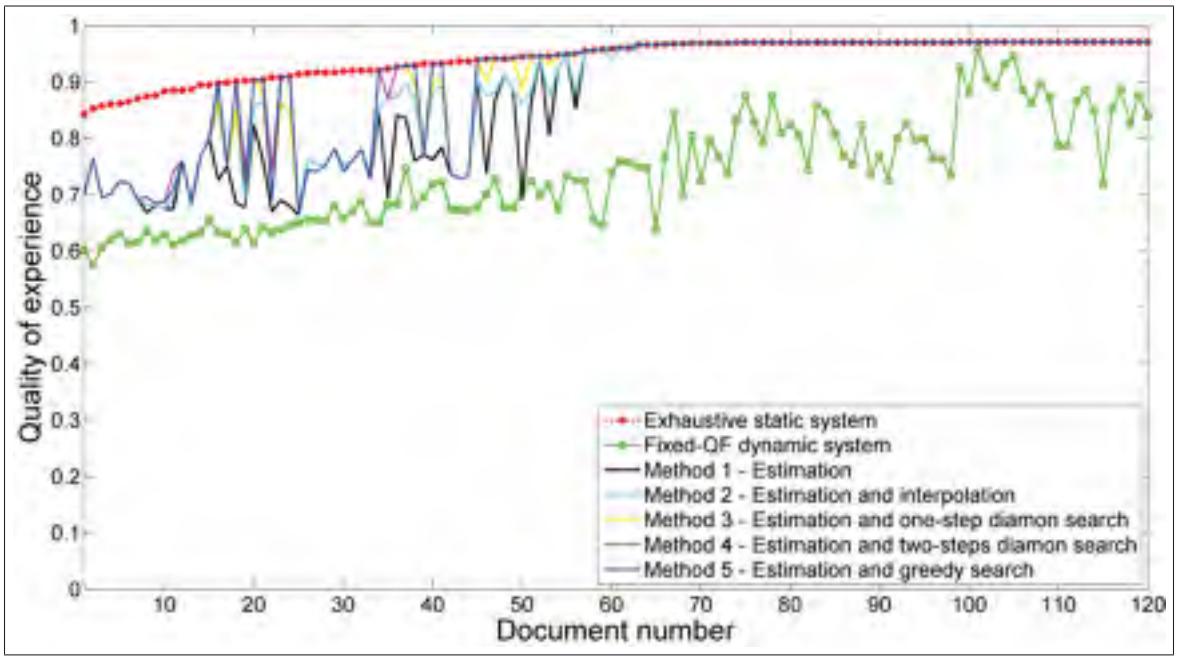

Figure D-5 JPEG optimal $\mathcal{Q}_E$ obtained by our framework vs. that of the exhaustive static and fixed-QF dynamic systems computed with a file size ratio of 1, $N_B(D) = 240$ kbps and $N_L(D) = 504$ ms.
The slides are sorted according to the $\mathcal{Q}_E$ of the exhaustive static system.

$\mathcal{Q}_T$ and $\mathcal{Q}_E$ for JPEG, respectively. Those obtained in the case of XHTML are presented in Figures D-10, D-11 and D-12.

As mentioned earlier, the weights are interpreted as penalties. For instance, when a weight value of $\mathcal{W}_V(D) = 0.5$ is used instead of 1, the weighted $\mathcal{Q}_V$ is increased and the obtained optimal $\mathcal{Q}_V$ should be decreased (see section 5.4). As shown in these figures, when the combination $(\mathcal{W}_V(D) = 0.5, \mathcal{W}_T(D) = 1)$ is used, overall the obtained $\mathcal{Q}_V$ is smaller than that obtained with no change in the weights (or using 1 as a weight value for both $\mathcal{Q}_V$ and $\mathcal{Q}_T$). Similarly, using the combination $(\mathcal{W}_V(D) = 1, \mathcal{W}_T(D) = 0.5)$, we obtained a lower $\mathcal{Q}_T$ compared to that obtained without applying the weights. These observations are quite visible in the case of JPEG. In the case of XHTML, we observed the same change in $\mathcal{Q}_V$ and $\mathcal{Q}_T$ and of course in $\mathcal{Q}_E$. This is an interesting and particular case, in which the estimated transcoding parameters are the same when the two weight combinations are used. This is quite reasonable, as the estimated transcoding parameters are the same, their corresponding computed (not

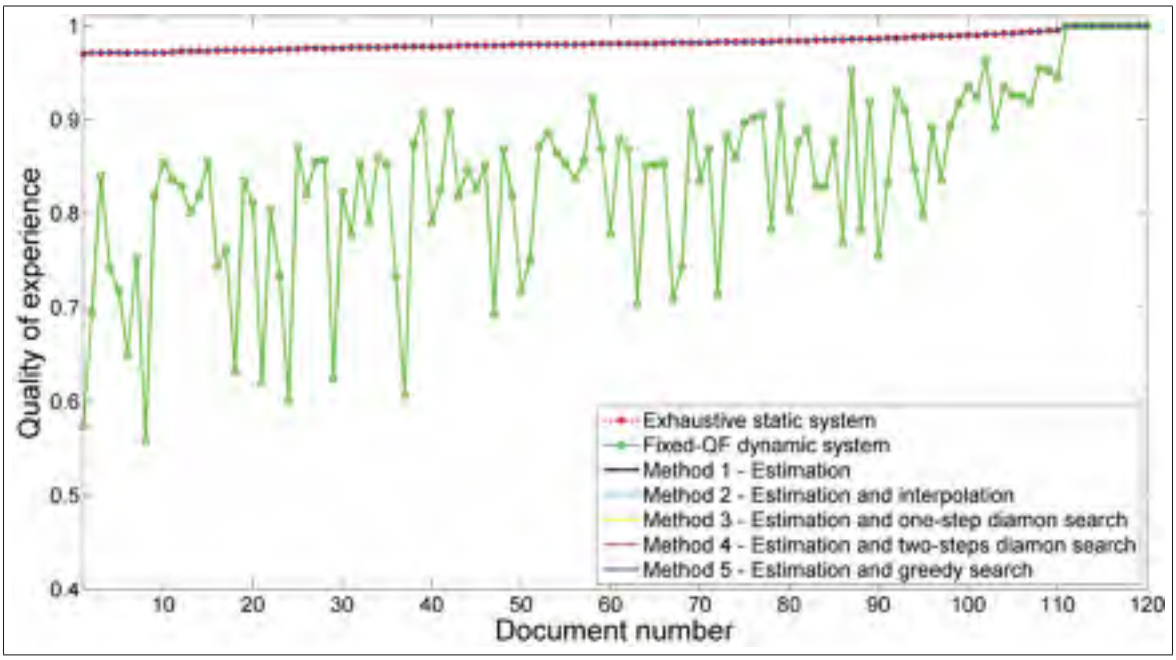

Figure D-6 XHTML optimal $\mathcal{Q}_E$ obtained by methods 1 to 5 vs. that of the exhaustive static and fixed-QF dynamic systems computed with a file size ratio of 1, $N_B(D) = 240$ kbps and $N_L(D) = 504$ ms.
The slides are sorted according to the $\mathcal{Q}_E$ of the exhaustive static system.

estimated) $\mathcal{Q}_V$, $\mathcal{Q}_T$ and $\mathcal{Q}_E$ should be the same. Though, we obtained in this case the same estimated transcoding parameters, their corresponding estimated $\mathcal{Q}_V$, $\mathcal{Q}_T$, however, are different. This is shown by Figure D-13, in which the differences between the estimated $\mathcal{Q}_V$, $\mathcal{Q}_T$ and $\mathcal{Q}_E$ obtained using the two weight combinations under consideration, are shown. From this figure, we see that the estimated $\mathcal{Q}_E$ has not changed (which is behind the obtained similar transcoding parameters), but there are actually differences in $\mathcal{Q}_V$ and $\mathcal{Q}_T$. In other words, the estimated $\mathcal{Q}_V$ and $\mathcal{Q}_T$ were compensated in both weight combination cases.

We note that, using the user's preferences, we observed that the obtained $\mathcal{Q}_E$ change following the weight combination in use, as shown in Figures D-9 and D-12. Moreover, the change in one quality is compensated by the other one. These observation is clearly visible in the case of JPEG (Figure D-9). In the case of XHTML, the compensation is not visible, since we obtained the same $\mathcal{Q}_E$ when the two weight combinations where used.

Lastly, we can say that the future work can be carried out to evaluate the usability and effectiveness of the proposed user's preferences model in other contexts, such as Web or video content adaptation.

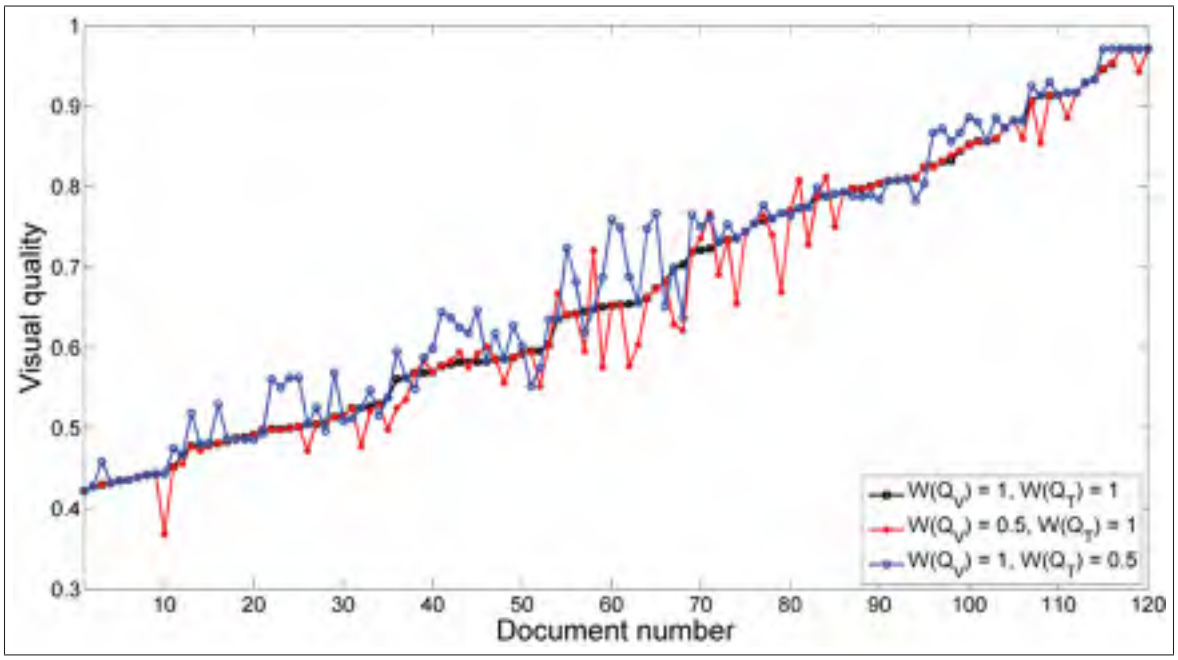

Figure D-7 JPEG optimal $\mathcal{Q}_V$ obtained by our framework (method 1) computed with a file size ratio of 1.15, $N_B(D) = 50$ kbps and $N_L(D) = 488$ ms.
The slides are sorted according to the $\mathcal{Q}_V$ obtained using $\mathcal{W}_V(D) = 1$, $\mathcal{W}_T(D) = 1$.

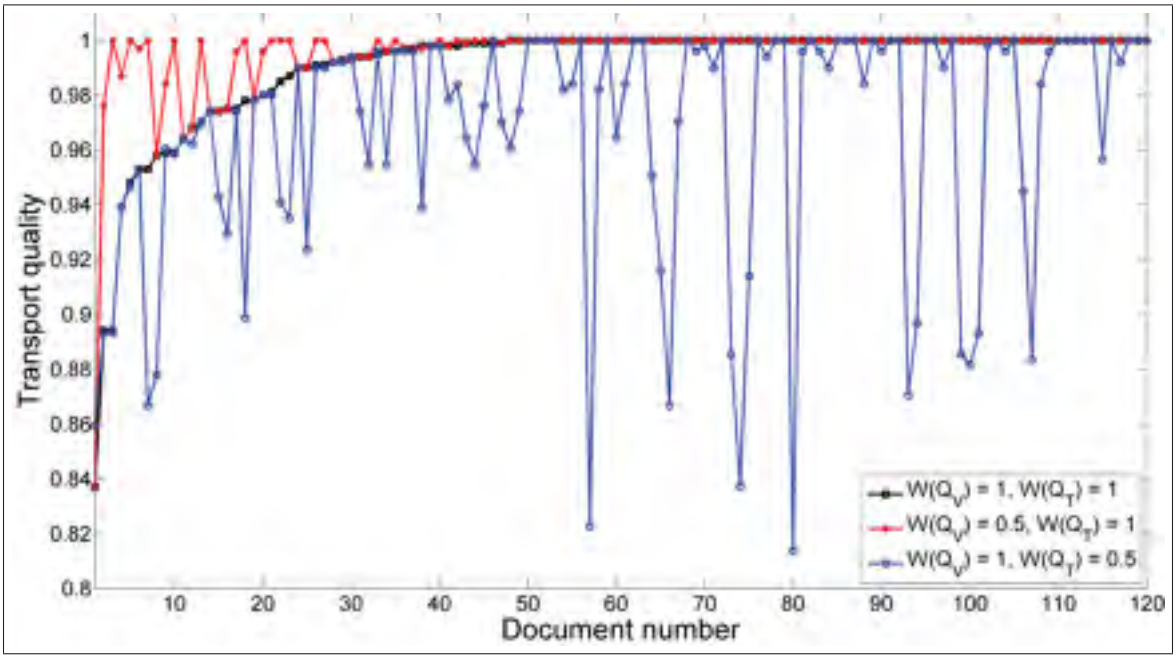

Figure D-8 JPEG optimal $\mathcal{Q}_T$ obtained by our framework (method 1) computed with a file size ratio of 1.15, $N_B(D) = 50$ kbps and $N_L(D) = 488$ ms.
The slides are sorted according to the $\mathcal{Q}_T$ obtained using $\mathcal{W}_V(D) = 1$, $\mathcal{W}_T(D) = 1$.

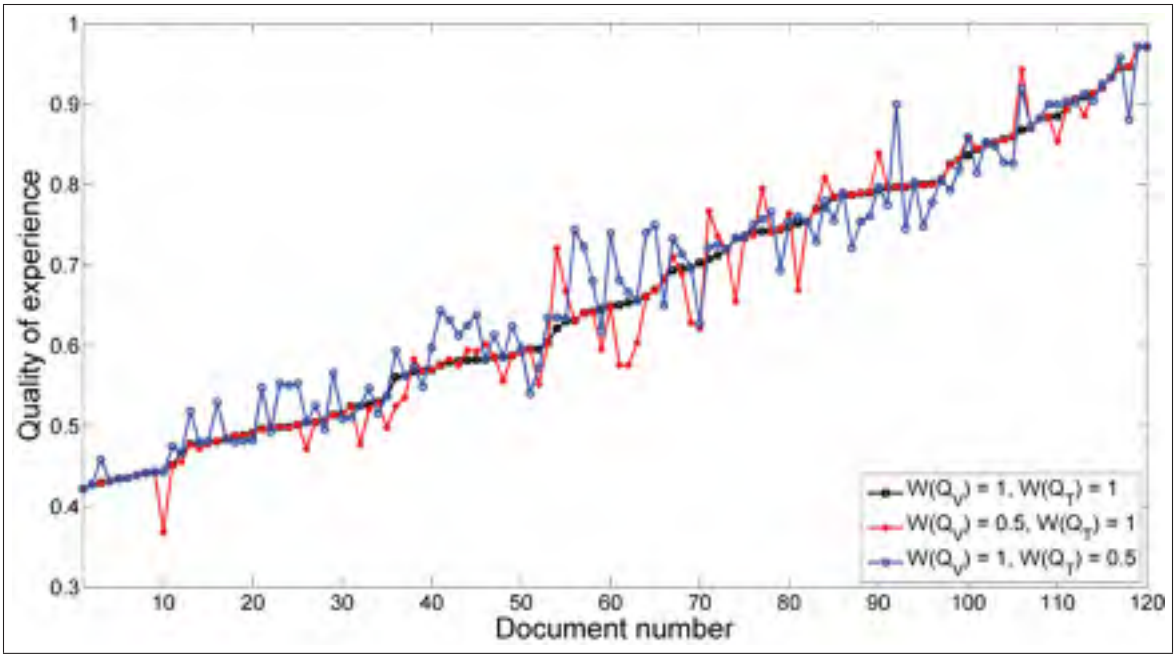

Figure D-9 JPEG optimal $\mathcal{Q}_E$ obtained by our framework (method 1) computed with a file size ratio of 1.15, $N_B(D) = 50$ kbps and $N_L(D) = 488$ ms.
The slides are sorted according to the $\mathcal{Q}_E$ obtained using $\mathcal{W}_V(D) = 1$, $\mathcal{W}_T(D) = 1$.

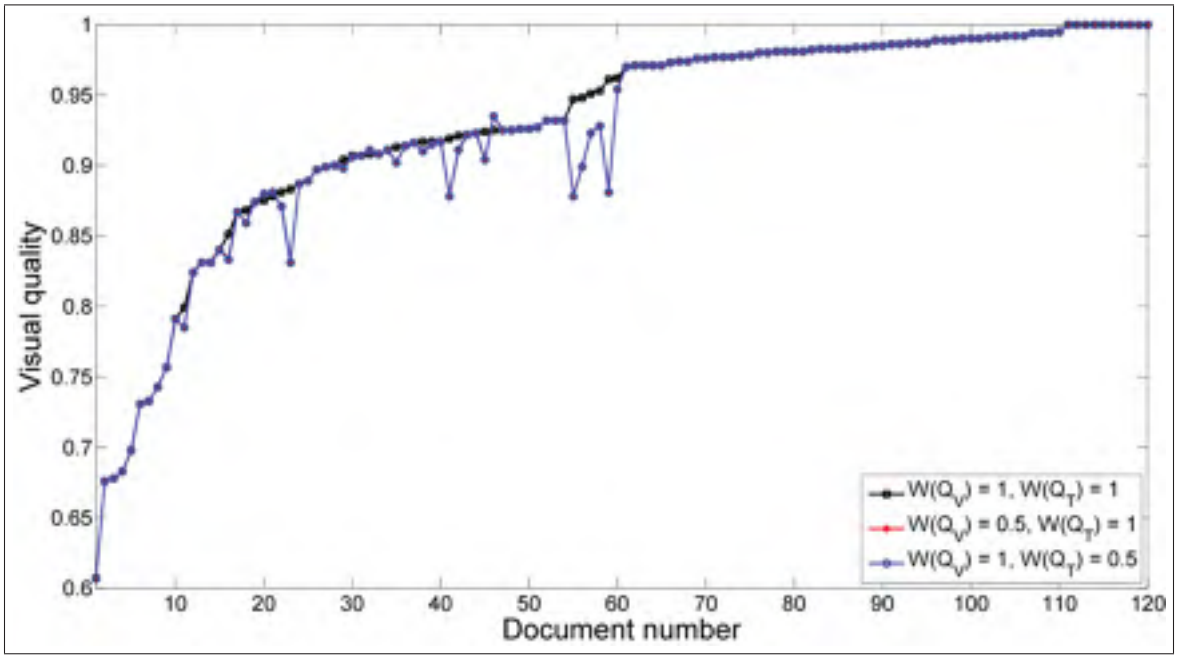

Figure D-10 XHTML optimal $\mathcal{Q}_V$ obtained by our framework (method 1) computed with a file size ratio of 1, $N_B(D) = 50$ kbps and $N_L(D) = 488$ ms. The slides are sorted according to the $\mathcal{Q}_V$ obtained using $\mathcal{W}_V(D) = 1$, $\mathcal{W}_T(D) = 1$.

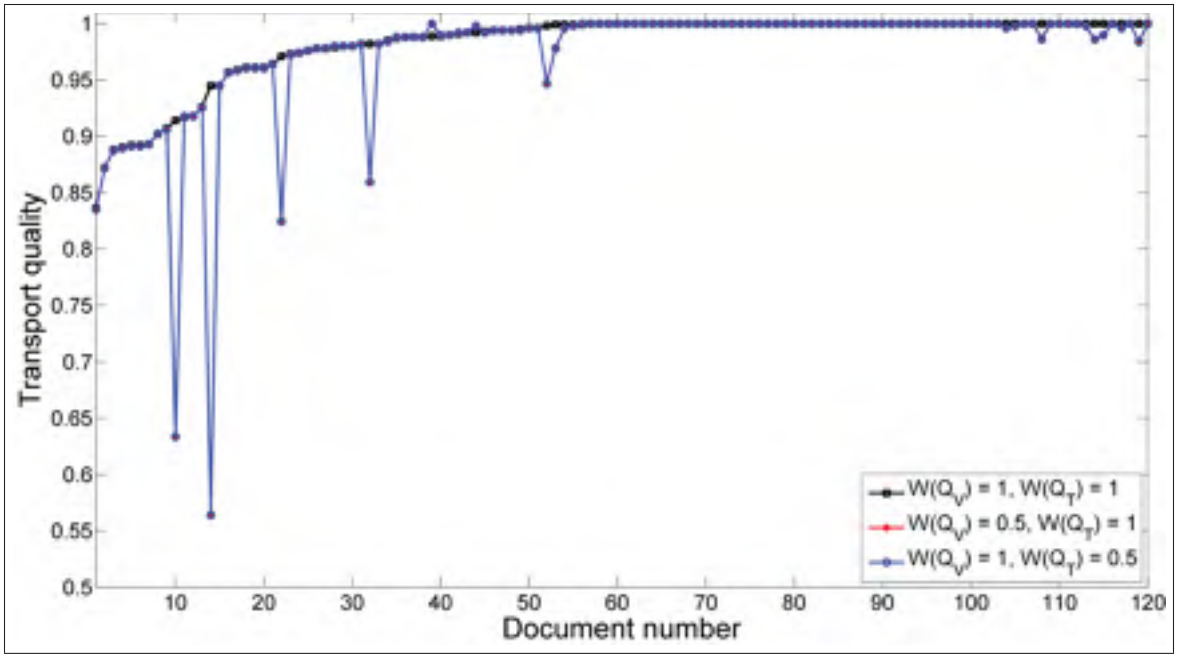

Figure D-11 XHTML optimal $\mathcal{Q}_T$ obtained by our framework (method 1) computed with a file size ratio of 1, $N_B(D) = 50$ kbps and $N_L(D) = 488$ ms. The slides are sorted according to the $\mathcal{Q}_T$ obtained using $\mathcal{W}_V(D) = 1$, $\mathcal{W}_T(D) = 1$.

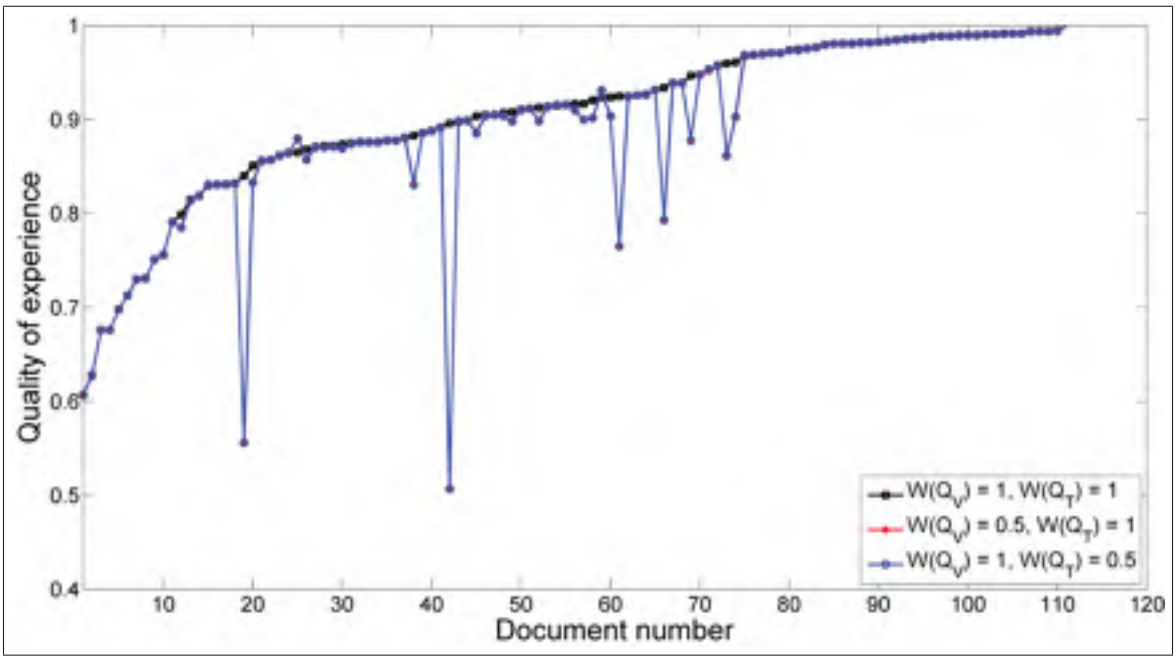

Figure D-12 XHTML optimal $\mathcal{Q}_E$ obtained by our framework (method 1) computed with a file size ratio of 1, $N_B(D) = 50$ kbps and $N_L(D) = 488$ ms.
The slides are sorted according to the $\mathcal{Q}_E$ obtained using $\mathcal{W}_V(D) = 1$, $\mathcal{W}_T(D) = 1$.

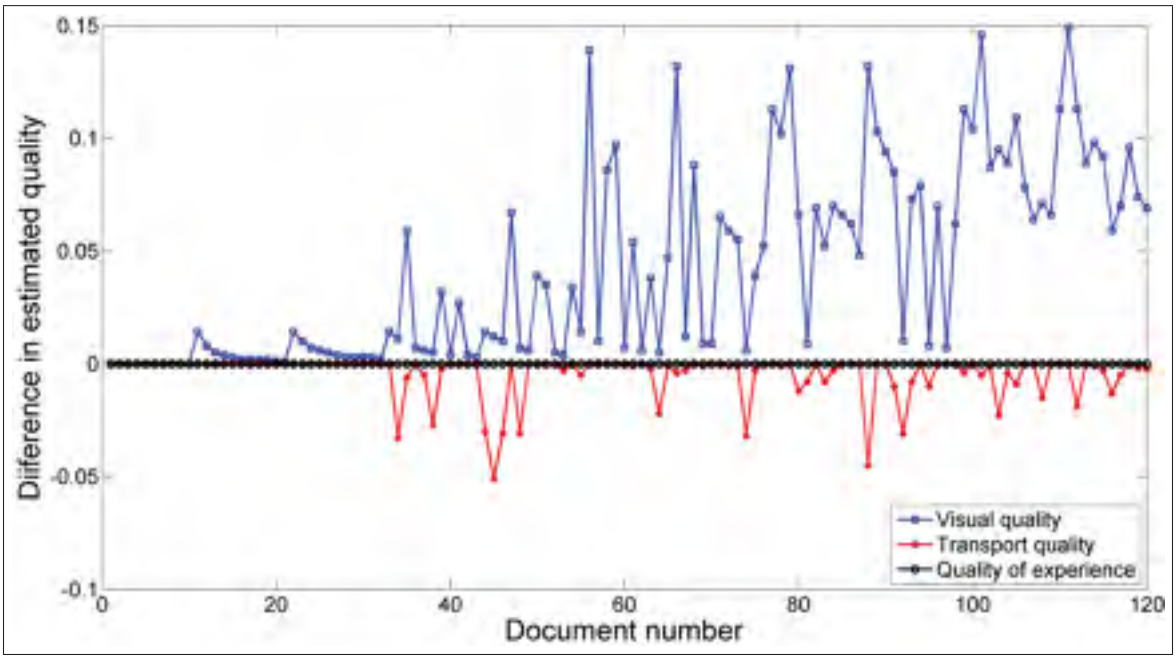

Figure D-13 Differences between the XHTML estimated $\mathcal{Q}_V$, $\mathcal{Q}_T$ and $\mathcal{Q}_E$ obtained with the two weight combinations $(\mathcal{W}_V(D) = 1, \mathcal{W}_T(D) = 0.5)$ and $(\mathcal{W}_V(D) = 0.5, \mathcal{W}_T(D) = 1)$, computed with a file size ratio of 1, $N_B(D) = 50$ kbps and $N_L(D) = 488$ ms.

BIBLIOGRAPHY

Adobe Systems Incorporated. 2012. "Adobe Flash Player 11". On line. <http://www.adobe.com/products/flashplayer.html>. Accessed on 19 September 2012. [24, 25]

Cavallaro, A., O. Steiger, and T. Ebrahimi. Jul 2003. "Semantic segmentation and description for video transcoding". In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*. p. 597-600. [20]

Chandra, S. and C. S. Ellis. 1999. "JPEG Compression Metric As A Quality-aware Image Transcoding". In *Proc. of the 2nd conference on USENIX Symposium on Internet Technologies and Systems*. p. 81-92. [4]

Chang, S. and D. G. Messerschmitt. 1995. "Manipulation and Compositing of MC-DCT Compressed Video". *IEEE Journal on Selected Areas in Communications*, vol. 13, n° 1, p. 1–11. [29]

Chao, L., 2011. *Open Source Mobile Learning: Mobile Linux Applications. Chapter 2: Adaptation Technologies in Mobile Learning* . IGI Global, 348 p. [31]

Chebbine, M. T., A. Obaid, S. Chebbine, and R. Johnston. Mar 2005. "Internet Content Adaptation System for Mobile and Heterogeneous Environments". In *Second IFIP International Conference on Wireless and Optical Communications Networks, WOCN 2005.* p. 346–350. IEEE. [21]

Chen, C., M. C. Chen, and Y. Sun. Aug. 2001. "PVA: a self-adaptive personal view agent system". In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*. (New York, New York, USA 2001), p. 257–262. ACM Press. [16]

Chen, J. and W. Ma. Jul. 2005. "Function-based object model for web page display in a mobile device". <http://www.freshpatents.com/-dt20090514ptan20090125800.php?type=description>. [15]

Chua, H. N., S. D. Scott, Y. W. Choi, and P. Blanchfield. Mar 2005. "Web-page Adaptation Framework for PC Mobile Device Collaboration". In *Advanced Information Networking and Applications, AINA 2005. 19th Int. Conference on*. p. 727-732. [15]

Chung-Sheng, L., R. Mohan, and J.R. Smith. May 1998. "Multimedia content description in the InfoPyramid". In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. p. 3789-3792. [32]

Converter, Free File. 2012. "Free File Converter". On line. <http://www.freefileconvert.com/>. Accessed on 22 September 2012. [23]

Coulombe, S. and S. Pigeon. Mar 2009. “Quality-aware Selection of Quality Factor and Scaling Parameters in JPEG Image Transcoding”. In *2009 IEEE Symp. on Computational Intelligence for Multimedia Signal and Vision Processing*. p. 68-74. [6, 29, 59, 61, 88, 89, 90, 100, 106, 115, 118, 139, 141]

Coulombe, S. and S. Pigeon. Mar 2010. “Low-complexity Transcoding of JPEG Images With Near-optimal Quality Using a Predictive Quality Factor and Scaling Parameters.”. *Image Processing, IEEE Transactions on*, vol. 19, n° 3, p. 712-721. [6, 29, 59, 61, 118, 139, 141]

Dieckmann, A., K. Dippold, and H. Dietrich. 2009. “Compensatory versus Noncompensatory Models for Predicting Consumer Preferences”. *Judgment and Decision Making*, vol. 4, n° 3, p. 200-213. [48, 83]

Ding, Y. 2007. “A Simple Picture of Web Evolution”. On line. <http://yihongs-research.blogspot.ca/2007/09/simple-picture-of-web-evolution.html>. Accessed on 10 October 2012. [10]

Ding, Y. and L. Xu. 2007a. “Evolution of the World Wide Web. A historical view and analogical study”. On line. <http://www.deg.byu.edu/ding/WebEvolution/evolution-prelude.html>. Accessed on 10 October 2012. [10]

Ding, Y. and L. Xu. 2007b. “Evolution of the World Wide Web. Part1: Past and Present of WWW”. On line. <http://www.deg.byu.edu/ding/WebEvolution/evolution-review.html>. Accessed on 10 October 2012. [10]

Ding, Y. and L. Xu. 2007c. “Evolution of the World Wide Web. Part 2: Web Evolution Theory and the Next Stage”. On line. <http://www.deg.byu.edu/ding/WebEvolution/evolution-dream.html>. Accessed on 10 October 2012. [10]

DiNucci, D. 1999. “Fragmented Future”. On line. <http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications>. Accessed on 16 October 2012. [10]

EffectMatrix Ltd. 2011. “E.M. PowerPoint Video Converter”. On line. <http://www.effectmatrix.com/PowerPoint-Video-Converter/Free-PowerPoint-Video-Converter.htm>. Accessed on 29 October 2012. [23]

Esenther, A. W. 2002. “Instant Co-Browsing: Lightweight Real-Time Collaborative Web Browsing”. In *In Proc. Of the 11th Int*. p. 7–11. Press. [15]

Flanagan, D., 2006. *avaScript: The Definitive Guide*. O’Reilly Media, Inc., 1032 p. [11]

Garrett, J. J. 2005. “Ajax: A New Approach to Web Applications”. On line. <http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications>. Accessed on 16 October 2012. [11]

Glover, T. and J. Davies. Jul 2005. “Integrating device independence and user profiles on the Web”. *BT Technology Journal*, vol. 23, n° 3, p. 239–248. [31]

Google Inc. 2011. “GoogleDocs Mobile”. On line. <http://www.google.ca/mobile/docs/index.html>. Accessed on 19 September 2012. [4]

Google Inc. 2012a. “GoogleDocs”. On line. <http://www.google.com/google-d-s/documents/>. Accessed on 19 September 2012. [1, 26]

Google Inc. 2012b. “Minimum requirements for Google Docs on a mobile browser”. On line. <http://support.google.com/drive/bin/answer.py?hl=en&answer=77421>. Accessed on 9 October 2012. [4]

Google Inc. 2012c. “Convert a synced file to Google Docs format”. On line. <http://support.google.com/drive/bin/answer.py?hl=en&answer=2407404>. Accessed on 9 October 2012. [4]

Han, R., P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas. 1998. “Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing”. *IEEE Personal Communications*, vol. 5, n° 6, p. 8-17. [3, 29, 54, 55, 58]

Hong, J., E. Suh, and S. Kim. 2009. “Context-aware Systems: A Literature Review and Classification”. *Expert Systems with Applications*, vol. 36, n° 4, p. 8509-8522. [3]

Hwang, C. L. and K. Yoon, 1981. *Multiple attribute decision making : Methods and applications*. Springer-Verlag. [48]

Hwang, Y., J. Kim, and E. Seo. 2003. “Structure-Aware Web Transcoding for Mobile Devices”. *IEEE Internet Computing*, vol. 7, n° 5, p. 14-21. [2, 3]

IBM. 2007. “IBM Lotus Sametime Unyte Meetings”. On line. <https://www.webdialogs.com/join/default.asp>. Accessed on 10 October 2012. [15]

ImageMagick. 1999. “ImageMagick Command Line Tools”. On line. <http://www.imagemagick.org/script/index.php>. Accessed on 19 September 2012. [100]

International Telecommunication Union. 2008. “H.264: Advanced video coding for generic audiovisual services”. On line. <http://www.itu.int/rec/T-REC-H.264>. Accessed on 19 September 2012. [21]

iSpring Solutions, Inc. 2012. “Free PowerPoint to Flash Converter”. On line. <http://www.ispringsolutions.com/free_powerpoint_to_flash_converter.html>. Accessed on 29 October 2012. [25]

Jacobs, S., 2006. *Beginning XML with DOM and Ajax: From Novice to Professional* . Apress, 456 p. [70, 72]

Jan, R., C. Lin, and M. Chern. 2006. “An Optimization Model for Web Content Adaptation”. *Computer Networks*, vol. 50, n° 7, p. 953–965. [3, 5, 52]

Joint Photographic Experts Group. 2007. “JPEG2000”. On line. <http://www.jpeg.org/jpeg2000/index.html>. Accessed on 19 September 2012. [21]

Kern, W. 2008. “Web 2.0 - End of Accessibility? Analysis of Most Common Problems with Web 2.0 Based Applications Regarding Web Accessibility”. *International Journal of Public Information Systems (IJPIS)*, vol. 2, p. 131–154. [10]

Kim, W., D. Jang, and T. Kim. Dec. 2007. “Improved Web Content Adaptation for Visual Aspect of Mobile Services”. In *2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*. p. 402–408. IEEE. [15]

Kitayama, F., S. Hitose, G. Kondoh, and K. Kuse. 1999. “Design of a Framework for Dynamic Content Adaptation to Web-enabled Terminals and Enterprise Applications”. *Proceedings Sixth Asia Pacific Software Engineering Conference ASPEC99 Cat NoPR00509*, p. 72-79. [3]

Kolich, M. S. 2009. “High Performance JavaScript Vector Graphics Library”. On line. <http://mark.koli.ch/2009/07/howto-include-binary-image-data-in-cascading-style-sheets-css.html>. Accessed on 04 April 2013. [160]

Kuipers, F., R. Kooij, D. De Vleeschauwer, and K. Brunnström. 2010. Techniques for Measuring Quality of Experience. *Wired/Wireless Internet Communications*, volume 6074 of *Lecture Notes in Computer Science*, p. 216-227. Springer-Verlag Berlin/Heidelberg. ISBN 978-3-642-13314-5. [40, 56, 59, 82, 83, 105, 139, 141]

Lee, L. and R. Anderson. 2009. “A Comparison of Compensatory and Non-Compensatory Decision Making Strategies in IT Project Portfolio Management”. On line. <http://aisel.aisnet.org/irwitpm2009/9>. Accessed on 10 October 2012. [48, 83]

Lee, S. O. K. and A. H. W. Chun. Apr. 2007. “Automatic Tag Recommendation For The Web 2.0 Blogosphere Using Collaborative Tagging and Hybrid ANN Semantic Structures”. In *Proceedings of the 6th Conference on WSEAS International Conference on Applied Computer Science*. (Stevens Point, Wisconsin, USA 2007), p. 88–93. World Scientific and Engineering Academy and Society (WSEAS). [9]

Li, D. and U. Chandra. May 2008. “Building Web-based Collaboration Services on Mobile Phones”. In *Int. Symp. on Collaborative Technologies and Systems*. p. 295-304. IEEE. [2, 4, 102]

Liang, A. Liang, C. Li, and S. Guo. Aug 2006. “Dynamic Mobile Content Adaptation Abstracting in Device Independent Web Engineering”. In *Communications, 2006. APCC ’06. Asia-Pacific Conference on*. p. 1-4. [57]

Louafi, H., S. Coulombe, and U. Chandra. 2012. “Quality Prediction-Based Dynamic Content Adaptation Framework Applied to Collaborative Mobile Presentations”. *IEEE Transactions on Mobile Computing*, vol. 99, p. 1-9. [79, 118, 140]

Louafi, H., S. Coulombe, and U. Chandra. Mar 2013a. “Efficient Near-Optimal Dynamic Content Adaptation Applied to JPEG Slides Presentations in Mobile Web Conferencing”. In *The 27th IEEE International Conference on Advanced Information Networking and Applications (AINA-2013)*. p. 724-731. [117, 118, 140]

Louafi, H., S. Coulombe, and U. Chandra. 2013b. "Robust QoE-aware Prediction-based Dynamic Content Adaptation Framework Applied to Slides Documents in Mobile Web Conferencing". *IOS Mobile Information Systems (MIS), (Submitted)*. [117, 140]

Lum, W. Y. and F.C.M. Lau. 2002. "On Balancing Between Transcoding Overhead and Spatial Consumption in Content Adaptation". *Proceedings of the 8th annual international conference on Mobile computing and networking MobiCom 02*, p. 239. [4, 40]

Lum, W. Y. and F.C.M. Lau. Dec 2003. "User-Centric Content Negotiation for Effective Adaptation Service in Mobile Computing". *IEEE Transactions on Software Engineering*, vol. 29, n° 12, p. 1100-1111. [2, 3, 14, 19, 40, 42, 43, 46, 47, 50, 52, 54, 86, 97, 103]

Lum, W. Y. and F.C.M. Lau. 2005. "User-Centric Adaptation of Structured Web Documents for Small Devices". In *19th Int. Conf. on Advanced Information Networking and Applications (AINA'05)*. p. 507-512. IEEE. [2, 5, 14, 15]

Merhav, N. and V. Bhaskaran. 1996. "A Transform Domain Approach to Spatial Domain Image Scaling". In *IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*. p. 2403–2406. IEEE. [29]

Mérida, D., R. Fabregat, X. Prat, D. Huerva, and J. Velez. 2008. "A Dynamic Content Generator for Adaptation in Hypermedia Systems". In *Proceedings of the 5th international conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. (Berlin, Heidelberg 2008), p. 320-323. Springer-Verlag. [57, 58]

Meyer, E. A. and K. S. Meyer. 2012. "S5: A Simple Standards-Based Slide Show System". On line. <http://meyerweb.com/eric/tools/s5/>. Accessed on 19 September 2012. [21]

Microsoft. 2003. "Publish a presentation to the Web". On line. <http://office.microsoft.com/en-us/powerpoint-help/publish-a-presentation-to-the-web-HP005266845.aspx>. Accessed on 19 September 2012. [22]

Microsoft. 2004. "Microsoft NetMeeting". On line. <http://www.microsoft.com/en-us/download/details.aspx?id=23745>. Accessed on 10 October 2012. [15]

Mohan, R. and J.R. Smith. Mar 1999. "Adapting Multimedia Internet Content for Universal Access". *IEEE Transactions on Multimedia*, vol. 1, n° 1, p. 104-114. [3, 5, 14, 32, 33, 34, 103]

Nah, F. F. Jan 2004. "A study on Tolerable Waiting Time: How Long Are Web Users Willing to Wait?". *Behaviour & Information Technology*, vol. 23, n° 3, p. 153-163. [86, 107]

Ningning, H. and P. Steenkiste. Aug 2003. "Evaluation and characterization of available bandwidth probing techniques". *IEEE Journal on Selected Areas in Communications*, vol. 21, n° 6, p. 879-894. [92]

Noble, B. D., M. Price, and M. Satyanarayanan. 1995. "A Programming Interface for Application-Aware Adaptation in Mobile Computing". *2nd USENIX Symposium on Mobile and Location Independent Computing*, vol. 8, n° 4, p. 57-66. [3]

O'Donoghue, R. 2006. "Getting started with Mobile AJAX". On line. <http://mobiforge.com/developing/story/getting-started-with-mobile-ajax>. Accessed on 10 October 2012. [11, 12]

Open Mobile Alliance. 2006. "User Agent Profile - Approved Version 2.0". On line. <http://www.w3.org/Protocols/rfc2616/rfc2616.html>. Accessed on 19 October 2012. [31]

OpenOffice. 2008. "Supported Fonts". On line. <http://wiki.openoffice.org/wiki/Font-FAQ#Supported_Fonts>. Accessed on 2 October 2012. [75]

OpenOffice. 2010. "The OpenOffice.org API Project". On line. <http://api.openoffice.org>. Accessed on 19 September 2012. [68, 99]

O'Reilly, T. 2005. "What Is Web 2.0". On line. <http://oreilly.com/web2/archive/what-is-web-20.html>. Accessed on 10 October 2012. [10]

Pigeon, S. and S. Coulombe. Jun 2008. "Computationally Efficient Algorithms for Predicting the File Size of JPEG Images Subject to Changes of Quality Factor and Scaling". In *2008 24th Biennial Symposium on Communications*. p. 378-382. IEEE. [6, 29, 59, 61, 91, 92, 93, 94, 95, 100, 115, 118, 127, 139, 141]

Pigeon, S. and S. Coulombe. Dec 2011. "Efficient Clustering-based Algorithm for Predicting File Size and Structural Similarity of Transcoded JPEG Images". *Multimedia, Int. Symp. on*, p. 137-142. [6, 29, 59, 61, 89, 93]

Potter, S. and J. Nieh. May 2005. "WebPod: Persistent Web Browsing Sessions With Pocketable Storage Devices". In *Proceedings of the 14th International Conference on World Wide Web - WWW '05*. (New York, New York, USA 2005), p. 603–612. ACM Press. [14]

Raggett, D. 2010. "HTML Slidy: Slide Shows in HTML and XHTML". On line. <http://www.w3.org/Talks/Tools/Slidy2/Overview.html>. Accessed on 19 September 2012. [21]

Richards, A., M. Antoniades, V. Witana, and G. Rogers. Nov 1998. "Mapping User Level QoS from a Single Parameter". In *Proceedings of the International Conference on Multimedia Networks and Services (MMNS '98)*. [44, 45]

Ryan, G. and M. Valverde. Apr 2006. "Waiting in line for Online Services: A Qualitative Study of The User's Perspective". *Information Systems Journal*, vol. 16, n° 2, p. 181-211. [86, 107]

Schubert, Manfred. 2010. "Word Browser Plugin". On line. <http://schubert-it.com/free/>. Accessed on 19 September 2012. [26]

Scientiamobile. 2011. “WURFL”. On line. <http://www.scientiamobile.com/>. Accessed on 19 September 2012. [3]

Shahabuddin, S., R. Iqbal, A. Nazari, and S. Shirmohammadi. Oct. 2009. “Compressed domain spatial adaptation for H.264 video”. In *Proceedings of the seventeen ACM international conference on Multimedia - MM ’09*. p. 797. ACM Press. [29]

Sheikh, H. R., Z. Wang, L. Cormack, and A. C. Bovik. 2012. “LIVE Image Quality Assessment Database Release 2”. <http://live.ece.utexas.edu/research/quality/subjective.htm>. [105]

Smith, B. C. and L. A. Rowe. Sep. 1993. “Algorithms for Manipulating Compressed Images”. *IEEE Computer Graphics and Applications*, vol. 13, n° 5, p. 34–42. [29]

Software Mechanics Pty Ltd. 2011. “chinook, SVG export from powerpoint”. On line. <http://www.svgmaker.com/product_standard.html>. Accessed on 29 October 2012. [24]

Soo-Chang, P. and W. Yu-Ying. Nov 2011. “Census-based vision for auditory depth images and speech navigation of visually impaired users”. *Consumer Electronics, IEEE Transactions on*, vol. 57, n° 4, p. 1883-1890. [20]

SourceForge. 2009. “chinook, SVG export from powerpoint”. On line. <http://chinook.sourceforge.net/>. Accessed on 29 October 2012. [24]

Spivack, N. 2006. “The Third-Generation Web is Coming”. On line. <http://www.kurzweilai.net/the-third-generation-web-is-coming>. Accessed on 10 October 2012. [12]

Sudhir, D. and W. Tao, 2004. *Content Networking in the Mobile Internet, Chapter 7: Content Adaptation for the Mobile Internet*. John Wiley and Sons, 547 p. [2, 19, 20, 28, 30, 32, 33, 34, 37]

Svoboda, P., F. Ricciato, W. Keim, and M. Rupp. Jun 2007. “Measured WEB Performance in GPRS, EDGE, UMTS and HSDPA with and without Caching”. In *IEEE Int. Symp. on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. p. 1-6. [92, 106, 170]

The Free Dictionary. 2012a. “Web 2.0”. On line. <http://encyclopedia.thefreedictionary.com/Web+2.0>. Accessed on 10 October 2012. [10]

The Free Dictionary. 2012b. “Web 2.0”. On line. <http://encyclopedia2.thefreedictionary.com/web+2.0>. Accessed on 10 October 2012. [11]

The Free Dictionary. 2012c. “Web 3.0”. On line. <http://encyclopedia.thefreedictionary.com/Web+3.0>. Accessed on 10 October 2012. [12]

The MathWorks. 2012. “Z-shaped built-in membership function”. On line. <http://www.mathworks.com/help/toolbox/fuzzy/zmf.html>. Accessed on 19 September 2012. [86]

The Moving Picture Experts Group (MPEG). 1988. “MPEG”. On line. <http://mpeg.chiariglione.org/>. Accessed on 19 September 2012. [21]

The USC-SIPI. 2012. “The USC-SIPI Image Database”. On line. <http://sipi.usc.edu/database/database.cgi?volume=misc\&image=11>. Accessed on 19 September 2012. [99]

Tsai, D. W. and Y. Zhang. Sep. 2009. “A Frequency Sensitivity-Based Quality Prediction Model for JPEG Images”. In *Fifth International Conference on Image and Graphics*. p. 28–32. IEEE. [29]

VeryDOC. 2012. “VeryDOC DOC to Any Converter”. On line. <http://www.verydoc.com/doc-to-any.html>. Accessed on 29 October 2012. [25]

VeryPDF Knowledge Base. 2012. “How to convert MS Office PowerPoint documents of PPT to SVG?”. On line. <http://www.verypdf.com/wordpress/201201/how-to-convert-ms-office-powerpoint-documents-of-ppt-to-svg-19695.html>. Accessed on 29 October 2012. [25]

W3C. 1987. “Graphics Interchange Format”. On line. <http://www.w3.org/Graphics/GIF/spec-gif87.txt>. Accessed on 28 October 2012. [21]

W3C. 1999. “Hypertext Transfer Protocol”. On line. <http://www.w3.org/Protocols/rfc2616/rfc2616.html>. Accessed on 19 October 2012. [30]

W3C. 2002. “XHTML 1.0 The Extensible HyperText Markup Language”. On line. <http://www.w3.org/TR/xhtml1/>. Accessed on 16 October 2012. [11]

W3C. 2012. “HTML and CSS”. On line. <http://www.w3.org/standards/webdesign/htmlcss#whatcss>. Accessed on 16 October 2012. [11]

W3C. 2012a. “Scalable Vector Graphics (SVG)”. On line. <http://www.w3.org/Graphics/SVG/>. Accessed on 19 September 2012. [24, 25]

W3C. 2012b. “Amaya”. On line. <http://www.w3.org/Amaya/Overview.html>. Accessed on 19 September 2012. [22]

Wang, H., A. Divakaran, A. Vetro, S. Chang, and H. Sun. Jun 2003. “Survey of Compressed-Domain Features Used in Audio-Visual Indexing and Analysis”. *Journal of Visual Communication and Image Representation*, vol. 14, n° 2, p. 150–183. [29]

Wang, Z., A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. “Image Quality Assessment: From Error Visibility to Structural Similarity.”. *IEEE Transactions on Image Processing*, vol. 13, n° 4, p. 600-612. [59, 85]

Wikipedia. 2013. “Data URI scheme”. On line. <http://en.wikipedia.org/wiki/Data\URI\scheme>. Accessed on 04 April 2013. [161]

Xiao, Y., Y. Tao, and W. Li. Dec. 2008. "A Dynamic Web Page Adaptation for Mobile Device Based on Web2.0". In *Advanced Software Engineering and Its Applications*. p. 119–122. IEEE. [56]

Xilisoft Corp. 2012. "Xilisoft PowerPoint to Video Converter". On line. <http://www.xilisoft.com/powerpoint-to-video-converter.html>. Accessed on 29 October 2012. [23]

Yoon, K. P. and C. Hwang, 1995. *Multiple attribute decision making : An Introduction*. Sage University Papers. [97]

Zamzar. 2012. "ZAMZAR, Free File Conversion". On line. <http://www.zamzar.com/>. Accessed on 22 September 2012. [23]

Zhang, Y., S. Zhang, and S. Han. 2006a. A New Methodology of QoS Evaluation and Service Selection for Ubiquitous Computing. *Wireless Algorithms, Systems, and Applications*, volume 4138 of *LNCS*, p. 69-80. Springer Berlin / Heidelberg. [2, 3, 40, 41, 42, 43, 49, 50, 51, 52, 54, 86, 97, 103]

Zhang, Y., S. Zhang, and S. Han. Aug 2006b. "Context-Based Qos Model and its Application in Ubiquitous Computing". In *International Conference on Machine Learning and Cybernetics*. p. 1517–1521. IEEE. [40, 41]

Zhang, Y., S. Zhang, and S. Han. 2006c. "Context-Aware Service Selection Engine for Ubiquitous Computing Application". In *6th World Congress on Intelligent Control and Automation*. p. 4269–4273. IEEE. [40, 41]

Zhang, Y., S. Zhang, and H. Tong. 2006d. Adaptive Service Delivery for Mobile Users in Ubiquitous Computing Environments. Ma, J., Hai Jin, Laurence Yang, and Jeffrey Tsai, editors, *Ubiquitous Intelligence and Computing*, volume 4159 of *Lecture Notes in Computer Science*, p. 209–218. Springer Berlin / Heidelberg. [40, 41]

Zoho Corp. 2012. "Zoho Show". On line. <https://show.zoho.com/login.do>. Accessed on 19 September 2012. [1, 26]

Zorn, W. 2008. "High Performance JavaScript Vector Graphics Library". On line. <http://www.walterzorn.de/en/jsgraphics/jsgraphics_e.htm>. Accessed on 04 April 2013. [162]