AN ICAP-BASED CONTENT REPURPOSING SYSTEM FOR UBIQUITOUS ACCESS TO MULTIMEDIA CONTENT

BY

TAM WING-LAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF PHILOSOPHY
IN
INFORMATION ENGINEERING

JULY 2003

Acknowledgments

I would like to express my deep and sincere gratitude to my supervisor, Professor Peter Tak-shing Yum, who has commented extensively and intensively on successive drafts of this thesis. His understanding and guidance have provided a good basis for the present thesis.

I am deeply grateful to Kington Chan for his detailed and constructive comments, and for his important and continuous support and encouragement throughout this work.

During this work I have collaborated with many colleagues and undergraduate students for whom I have great regard. I wish to extend my warmest thanks to all those who have helped me with my work in the Communication Technology Laboratory: Prof Yang Yang, Cheng Li, Cunqing Hua, Forest Shen, Li Zhang, Xinyan Zhang, Lin Zhang, and Ling Li.

I owe my loving thanks to my family and friends, for all of their encouragement and caring throughout the years.

Finally, I wish to thank the funding of my research assistantship from Area of Excellence in Information Technology (AoE-IT) and Hong Kong RGC Grant 4371/99E.

Abstract

Until recently, the services of delivering variety of multimedia content over Internet are fast evolving to a symbiosis of pervasive computing devices and ubiquitous network infrastructure. However, the wide variety of device capabilities, network bandwidth, and user preferences will require a new approach for content repurposing. The mismatch between rich multimedia content and diversity of usage scenarios creates new research opportunities and challenges. In this thesis, we propose a real-time *ICAP-based Content Repurposing System*. This system includes the following components: (a) a *generic service-enabling platform* for content repurposing applications to build on; (b) a *rule engine* that collects the information of device profile, network bandwidth and user preferences and generates the transcoding parameters based on the decision function; and (c) a set of *ICAP-enabled applications* that repurpose the multimedia content based on the transcoding parameters.

The *generic service-enabling platform* provides a scalable and flexible platform for adaptation services to run on. The *rule engine* decides on adaptation policies. It controls the extent of compression and types of transcoding that are performed by *ICAP-enabled applications*. We provide a delay analysis on the system for how the *rule engine* makes decision.

滲透式網絡中介內容修改系統

譚詠琳

香港中文大學訊息工程學系

哲學碩士論文摘要

滲透式網絡中介軟體及平台技術的主要目標是在異質之有線／無線網路上建構能支援多個移動使用者的普及計算環境。支援不同的使用者偏好、計算機設備能力及網路頻寬需要中介平台技術以重新格式化內容。在這篇論文中，我們通過使用「互聯網內容修改協定」集成實時中介內容修改系統。這個內容修改系統由三個組件組成：

一、「泛型服務平台」，負責集成不同的增值服務伺服器；
二、「規則引擎」，收集及分析使用者偏好、計算機設備能力及網路頻寬用以生成內容修改參數；及
三、一系列的「內容修改應用程式」，利用內容修改參數以重新格式化多媒體數據。

「泛型服務平台」確保集成「內容修改應用程式」後系統的性能、可靠性和可擴展性。「規則引擎」負責判定傳輸作業符合哪些規則的規定，用以制定決策內容修改的技術及壓縮程度。我們對系統延遲進行分析以得知「規則引擎」如何進行決策制定。

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

Number *Page*

Chapter 1

Introduction

Recent mobile communications systems and wireless computing devices (e.g. notebook PCs, hand-held computers, PDAs, smart phones, TV browsers and wearable computers) enable us to access Web content on the Internet. However, for multimedia applications, such as a digital library, most media-rich content was designed and organized for desktop computers with high-speed broadband network. Web content usually contains media-rich data such as image, audio and video, which are not suitable for mobile devices with limited network bandwidth and device capabilities. Compared to wireline access, the available bandwidth is narrow. The communication channel is less reliable, for example the error rate changes dynamically due to the effect of fading. On small devices such as PDAs and smart phones, device capabilities, such as computational and rendering power, storage capacity and display capabilities, are restricted for portability. Therefore, for better media presentation on mobile devices, the quality of multimedia content should be adjusted according to network characteristics and device capabilities by intermediary transcoding services. Besides, in order to add value to the information delivered to users, multimedia presentation should be personalized according to the preferences of individual user. Therefore, it is a challenging research study to eliminate the mismatch between rich multimedia content and the diversity of usage scenarios.

In order to provide appropriate multimedia presentation to different devices, many issues need to be addressed.

(A) Contextualization

To design a good adaptation service, we must understand the client environment sufficiently. We can gain an understanding through a contextualization framework that facilitates the expression and capturing of context information. Context includes any information that can characterize the client's situation. Client device's characteristics and capabilities are part of the context of a client environment where Web content rendering occurs.

(B) Context Analysis and Adaptation Decision

To generate the best content version for presentation in a content adaptation system, a decision engine is needed to analyze the context information, and selects different adaptation strategy. The decision engine aims to increase users' satisfaction in the rendered content in a constrained mobile computing environment. It automatically negotiates for the appropriate content adaptation decisions that the transcoder will use to generate the content version.

(C) Content Adaptation Techniques

In order to increase accessibility of multimedia information, many media processing techniques can be used to enable media adaptation. According to the adaptation target context, multimedia content is adapted to device capabilities, network characteristics and user preferences.

(D) System Integration

A well-designed adaptation framework is needed to integrate all of the above technologies and interface to the network topology. It should provide high scalability, reliability and performance for content adaptation.

1.1 Research Background

A considerable amount of work has been done in the area of enabling ubiquitous access to Web content from pervasive computing devices. Lei and Georganas [3]

presented some related issues for building a general context-based media adaptation framework. These issues include context description and exchange schemes, adaptation model for managing and manipulating multimedia content, adaptation techniques and architectural issues. They provided a classification of media adaptation techniques that can be used to guide research in this area. In this thesis, the following issues will be discussed.

(A) Context Descriptions and Exchange

In order to produce the optimized content for a specific user agent, it is important to gain an understanding of context information. The HTTP/1.1 content negotiation capability [4] and the W3C Composite Capability/Preference Profile (CC/PP) [5] are standards for describing device capability and user preferences as part of an HTTP request. CC/PP can specify client capability and user preferences as a collection of URIs (Uniform Resource Identifier) and RDF (Resource Description Framework) text. Both the HTTP/1.1 content negotiation capability and CC/PP provide the declarative semantics needed to determine the adaptation settings for transcoding.

(B) Decision Engine

Lum and Lau [6] offer good insight in designing a context-aware decision engine for content adaptation of text and images. It is aware of different types of context information, such as user preferences, device's rendering capability and the network characteristics. It tries to arrive at the best trade-off for content adaptation while minimizing content degradation due to lossy transcoding. The engine relies on a user's indication of preferences according to his or her perception in different quality domains. On the basis of these preferences, the engine can devise a method to express quantitatively any given content's quality along various quality axes and algorithms to negotiate for an optimal content version with some guarantee on the returned objects' QoS.

(C) Adaptation Techniques for Multimedia Content

In order to increase accessibility of multimedia content, various media transcoding techniques are applied to adapt multimedia content according to device capabilities, network characteristics and user preferences. If a client device cannot support a type of format, format conversion can be carried out between media types, for example, speech-to-text (speech recognition), video-to-image (video mosaicing) or image-to-text (embedded caption recognition); and within media types, for example transformation of HTML to WML, JPG/GIF to WBMP and WAV to MP3. Many research works have been done on applying image-processing techniques to adapt the size (in pixel) of embedded images on a Web page in order to fit the tiny display size of mobile devices. Also, to cope with the variety of effective bandwidth and network latency, some image transcoders compress the data size (in bytes) of an image through color depth reduction, image scaling and image cropping. For audio content, compression can be done by reducing the bit-rate and the number of channels of the media file or performing format conversion (e.g. WAV to MP3). Compression of video, which results in reduction in size (in bytes), quality and data rate, can be done by video frame dropping, color conversion, DCT coefficient scaling and reduction in audio quality. Details can be found in [23].

(D) Transcoding Framework

One important design issue is to decide where the adaptation resides. Three locations are available: client device, proxy, and original server. Client-based adaptation is not suitable for mobile devices because the devices are connected through low network bandwidth, which results in slow access to rich media content; and they are restricted in their computational power, which makes content adaptation slow or even impossible. In server-based adaptation, the media server is responsible for analyzing the context profile and discovering how much bandwidth is available. It then selects the appropriate adaptation technique. Using the server-based adaptation has the advantage that the adaptation may have the best effect. It is because the content authors can preview the adapted result under different viewer preference and conditions. Placing the adaptation on the server also has drawbacks. It complicates the implementation of

an origin server and the algorithms for generating appropriate presentation to a request, which results in additional computational load and resource consumption on the server. Most content adaptation systems for mobile devices are HTTP proxy-based. In proxy-based adaptation, the transcoding proxy is placed at the edge of the network. It intercepts client device's requests for Web content, fetches the requested content, adapts it, and sends the adapted version to the client. Adaptation at the proxy means that there is no need to change existing clients and servers. It reduces the computation load of the origin server. By caching at the proxy, it improves performance and end-to-end latency perceived by client. To improve performance and alleviate network bottleneck on the transcoding proxy, the proxy distributes heavy loading transformation processing to application servers. To ensure scalability and flexibility of the transcoding framework, application servers should be built on top of standard and open remote call-out protocols and flexible APIs.

There is a wide variety of existing or proposed protocols for implementing value-added services at Web intermediary, e.g. ICAP (Internet Content Adaptation Protocol) [2] and SOAP (Simple Object Access Protocol) [7]. In the content repurposing system, we utilize ICAP as the communication platform between Web proxy and transcoding servers for content adaptation. ICAP is an open protocol which enables communication between edge content devices (Web caches and Internet content origin servers) and application servers that modify or process web contents before delivering them to the Internet access devices. It empowers edge devices like a cache to distribute application services without overloading the cache or slowing the response. It is HTTP-based and uses existing infrastructure and communication methods on the Internet. It can improve performance, scalability and flexibility of transcoding framework [1].

1.2 Contribution of the Thesis

In the thesis, we propose a real-time *ICAP-based Content Repurposing System*. It provides proxy-based adaptation services. It utilizes ICAP (Internet Content Adaptation Protocol) [2] as a common communication platform for improving performance and reducing latency.

This system includes the following components: (a) a *generic service-enabling platform* for content repurposing applications to build on; (b) a *rule engine* that collects the information of device profile, network bandwidth and user preferences and generates the transcoding parameters based on the decision function; and (c) a set of *ICAP-enabled applications* that repurpose the multimedia content based on the transcoding parameters.

The system provides content adaptation for various types of multimedia content, including text, image, audio and video. It supports two modes of adaptation, *store-and-forward* transcoding and *streamed* transcoding.

The content repurposing system aims to provide adaptation on two dimensions, *content reformatting* and *content compression* for different application cases. For *content reformatting*, the transcoder reformats and transforms multimedia content to a format that can be supported by the device. For *content compression*, the transcoder reduces the end-to-end latency perceived by the clients by reducing the quality of content.

The end-to-end latency is regarded as the metric of evaluating the system performance. We provide delay analysis of different transcoding policies for *rule engine* to control the extent of compression and types of transcoding that are performed by *ICAP-enabled applications*.

1.3 Organization of the Thesis

The rest of this thesis proceeds as follows: Chapter 2 presents the architecture of the *ICAP-based Content Repurposing System*. Chapter 3 discusses the transcoding techniques. Chapter 4 presents how the system makes decision on adaptation policies based on experimental results. Chapter 5 concludes our work.

Chapter 2

Content Repurposing System Architecture

The *ICAP-based Content Repurposing System* provides a flexible middleware platform for distributing transcoding services for ubiquitous access to Internet content. It is a transcoding proxy server system which consists of a proxy attached to a farm of transcoding servers. Figure 2-1 shows the overview of the infrastructure of the content repurposing system. The system consists of an *ICAP-enabled Proxy* and a set of *ICAP-enabled Application Servers*.

2.1 Introduction to ICAP-based Content Repurposing System

The content repurposing system consists of three main components. Each of them performs different tasks and features different functionalities.

(A) Generic Service-enabling Platform

The *generic service-enabling platform* is based on ICAP, with well-defined and flexible API's (Application Programming Interfaces) for value-added edge services to build on. The platform aims to provide the following features:

- Scalability – When the offered load to the transcoding server increases, a proportional increase in hardware can maintain the same per-user level of service.
- Reusability – New services can be developed from common software modules.

Details are discussed in Section 2.2.

(B) Rule Engine

The *rule engine* collects the information of user preferences, device profile, and network condition and generates *transcoding parameters* based on the decision function. It provides a mechanism for processing and parsing of transcoding rules. They are the conditions which tell the proxy when and how the services should be executed. Details are discussed in Section 2.3.

(C) ICAP-enabled Applications

ICAP-enabled applications repurpose the hypermedia content (including text, image, audio and video) in real time based on the *transcoding parameters*. These applications involve techniques in transcoding multimedia content, including text, image, audio and video. Content adaptation provides the following functions:

- Different transcoding result will be generated for the same resource upon request. Content is adapted for optimizing network condition, device capabilities and preferences of individual user.
- Content is displayed in a manner for the best visualization according to device's display size and color depth.
- Content is distilled to an optimized level of quality for reducing the computing time for display at device. The smaller and more efficient data representation reduces transmission time to the client.

Details are discussed in Section 2.4.

2.2 Generic Service-enabling Platform

Figure 2-2 shows the architecture and the message flow between the client device, proxy, application servers and original server. The *Remote Execution Module* and the *API* at the *ICAP-enabled Application Server* form a *generic service-enabling platform* for transcoding services to build on.

The *generic service-enabling platform* enables communication between Web proxy and remote application servers. It is based on ICAP, with well-defined and flexible APIs for plug-in of application servers. ICAP is a client-server model [1]. It allows ICAP clients, the proxy, to pass HTTP messages (HTTP requests or responses) to ICAP servers, the application servers, for content adaptation. It defines four adaptation techniques [1, 8] which is able to facilitate all the scenarios in content adaptation. The techniques include *Request Modification*, *Request Satisfaction*, *Response Modification* and *Result Modification*. In this thesis, we only utilize two modification modes for adaptation on multimedia content: *Request Satisfaction* and *Response Modification*.

For *Request Satisfaction*, the message flow between client device, proxy, application server, and original server is shown in Figure 2-3. The client device sends a request to the proxy. The proxy redirects the request to an ICAP server. The ICAP server modifies the message and sends it straight to the origin server for fulfillment. The origin server sends the response content to the client device through the ICAP server and proxy. This mode will be used for the adaptation of streaming media, e.g. audio and video, which will be discussed in the later section.

Figure 2-4 illustrated the message flow in *Response Modification*. The response located by the client device's HTTP request is fetched from the original server. This original response is modified by the ICAP server and the adapted response is returned to the end-user through the proxy. This mode will be used for the adaptation of Web pages and image files.

Upon client's connection, the *Rule Engine* determines the appropriate application server and the ICAP modification mode. The *Remote Execution Module* invokes *ICAP-enabled Proxy Server* through ICAP for content adaptation. The *Remote Execution Module* at ICAP client together with API at server-side form a *generic service-enabling platform* for flexibly insertion of transcoding service for different adaptation purposes. The API provides encapsulation of the protocol semantics of ICAP. Developers of transcoding services can make use of the set of publicly available API method calls for integrating their transcoding technology with ICAP. The API

provides a flexible and scalable solution for plug-in of new transcoding modules in the content repurposing system.

2.3 Rule Engine

The *Rule Engine* performs first-stage negotiation between client request and context information to decide the appropriate *ICAP-enabled Application Server* for adaptation. The negotiation depends on pre-defined decision function. As shown in Figure 2-5, to plug-in an application server into the content repurposing system requires provision of transcoding rule. The transcoding rule specifies the conditions under which the adaptation needs to be performed by a particular application server. The *Rule Update Module* collects the set of rules provided by the farm of application servers and updates the decision function. The decision function negotiates between client request and context information to generate *transcoding parameters*. The *transcoding parameters* specify the ICAP modification mode, the location where transcoding service takes place and other useful information for transcoding. The *Remote Execution Module* generates and executes ICAP request for adaptation on remote application server based on *transcoding parameters*.

2.4 ICAP-enabled Application Server

As shown in Figure 2-2, the *ICAP-enabled Application Server* consists of *API*, *Transcoding Control Module* (*TCM*), and *Transcoding Modules* (*TM*). Discussed in Section 2.2, the *API* is responsible for performing ICAP communication with the proxy. In this section, we will describe the functionalities of *TCM* and *TM*.

TCM performs two functionalities: (a) generation of transcoding rule and (b) content analysis. It generates transcoding rules based on the adaptation policies and sends them to the proxy through the *API*. The transcoding rule states condition under which the proxy should send a transcoding request to the particular application server. For example, image transcoder accepts only image file format; some transcoders provides services to only low-bandwidth users (e.g. video transcoder).

Upon request for adaptation service, *TCM* performs content analysis on the original HTTP message based on *context information*, its adaptation policy and transcoding rules. It chooses the appropriate *Transcoding Modules* (*TM*) for adaptation.

TMs are the modules which actually perform adaptation. *TM* receives the original HTTP message and *transcoding parameters* from *TCM*. It adapts multimedia content and sends the adapted content to the proxy through the *API*.

2.5 Store-and-forward Transcoding and Streamed Transcoding

In the *Content Repurposing System*, the *ICAP-enabled Application Server* supports two different transcoding strategies: *store-and-forward* transcoding and *streamed* transcoding. For *store-and-forward* media transcoder, it must wait to accumulate an entire media object before transcoding can begin and then must wait to generate a transcoded media object before it is made available to be output. For example, typical command-line interfaces to image conversion libraries are *store-and-forward* transcoders that require the input image in its entirety before image processing can commence, and internally generate a transcoded image in its entirety before making it available as output.

For *streamed* media transcoder, it starts writing out media data encoded in an output format before having fully read in the complete input stream of bytes corresponding to the whole media object encoded in the input format.

For different transcoding techniques, we apply different transcoding strategies as shown in following:

(A) Text Transcoding

In text transcoding, the ICAP modification mode is *Response Modification*. The text page is fetched by the *ICAP-enabled Proxy*. And it is streamed to the *ICAP-enabled Application Server* with the delay of one buffer block which is 4096 bytes.

For adaptation of text, the transcoder needs to analyze the whole text page before transcoding. Therefore, at the *ICAP-enabled Application Server*, adaptation only starts until the entire text page is fetched from the proxy. We perform *store-and-forward* transcoding for text. After the transcoded text page is generated, it is forwarded to the proxy, where it will be returned to the mobile device.

(B) Image Transcoding

In image transcoding, the ICAP modification mode is *Response Modification*. *Store-and-forward* transcoding is performed.

(C) Audio and Video Transcoding

We use *streamed* transcoding for both audio and video transcoding. After one buffer block of media data is streamed to the *ICAP-enabled Application Server*, transcoding starts and the server begins to write transcoded media data. After one buffer block of transcoded image data bytes is accumulated, it is automatically streamed to the proxy. The proxy will immediately return the byte stream to the mobile device. Blocks of transcoded data are continuously streamed to the device on-the-fly when transcoding occurs.

The ICAP modification mode can either be *Request Satisfaction* or *Response Modification*. If we need to cache the original media object at the proxy, *Response Modification* is used. For example, small original audio and video clips can be cached at the proxy. Therefore, if the same object is requested next time, transcoder can use the cached copy for adaptation instead of fetching it from the remote HTTP server. However, large original video objects will not be replicated due to the limited storage capacity.

If caching of the original media object is not required, we use *Request Satisfaction*. The media object is fetched directly from the *ICAP-enabled Application Server* instead of the proxy. For *Request Satisfaction*, the media source can be placed locally at the

application server or remotely at original HTTP server; while for *Response Modification*, the media source is usually placed at remote origin server.

Audio and Video Streaming

The content repurposing system supports streaming of media files. Video and audio can be streamed across the Internet from the server to the mobile device. The client can playback the incoming multimedia stream in real time as the data is received.

There are generally two types of streaming: (a) HTTP streaming and (b) true streaming. HTTP streaming, also known as progressive streaming is an approach to serve media files on the Web without the added management requirements and expense of server-side streaming software. This technique is not well-suited for high-volume sites serving numerous simultaneous streams, but many smaller Web sites can benefit tremendously from this simple and inexpensive approach. The server will not detect user's connection speed. Instead, files optimized for each of the various connection speeds are made available for users to select themselves. Currently, HTTP-based approach does not allow for live streaming audio or video presentations because complete files must be stored on the Web server before they can be accessed. HTTP does not make efficient use of server resources, and as a result doesn't perform well under heavy server loads.

In server-based true streaming, it is able to guarantee continuous delivery of media even if the networks' performance degenerates. If this happens, the video streaming server will automatically send less video data (thus reducing the quality). If the amount of available bandwidth decreases more, the video streaming server will degrade video quality further, until only the audio is left.

Since the system is a HTTP proxy-based adaptation framework, we use HTTP streaming to serve audio and video data.

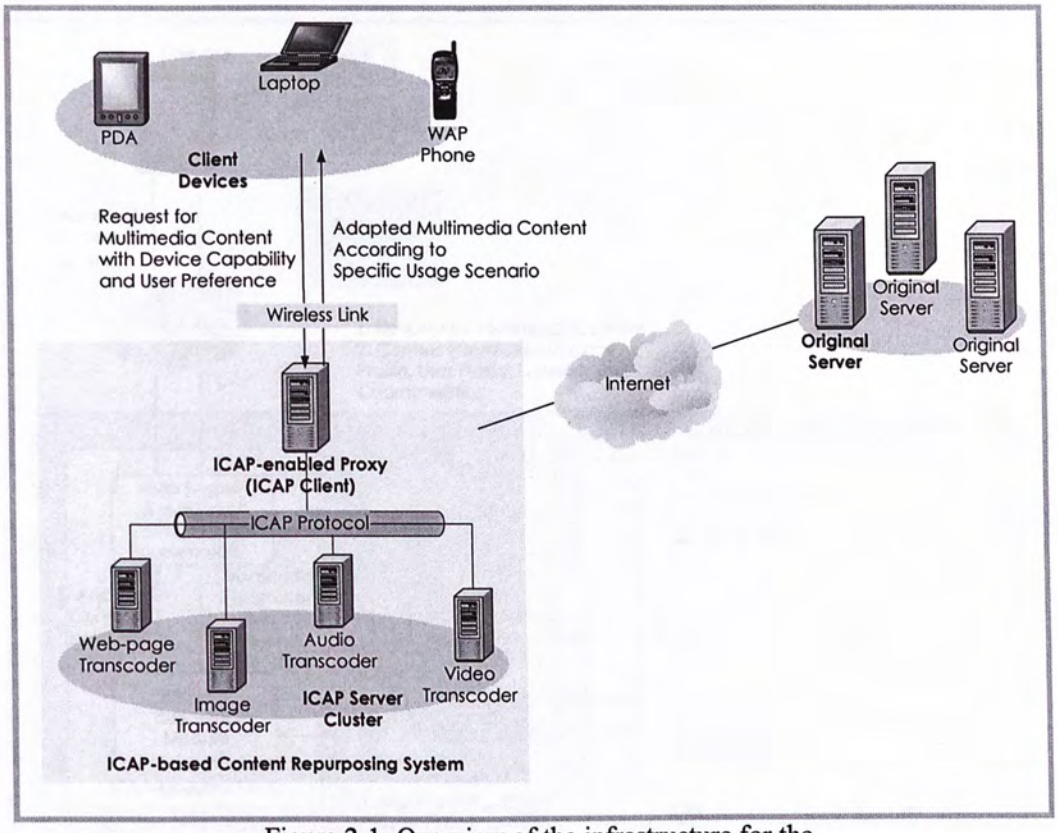

Figure 2-1: Overview of the infrastructure for the *ICAP-based Content Repurposing System*. Proxy cache distributes transcoding servers through ICAP.

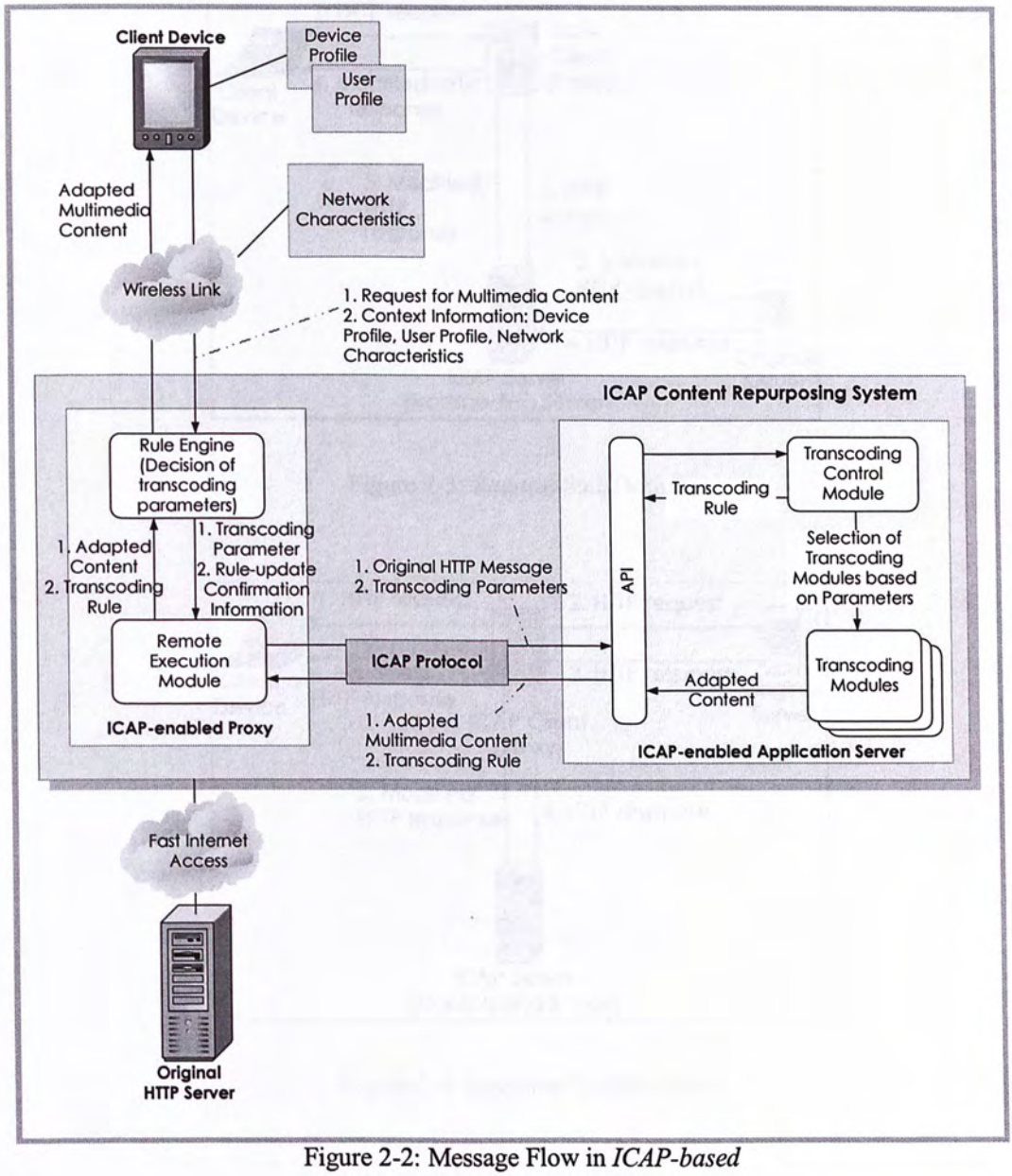

Figure 2-2: Message Flow in *ICAP-based Content Repurposing System.*

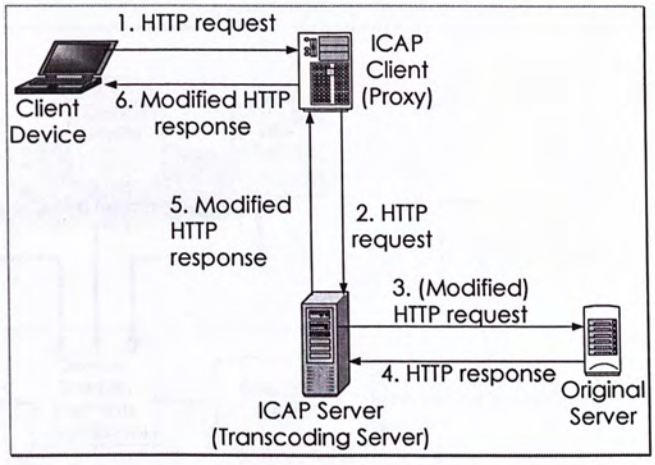

Figure 2-3: Request Satisfaction

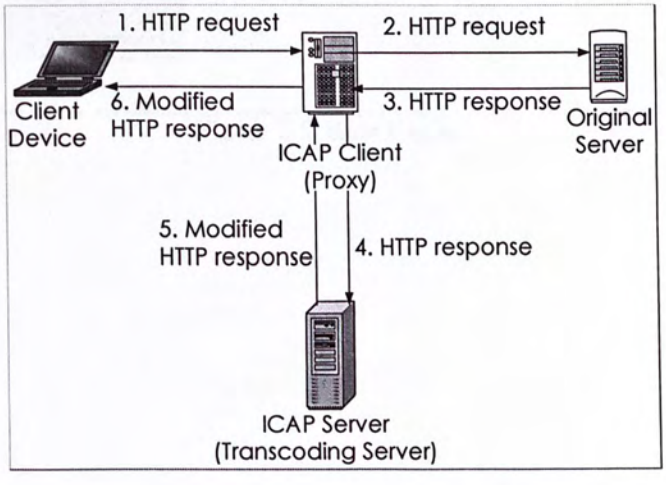

Figure 2-4: Response Modification

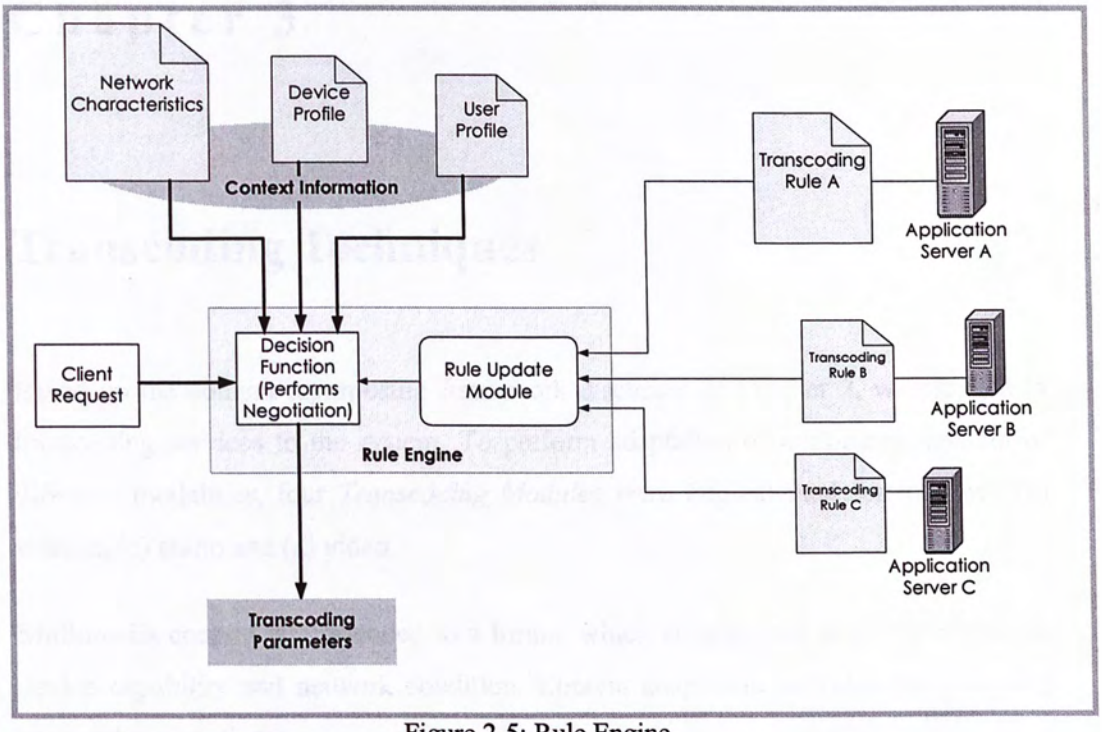

Figure 2-5: Rule Engine

Chapter 3

Transcoding Techniques

Based on the content repurposing framework discussed in Chapter 2, we plugged in transcoding services to the system. To perform adaptation of multimedia content of different modalities, four *Transcoding Modules* were implemented for: (a) text, (b) images, (c) audio and (d) video.

Multimedia content is transcoded to a format which is optimized to user preferences, device capability and network condition. Content adaptation provides the following transcoding functions:

- The encoding format of the media data could be altered, for example translating HTML page to WML for WAP-enabled device, and transcoding WAV audio to MP3 format for compression.
- The displayable area of media data is transcoded to fit the display size of tiny devices.
- The quality level of the media data is adapted, for example compressing the file size of JPEG images by reducing JPEG Quality Factor, and reducing the bit rate of audio and video files.

We provide transcoding services of text and images for mobile device with only WAP platform [9]. Wireless Application Protocol (WAP) is a specification for a set of communication protocols to standardize the way that wireless devices can be used for Internet access. It represents more than 95 percent of the handsets in the world market [10]. To display Web content on WAP platform, Wireless Markup Language (WML) is used for encoding the page and its text; while Wireless Bitmap (WBMP), which is

monochromic, is used for displaying the images. The *Text Transcoding Module* and *Image Transcoding Module* translate Web page and the embedded images to WML and WBMP respectively.

We provide transcoding services to multimedia content of text, images, audio and video according to client *context information*.

3.1 Text Transcoding

The *Text Transcoding Module* (*TM*) has three main functionalities:

- To transcode documents from HTML to WML to provide mobile devices with Internet access in the WAP environment;
- To perform document segmentation for better display on small screens; and
- To reformat page to limit the number of embedded image links for slow networks.

The *rule engine* at the proxy analyses the client *context information* to generate the *transcoding parameters* and decides the transcoding methodology. *Device profile* stores whether the device supports the WAP platform. *User profile* stores the user preferences on whether to receive WML or HTML pages. *Network Characteristics* stores the network information which is useful to page reformatting. The *Transcoding Control Module* (*TCM*) at server-side performs content analysis on text page. *TM* performs adaptation based on *transcoding parameters*.

(A) HTML to WML

Strict one-to-one mapping of tags between the two markup languages is not necessary and impossible as well. HTML document is parsed in the system. All tags are classified into one of the five types (as shown in Figure 3-1):

- *Type I Discarded Tags* – These tags are discarded together with the content enclosed inside them. (e.g. <applet>, <param>, <map>, etc)

- *Type II Discarded Tags* –These tags are discarded but the content encapsulated are mapped to WML format. (e.g. <nobr>, <Tt>, <Kbd>, etc)
- *Exact Mapping Tags* – These are the common tags of both HTML and WML. (e.g. <I>, <select>, <meta>, etc)
- *Rendered Tags* – Here, the HTML tag is changed into a similar tag in WML (e.g. <html> to <wml>).
- *Emulated Tags* – Each of these tags is emulated by a series of WML tags. (e.g. for tag <frame>, a series of WML tags is used to emulate the layout.)

3.2 Image Transcoding

To perform graphical manipulation such as image transcoding, we used ImageMagick 5.5.7 [11]. The *Image Transcoding Module* (*TM*) performs the following functions:

- To transcode Web images to WBMP to provide mobile devices with Internet access in the WAP environment;
- To adapt the image area of delivered WBMP, JPEG, GIF and PNG to fit the displayable area of device.
- To compress GIF, JPEG and PNG files to adaptive quality level to reduce bandwidth consumption for slow network.

(A) WBMP

A WBMP graphic features 1-bit color (either black or white) and, unlike the popular GIF and JPEG formats prevalent on the Web, is not compressed. To display WBMP graphic on WAP phones, its size cannot exceed 1,461 bytes because of memory limitations on current-generation WAP phones. *TM* performs the mapping of JPEG or GIF formats into WBMP. Specifically, an input image file is analyzed and classified as still images and animated GIF files.

- *Still Images* – The original image is first translated to an intermediate image format, BMP (Bitmap). Color de-quantization and resolution adjustment is

performed based on the parameters from *Transcoding Control Module*. Finally, the BMP format is converted to the WBMP format.

- *Animated GIF Files* – Animated GIF files are first disassembled into several still GIF files where each of them is converted to a WBMP file (Figure 3-2). Then a WMLScript is used for rendering the animation of these still WBMP images.

Since WBMP graphic is not compressed, the file size is solely dependent on image area. To generate WBMP image of smaller size, we scale down the image to reduce the number of pixels. The transcoding of graphics to WBMP format features the following functions:

- Enabling mobile users with only WAP platform to access Web content.
- Transcoding the Web content in different manner based on user preferences and the device capabilities. The transcoded image area is adapted to fit the device display size.
- Improving the response time by reducing the content size. It converts the embedded images to WBMP format of smaller size by scaling down the image.

(B) Other Web Images

Embedded images on Web pages mainly comprise of GIF, JPEG and small amount of PNG images. The *Image Transcoding Module* reduces the file size of GIF images in two ways. First, it reduces the number of colors of color-rich images, thereby reducing the number of bits needed per pixel. Second, the image area is adapted to smaller display size and lower transmission speed.

For JPEG image, the *Image Transcoding Module* adapts the file size in two ways. First, it alters the JPEG Quality Factor. The details in transcoding of images using JPEG Quality Factor as a transcoding metric will be discussed in the following section. Second, it alters the image area.

(C) JPEG Quality Factor as a Transcoding Metric

In JPEG, the compression ratio of the output image is controlled solely by the quantization tables used during quantization. An all-1's quantizer achieves a compression ratio that is comparable to a lossless compression algorithm. The JPEG specification [12] provides standard luminance and chrominance quantization tables that provide good results for a variety of images. The standard quantization table can be scaled to vary the compression ratios. Most JPEG compressors allow the user to specify a range of values for the scaling factor, by specifying a compression metric called Quality Factor. This Quality Factor is an artifact of JPEG compression. Different software implementations use different values for Quality Factor. Quality Factors are not standardized across JPEG implementations. Most of the image software follows the standard as IJG Library [13] use. IJG Library uses a 0-100 scale, where 0 is the lowest quality and 100 is the highest. As the Quality Factor is decreased from 100, image compression improves, but the quality of the resulting image is significantly reduced. We use ImageMagick to transcode an image to different quality factors. The resulted images and their file size are shown on Figure 3-3.

In JPEG, the compression ratio of the output image is controlled solely by the quantization tables used during quantization. An all-1's quantizer achieves a compression ratio that is comparable to a lossless compression algorithm. The JPEG specification [12] provides standard luminance and chrominance quantization tables that provide good results for a variety of images. The standard quantization table can be scaled to vary the compression ratios. Most JPEG compressors allow the user to specify a range of values for the scaling factor, by specifying a compression metric called Quality Factor. This Quality Factor is an artifact of JPEG compression. Different software implementations use different values for Quality Factor. Quality Factors are not standardized across JPEG implementations. Most of the image software follows the standard as IJG Library [13] use. IJG Library uses a 0-100 scale, where 0 is the lowest quality and 100 is the highest. As the Quality Factor is decreased from 100, image compression improves, but the quality of the resulting image is significantly reduced. We use ImageMagick to transcode an image to different quality factors. The resulted images and their file size are shown on Figure 3-3.

JPEG Quality Factor is a good representation of the subjective perceived image quality. Reduction in JPEG Quality Factor directly translates to loss of image information quality [14].

ImageMagick 5.5.7 offers package for altering the JPEG Quality Factor of a JPEG image. But the JPEG Quality Factor is not stored along with the image and the original JPEG Quality Factor is lost. Without the knowledge of the initial Quality Factor used to produce an image, ImageMagick might transcode an image with a low initial Quality Factor to an apparently higher Quality Factor value, even though such an operation does not increase the information quality of the output transcoded image. Systems that have used JPEG Quality Factor as a transcoding metric [16, 17], have avoided this problem of not knowing the initial JPEG Quality Factor by transcoding the images to a sufficiently low quality value, such as Q=5, so that they can transcode all images to a smaller size.

Surendar Chandra and Duke University develops software, *JPEG Quality Predictor*, with simple algorithm to estimate the initial IJG equivalent Quality Factor of a JPEG image [14, 15]. We tested the accuracy of the predictor for images produced using ImageMagick 5.5.7. We compressed a reference BMP images using ImageMagick to five Quality Factors: 5, 25, 50, 75 and 95 respectively. The predicted Quality Factors are 5, 25, 52, 77, and 97 respectively. They match closely with the original Quality Factors.

The *Image Transcoding Module* utilizes *JPEG Quality Predictor* to reduce the Quality Factor of original JPEG image to adaptive quality level.

3.3 Audio Transcoding

We provide transcoding services for MP3 and WAV files. All delivered audio are in the format of MP3 (MPEG Audio Layer-III) [21]. MP3 is a standard technology and format for compression a sound sequence into a very small file (about one-twelfth the

size of the original file) while preserving the original level of sound quality when it is played.

MP3 was designed for better quality of audio at lower bit rate. It can be encoded as MPEG-1 and MPEG-2 Audio Layer-III. MPEG-1 provides mono and stereo channel encoding of audio at 32, 44.1 and 48 kHz sampling rates and bit-rates from 32 to 448 kbps. MPEG-2 is a backwards compatible extension to MPEG-1. It was specifically created for effective compression at lower sampling rates and bit-rates, with up to five channels, plus one low frequency enhancement channel. MPEG-2 provides encoding of audio at 16, 22.05 and 24kHz sampling rates for bit rates between 8 to 320 kbps for Layer-III.

The *Audio Transcoding Module* uses lame-3.93.1 [18] to transcode audio files to MP3 format of adaptive quality. Lame provides MPEG-1, 2 and 2.5 Layer-III encoding. "MPEG-2.5" is a proprietary non-ISO extension which was created by the Fraunhofer Institute [22] to improve performance at lower bit rates. At lower bit-rates, this extension allows sampling rates of 8, 11.025 and 24kHz. A high sampling rate at a very low bit-rate requires a trade-off in reduced resolution. Lowering the sampling rate reduces the frequency response but allows the frequency resolution to be increased, so the result is a file with significantly better quality.

The output audio files could be encoded to CBR (constant bitrate) or either one of the two types of variable bitrate, VBR (variable bitrate) and ABR (average bitrate, as known as safe bitrate).

The transcoding module aims to provide the following features:

- To transform WAV files to MP3 format for compression.
- To alter the bitrate of MP3 files for adaptation to network condition for smooth playback of audio.
- To utilize *streamed* transcoding for the *ICAP-enabled Application Server* to provide HTTP streaming of audio.

3.4 Video Transcoding

To adaptive control the bit rate of MPEG-1 video, we integrate the technology of Video Transcoder version 1.0 [19] into *Video Transcoding Module* (*TM*). Video Transcoder version 1.0 is a command-line program which transcodes an MPEG-1 file in a file-in-file-out way. It provides bit rate control of MPEG-1 video using multiple transcoding techniques. It combines techniques of requantization and subsampling to achieve a wider range of output bit rates.

We embed the transcoder into the *Video Transcoding Module* to support *streamed* transcoding. The *TM* aims at providing real-time video which can be streamed smoothly to the client device for better display quality. It has three main functionalities:

- To adapt the resolution of MPEG-1 video to fit the displayable area of device and save bandwidth by subsampling.
- To compress MPEG-1 video by automatic bit rate control using adaptive quantization for adaptation to network bandwidth for smooth playback.
- To utilize *streamed* transcoding to for the *ICAP-enabled Application Server* to provide HTTP streaming of video.

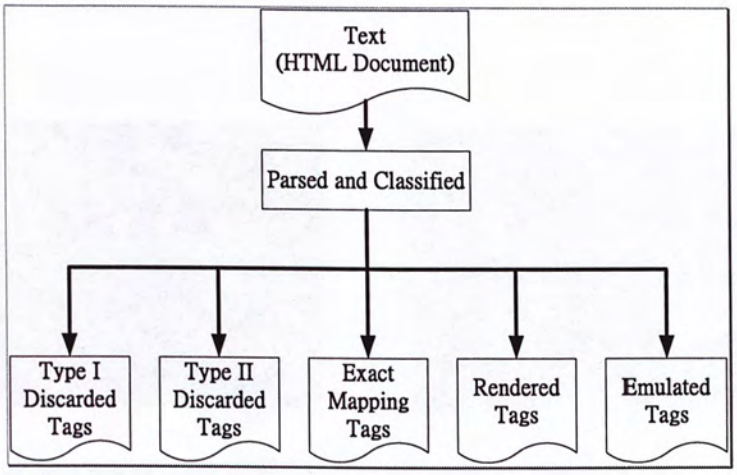

Figure 3-1: Content Adaptation Decision Tree

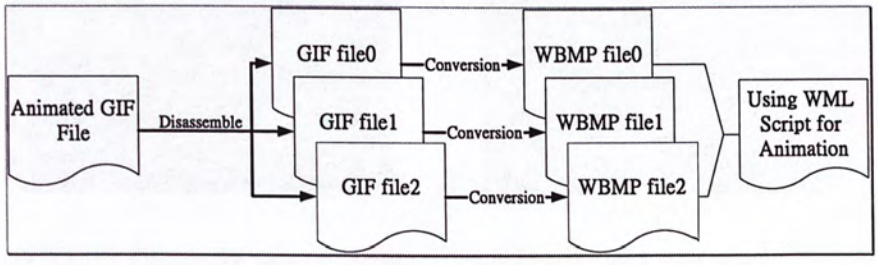

Figure 3-2: Adaptation of an animated GIF file in the WAP environment

(a) Original Image, Size: 1.96KB

(b) Quality Factor: 75, Size: 1.69 KB

(c) Quality Factor: 50, Size: 1.13 KB

(d) Quality Factor: 25, Size: 868 bytes

Figure 3-3: Transcoding of a JPEG image to different Quality Factors

Chapter 4

Adaptation Policy

The content repurposing system aims to provide adaptation on two dimensions:

(A) Content Reformatting: For *weak* mobile device which does not support an encoding format of the requested content or cannot display the content appropriately, the transcoder transforms the content to a format which is supported by the device. For example, for mobile devices with only WAP platform, text and images are transformed to WML and WBMP; and the resolution of images and video are scaled down to match with the small display size of the mobile device.

(B) Content Compression: For mobile device which connects through low bandwidth, the system transcodes and reformats content to a lower quality to compress the file size (or bit-rate for real-time media). The reduction in file size may result in shorter end-to-end latency of retrieving multimedia content. The reduction in latency requires the trade off in the loss in quality.

Table 4-1 summarizes the three application cases of the system. We use different adaptation policy on each case. A *weak* mobile device is a device which cannot present the requested content appropriately or cannot support an encoding format of the content. A *strong* device is one which can present the content appropriately and support the encoding format. In this chapter, we focus on the analysis of the end-to-end latency of the content repurposing system. For low-bandwidth clients, as in Case 1 and 3, the system transcodes content to a smaller file size (or bit-rate for real-time media). This results in the reduction in latency. In Case 2, the client connects through a high bandwidth. Content is reformatted only for better display at the device. Transcoding

may enlarge the end-to-end latency. In Section 4.4, we provide the three application case studies.

	Application Cases	Adaptation Policies
1	*Weak* device with insufficient bandwidth	*Content reformatting* and *content compression*
2	*Weak* device with sufficient bandwidth	*Content reformatting*
3	*Strong* device with insufficient bandwidth	*Content compression*

Table 4-1: Application cases and adaptation policies

To perform adaptation, the *rule engine* generates a set of *transcoding parameters* that controls the extent and types of compression performed by the *Transcoding Module* (*TM*). The decision making on the *transcoding parameters* is based on client's *context information* (*user profile*, *device profile*, and *network characteristics*). For example, the *scaling parameter* determines how much an image or video is downsampled; *number-of-color parameter* determines the output number of colors in a colormapped image file; the *encoding-format parameter* determines whether the encoding format of the original media file should be altered (HTML to WML, GIF to WBMP, WAV to MP3, etc); the *quality-factor parameter* determines the output Quality Factor of the JPEG image; the *bitrate parameter* determines the output bitrate of audio and video files. Given *N transcoding parameters*, the *N*-tuple space of possible combinations becomes quite large, which poses a problem for optimized transcoding.

For low-bandwidth clients, to maximize the benefits of transcoding, the *rule engine* needs to consider each of the above criteria to determine under which conditions transcoding is able to reduce the response time. Transcoding of media data is computational expensive and introduces delay. For slow wireless connection proxy-client link, the transcoding delay could be compensated by the reduction in transmission time through compression. The reduction in response time due to media

compression significantly outweighs the delay due to transcoding operations. But, for fast proxy-client link, it is not beneficial to transcode content for compression. The reduction in response time for compression could no longer compensate the expensive transcoding computations. Therefore, in this case, we only perform *content reformatting* for *weak* devices.

In this Chapter, we perform delay analysis for different transcoding policies. We also formulate conditions for *rule engine* to make decision when it is beneficial for the *TM* to transcode a media data for low-bandwidth clients. The delay due to text transcoding is rather static and small when compared with transcoding of media-rich data. Therefore, we will focus on image, audio and video transcoding in this section.

4.1 Delay Analysis of Content Repurposing System

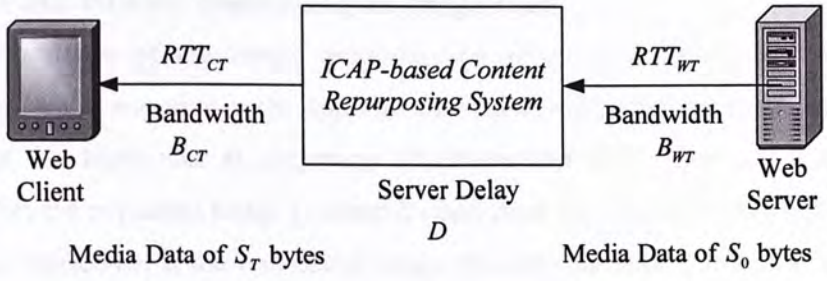

Figure 4-1: Delay Analysis of *ICAP-based Content Repurposing System*

Figure 4-1 shows the model for delay analysis of the system. All analysis is based on *Response Modification* where the *ICAP-enabled Proxy Server* first retrieves the media data from the *Web Server*. The *ICAP-enabled Application Server* performs adaptation on the media data and the transcoded result is forwarded to the *Web Client* through the *ICAP-enabled Proxy Server*.

The original media is of data size S_0 (in bytes). It is downloaded from the Web server to the *ICAP-enabled Proxy Server* through a link of bandwidth B_{WT} and network roundtrip time latency RTT_{WT}. The transmission link between the proxy server and the

ICAP-enabled Application Server is fast. Besides, ICAP is a lightweight protocol. Therefore the delay due to message passing between them is negligible. And we assume the server delay D is dominated by the transcoding latency at *ICAP-enabled Application Server*. The transcoded media is of data size S_T. It is downloaded to the mobile client through a link of effective bandwidth B_{CT} and network roundtrip time latency RTT_{CT}.

In the content repurposing system, we implement *store-and-forward* and *streamed* transcoding. *Store-and-forward* transcoding is used in adaptation of text and images; while *streamed* transcoding is for audio and video streaming media. The two transcoding strategies would be analyzed separately and discussed in Section 4.2 and 4.3 respectively.

4.2 Store-and-forward Transcoding for Image Files

The performance of the image transcoder (*ICAP-enabled Application Server*) is evaluated by the reduction in the response time perceived by the mobile client after we have put the transcoder at the proxy (*ICAP-enabled Proxy Server*). Without the transcoder, the requested image is either fetched from the original server or the cache. With the transcoder, if the transcoded image has been cached, it is forwarded to the client. If the transcoded image has not been cached, the original page, which is fetched from the original server or the cache, is transcoded. The transcoded image is directed to the client and cached.

When there is no transformation of content occurs, we assume the processing delay of the *ICAP-enabled Proxy Server* is negligible. We analyze the system by the following five cases:

(A) No Adaptation, No Caching:

Without transcoding, if the requested image has not been cached at the proxy, the end-to-end latency R_0 perceived by the mobile client is given by [17]:

$$R_0 = 2RTT_{WT} + 2RTT_{CT} + \frac{S_0}{\min(B_{CT}, B_{WT})} \quad (1)$$

Here, we assume the *ICAP-enabled Proxy Server* does not introduce any delay on the message passing between the mobile client and the Web server.

(B) No Adaptation, Original Image Cached:

Without the transcoder, if the requested image is cached at the proxy, the end-to-end latency R_1 perceived by the mobile client is given by [17]:

$$R_1 = 2RTT_{CT} + \frac{S_0}{B_{CT}} \quad (2)$$

(C) Adaptation, Cache Misses:

With the transcoder, if both the required transcoded image and the original image have not been cached, the end-to-end latency R_2 is given by:

$$R_2 = 2RTT_{WT} + 2RTT_{CT} + D + \frac{S_T}{B_{CT}} + \frac{S_0}{B_{WT}} \quad (3)$$

The *ICAP-enabled Proxy Server* stores the requested image fetched from the Web server, transcodes the content and forwards it to the client.

(D) Adaptation, Original Image Cached:

With the transcoder, if the required transcoded image has not been cached but the proxy has cached the original one, the end-to-end latency R_3 is given by:

$$R_3 = 2RTT_{CT} + D + \frac{S_T}{B_{CT}} \quad (4)$$

(E) Adaptation, Transcoded Image Cached:

With the transcoder, if the required transcoded image has been cached, the end-to-end latency R_4 is given by:

$$R_4 = 2RTT_{CT} + \frac{S_T}{B_{CT}} \tag{5}$$

Typically, mobile device connects outside through a much slower link than the *Transcoding Proxy Server* does (i.e. $B_{WT} >> B_{CT}$).

For the case that the original content is cached, since $R_2 > R_3 > R_4$ and $R_0 > R_1$, comparing equation (2) and (3), the content repurposing system reduces the end-to-end latency (i.e. $R_1 > R_2$) if and only if

$$D < \frac{S_0 - S_T}{B_{CT}} - \frac{S_0}{B_{WT}} - 2RTT_{WT} \tag{6}$$

For the case that the original content is not cached, the content repurposing system reduces the end-to-end latency (i.e. $R_0 > R_2$) if and only if

$$D < \frac{S_0}{\min(B_{CT}, B_{WT})} - \frac{S_T}{B_{CT}} - \frac{S_0}{B_{WT}} \tag{7}$$

Therefore, for low-bandwidth client, the content repurposing system should perform transcoding to satisfy inequality (6) or (7). *Rule Engine* could find the value of RTT_{WT}, B_{CT} and B_{WT} from the *network characteristics* of client's *context information.* S_0 is also known. Therefore, for *Rule Engine* to make decision on adaptation policies, it should have the ability to predict the output file size S_T and the transcoding latency D for a particular set of *transcoding parameters*.

Transcoding latency depends not only on media processing time (CPU time), but also on queuing delay caused by the operating system's sharing of the CPU among multiple processes and threads. The variable delay introduced by the operating system is especially difficult to predict in non-real-time operating systems. Therefore we predict the transcoding latency due to media processing only.

In our experiments, the *ICAP-enabled Proxy Server* runs on Dell System PowerEdge 2400, with dual CPUs, each of Pentium III 667MHz. System memory is 256MB and

processor bus runs on 133MHz. The image transcoder and audio transcoder runs on two different machines with the same configuration: Pentium IV 1400MHz with 256MB memory. The video transcoder runs on dual CPUs, each of Pentium III 851MHz. System memory is 512MB.

In the following sections, we perform experiments on transcoding of images and obtain results for *Rule Engine* to learn how to predict S_T and D.

4.2.1 Distribution of Input Web Images

In our experiments, we only use JPEG images as the input images to the image transcoder. We collect 3000 JPEG images from the first ten top-rated global Web sites (by properties) according to Nielsen's Net-ratings in March, 2003 [20]. Figure 4-2 shows the cumulative distribution of the collected input images as a function of their input number of pixels. 80% of the input images are less than 10K pixels. Figure 4-3 shows the cumulative distribution of the file size of the input images. 90% of the input images are less than 5KB. Most of the images in Web sites have small image geometries and small file size.

The *Rule Engine* determines whether the client browser runs on WAP platform. If yes, the *encoding-format* parameter will be set for *TM* to transcode the Web image to WBMP format. Otherwise, the output image would remain in its original format. In this thesis, we would only analyze the performance of the image transcoder for adaptation of WBMP and JPEG only.

4.2.2 Transcoding Web images to WBMP

We define P_0 as the number of pixels in the original image and P_T as the output number of pixels. We transform the set of input images into WBMP. We do not put a scaling on the output image area. Therefore, P_T is the same as P_0. Figure 4-4 and Figure 4-5 show the scatter plots of the transcoding latency D versus original image size S_0 and the number of pixels P_0 respectively. There is no close relationship between the original image size and the transcoding latency. An input image of larger size does not necessarily take longer time to transcode. But a strong linear relationship exists between D and P_0. On Figure 4-5, the dotted straight line is the fitting of the data into a

linear least-square regression model. The correlation coefficient for the linear relationship between D and P_0 is 0.99. Therefore, we express D as a linear function of P_0, $D(P_0)$.

Figure 4-6 shows the scatter plots of output image data size S_T versus original image pixel number P_0. S_T does not depend on S_0. It depends only on P_0. Figure 4-7 shows a strong linear relationship between P_0 and the output image data size S_T. This can be explained by the nature of WBMP images. They are not compressed and its file size reflects its image area. We express S_T as a linear function of P_0, $S_T(P_0)$.

In the transcoder, the server delay $D(P_0)$ depends on: (a) the number of pixels of the original message P_0 and (b) the scaling ratio of the pixel number between the original and transcoded images. It takes longer to transcode an image of higher resolution with the same scaling ratio. The output size of transcoded image $S_T(P_0)$ relates tightly to its original number of pixels P_0. Both $D(P_0)$ and $S_T(P_0)$ are independent of the data size of the original image S_0.

Figure 4-8 shows the cumulative distribution of the ratio of the file size of output WBMP to that of the original Web image. 93% of the output images have their file size smaller than the original file size. Some output WBMP files have their file size increased by several times. This occurs for those GIF and JPEG images which have been highly compressed. They have large image geometries but small file size. This results in generating large WBMP file. We repeat the experiments by resizing the geometry of the original images during the transformation to WBMP. We scale down their width and height both by (a) 25%; (b) 50%; and (c) 75%. From Figure 4-9, by scaling down the width and height by 25% respectively, 90% of the output WBMP images have their file size less than 8% of their original file size and all output WBMP images have their file size less than 45% of the original value. Figure 4-10 shows the transcoding latency versus the number of pixels in the original images for different level of downsampling. Only the linear regression fitting lines are shown in the diagram. It shows that the transcoding latency is greatly reduced after we resize the image to a smaller area for the same input number of pixels.

4.2.3 Adaptation policy of Transformation to WBMP

From Figure 4-5 and Figure 4-7, both transcoding latency D and output file size of WBMP S_T have high linear correlation with P_0. We express D and S_T as linear functions of P_0. *Rule Engine* could predict D and S_T from P_0 for different level of downsampling the image. We provide three levels of downsampling the images. We scale down their width and height both by (a) 25%; (b) 50%; and (c) 75%. For low-bandwidth clients, *Rule Engine* determines how much the Web image should be scaled so that the inequality (6) or (7) is satisfied.

4.2.4 Adaptation policy of JPEG images

The image transcoder provides three measures for adaptation of JPEG files: (a) reducing the Quality Factor of the original JPEG by 25%, 50% or 75%; (b) downsampling both the image width and height by 25%, 50% or 75%; and (c) applying measures (a) and (b) together, that is reducing both the Quality Factor and the image area.

In this section, we first define transcoding efficiency of a transcoding operation and present the results. Then we discuss the transcoding latency for different transcoding measures.

Once the Quality Factor is determined, we can check if the file is sufficiently compressed. S. Chandra and C. Ellis define transcoding efficiency of a transcoding operation by the ability to lose more in file size for a particular loss in information quality [14]. For example, if an image was transcoded to lose 50% of Quality Factor, the output image size should be less than 50% of the original image for it to be efficiently transcoded. For some applications, this restriction may be too strict and these applications may relax the constraint to accept output size of, say, 55% instead of 50%.

We performed experiments on the image transcoder to measure the transcoding efficiency. Figure 4-11 shows the cumulative distribution of the Quality Factor of the input images. Figure 4-12 shows the cumulative distribution of the ratio of the output file size to that of the original JPEG image. We reduce the initial Quality Factor to

25%, 50% and 75%. By reducing their Quality Factor to 25%, 50% and 75% respectively, all of the input image could reduce their file size to 55%, 75% and 90% of their original size. For a given reduction in quality, any change in file size that is less than the change in quality is an efficient transcoding as the resulting file size is smaller than the corresponding loss in quality. From Figure 4-12, we can see that reducing the initial quality to 25%, 50% and 75% was efficient for 5%, 24% and 56% of the images respectively.

We find that the percentage of images that can be transcoded efficiently may differ significantly for different Web sites. Different Web sites have different nature in their embedded images. Commerce sites, such as ebay.com and amazon.com, would like to deliver big, high quality images for promoting the listed products to attract subscribers. News sites, such as CNN.com and abc.com, use images to reinforce some news story. The images are secondary to the news being delivered. Hence we expect the images to be small and of low quality. From the experiments, we find that the initial Quality Factor affects much on the ratio of the output file size to the original file size. We plotted the average ratio of output file size to original file size against the initial Quality Factor in Figure 4-13. The higher the quality of the image, the more the JPEG image could be compressed. Therefore, we would expect the compression ratio of images from News sites is lower than that from Commerce sites which consist of high quality images.

We repeat the experiment on another transcoding measure of resizing the image geometries. From Figure 4-14, by reducing both the image height and width by 25%, 50% and 75% respectively, all of the images could be reduced to 50%, 55% and 80%.

We then combine the above two transcoding measures by both reducing the Quality Factor and resizing the image geometries. From Figure 4-15, by reducing the image height, image width and Quality Factor to 25%, 50% and 75% respectively, all images could be reduced to 40%, 45% and 60% of the original file size.

The results obtained from Figure 4-12, 4-14 and 4-15 provide a good basis for *Rule Engine* to gain information of the lower bound of file size reduction after transcoding for different transcoding measures.

We proceed to the experiments for predicting the transcoding latency for different adaptation policies. In previous experiment (Figure 4-5), we find that for transformation of Web images to WBMP format, there is a high correlation between the transcoding latency and the number of pixels in original image. The experiment shows the same result in adaptation of JPEG images. Figure 4-16 plots the linear regression fitting line of transcoding latency against the original number of pixels for reducing the Quality Factor to 75%, 50% and 25% respectively. We find that the transcoding latency only differs very slightly for different percentage of Quality Factor that the image is reduced to. The number of processing blocks in the image does not change for different reduction in the Quality Factor. However, for transcoding the image to a lower ratio of the initial Quality Factor, say 25%, the number of bits that the transcoder needs to output is fewer. Therefore we would expect a slightly higher latency for transcoding the Quality Factor to 25% of the initial value than to 75%.

Figure 4-17 plots the transcoding latency for different adaptation policies. The processing time is higher if we need to resize the image compared with reducing the Quality Factor alone. If we resize an image more on its geometry, the transcoding latency would be lowered. This is because the number of processing blocks is reduced if we perform downsampling on the image and the number of bits that the transcoder needs to output is fewer. The transcoding latency of reducing both the Quality Factor and the image geometry is slightly less than that of reducing only the image geometry. This is because the number of bits that the transcoder needs to output is more. Resizing the image shows a greater effect on affecting the transcoding latency.

The image transcoder provides nine adaptation policies for adaptation of JPEG images: (i) reducing the Quality Factor by (a) 25%, (b) 50%, and (c) 75%; (ii) reducing the image width and height both by (a) 25%, (b) 50%, and (c) 75%; (iii) reducing the Quality Factor, image width and height by (a) 25%, (b) 50%, and (c) 75%.

Figure 4-16 and 4-17 show that the *Rule Engine* can predict transcoding latency D from P_0 for different adaptation policies. From Figure 4-12, 4-14 and 4-15, the *Rule Engine* can predict an upper bound of S_T from S_0. For low-bandwidth client, the *Rule Engine* can determine the adaptation policy for satisfying the inequality (6) or (7).

4.3 Streamed Transcoding for Audio/Video Files

We perform *streamed* transcoding for low-bandwidth clients only. In this section we derive conditions under which it is beneficial for a streamed audio/video transcoder to perform adaptation. A *streamed* transcoder starts writing out image data encoded in an output format before having fully read in the complete input stream of bytes corresponding to the whole file encoded in the input format. If the media encoding format supports streaming, media file could be playback at the client's media player while transcoding is processing on-the-fly at the server side. We utilize the model presented in [17] as a basis in the analysis of *streamed* transcoding.

In *streamed* transcoding, the input image arrives as a stream of bits spaced apart by $1/B_{WT}$. The *streamed* transcoder will take a buffer of G_0 bits for transcoding, incurring a small *store-and-forward* delay D_1.

$$D_1 = \frac{G_0}{B_{WT}} \tag{7}$$

The group of bits is then transcoding into a group of G_T output bits, incurring a transcoding delay D_2.

$$D_2 = \frac{D}{S_0 / G_0} \tag{8}$$

D is the transcoding latency for transcoding S_0 bits of the media file (time for transcoding the whole file).

If $D_2 < D_1$, the transcoder can convert each input group of G_0 bits to its corresponding output group of G_T bits before the next input group needs to be processed. In this case

the internal memory requirement of the *streamed* transcoder is bounded. But, if $D_2 > D_1$, the transcoder is not able to process the input bits fast enough and the finite-length internal RAM buffers of the transcoder will overflow given a continuous input stream.

For $D_2 < D_1$, the group transcoding latency must satisfy

$$\frac{D}{S_0 / G_0} < \frac{G_0}{B_{WT}} \text{ or}$$

$$D < \frac{S_0}{B_{WT}} \tag{9}$$

To transmit the output transcoded groups of G_T bits to the client, it takes a transmission delay D_3.

$$D_3 = \frac{G_T}{B_{CT}} \tag{10}$$

We assume $D_2 < D_1$. If $D_3 < D_1$, each output group of G_T bits can be sent before the next output group is ready for transmission. But, if $D_3 > D_1$, the output transmission link cannot send the produced bits fast enough to keep the output queue empty. The output queue of the transmission link grows without bound given a continuous input stream.

For $D_3 < D_1$, the inequality (11) must be satisfied

$$\frac{G_T}{B_{CT}} < \frac{G_0}{B_{WT}} \text{ or}$$

$$\gamma > \frac{B_{WT}}{B_{CT}} \tag{11}$$

where γ is the *group image compression ratio* G_T/G_0, which we assume to be on average equivalent to the overall image compression ratio S_T/S_0.

Therefore, *streamed* transcoding of audio and video files could only perform when both inequalities (9) and (11) is satisfied. For *Rule Engine* to make decision on adaptation policies, it should have the ability to predict the output file size S_T and the transcoding latency D for a particular set of *transcoding parameters*.

In the following sections, we perform experiments on transcoding of audio and video files and obtain results for *Rule Engine* to learn how to predict S_T and D.

4.3.1 Audio Transcoding

The audio transcoder transcode original MP3 files into MPEG-1, 2, 2.5 Layer III audio. We provide transcoding of MP3 files from a variety of bit-rates ranging from 8 kbps to 128 kbps, with joint stereo channels. Traditional stereo audio encodes each left and right channel as a separate stream. Joint stereo takes advantage of the correlations between two stereo channels. It enables better audio compression by essentially combining redundant audio information in both stereo channels into a single data stream. Differences in the two audio channels that are detectable to human hearing are not combined, so stereo sound is maintained, but in a smaller file size than traditional stereo would require.

We use Constant Bit-rate (CBR). The compression ratio S_0 / S_T is the same as the ratio of original bit-rate to output bit-rate. Table 4-2 shows the sampling rate, coding standard, transcoding latency per audio frame for different output bit-rates. Except for encoding audio to 96 kbps, all files could be transcoded within 5 ms per frame.

The MP3 stream is encoded as frames, which are fixed-length block of audio data. Each frame contains 1152 samples for MPEG-1 audio and 576 for MPEG-2 and MPEG-2.5. From Table 6-2, given the sampling rate S_r and output bit-rate $B_{r'}$, we could estimate the total transcoding latency D.

s is the number of samples in a frame. Size of each frame b (in bits) is given by:

$$b = \frac{s}{S_r} B_{r'} \tag{12}$$

S_T(in bits) is the estimated output file size which is predicted by the output bit-rate $B_{r'}$ original bit-rate B_r and original file size S_0.

$$S_T = \frac{S_0 B_{r'}}{B_{r'}} \tag{13}$$

Then total number of frames n is given by:

$$n = \frac{S_T}{b} = \frac{sS_0}{B_r S_r} \tag{13}$$

From Table 6-2, transcoding latency per frame is given by d. The total transcoding latency for whole stream D is predicted as:

$$D = nd = \frac{nsS_0}{B_r S_r} \tag{14}$$

Output Bitrate (kbps)	Coding Standard	Transcoding Latency per frame (ms)	Sampling Rate (kHz)
128	MPEG-1	4.11	44.1
96	MPEG-1	7.63	32.00
64	MPEG-2	4.08	22.05
48	MPEG-2	4.33	16.00
32	MPEG-2.5	4.53	11.03
16	MPEG-2.5	4.87	8.00
8	MPEG-2.5	4.97	8.00

Table 4-2: Transcoding Measures of MP3 files.

4.3.2 Video Transcoding

According to [19], *Video Transcoder Version 1.0*, which was run on a Pentium IV 2.0GHz machine, the processing time per frame of transcoding an MPEG-1 file was constant and was about 11ms, which is equivalent to 91fps, in the whole range of achievable bit rates. For VCD-quality MPEG-1 movies, the frame rate is at 30 frames

per second for NTSC, and 25 frames per second for PAL. We could use the results in [19] to predict D and S_T given that S_0 is known. Details are not discussed in this thesis.

4.4 Case Study

As discussed before, there are four application cases as shown in Table 4-1. We provide each of them with a case study.

4.4.1 Weak Device with Insufficient Bandwidth

We performed experiments to test the performance of the image transcoder for a mobile client connecting to the proxy by a GPRS (General Packet Radio Services) WAP-enabled phone with downstream data transmission speed of up to 40.2kbps. The phone supports WAP only. We tested the client by retrieving 1000 JPEG files from a commerce site, ebay.com. All original and transcoded content are not cached at the proxy. We compare the end-to-end latency with and without image transcoding. The transcoder performs (a) transformation of JPEG images to WBMP (b) reduction of image height and width both by 25% for better display on small screen and reduction in file size. We plot the percentage of end-to-end latency reduced as cumulative distribution functions, as shown in Figure 4-18. A small percentage of 0.17% of the images have their end-to-end latency increased after transcoding. Most of the images, 99.83%, have their latency improved. 50% of the images have their latency reduced by more than 49%.

4.4.2 Weak Device with Sufficient Bandwidth

We test the performance of the image transcoder for a PDA client connecting to the proxy through high bandwidth. The PDA connects through 802.11b (Wi-Fi) access point with downstream data transmission speed of up to 11Mbps. Again, we tested the client by retrieving 1000 JPEG files from a commerce site, ebay.com. Due to the limited display size of the PDA, the image transcoder adapts images to smaller image geometries, reduction of image height and width by 25%. The transcoded images remain in JPEG format and their original Quality Factor. We plot the percentage of end-to-end latency reduced as cumulative distribution functions, as shown in Figure 4-

19. 36.8% of the images have their end-to-end latency increased after transcoding. The computational expense in transcoding could not be compensated by the reduction in transmission latency by reducing the file size for high-bandwidth clients.

4.4.3 Strong Device with Insufficient Bandwidth

A notebook PC connects through 802.11b (Wi-Fi) access point with downstream data transmission speed of up to 11Mbps. The client retrieves audio and movie from busy Web sites. The slow proxy-server link does not support the delivery of high-quality multimedia content. Therefore the audio/video transcoder adapts the bit-rate of the multimedia data to a lower level.

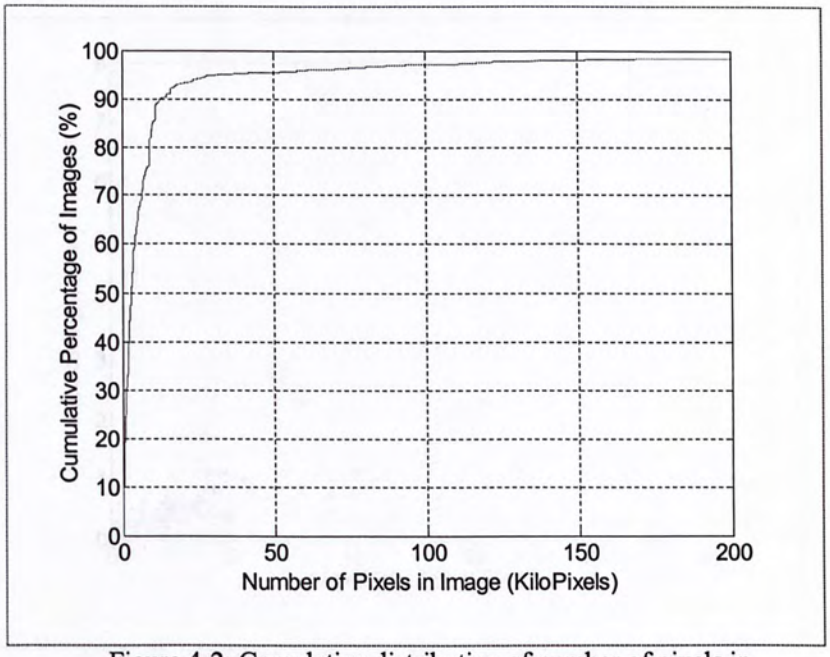

Figure 4-2: Cumulative distribution of number of pixels in image

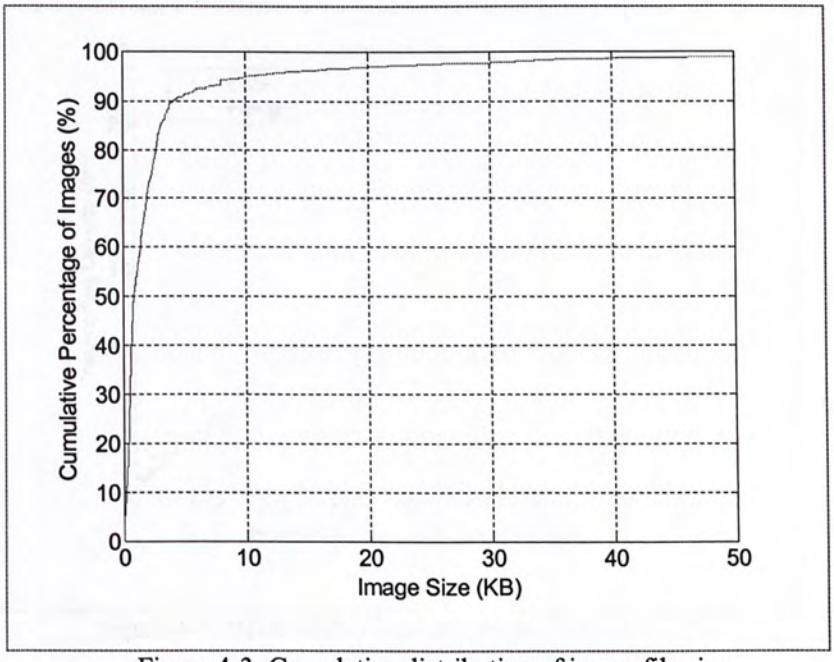

Figure 4-3: Cumulative distribution of image file size

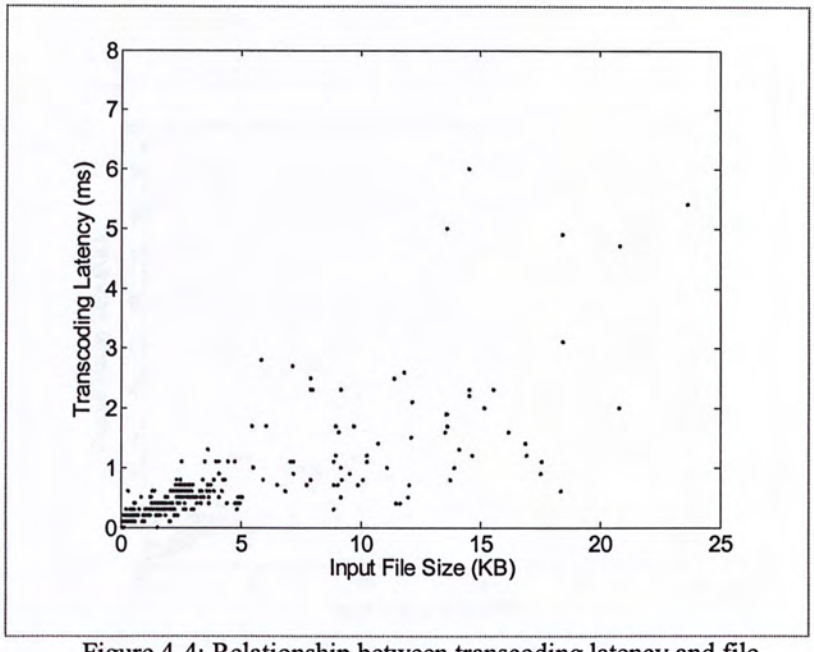

Figure 4-4: Relationship between transcoding latency and file size of input images for transcoding to WBMP format

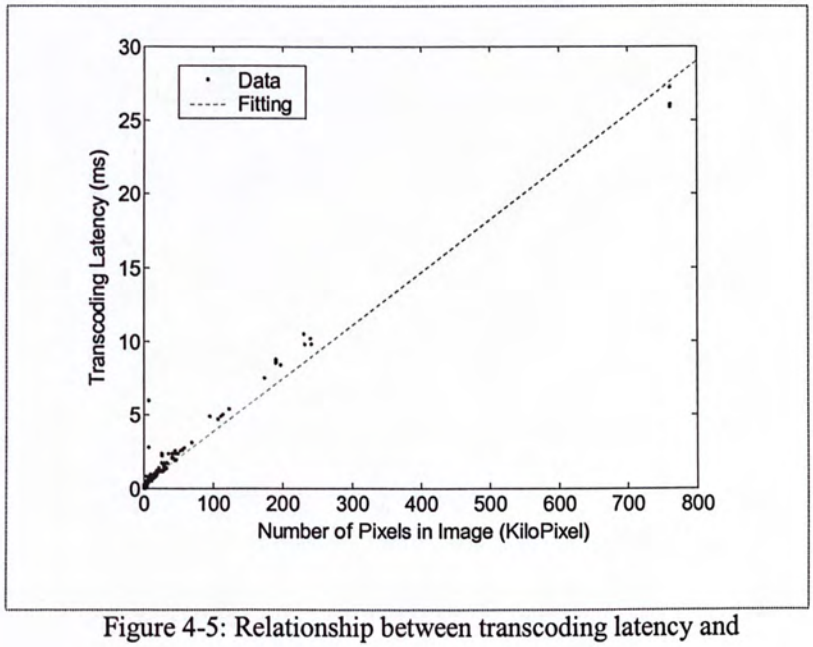

Figure 4-5: Relationship between transcoding latency and number of pixels in input images for transcoding to WBMP format

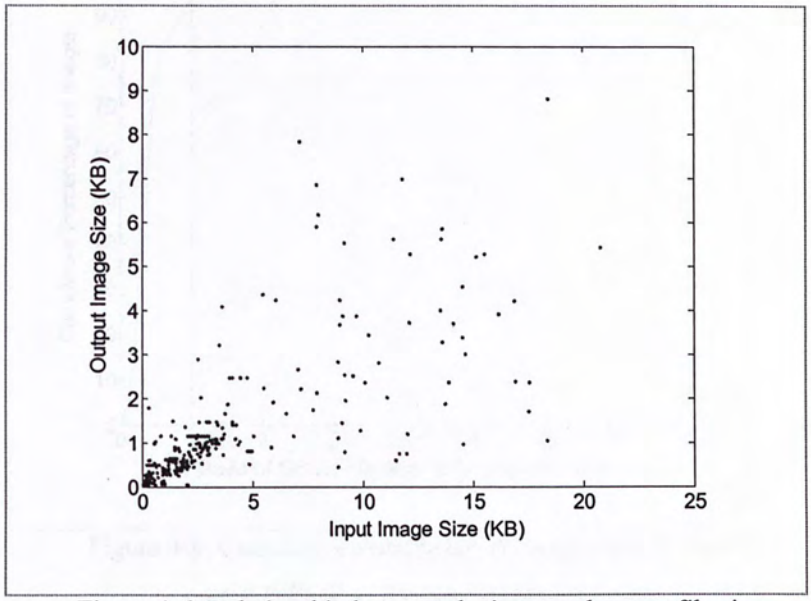

Figure 4-6: Relationship between the input and output file size

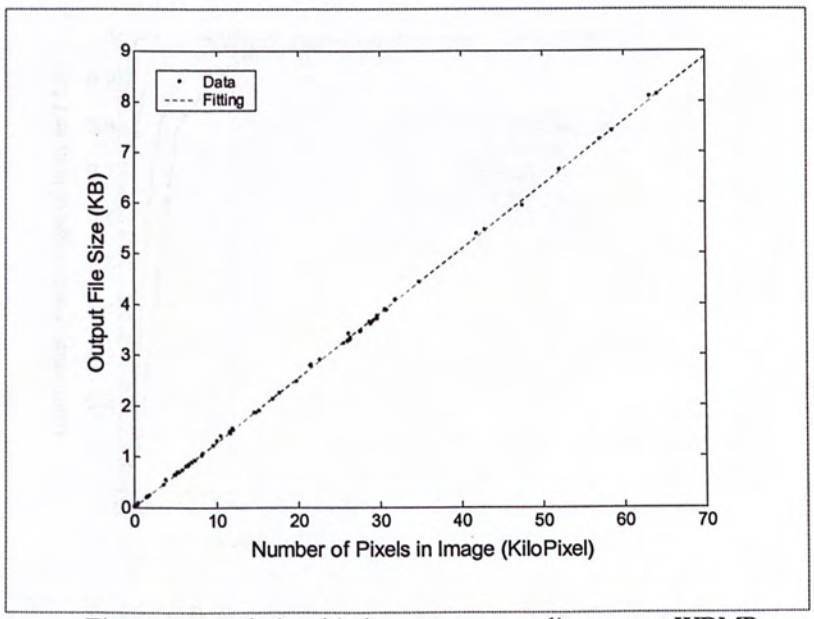

Figure 4-7: Relationship between transcoding output WBMP file size and number of pixels in the original image

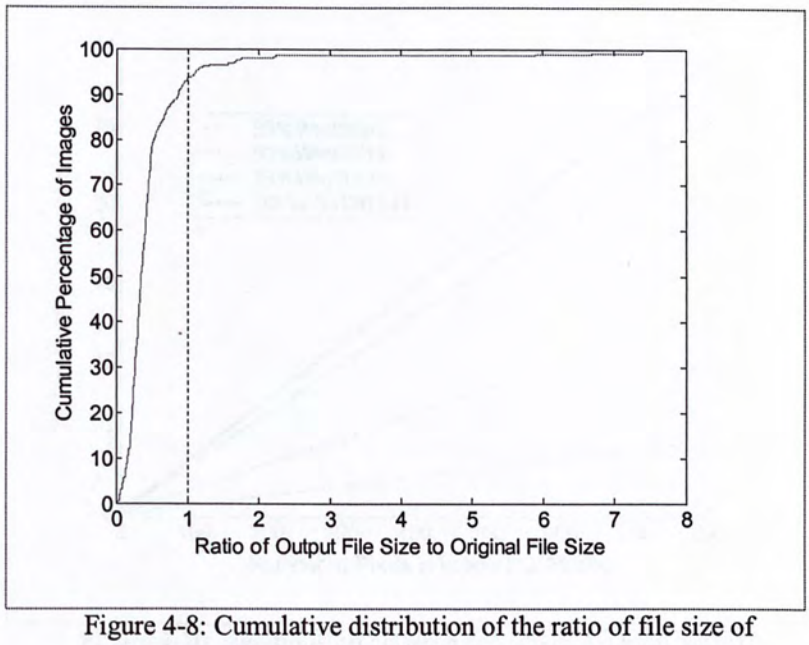

Figure 4-8: Cumulative distribution of the ratio of file size of output WBMP to file size of original image

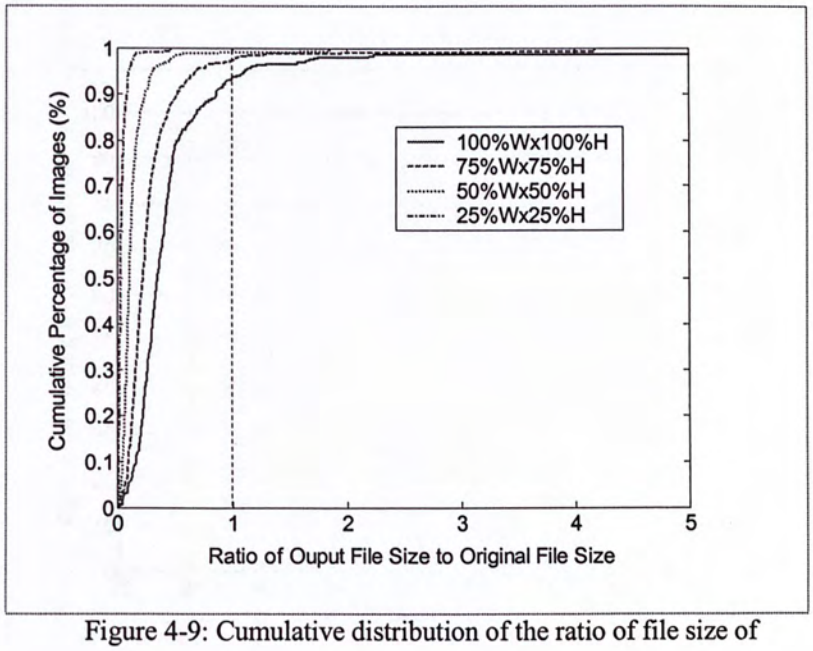

Figure 4-9: Cumulative distribution of the ratio of file size of output WBMP to file size of original image with scaling

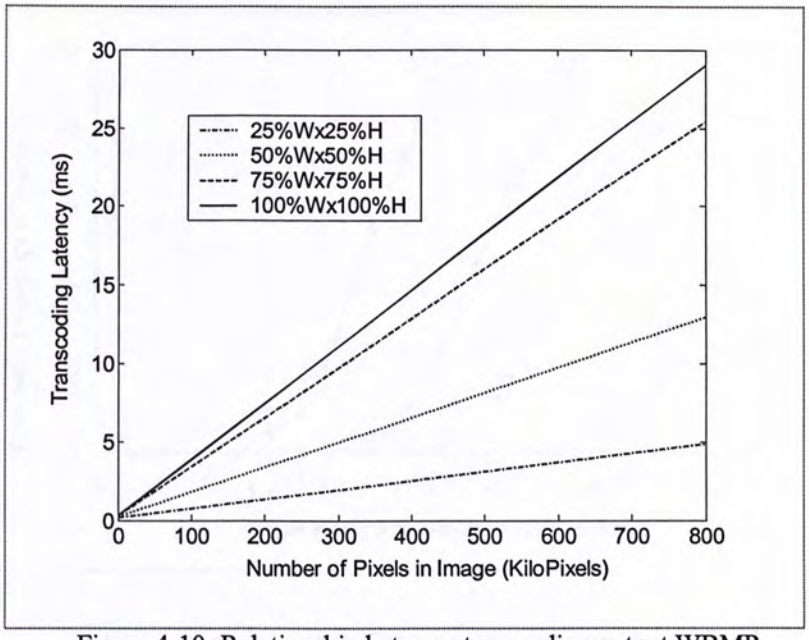

Figure 4-10: Relationship between transcoding output WBMP file size and number of pixels in the original image with scaling

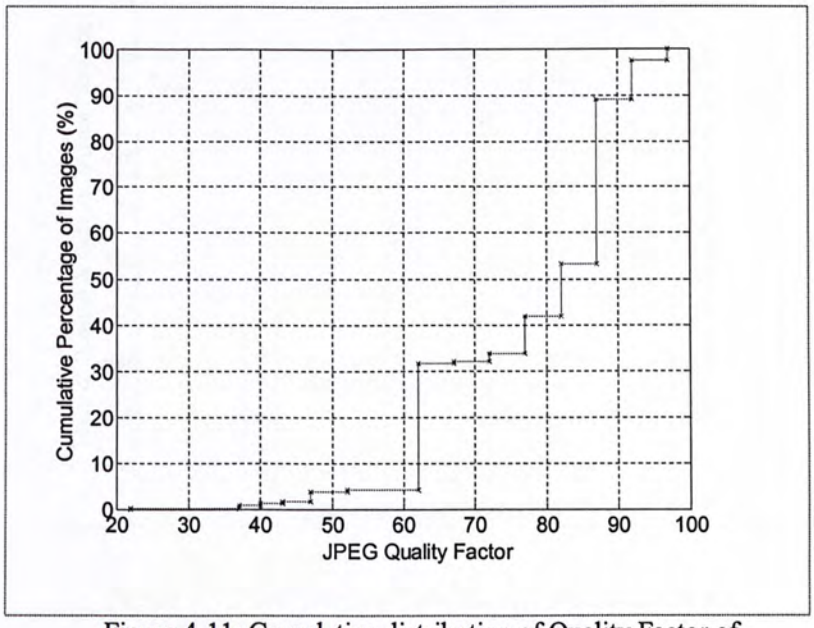

Figure 4-11: Cumulative distribution of Quality Factor of JPEG images

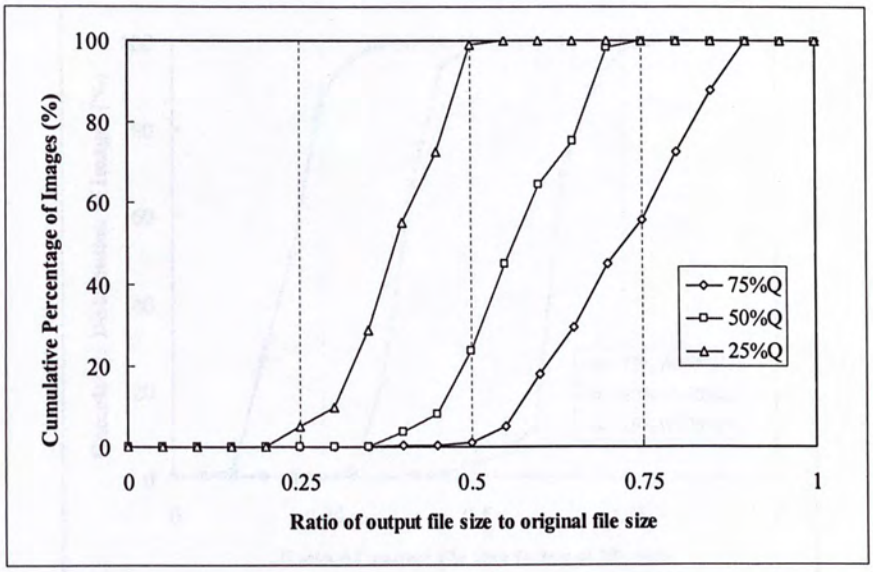

Figure 4-12: Cumulative distribution of Quality Factor of JPEG images for reducing Quality Factor

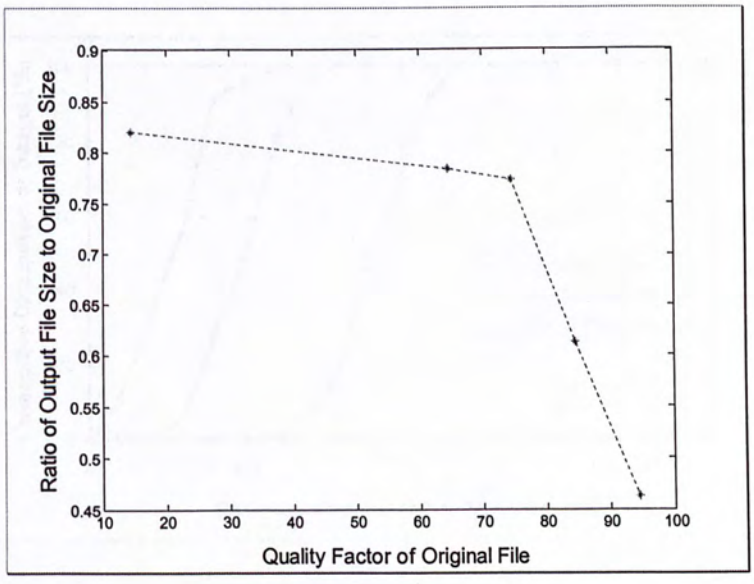

Figure 4-13: Relationship between compression ratio and quality factor of original JPEG file

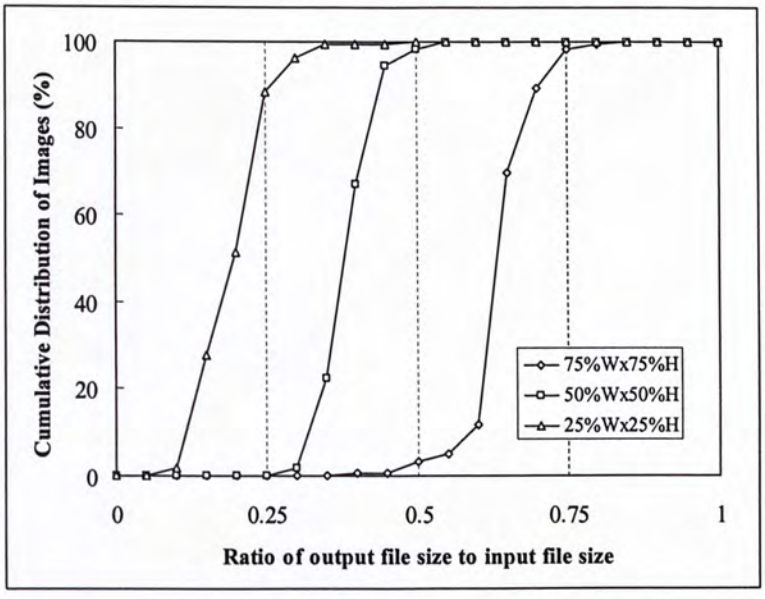

Figure 4-14: Cumulative distribution of ratio of output file size to input file size for resizing the JPEG

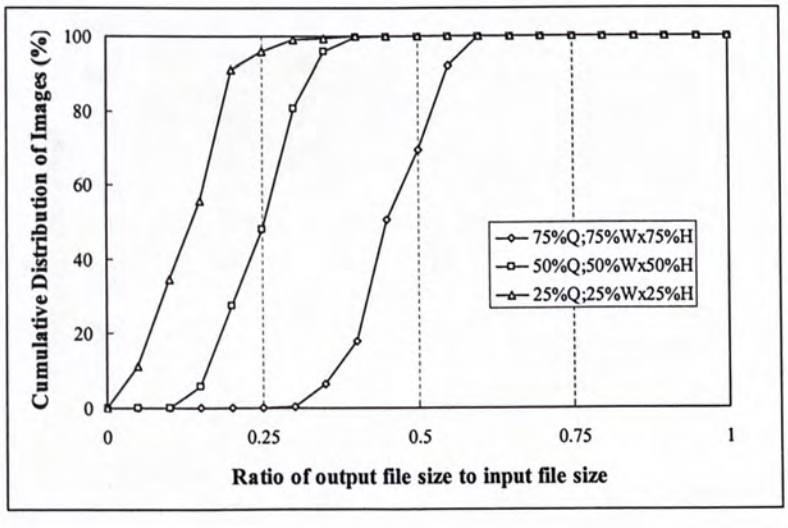

Figure 4-15: Cumulative distribution of ratio of output file size to input file size for both reducing Quality Factor and resizing the JPEG

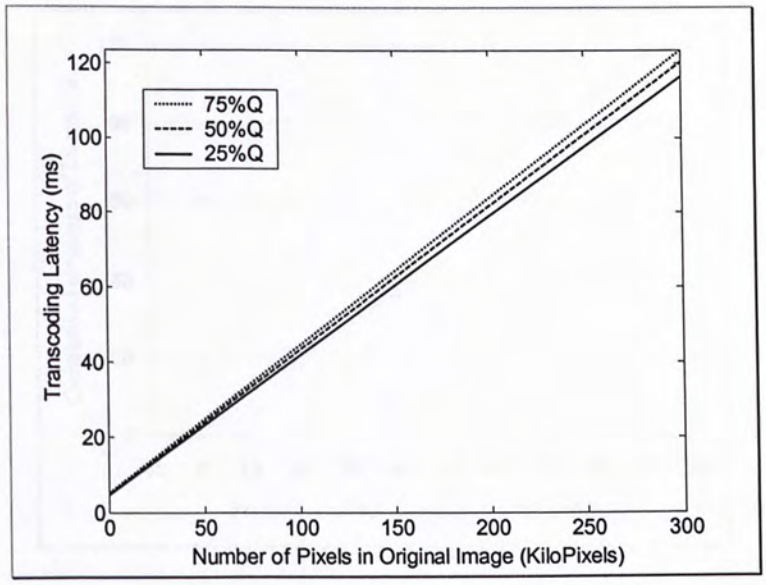

Figure 4-16: Relationship between transcoding latency and number of pixels in original image for reducing Quality Factor

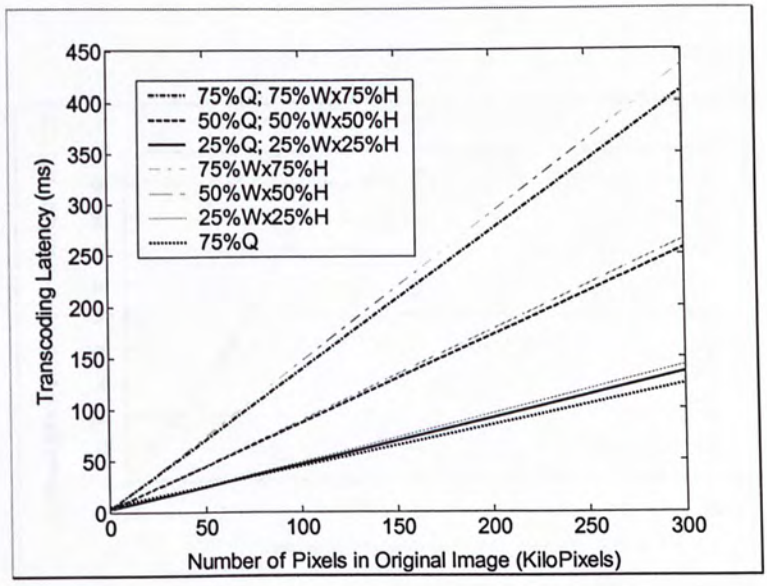

Figure 4-17: Relationship between transcoding latency and number of pixels in original image for different transcoding latency

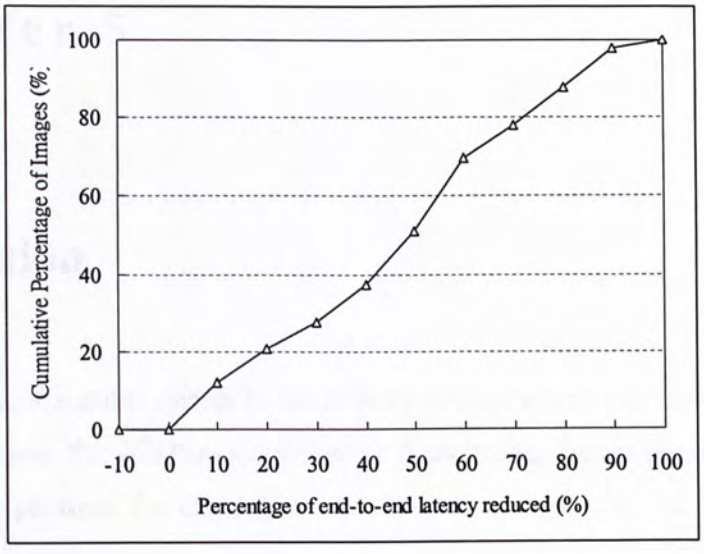

Figure 4-18: Cumulative distribution of percentage of end-to-end latency reduced for transforming JPEG to WBMP format and reducing image geometries by 25% of both image height and width.

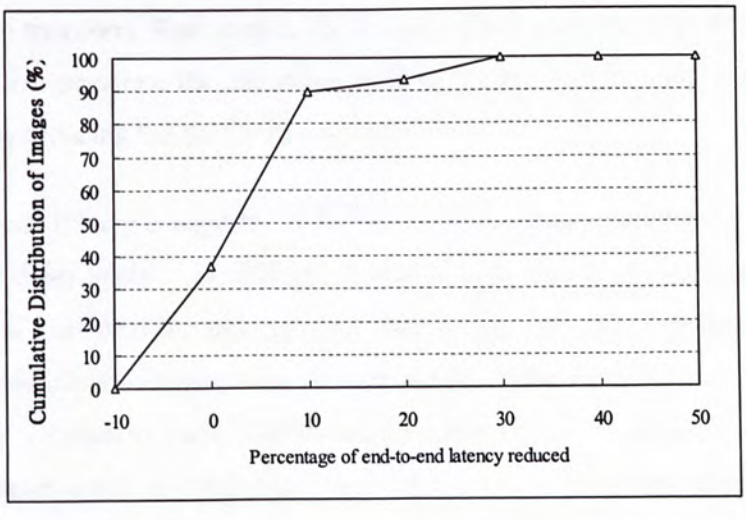

Figure 4-19: Cumulative distribution of percentage of end-to-end latency reduced for reducing JPEG image geometries by 25% of both image height and width.

Chapter 5

Conclusion

The provision of scalable system in the delivery of edge services is of great interest to service providers. The *ICAP-enabled Content Repurposing System* provides a flexible and scalable platform for distribution of application servers at the network edge through Web proxy.

The system provides real-time adaptation services for various types of multimedia content, including text, image, audio and video. The content repurposing system aims to provide adaptation on two dimensions, *content reformatting* and *content compression* for different application cases. For *content reformatting*, the transcoder transforms multimedia content to a format that can be supported by the device. For example, we transform Web content for devices which supports only WAP platform. For *content compression*, the transcoder reduces the end-to-end latency perceived by the clients by reducing the quality of content.

The end-to-end latency is regarded as the metric of evaluating the system performance. We provide delay analysis of different transcoding policies for *rule engine* to control the extent of compression and types of transcoding that are performed by *ICAP-enabled applications*. We categorize the application cases into four classes and provide a case study for each of them. The transcoding system could lengthen the end-to-end latency in some cases at which high bandwidth is available at both client and server sides. For the case that there is not sufficient bandwidth for the client to receive high-quality multimedia content, the system reduces the content size or bit-rate and the end-user enjoys a faster and smoother presentation of multimedia content.

Bibliography

[1] Network Appliance, "Internet Content Adaptation Protocol (ICAP)", *ICAP White Paper, Version 1.01,* July, 2001.

[2] ICAP Forum, http://www.i-cap.org/

[3] Zhijun Lei, Nicolas D. Georganas, "Context-based media adaptation for pervasive computing", *Proceedings of Canadian Conference on Electrical and Computer Engineering (CCECE 2001),* Toronto, May 2001.

[4] R. Fielding, UC Irvine, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, MIT/LCS, "Hypertext Transfer Protocol -- HTTP/1.1 (RFC: 2068)", Network Working Group, http://www.w3.org/Protocols/rfc2068/rfc2068, January 1997.

[5] Composite Capabilities/Preferences Profile, http://www.w3.org/Mobile/CCPP/

[6] Wai Yip Lum; Lau, F.C.M., "A context-aware decision engine for content adaptation", *IEEE Pervasive Computing*, Volume: 1 Issue: 3 , July-Sept. 2002, Page(s): 41 -49.

[7] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Frystyk Nielsen, H., Thatte, S. and D. Winer, "Simple Object Access Protocol (SOAP) 1.1", W3C Note http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, May 2000.

[8] J. Elson, A. Cerpa, UCLA, "Internet Content Adaptation Protocol (RFC: 3507)", Network Working Group, http://www.faqs.org/rfcs/rfc3507.html, April 2003.

[9] WAP Forum, http://www.wapforum.org/

[10] Press release from WAP Forum, http://www.wapforum.org/new/20000821149WAP.htm

[11] ImageMagick, http://www.imagemagick.org/

[12] CCITT Recommendation T.81, "Digital compression and coding of continuous-tone still images – requirements and guidelines", International Telecommunication Union (ITU), Geneva. Sep 1992.

[13] Independent JPEG Group, http://www.ijg.org/

[14] Surendar Chandra, Carla Schlatter Ellis, "JPEG compression metric as a quality aware image transcoding", USENIX Symposium on Internet Technologies and Systems 1999, October 1999.

[15] Quality Aware Transcoding, http://www.cse.nd.edu/~csesys/qat/qat.shtml.

[16] A. Fox, E. A. Brewer, "Reducing www latency and bandwidth requirements via real-time distillation", *Proceedings of Fifth International World Wide Web Conference*, pages 1445-1446, Paris, France, May 1996.

[17] R. Han, P Bhagwat, R. LaMaire, T. Mummert, V. Perret, J. Rubas, IBM T.J. Watson Research Centre, "Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing", *IEEE Personal Communications*, pp. 8-17, December 1998.

[18] The Lame Project, http://lame.sourceforge.net/

[19] L. S. Lam, "Design and Implementation of Video Transcoding System", *Final Year Project Report*, Department of Information Engineering, The Chinese University of Hong Kong, May 2002.

[20] Nielsen's netratings, http://www.nielsen-netratings.com/

[21] Moving Picture Experts Group (MPEG), http://mpeg.telecomitalialab.com/

[22] Fraunhofer Institute, http://www.iis.fraunhofer.de/amm/

[23] Zhijun Lei and Nicholas D. Georganas, "Context-based Media Adaptation in Pervasive Computing", *Proceedings of Canadian Conference on Electrical and Computer Engineering*, CCECE 2001, 2001.