A DEVELOPMENT OF SCALING CONTROL MECHANISM FOR FASTER ACCESSING WORLD WIDE WEB SYSTEMS

SUPARERK PISUCHPEN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE (TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT)
FACULTY OF GRADUTE STUDIES
MAHIDOL UNIVERSITY
2000

ISBN 974-664-397-5

Thesis
entitled

A DEVELOPMENT OF SCALING CONTROL MECHANISM FOR FASTER ACCESSING WORLD WIDE WEB SYSTEMS

P. Suparerk
.....
Mr.Suparerk Pisuchpen
Candidate

.....
Dr.Thanatorn Uan-on, D.Engr.
Major-advisor

.....
Assoc.Prof.Wanchai Rivepiboon, Ph.D.
Co-advisor

.....
Mr.Chumchok Namsrisakurat, M.S.
Co-advisor

.....
Prof.Liangchai Limlomwongse, Ph.D.
Dean
Faculty of Graduate Studies

.....
Dr.Thanatorn Uan-on, D.Engr.
Chairman
Master of Science Programme
in Technology of Information System
Management
Faculty of Engineering

Thesis
entitled

A DEVELOPMENT OF SCALING CONTROL MECHANISM FOR FASTER ACCESSING WORLD WIDE WEB SYSTEMS

was submitted to the Faculty of Graduate Studies, Mahidol University for the degree of Master of Science (Technology of Information System Management)
on
June 9, 2000

P. Suparerk
.....
Mr.Suparerk Pisuchpen
Candidate

.....
Dr.Thanatorn Uan-on, D.Engr
Chairman

R. Wanchai
.....
Assoc.Prof.Wanchai Rivepiboon, Ph.D.
Member

.....
Mr.Unnop Viriyavit, B.Econ
Member

.....
Mr.Chumchok Namsrisakurat, M.S.
Member

.....
Prof.Liangchai Limlomwongse, Ph.D.
Dean
Faculty of Graduate Studies
Mahidol University

.....
Dr.Thanatorn Uan-on, D.Engr
Dean
Faculty of Engineering
Mahidol University

ACKNOWLEDGEMENT

First of all, I would like to express my sincere gratitude and appreciation to the member of the supervisory committee, Dr.Thanatorn Uan-on, Assoc.Prof.Dr.Wanchai Rivepiboon, Mr.Chumchok Namsrisakurat, guidance and valuable suggestions which providing me broader perspective and help me in refining output.

I would like to thank all my friends for their support and wonderful friendship.

Finally, I would like to acknowledge my appreciation and gratitude to my family for their continuous support and love through my life.

Suparerk Pisuchpen

4037692 EGTI/M : MAJOR : TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT;M Sc. (TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT)
KEY WORDS : PROXY SERVER / WORLD WIDE WEB / SCALING

SUPARERK PISUCHPEN: A DEVELOPMENT OF SCALING CONTROL MECHANISM FOR FASTER ACCESSING WORLD WIDE WEB SYSTEMS. THESIS ADVISORS: THANATORN UAN-ON D.ENGR., WANCHAI RIVEPIBOON Ph.D., CHUMCHOK NAMSRISAKURAT M.S., 110 P. ISBN 974-664-397-5

The objective of this research was to develop a program, which accelerates access time for searching and viewing image information on the WORLD WIDE WEB. The program can help people to quickly search and view image information on the WORLD WIDE WEB by reducing the quality of images (JPEG) and keeping them in cache.

This program consists of 5 parts. The first part is the main control part, which performs a setup system, waiting client connection, connection to client socket, and controlling the number of connections. The second part is the http-header handle part, which performs: getting a requested message, intercepting a requested message, verifying a message, deciding to modify a message, and deciding to get an image from the cache control part or web reader part. The third part is the cache control part, which performs creating image index and keeping information of images, searching image filename from cache image index, and getting an image from the cache. The fourth part is the web reader part, which performs creating a socket connection to a destination (web server or another proxy server), sending a requested message to a destination, getting a response message from destination, verifying a response message, deciding to convert quality and cache JPEG images, sending JPEG images to client, and controlling thread queue to use quality convert program. The fifth part is the image quality convert part. The part performs reading JPEG image from the web server or proxy server, verifying JPEG image size, deciding to convert image quality, calling quality convert program, sending image to client and write it to file.

The program reduces used time for searching image information such as knowledge base of herbage, animals, astronomy, etc. User can search image information on the WORLD WIDE WEB by primary method and easy to get original image.

4037692 EGTI/M : สาขาวิชา : สาขาเทคโนโลยีการจัดการระบบสารสนเทศ; วท.ม.
(เทคโนโลยีการจัดการระบบสารสนเทศ)

ศุภฤกษ์ ภิสัชเพ็ญ : การพัฒนาระบบควบคุมการย่อส่วนสำหรับ WORLD WIDE WEB (A DEVELOPMENT OF SCALING CONTROL MECHANISM FOR FASTER ACCESSING WORLD WIDE WEB SYSTEMS). คณะกรรมการควบคุมวิทยานิพนธ์ : ธนากร อ้วนอ่อน D.Engr, วันชัย ริ้วไพบูลย์ Ph.D., ชุมโชค นำศรีสกุลรัตน์ M.S., 110 หน้า. ISBN 974-664-397-5

ในวิทยานิพนธ์ฉบับนี้จะเป็นการศึกษาโปรแกรมสำหรับเร่งเวลาการเข้าถึงข้อมูลชนิดภาพบน WORLD WIDE WEB โปรแกรมนี้สามารถช่วยในการค้นหาข้อมูลบน WORLD WIDE WEB ได้อย่างรวดเร็ว โดยการลด quality ของภาพ (JPEG) และเก็บไว้ใน cache

โปรแกรมนี้ประกอบด้วย 5 ส่วน ส่วนที่ 1 เป็นส่วนควบคุมหลักทำหน้าที่ตั้งค่าเริ่มต้นให้ระบบ, คอยการเชื่อมโยงจาก client, เชื่อมโยงกับ client socket, ควบคุมจำนวนของการเชื่อมโยง. ส่วนที่ 2 เป็นส่วนจัดการ http-header ทำหน้าที่รับและจับข้อความร้องขอ, ตรวจสอบข้อความร้องขอ, ปรับปรุงข้อความร้องขอ, รับไฟล์ภาพจากส่วนควบคุม cache หรือ ส่วนอ่านข้อมูลจาก web. ส่วนที่ 3 เป็นส่วนควบคุม cache ทำหน้าที่สร้างดัชนีภาพเก็บข้อมูลของภาพ, ค้นหาชื่อไฟล์ภาพจากดัชนีภาพ, รับไฟล์ภาพจาก cache. ส่วนที่ 4 เป็นส่วนอ่านข้อมูลจาก web ทำหน้าที่สร้าง socket การเชื่อมโยงไปยังเป้าหมาย , ส่งข้อความร้องขอไปที่เป้าหมาย, รับข้อความตอบรับจากเป้าหมาย, ตรวจสอบข้อความตอบรับ, เลือกอ่านข้อมูล, แปลง quality และเก็บลง cache, ส่งข้อมูลไปให้ client, ควบคุมแถวคอยของ thread เพื่อใช้โปรแกรมแปลง quality ของภาพ. ส่วนที่ 5 เป็นส่วนเรียกโปรแกรมแปลง quality ของภาพ ทำหน้าที่รับข้อมูลภาพ (JPEG) จาก web server หรือ proxy server, ตรวจสอบขนาดของภาพ (JPEG), เลือกที่จะแปลง quality ของภาพ, เรียกโปรแกรมแปลง quality ของภาพ, ส่งข้อมูลภาพไปที่ client

โปรแกรมนี้ช่วยลดเวลาในการค้นหาข้อมูลภาพเช่น knowledge base พืชสมุนไพร, สัตว์ชนิดต่าง ๆ, ดาราศาสตร์ โดยใช้การค้นหาข้อมูลแบบเดิมและง่ายในการภาพดั้งเดิม

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Background and Problem

Almost HyperText Markup Language pages contain many image objects. One investigation by crawler robots[9] reports that more than 50% of pages contain at least one image. Although image objects are useful for visual data communication and media-rich contents are fun for users, the transfer time for such objects from the server to the browser is longer. This may lead to the user being dissatisfied by the long response time even if he is not in a hurry.

On the other hand, images can be reduced with a downgrade in quality. For instance, JPEG image can be scaled down in quality in order to reduce their size. The scaled down objects are useful for preview functions. These scaled down objects are useful for quick search of information. Applying image scaling to WWW systems. If we scale down same image every searching, we lose many times. This problem can be solved by keeps it in file (cache) same as proxy function therefore, we will assemble proxy server and scaling down JPEG image.

This study proposes assembly a scaling control mechanism and proxy server that provides useful and careful interactive user interfaces in order to make the most of media scaling effects. WWW latency[1] can be reduced by scale down large-sized of

image objects. Our goal is to reduce WWW latency[1], and used time to get image information.

1.2 Objectives

The objectives of this study are :

1.2.1 To develop a program accelerate access time of HTML page that contain many JPEG[8] image objects

1.2.2 To apply image scaling methods to reduce size of the image

1.3 Scope of work

1.3.1 Scope of programs that will development

1.3.1.1 To control program for intercept messages.

1.3.1.2 Scaling control program to control and manage thread queue to call scaling JPEG images program and sends them to client.

1.3.2 Scope of testing

1.3.2.1 To measure times to preview original JPEG images and HTML pages, that contain original JPEG images.

1.3.2.2 To measure times to preview original JPEG images and HTML pages, that contain original JPEG images, using the scaling control program.

1.4 Expected Results

1.4.1 The model of scale down proxy server, that can be determine to reduce size of appropriate JPEG images and keeps it in cache for next request.

1.4.2 Preview images and searches images information, although information source contain many JPEG image objects.

1.5 Definitions of Terms

HTML : (HyperText Markup Language) HTML is $SGML_5$ DTD_5. In practical terms, HTML is a collection of styles (indicated by markup tags) that define the various components of a World Wide Web document.

$SGML_5$: (Standard Generalized Markup Language) This is a standard for describing markup languages.

DTD_5 : (Document Type Definition) This is a specific markup language, written using $SGML_5$.

JPEG : (pronounced "jay-peg") This is a type of image format. JPEG is a standardized compression method for full-color and gray-scale images.

Server : Server is a host computer.

Browser : This is a application program, which is used to send and receive message and receive image file and $HTML_5$ page.

WWW : (World Wide Web or Web, for short) One type of system run on internet use HTTP protocol for connection.

HTTP : (HyperText Transfer Protocol) This is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods.

Proxy server : An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, with possible translation, to other servers. A proxy must implement both the client and server requirements of this specification. This internet application uses cache to keep data and get it very fast.

Cache : This is a storage. It uses to keep data and create index for searching data in cache.

CHAPTER II

LITERATURE REVIEW

This chapter presents literatures, those are relative with this study. Those literatures show the advantage and disadvantage of existing system and they can help to design system of this study.

2.1 System model of Proxy server

Proxy server model has many functions, but in this study, we use proxy for verify request message, response message and cache control. Proxy server can cache many objects for fast access in the next request time. Usually, If not necessary, $Proxy_2$ will not modify request message and response message. Proxy server will pass request message to web server, pass response message and object to browser. $Proxy_2$ server can cache every object. But this proxy server can cache JPEG image only. $Proxy_2$ server can run on 2 situations. The first situation is normal situation, which has one proxy server. The second situation has more than one proxy server, therefore proxy server will send request message to another proxy server, it doesn't get object by itself.

2.1.1 Normal situation

This situation is shown in figure 2.1. Proxy server is between WWW server, and browser.

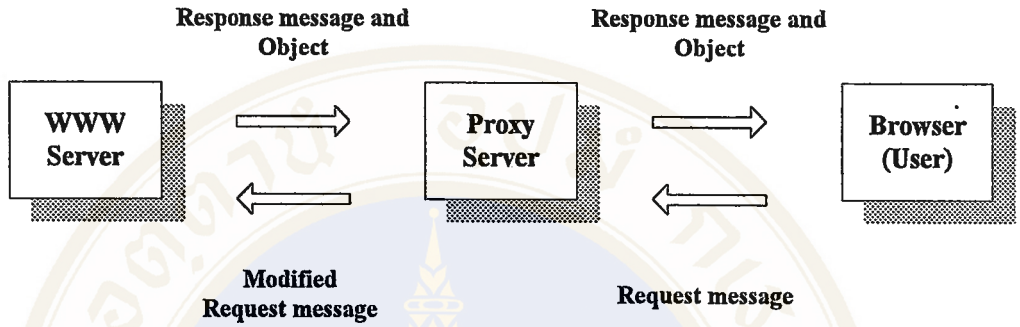


Figure 2.1 A system model in normal situation

In operation, client browser sends request message to proxy server, then proxy server verifies and modifies request message. After that, proxy server sends modified message to WWW server and waits for response message and object, if proxy server receives response message and object. Proxy server will cache object and send object to browser.

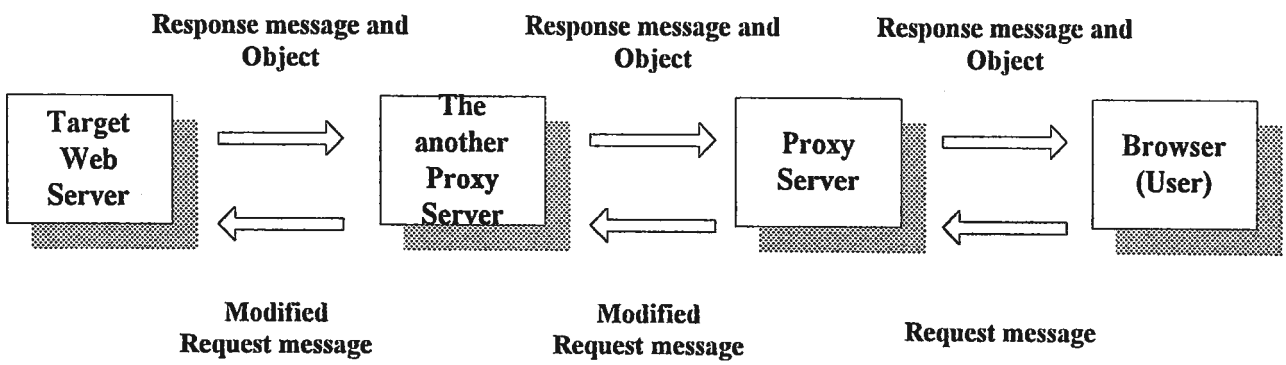


Figure 2.2 A system model in Connected the another Proxy server situation

2.1.2 Connected the another Proxy server situation

This situation is shown in figure 2.2. Proxy server is between the another proxy server, and browser.

In operation, client browser sends request message to proxy server, then proxy server verifies and modifies request message. After that proxy server sends modified message to the another proxy server and waits for response message and objects, if proxy server receives response message and objects. Proxy server will store object to cache, and send objects to browser same as operation in normal situation. But this situation, proxy server doesn't get object from target web server, the another proxy server gets object from target web server and sends it to proxy server.

2.2 Existing systems

Let us review some traditional researchs. $Pythia_1$ is an HTTP proxy server that scales down the JPEG, GIF, and Postscript objects at Internet telephone access points. $Pythia_1$ significantly reduces WWW transfer time by downgrading the objects on demand. However, $Pythia_1$ has the following limitation in usability :

2.2.1 $Pythia_1$ scales down the inline images of pages, and provides anchors linked to the originals automatically. Although there are many objects that are not embedded in the pages but accessible by anchors, no user interface is provided to control scaling these objects.

2.2.2 Pythia$_1$ enables the user to access the original of the reduced inline images with each extra anchor around the reduced images. However, the user must change the proxy setting in order to get the entire original page. Regarding a page as important, a user wishes to get the original page by a simple operation.

2.2.3 Since Pythia$_1$ decides the scaling and its parameter at the time of the object transfer, it is difficult for a user to judge whether the received object is the original or not. If a user takes the original for the scaled down, he may access the same object again expecting the detail. Then he may be disappointed with the result. Additionally the bounding box dimensions of the images may be changed to unexpected size, causing coordinates mismatch of image maps.

Another research project, OreO$_2$, proposes a general notion of application-specific processing of proxy servers and describes media scaling as an example. However, there are few specific proposals for interactive control of the media scaling

2.3 Media Scaling Server (MSS)$_3$

2.3.1 Requirements of Media Scaling Server

The requirements of MSS$_3$ are listed as follows :

2.3.1.1 Interactive selection of the scaling parameter and accessing the original by simple operations. MSS$_3$ should enable users to select receiving either the scaled down objects or the original according to their situation. For instance, when accessing WWW server containing large image objects, a user might first wish to

survey all of images in order to select some of them. Then he might want to check the details of these images easily.

2.3.1.2 Confusion prevention by distinguishable representation and operation compatibility.

To prevent media scaling from confusing users, MSS_3 is required to represent scaled down objects as "scaled down" and to maintain the original operation as much as possible. For instance, a reduced image map should be represented as "shrunk" and mapped to the proper objects as well as to the originals.

2.3.2 Desired user interface of image scaling

This study focuses on the shrinking of images as a scaling example. According to the requirements, desired user interface of MSS_3 is designed as follows :

2.3.2.1 Automatic scaling of inline images. Since a user can not detect the existence of inline images in advance, MSS_3 should scale these inline images down automatically to provide quick page navigation.

2.3.2.2 Choice of scaled down images from the images linked by anchors. On "well built" pages, anchors for image objects are often designed as thumbnail images of the original. Although these thumbnails are useful for searching, not all authors design them. MSS_3 should allow the user to preview these anchor-linked images on pages by providing the choice of scaled down images from the original.

2.3.2.3 Easy access to the original pages and images. Sometime, a user wishes to access the original objects in order to check details. MSS_3 should enable browser users to access the original by simple operation.

2.3.2.4 Signify the originality of pages and images. MSS_3 should inform or suggest to the user whether the object is the original or not.

2.4 Virtual URL-based scaling control mechanism

We propose Virtual URL based Scaling Control Mechanism ($VSCM_3$) for MSS_3 to provide the desired user interface.

2.4.1 Mechanism

$VSCM_3$ controls scaling by Virtual URL and four elemental processings. MSS_3 should work with a variety of browsers when it is located at a server site and with all servers used globally when located at an Internet intermediary point. Therefore, $VSCM_3$ uses only HTTP 1.0 and HTML 2.0.

2.4.2 Virtual URL

For the interactivity, a browser needs to pass scaling information to MSS_3 with every user access, where scaling information determines whether scaling will occur or not, and if scaling is to occur, the method and parameter of the scaling. Since the scaling information would be changed by every HTTP connection, it is rational to embed scaling information in the HTTP request.

We define Virtual URL (VURL), which is extended URL containing the scaling information for MSS_3. URL is extended to $VURL_3$ by inserting the command extension to the left of its extension (.html, .gif, etc.) as shown in figure 2.3

The command extension is written as "_ce(method, parameter)" where the method and parameter indicate the scaling method and its parameter at MSS_3. For example, http://www…/car_ce(zoom, 0.5).gif is the virtual address of the object scaled down to one-half length along each dimension by MSS_3> For simplicity. a $VURL_3$ based on a URL is described as VURL(URL, method, parameter) and the example $VURL_3$ is written as VRUL (http://www…/car.gif, zoom, 0.5). MSS_3 detects VURL in the request sent by browsers and extracts scaling information and stores it so that MSS_3 will scale down the object according to the information. Scaling information is not only used for media scaling, but also for other $VSCM_3$ processing.

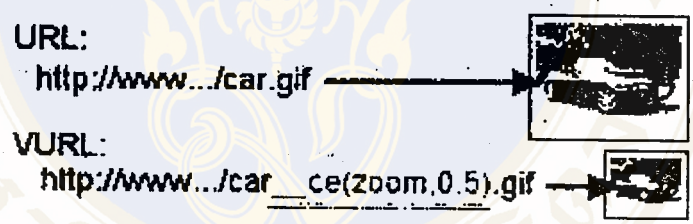


Figure 2.3 An example of VURL

2.4.3 Elemental processings

These processing are work with their rules. Illustrative rules are defined at the next subsection.

2.4.3.1 URL rectification : When MSS_3 receives a request containing VURL (VURL request) from the browser, MSS_3 rectifies the $VRUL_3$ to URL and store the scaling information of the $VURL_3$ for another processing later. This is a basic rule

of URL rectification. Additionally, MSS_3 modifies other part of $VURL_3$ (e.g., coordinates on the imagemap).

2.4.3.2 HTML transposition : For the applicability, browsers are expected to send $VURL_3$ requests without their additional functions. In order for browsers to send $VURL_3$ on HTTP requests, MSS_3 transposes HTML text sent from the servers by embedding VURLs in the text.

2.4.3.3 HTML composition : MSS_3 composes extra HTML text containing VURLs to provide extra representation of the object to improve the usability. For instance, a scaled down image can be represented as a thumbnail anchor linked to the original (shown in next subsection).

2.4.3.4 Media scaling : MSS_3 scales down the object sent from the server according to the scaling information stored by URL rectification.

2.4.4 A typical processing of MSS_3

Figure 2.4 shows a typical processing of MSS_3 after a browser sends a request that contains VURL (an example is in brackets). MSS_3 processes the request as follows

2.4.4.1 Receives $VURL_3$ request [GET http://…/car_si(zoom, 0.5).gif] from browser,

2.4.4.2 URL rectification : Extracts scaling information from $VURL_3$ and stores it [(zoom, 0.5)], and rectifies $VURL_3$ to the original URL [http://…/car.gif] removing the command extension,

2.4.4.3 Sends request [GET http://…/car.gif] which contains the original URL to the server,

2.4.4.4 Receives content object from the server,

2.4.4.5 Media scaling : Scales down the object according to the scaling information stored in step 2 (scale down the resolution of the image to 50 %),

2.4.4.6 Sends the scaled down object to the browser.

By this sequence, MSS_3 scales down the objects sent from the server to the browsers according to the scaling information from browsers. Since MSS_3 sends the URL request, MSS is accessible for any WWW servers.

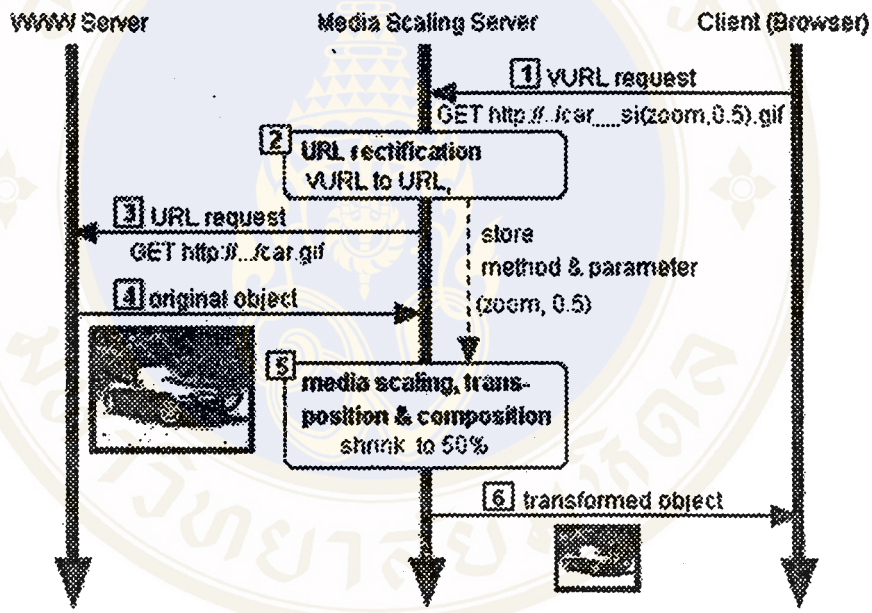


Figure 2.4 A typical sequence of the VURL request.

2.4.5 Fixed media scaling parameter for scaling detection

Although optimization of scaling parameter on demand can bound the latency, it is difficult for MSS to detect the object scaling in advance. Thus, the scaling parameter is fixed at either 50% or 25%. The method for image scaling is "zoom" and the parameter is either "0.5" or "0.25".

2.4.6 Location of the command extension

There are two factors necessary for $VURL_3$ specification: location and format of the command extension. We have settled on the former, while not the latter. The location may be selected from the following:

2.4.6.1 top of the URL,

2.4.6.2 end of the URL (after the extension),

2.4.6.3 end of the URL (before the extension).

If MSS_3 inserts the command extension at (2.4.6.1), the HTML transposition process becomes complex, because there are many relative URLs and MSS should transpose them to absolute URLs to insert the command extension. Next, if MSS_3 inserts the command extension at (2.4.6.2), it may cause an error for another module on WWW that is checking extension of URL. For example, because MSS_3 selects URL to transpose by checking its extension, it would he hard to detect extensions of URLs when MSS_3 is cascaded. To avoid these problems, we locate the command extension at 2.4.6.3) and the extension is not hidden.

2.5 HTTP protocol operation

HTTP protocol has been in use by the World Wide Web global information initiative since 1990. In this model system operate with World Wide Web, it will relative with HTTP protocol. This model system will wait for connection and receives HTTP-Header request. Then, verify HTTP-Header and determine to modify it. After that, may be send to Web server or another Proxy server or Cache control part.

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. The first version of HTTP, referred to as HTTP/0.9, was a simple protocol for raw data transfer across the Internet. HTTP/1.0 improved the protocol by allowing messages to be in the format of MIME-like messages, containing metainformation about the data transferred and modifiers on the request/response semantics. However, HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, and virtual hosts. In addition, the proliferation of incompletely implemented applications calling themselves "HTTP/1.0" has necessitated a protocol version change in order for two communicating applications to determine each other's true capabilities. This specification defines the protocol referred to as "HTTP/1.1". This protocol includes more stringent requirements than HTTP/1.0 in order to ensure reliable implementation of its features. Practical information systems require more functionality than simple retrieval, including search, front-end update, and annotation. HTTP allows an open-ended set of methods that indicate the purpose of a request. It builds on the discipline of reference provided by the Uniform Resource Identifier (URI), as a location (URL) or name (URN_6), for indicating the resource to which a method is to be applied. Messages are passed in a format similar to that used by Internet mail as defined by the Multipurpose Internet Mail Extensions ($MIME_5$).

HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet systems. In this way, HTTP allows basic hypermedia access to resources available from diverse applications.

2.5.1 Overall operation

The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible entity-body content.

Most HTTP communication is initiated by a user agent and consists of a request to be applied to a resource on some origin server. In the simplest case as shown in figure 2.5, this may be accomplished via a single connection (v) between the user agent (UA) and the origin server (O).

A more complicated situation occurs when one or more intermediaries are present in the request/response chain. There are three common forms of intermediary: proxy, gateway, and tunnel. A proxy is a forwarding agent, receiving requests for a URI in its absolute form, rewriting all or part of the message, and forwarding the reformatted request toward the server identified by the URI.

A gateway is a receiving agent, acting as a layer above some other server(s) and, if necessary, translating the requests to the underlying server's protocol. A tunnel acts as a relay point between two connections without changing the messages, tunnels are used when the communication needs to pass through an intermediary (such as a firewall) even when the intermediary cannot understand the contents of the messages.

The figure 2.6 shows three intermediaries (A, B, and C) between the user agent and origin server. A request or response message that travels the whole chain will pass through four separate connections.

This distinction is important because some HTTP communication options may apply only to the connection with the nearest, non-tunnel neighbor, only to the end-points of the chain, or to all connections along the chain. Although the diagram is linear, each participant may be engaged in multiple, simultaneous communications.

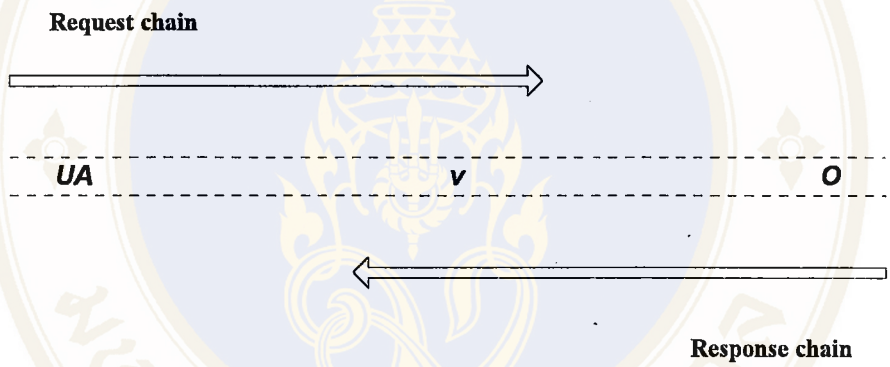


Figure 2.5 The simplest case of HTTP communication

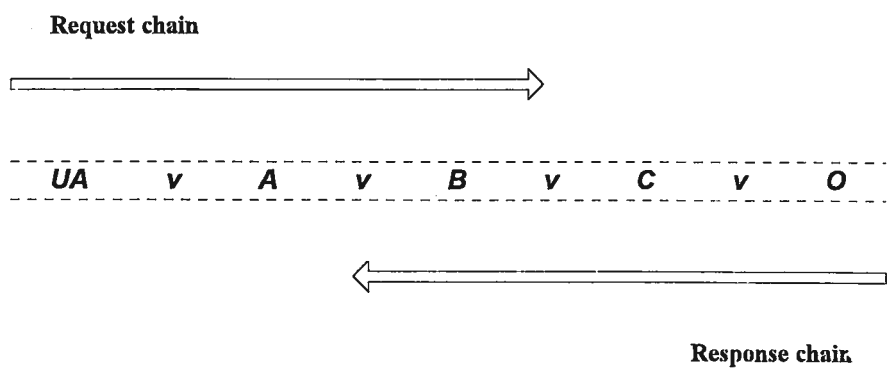


Figure 2.6 The case of HTTP communication with three intermediaries (A, B, C)

For example, B may be receiving requests from many clients other than A, and/or forwarding requests to servers other than C, at the same time that it is handling A's request.

Any parties to the communication, which is not acting as a tunnel may employ an internal cache for handling requests as shown in figure 2.7. The effect of cache is that, the request/response chain is shortened if one of the participants along the chain has a cached response applicable to that request. The following illustrates the resulting chain if B has a cached copy of an earlier response from O (via C) for a request, which has not been cached by UA or A.

Not all responses are usefully cacheable, and some requests may contain modifiers, which place special requirements on cache behavior.

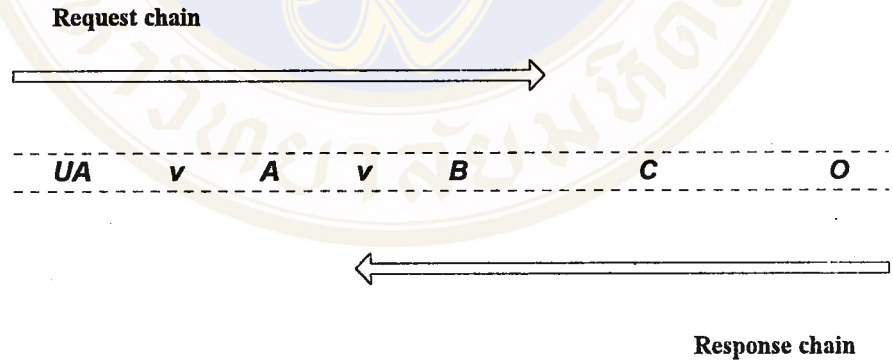


Figure 2.7 Effect of cache to the case of HTTP communication with three intermediaries (A, B, C)

HTTP communication usually takes place over TCP/IP connections. The default port is TCP 80, but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks.

HTTP only presumes a reliable transport, any protocol that provides such guarantees can be used; the mapping of the HTTP/1.1 request and response structures onto the transport data units of the protocol in question is outside the scope of this specification.

2.5.2 Protocol parameters

In HTTP-Header has many parameters. We will show specially important parameters, that relative with this study as follow :

2.5.2.1 HTTP version

HTTP uses a "<major>.<minor>" numbering scheme to indicate versions of the protocol. The protocol versioning policy is intended to allow the sender to indicate the format of a message and its capacity for understanding further HTTP communication, rather than the features obtained via that communication. No change is made to the version number for the addition of message components which do not affect communication behavior or which only add to extensible field values. The <minor> number is incremented when the changes made to the protocol add features, which do not change the general message parsing algorithm, but which may add to the message semantics and imply additional capabilities of the sender. The <major> number is incremented when the format of a message within the protocol is changed.

The version of an HTTP message is indicated by an HTTP-Version field. It is in the first line of the message. The scheme-specific syntax for HTTP-Version field is

"HTTP" "/" <major> "." <minor>

The examples of HTTP-Version are HTTP/1.0, HTTP/1.1, etc.

That the major and minor numbers must be treated as separate integers and that each may be incremented higher than a single digit. Thus, HTTP/2.4 is a lower version than HTTP/2.13, which in turn is lower than HTTP/12.3. Leading zeros must be ignored by recipients and must not be sent.

Applications sending Request or Response messages, as defined by this specification, must include an HTTP-Version of "HTTP/1.1". Use of this version number indicates that the sending application is at least conditionally compliant with this specification.

Proxy and gateway applications must be careful when forwarding messages in protocol versions different from that of the application. Since the protocol version indicates the protocol capability of the sender. A proxy/gateway must never send a message with a version indicator, which is greater than its actual version, if a higher version request is received, the proxy/gateway must either downgrade the request version, respond with an error, or switch to tunnel behavior. Requests with a version lower than that of the proxy/gateway's version may be upgraded before being forwarded; the proxy/gateway's response to that request must be in the same major version as the request.

2.5.2.2 Uniform Resource Identifiers

URIs have been known by many names: WWW addresses, Universal Document Identifiers, Universal Resource Identifiers , and finally the combination of Uniform Resource Locators (URL) and Names (URN). As far as HTTP is concerned, Uniform Resource Identifiers are simply formatted strings which identify--via name, location, or any other characteristic--a resource.

The "http" scheme is used to locate network resources via the HTTP protocol. The scheme-specific syntax and semantics for http URLs is

"http:" "//" host [":" port] [absolute path]

host = A legal Internet host domain name or IP address (in dotted-decimal form).

port = number , example : 80, 8080, 9666 etc

The example of http URLs is : http://www.mahidol.ac.th:80/exam.html

host = www.mahidol.ac.th

port = 80

absolute path = exam.html

If the port is empty or not given, port 80 is assumed. The semantics are that the identified resource is located at the server listening for TCP connections on that port of that host, and the Request-URI for the resource is absolute path. The use of IP addresses in URL's should be avoided whenever possible. If the absolute path is not present in the URL, it must be given as "/" when used as a Request-URI for a resource.

2.5.3 HTTP message

HTTP message has 2 types. HTTP message consist of requests from client to server (Request) and responses from server to client (Response). Both types of message consist of a start-line, header fields, an empty line (indicating the end of the header fields, and an optional message-body.

2.5.3.1 Start-line

Start-line of Request message difference from Response message. Start-line of Request message is Request-Line and Start-line of Response message is Status-Line.

2.5.3.2 General header fields

Header fields of Request message have some parts difference from Response message same as Start-line. But there are a few header fields which have general applicability for both request and response messages.

2.5.3.2.1 Cache-Control : The Cache-Control general-header field is used to specify directives that must be obeyed by all caching mechanisms along the request/response chain.

2.5.3.2.2 Connection : The Connection general-header field allows the sender to specify options that are desired for that particular connection and MUST NOT be communicated by proxies over further connections.

2.5.3.2.3 Date : The Date general-header field represents the date and time at which the message was originated.

2.5.3.2.4 Pragma : The Pragma general-header field is used

to include implementation- specific directives that may apply to any recipient along the request/response chain.

2.5.3.2.5 Transfer-Encoding : The Transfer-Encoding general-header field indicates what (if any) type of transformation has been applied to the message body in order to safely transfer it between the sender and the recipient.

2.5.3.2.6 Upgrade : The Upgrade header field is intended to provide a simple mechanism for transition from HTTP/1.1 to some other, incompatible protocol. The Upgrade allows the client to specify what additional communication protocols it supports and would like to use if the server finds it appropriate to switch protocols.

2.5.3.2.7 Via : The Via general-header field must be used by gateways and Proxies to indicate the intermediate protocols and recipients between the user agent and the server on requests, and between the origin server and the client on responses.

2.5.3.3 Message body

The message-body (if any) of an HTTP message is used to carry the entity-body associated with the request or response.

The rules for when a message-body is allowed in a message differ for requests and responses.

The presence of a message-body in a request is signaled by the inclusion of a Content-Length or Transfer-Encoding header field in the request's message-headers. A message-body may be included in a request only when the request method allows an entity-body.

For response messages, whether or not a message-body is included with a message is dependent on both the request method and the response status code. All responses to the HEAD request method must not include a message-body, even though the presence of entity- header fields might lead one to believe they do. All 1xx (informational), 204 (no content), and 304 (not modified) responses must not include a message-body. All other responses do include a message-body, although it may be of zero length

2.5.4 Request message

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

2.5.4.1 Request-Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP (space) characters. No CR or LF are allowed except in the final CRLF sequence.

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

2.5.4.1.1 Method

The Method token indicates the method to be performed on the resource identified by the Request-URI. The method is case-sensitive. The examples of method are OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, etc.

The list of methods allowed by a resource can be specified in an Allow header field. The return code of the response always notifies the client whether a method is currently allowed on a resource, since the set of allowed methods can change dynamically. Servers should return the status code 405 (Method Not Allowed) if the method is known by the server but not allowed for the requested resource, and 501 (Not Implemented) if the method is unrecognized or not implemented by the server. The list of methods known by a server can be listed in a Public response-header field.

The methods GET and HEAD must be supported by all general-purpose servers. All other methods are optional.

2.5.4.1.2 Request-URI

The Request-URI is a Uniform Resource Identifier and identifies the resource upon which to apply the request.

The absoluteURI$_5$ form is required when the request is being made to a proxy. The proxy is requested to forward the request or service it from a valid cache, and return the response. The proxy may forward the request on to another proxy or directly to the server specified by the absoluteURI$_5$. The most common form of Request-URI$_5$ is that used to identify a resource on an origin server or gateway. In this case the absolute path of the URI must be transmitted as the Request-URI$_5$, and the network location of the URI must be transmitted in a Host header field. For example, a client wishing to retrieve the resource above directly from the origin server would create a TCP connection to port 80 of the host "www.w3.org" and send the lines:

GET /pub/WWW/TheProject.html HTTP/1.1

Host: www.w3.org

followed by the remainder of the request. The absolute path cannot be empty; if none is present in the original URI, it MUST be given as "/" (the server root).

2.5.4.2 Request header fields

The request-header fields allow the client to pass additional information about the request, and about the client itself, to the server. These fields act as request modifiers, with semantics equivalent to the parameters on a programming language method invocation. The examples of Request header fields are Accept, Accept-Charset, Accept-Encoding, Accept-Language, Authorization, From, Host, If-Modified-Since, Proxy-Authorization, Referer, User-Agent, etc$_5$. Request-header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields may be given the semantics of request- header fields if all parties in the communication recognize them to be request-header fields. Unrecognized header fields are treated as entity-header fields.

2.5.5. Response message

After receiving and interpreting a request message, a server responds with an HTTP response message.

2.5.5.1 Status-Line

The first line of a Response message is the Status-Line, consisting of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by SP (space) characters. No CR or LF is allowed except in the final CRLF sequence.

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

2.5.5.2 Status-Code and Reason Phrase

The Status-Code element is a 3-digit integer result code of the attempt to understand and satisfy the request. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata and the Reason-Phrase is intended for the human user. The client is not required to examine or display the Reason- Phrase.

The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. There are 5 values for the first digit:

1xx: Informational - Request received, continuing process

2xx: Success - The action was successfully received, understood, and accepted

3xx: Redirection - Further action must be taken in order to complete the request

4xx: Client Error - The request contains bad syntax or cannot be fulfilled

5xx: Server Error - The server failed to fulfill an apparently valid request

2.5.5.3 Response header fields

The response-header fields allow the server to pass additional information about the response which cannot be placed in the Status- Line. These header fields give information about the server and about further access to the resource identified by the Request-URI$_5$. The examples of Response header fields are Age, Location, Proxy-Authenticate, Public, Retry-After, Server, Vary, Warning, .WWW-Authenticate, etc. Response-header field names can be extended reliably only in combination with a change in the protocol version. However, new or experimental header fields may be given the semantics of response- header fields if all parties in the

communication recognize them to be response-header fields. Unrecognized header fields are treated as entity-header fields.

2.5.6 HTTP Connection

Prior to persistent connections, a separate TCP connection was established to fetch each URL, increasing the load on HTTP servers and causing congestion on the Internet. The use of inline images and other associated data often requires a client to make multiple requests of the same server in a short amount of time. Analyses of these performance problems are available, analysis and results from a prototype implementation are in.

Persistent HTTP connections have a number of advantages:

- By opening and closing fewer TCP connections, CPU time is saved, and memory used for TCP protocol control blocks is also saved.

2.5.6.1 HTTP requests and responses can be pipelined on a connection. Pipelining allows a client to make multiple requests without waiting for each response, allowing a single TCP connection to be used much more efficiently, with much lower elapsed time.

2.5.6.2 Network congestion is reduced by reducing the number of packets caused by TCP opens, and by allowing TCP sufficient time to determine the congestion state of the network.

2.5.6.3 HTTP can evolve more gracefully; since errors can be reported without the penalty of closing the TCP connection. Clients using future versions of HTTP might optimistically try a new feature, but if communicating with an older

server, retry with old semantics after an error is reported. Persistent connections provide a mechanism by which a client and a server can signal the close of a TCP connection. This signaling takes place using the Connection header field. Once a close has been signaled, the client must not send any more requests on that connection.

2.6 JAVA language

Java language is a good computer language, for object oriented programming. Now, Java used to create many program especially, network program. The reasons that, we choose this computer language, because the advantages of this computer language is appropriate for this study.

2.6.1 The advantages of Java language

The advantages of Java language for this study are :

2.6.1.1 Java language is a runtime library[12] that gives it platform independence, you can use the same code on Windows 98, Solaris, Unix, Macintosh, and so on. This certainly necessary for Internet programming

2.6.1.2 Java language has a similar syntax to C++. This means C++ programmer can easy learn Java language.

2.6.1.3 Memory management, Java language has automatic memory manager. That can solve insufficient memory problem, and other problem about memory.

2.6.1.4 Java language doesn't use pointer, it makes program no complex. It reduces reference memory problem.

2.6.1.5 Characteristic of Java language has inheritance, that is appropriate characteristic for programming.

2.6.2 Objected – Oriented programming

Object – Oriented programming$_{12}$ is programming by sending message to an object of unknown type then the exact response of the object cannot be predicate. That is a powerful programming technique and what are truly meant by the term object – oriented programming. Object – oriented programming$_{12}$ allows a complex program to be built out of much simpler constructs called objects, which interact by exchanging message. The following are principal terms and concepts of object- oriented approach: object, class, method, message, inheritance and polymorphism.

Before defining these terms it is probably helpful to relate these notions to familiar terms and concepts. In table 2.1, one such a rough mapping is given.

Object – Orientation Term	Programming Term
Object	Variable
Class	Type
Method	Function
Message	Call
Class hierarchy	Type hierarchy

Table 2.1 Mapping of notions between object – orientation and traditional concepts

2.6.3 Object

An object in object – oriented terminology[12] belongs to a class, just as an entity set. Simple objects are irreducible in that they cannot be decomposed into other objects. They are capable of an independent existence. Each object responds to a prescribed collection of messages that comprises that object's interface.

2.6.4 Class

After having determined the objects that will make up a program, they are grouped into classes according to share common properties. A base object can then be defined for each class. Special cases of class are handled by creating subclasses of objects from the base class. The base class defines general properties, the subclass or derived class defines the special properties. This scheme is called Class hierarchy[12].

The activities required to build object – oriented programming are; first, to create classes that define object representation and behavior, second, to create objects from class definitions and third; to arrange communication among them as a sequence of message. The reason for classifying objects is to simplify message passing protocol. This makes the software more tangible to non-implementers who have application knowledge. It also means that objects in similar application can be reused in the same area; reusing rather than reinventing software speeds the development and maintenance of large applications.

Abstract class is a class that you never make an instance of and contain the methods common to all subclasses. Abstract class makes it much easier to extend and modify programs that require different representations.

2.6.5 Methods and Messages

Only the methods of an object have access to its state, and method can only be invoked by sending a message to the object. The destination between a message and a method is subtle but important, since a method is a part of an object and not a global entity.

Message passing is the organization of the messages into public and private categories. Public messages are generally accessible. Private messages, however, can only be executed by the object itself. They are not available (visible) to outside users. When programming with an object – oriented language, the user is relieved of maintaining message passing. It is only responsibility is to place the new or updated code for the operation (method) in the appropriate class definition.

2.6.6 Inheritance

Inheritance is the ability to define a new object that is just like an old one except for a few minor differences. A new class can be derived from one or more existing classes and can inherit some or all of their properties. A single inheritance refers to a derived class having only one base class; multiple inheritance[12] allows the combination of the properties of existing classes. Single inheritance is more widely use than multiple inheritance.

Most languages perform representation inheritance at complied time and method inheritance at runtime. While it's generally a good idea to inherit common behavior, it's often undesirable to inherit representation. Inheritance is not the only way to share codes and promote reuse some parts are also let to construct compound

Object such as windows. Therefore, inheritance can be use as a mechanism to create objects that share properties with similar objects, and parts can be used as a mechanism. When describing composite objects, the scheme such how they are created and how components are add modified must also be described.

2.6.7 Polymorphism

Polymorphism is ability of different objects to respond differently in multiple Classes to the same message; such a message can be sent without knowing what kinds of objects are involved. An essential requirement of polymorphism is the ability to refer to objects without regard to their classes. We cannot use ordinary (non – pointer, non – reference) variables for this purpose. Every such variable is associate with a memory area that is just large enough to hold objects of one class where the class declared for the variable. Objects of different classes will generally have different size and so cannot be assured of fitting in the allocated memory space. Pointer and reference are refer to objects by their address and the memory space required for an address. The memory space required for an address is the same regardless of the size of the corresponding object, so a single pointer variable or reference can refer to object belonging to different classes.

Overloading is the possibility that an identifier or operator can simultaneously denote two or more distinct functions. Overloading is acceptable only where each message is unambiguous, i.e., where the message can be identified uniquely by using available type information.

2.6.8 Encapsulation

An object consist of an encapsulated representation (state) and a set of messages (operation or procedures) that can be applied to that object. Encapsulation is the technical name for information hiding. To achieve a narrow interface to other modules, a module typically makes only a few components visible outside. Such components are said to be exported by the module. There may be other components that remain hidden inside the module, being used only to assist the implementation of exported components.

The goal here is to separate the user of the object from its implementation. The user is no longer aware of how the object is implemented (using an array or linked lists, for example). User can only operate on an object using those messages that implementation provides. This has the obvious benefit that you can change the implementation of the encapsulated without affecting the public interface, and the object will always produce the expected results.

2.6.9 MultiTasking

Multitasking is the ability to have more than one program working at what seems like the same time. For example, you could print while editing or sending a fax. MultiThreading is same Multitasking. Multitasking can be done in two ways.

2.6.9.1 Preemptive multitasking. For examples, WindowNT, Windows 98. This system type difficult to use, but it has more efficiency. Preemptive time-slicing, the scheduler runs the current thread until it has used up a certain tiny fraction of a second, and then preempts it, suspends it, and resumes another thread for the next tiny

fraction of a second. Preemptive time-slicing is still not the ideal way to schedule threads, however. This approach gives a little too much control to the scheduler. The final touch many modern schedulers add is to allow you to assign each thread a priority. This priority creates a total ordering of all threads, making some threads more important than others. Being higher priority often means that a thread gets run more often (or gets more total running time), but it always means that it can interrupt other, lower-priority threads, even before their time-slice has expired.

2.6.9.2 Cooperative (or Non-preemptive multitasking). For examples Windows 3.11, Solaris, etc. Non-preemptive scheduling, the scheduler runs the current thread forever, requiring that thread explicitly to tell it when it is safe to start a different thread. Non-preemptive scheduling is courtly, always asking for permission to schedule, and is valuable in extremely time-critical, real-time applications where being interrupted at the wrong moment, or for too long, could be disastrous. However, most modern schedulers use preemptive time-slicing, because except for a few time-critical cases, it makes writing multithreaded programs much easier. For one thing, it does not force each thread to decide exactly when it should yield control to another thread. Instead, every thread can just run blindly on, knowing that the scheduler will be fair about giving all other threads their chance to run.

2.6.10 Multithreading

MultiThreading is the ability to have more than one process working at the same time. Process is called thread therefore, more than one process working at the same time called Multithreading. Each thread has own environment, that same as that

thread has own CPU, own register, own memory and own code. This advantage is used in web reader part of Scaling proxy server.

Each thread has many situation, for example, new, blocked, runnable, dead.

2.6.10.1 New situation : Thread is created.

2.6.10.2 Block situation : Thread is stoped processing. Thread is in block situation, if

2.6.10.2.1 Call function sleep() of thread by itself or other thread.

2.6.10.2.2 Call function suspend() of thread by itself or other thread.

2.6.10.2.3 Call wait by itself.

2.6.10.2.4 Some process can blocked by input or output.

2.6.10.3 Runnable situation : Thread is processing.

2.6.10.4 Dead situation : Thread is dead and send it's environment to operating system.

CHAPTER III

RESEARCH METHODOLOGY

This chapter presents tools and methodology, that used to make this study. The purpose of this chapter is identify a more efficiency program to develop the way for faster access HTML pages by reducing image size. So this study consists of three sections. The first section is the research tools. In second section is methodology and literature relevant to this study. In third section is schedule of work

3.1 Research tools

This study is developed by many support tools that they can .divide into hardware and software part as the following:

3.1.1 Hardware

Hardwares for support this study has 2 set. First set is a server, consists of personal–computer Pentium, 133 MHz. The characteristics of it are as follows: main board not less than BUS 100, network device 1 set, one hard disk not less than 2.0 GB, RAM must not less than 64 MB, CD-ROM, power supply must be more than 200 Watts, keyboard, printer, mouse and monitor. Second set is a client, consists of personal-computer Pentium III, 333 MHz. The characteristics of it are as follows: main

board not less than BUS 100, network device 1 set, one hard disk not less than 2.0 GB, RAM must not less than 64 MB, CD-ROM, power supply must be more than 200 Watts, keyboard, printer, mouse and monitor.

3.1.2 Software

Used Softwares for develop Scale down proxy server are Java Development Kit version 1.1.1, WinEdit 2000 for coding, debugging, ACDSee32 version 2.4.1 for quick view picture. The operating system for this study is MicroSoft Windows 98 and MicroSoft Windows 2000. The external program for convert image quality is ImageMagick, version 5.1.1

3.2 Methodology

The steps in this section are shown in figure 3.1. After we identify flowchart, then we develop methodology. We give the detail of 7 steps as the following.

3.2.1 Identify objectives and scope of the study :

Consider the merits and demerits of the study and assign scope of the study.

3.2.2 Collect data and requirement :

Collect data for design and create appropriate system model for quick view and searching image information on WWW. Search JPEG image quality convert program, that appropriate for system model.

3.2.3 Design system model :

Design system model is design operation chart of system model and flowchart of program, we can separate design system model to 3 part

3.2.3.1 Overall system model,

We use collected data (in section 3.2.2), system model of proxy server and existing system (in chapter 2) to design this overall system model

3.2.3.2 Thread control,

We use knowledge about data structure and multitasking to design thread control of this system model

3.2.3.3 Cache Control,

We use cache model of proxy server to apply with thread control (in section 3.2.3.2) to design cache control of this system model.

3.2.4 Programming, Testing and debugging each module:

This section is importance section, because this section use all collected information.

3.2.4.1 To coding the program by Java language with Object Oriented Programming (OOP).

3.2.4.2 Testing and debugging each module, In testing and debugging, we do it during programming, by log files of system. We store process information of program during running in 2 files access log file and error log file. We use both log files to test error and debug program

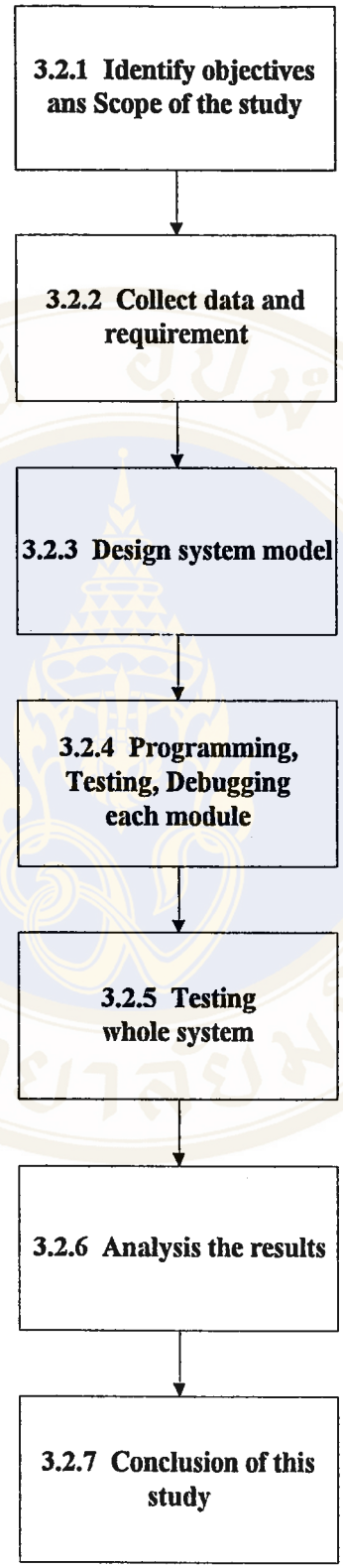


Figure 3.1 Step of the study

3.2.5 Testing whole system :

In testing whole system, we will test performance, efficiency of program by use browser to access original JPEG images, and HTML pages, that contain original JPEG images for compare with, scaled images and HTML pages that contain scaled images. We test system in 2 situation, normal situation,(has one proxy) and connected the another proxy server situation (has more than one proxy).

3.2.6 Analysis the result

Analysis result section use both quantitative, and qualitative, because of the nature of study. In the quantitative analysis, the program itself can be tested, for error by using exception handling in java program as well as validity test. The qualitative approach were base on completion time of getting images after, we run Scale down proxy server.

3.2.6.1 The method of estimate completion time

From statistical theory, when we don't know distribution, we will use sample size n >= 30 (from control limit theory). Then, we can estimate completion time of getting images from formula (1) as follows

$$\bar{X} - Z_{1-\frac{\alpha}{2}} \times \frac{\sigma}{\sqrt{n}} < \mu < \bar{X} + Z_{1-\frac{\alpha}{2}} \times \frac{\sigma}{\sqrt{n}} \quad \text{---------- (1)}$$

where

Z : The standard normal deviation

α : Confidence level, in this case we use 95 % confidence level

n : Sample size

σ : Standard deviation

u : The estimate completion time of get images

3.2.7 Conclusion

This step must begin after system complete, by create document which consist of used method, user manual, expected development, detail, methodology then, conclusion and recommendation of the research.

3.3 Schedule of work

We list many step, that schedule in table 3.1

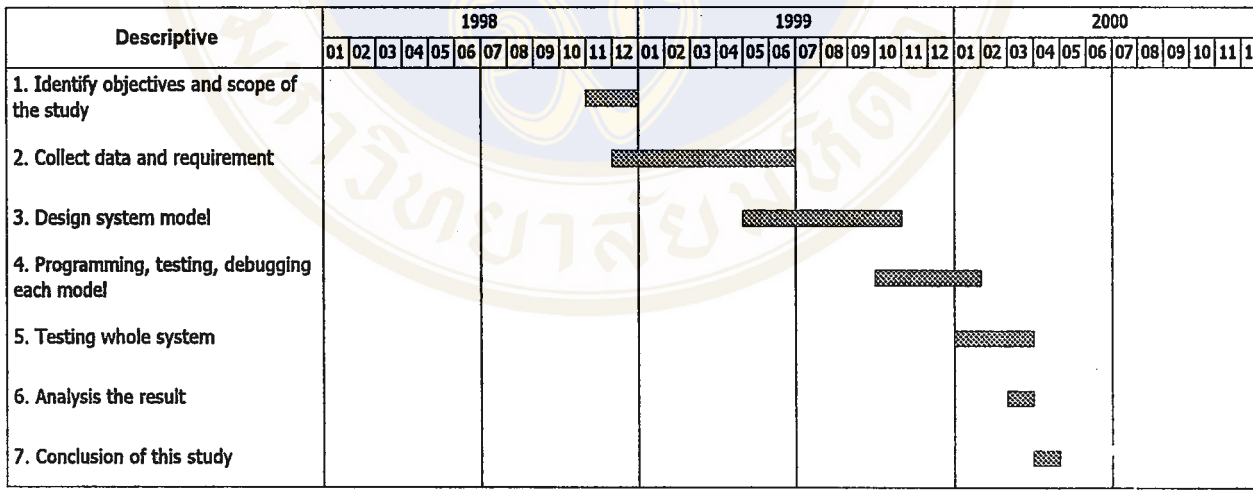


Table 3.1 Schedule of work

CHAPTER IV

RESULTS

This chapter presents the results of study. After we use literatures relevant (In chapter 2) and methodologies (In chapter 3) to collect data, find advantage, disadvantage, design overall operation chart, system model, flow chart. The final result of this research is the Scale down proxy server. This chapter shows detail of system operation and result of testing.

4.1 Overall operation of Scale down proxy server

In this study, we study scaling control program for quick view, and search JPEG image information on WWW systems. Scale down proxy server is proxy server and controller of quality convert program. Reducing image size can do many ways, for example, reducing resolution, reducing quality. But reduce quality is appropriate way for this study, because the other ways reduce width and height of image, therefore it difficult to positioning image on HTML page. Scale down proxy server works with one external program that performs JPEG image quality convert. Scale down proxy server can operate on 2 situation.

4.1.1 First situation is normal situation. System has one proxy server. Scale down proxy server is between Browser and Web Server. As shown in Appendix I.

4.1.2 Second situation is connected the another proxy server situation. System has more than one proxy server. Scale down proxy server is between Browser and the another proxy server. As shown in Appendix I.

Both situation, Scale down proxy server performs receive request message from browser and verify message, modify message and send it to Web server on first situation or the another proxy server on second situation. After that Scale down proxy server receive response message and object. Then, Scale down proxy server sends response message and object to browser.

However, Scale down proxy server has cache control function. It performs create cache index and store image to file. This function is efficiency of Scale down proxy server, it make operation to fast. Operation of Scale down proxy server with cache control shows in Appendix I. Scale down proxy server receives request message, then verify message and modify message (if necessary), search URL in cache index if image object have been request, get image from cache and send it to browser. If image object never requests, sends request message to Web server or another Proxy server and receives response message and object. If object isn't JPEG image, sends object to client. If object is JPEG image, convert quality JPEG image then store it in cache, after that sends it to browser

4.2 System model of Scale down proxy server

Scale down proxy server is a proxy server, which includes JPEG image scaling control program. It performs helping search image information on WWW, However, If

we have the another proxy, we will use the another proxy to travel on WWW. Because Scale down proxy server doesn't have full function of normal proxy server, it has some function of normal proxy server for only help searching image information on WWW. Scale down proxy server has scaling control part, but it doesn't have JPEG image scaling part.

Scale down proxy server consists of 5 parts as follows the main control part, the http-header handle part, the cache control part, the web reader part, image quality convert part.

4.2.1 The main control part

The main control part is a main program of Scale down proxy server. It performs setup system, waiting client connection, connection to client socket, control the number of connection. Flowchart of this part shows in Appendix I.

Detail of this part operation, before working, system must setup system, set all default value to prepare system to work. In setting up system, program will read all value for setup system from initial file. Initial file is " proxy.conf ". Detail of all values in initial file are name (program name), port (port number for system to wait for client connection), proxyhost (the another proxy server name or IP address of another proxy server), proxyport (port number of another proxy server), accesslog (log file to keep access information), errorlog (log file to keep error information), cachedir (directory name to keep cached file and cache index), cachemaxsize (cache size), cachetime (time to cache), convert (filename of JPEG image quality convert program), maxconnections (the maximum number of client connection). The example of initial file shows in

Appendix II. After that, wait for client connection and get the number of connection if it is more than maximum connection, block this connection and waits for the number of connection less than maximum connection. If this connection resume, it will work continue. System will limit the number of connection, because if the number of connection is more than maximum connection. System may be fail or speed very slow. Then, main control part call HTTP-Header handler part for operate client connection continue. After that system will wait for next client connection.

4.2.2 HTTP-Header handle part

The HTTP-Header handle part is important part of Scale down proxy server. It performs get request message, interception message, verification message, decision to modify message, decision to get image from cache control part or get object from web reader part. Flowchart of this part shows in Appendix I.

Detail of this part operation. At first, system gets input and output stream of client socket, then read header request and store it for verify operation. After that, verify URI in header, if URI starts with "http://noconvert", separate "noconvert." from URI and set noconvert variable to true. Then, check another Proxy server, if not separate "http://" from URI, after that, store URI for operation later. Then If noconvert variable equals true, separate "noconvert." from Host field in header message and set <CachedConverted> variable to false. Then, if header has body, read HTTP-Header body and store it for send to target. Then, if Pragma field and Cache-Control field in HTTP-Header message equals "no-cache", set <CachedConverted> to false, after that, verify the method in start-line in header, if method is HEAD method, set

<CachedConverted> variable to false. Then If <CachedConverted> variable equals false, call Web Reader part to continue. Else, search URL in cache index, if found URL in cache index, call Cache control part to continue. Else, call Web Reader part. After Cache Sender part and Web Reader part are finished, remove thread connection, close socket and return resource to operating system.

4.2.3 Web Reader part

The Web Reader part is a part, that performs connection to target Web server. This part performs creating socket connection to destination, sending request message to destination, getting response message from destination, verification response message, decision to convert image quality and cache it too, sending object to client, and control thread queue to use quality convert program call part. Flowchart of this part shows in Appendix I.

Detail of this part operation. At first, get inetaddress for connect to target, get port for connect to target after that, create socket from inetaddress and port and get input and output stream for send and receive message. Then write request to output stream and read HTTP-Header response from input stream. After that, if <CachedConverted> variable equals false, set <dontCacheConvert> variable to true. Then check Pragma field and Content-type field in header if Pragma equals "no-cache" or Content-type doesn't have "JPEG" word, set <dontCacheConvert> to true. After that, if <dontCacheConvert> variable equals true, read object from destination and send object to client and return to HTTP-Header handle part. Else, create new cache entry and get expired date from header. Then, add this thread to queue and that moment, if

image quality convert program is used, suspend this thread and wait for remove from queue and resume this thread. Else, remove this thread from queue and continue to call Image quality convert part for get image object. After Image quality convert part is finished, check queue, if queue isn't empty, remove first thread from queue and resume it. Else, store image object to cache and return to HTTP-Header handle part.

4.2.4 Image Quality Convert part

This part performs reading JPEG image from web server or proxy server, verification JPEG image size, decision to convert image quality, call quality convert program, sending image to client and write it to file. Flowchart of this part shows in Appendix I.

Detail of this part operation, get image size from HTTP-Header response, and check image size, if image size less than 2,000 bytes, read image from web server or proxy server and write image to file after that, return to Web Reader part. Else, read image from web server or proxy server and write image to file. Then calling Quality convert program (external program) and get original image size and converted image size to compare, if original image size more than converted image size, replace converted image with original image. Then sending HTTP-Header response to client and write it to file, then send image to client and write image to file too. After that return to Web Reader part.

4.2.5 Cache control part

This part performs all functions about cache. Cache control part has a thread,

which performs monitoring cache space, if it is changed. In addition Cache control part performs creating image index, and keeping information of images, creating new entry, add new entry to cache, removing entry from cache, searching image filename from cache index, getting image from cache.

4.2.5.1 Monitor cache thread

Monitor cache thread has low priority, because this thread performs monitoring cache space, if it is changed, this thread must update cached index and file in cache space. Cache is changed, when cached file is expired, cache size is more than maximum cache size, add new cached file. This thread will dead, when Scale down proxy server is stopped. Flowchart of monitor cache thread shows in Appendix I.

4.2.5.2 Cache sender

Cache sender is a part of Cache control part. It performs reading cached file and sending it to client, then close cached file and return to HTTP-Header handle part.

This part shows in Appendix I

4.2.5.3 Create New Entry

This part performs creating new entry and set file number, default expired date.

4.2.5.4 Add new entry to cache

This part performs creating directory to keep new entry, assigning cached file name, and changing converted file name to cached file name. Then, adding entry to index table and changing cache size and set <changed> variable to true.

4.2.5.5 Remove entry from cache

This part performs deleting cached file, changing cache size and removing entry from index. Then, if have no cached file in directory, that keeps this cached file, deletes that directory.

4.3 Result of testing Scale down proxy server

In testing Scale down proxy server, we can separate to 2 part, the first part is testing, and debug Scale down proxy server. The second part is testing whole system of scale down proxy server.

4.3.1 Testing and debug Scale down proxy server

In this part, we do it during coding program. We use log file to debug, when we are testing program or part of program, by use <if> command. We will set condition in <if> command , and write information, that we would like to know in log file. The log file in this system has 2 files, Access log and Error log. Both file can assigned in initial file.

4.3.1.1 Access log

The access log performs keep all information about getting object, default access log is "access.log". The example of access log file shows in Appendix II. This is a part of access log.

31.3.76.2 - 28/Apr/2000:23:21:11 PDT "GET http://31.3.76.1/default2.htm HTTP/1.0" 200 –

31.3.76.2

: Host address.

28/Apr/2000:23:21:11 PDT

: Time to write access log file.

"GET http://31.3.76.1/default2.htm HTTP/1.0"

: Request line from Request message.

200

: Response code from start line of Response message.

For getting JPEG image that is converted image quality, it has addition part.

31.3.76.2 - 28/Apr/2000:23:21:12 PDT "GET http://31.3.76.1/backgrnd.jpg HTTP/1.0" 200 - ImageRatio:277/2214=0.12511292

The addition part is " ImageRatio:277/2214=0.12511292 ". It explains ratio of converted image size and original image size.

For getting JPEG image that is cached, it has addition part.

31.3.76.2 - 28/Apr/2000:23:33:56 PDT "GET http://31.3.76.1/gg2.jpg HTTP/1.0" 200 - used cache

The addition part is " used cache ". It explains this image get from cache, not get from web server or proxy server.

In addition we can set other line, that you would like to test this system during testing and debug.

4.3.1.2 Error log

The error log performs keep all error messages during system running, default error log is "error.log". This log file is useful log file for debug. The error messages is written from all exception command in program.

The examples of error message that exception command in program write in error log file.

"Socketoptions failed?: "

: Can not accept client socket

"Couldnt close serversocket, shutting down"

: Can not close serversocket and shut down system. (extreme error)

"convert (file name) not found, is your path correct?"

"convert not found, imageprocessing disabled."

: Not found quality convert program

"Unknown host (message from exception command);

"couldnt find the specified host(message from exception command), is your URL correct?");

: Can not connect to host.

In addition we can set other line, that you would like to test this system during testing and debug same as access log, but usually we use this log file, because we almost use access log file for check request message.

4.3.2 Testing whole system of scale down proxy server

In this part, we do it after coding program finished. We use this testing for performance testing of Scale down proxy server. We can separate testing to 3 part. The first part, we will test getting original JPEG images and HTML page that contain original JPEG images, measure time to get them for compare. The second part, we will test getting converted JPEG images and HTML page that contain converted JPEG images, measure time to get them for compare. The third part, we will test getting converted JPEG images and HTML page that contain converted JPEG images, but in this part we will get them from cache of Scale down proxy server, measure time to get them for compare. All part must get same JPEG images and HTML page for good test and test in same environment too.

4.3.2.1 First part

At, first we will assign JPEG images and HTML page for all testing. List of image filename, size, and property of JPEG images for all testing are shown in table 4.1.

List of HTML page filename, JPEG image filename, that contained in HTML pages, and size, and property of JPEG images for all testing shown in table 4.2.

After that, begin testing get original images and HTML pages, and measure time. We will measure time to get each image, but for HTML page we will sum total of

OBJECT NAME	SIZE (Byte)	PROPERTY
Dog1.jpg	34K	399*275*16M
Dog2.jpg	4K	171*191*16M
300E.jpg	12K	320*240*16M
E320w.jpg	31K	465*4325*16M
Group.jpg	341K	2868*1747*16M

Table 4.1 Shows JPEG image filename, size, and property.

OBJECT NAME	SIZE (Byte)	PROPERTY
Testpage1.htm	1K	*HTML page contains 2 image*
BkFlower3.jpg	8K	240*240*16M
GG2.jpg	43K	640*480*16M
Testpage2.htm	4K	*HTML page contains 7 image*
H_logo.jpg	6K	209*88*16M
H_browse.jpg	3K	117*52*16M
H_samp.jpg	5K	293*91*16M
Powered.jpg	3K	114*43*16M
Docs.jpg	3K	97*54*16M
Tools.jpg	3K	139*55*16M
Backgrnd.jpg	3K	100*100*16M

Table 4.2 Shows HTML pages, JPEG image filename, size, and property.

time to get HTML pages and images that HTML pages contain. Mean time to get each images, and HTML shown in table 4.3.

Raw data of first part is used time to get images and HTML pages, and the results after use statistical theory, these results shown in Appendix III.

OBJECT NAME	SIZE (Byte)	Mean Time (Milliseconds)
Dog1.jpg	34K	1130.33
Dog2.jpg	4K	1048.67
300E.jpg	12K	1085.00
E320w.jpg	31K	1113.67
Group.jpg	341K	2141.67
Testpage1.htm	1K	*1384.67* **[Total 3851.01]**
GG2.jpg	43K	1271.67
BkFlower3.jpg	8K	1194.67
Testpage2.htm	4K	*1089.67* **[Total 9264.35]**
Docs.jpg	3K	1162.00
Tools.jpg	3K	1169.67
H_logo.jpg	6K	1184.00
Backgrnd.jpg	3K	1212.67
H_browse.jpg	3K	1171.67
H_samp.jpg	5K	1151.67
Powered.jpg	3K	1123.00

Table 4.3 The result of first part (get original JPEG images and HTML pages)

From the table 4.3, mean used time of Testpage1.htm include 2 JPEG images is (1384.67 + 1271.67 + 1194.67) = 3851.01 milliseconds. Mean used time of Testpage2.htm include 7 JPEG images is (1089.67 + 1162.00 + 1169.67 + 1184.00 + 1212.67 + 1171.67 + 1151.67 + 1123.00) = 9264.35 milliseconds.

4.3.2.2 Second part

In second part, we use JPEG images and HTML pages in table 4.1 and 4.2 for testing in this part. We will use Scale down proxy server to convert image quality after get images and HTML pages from web server or another proxy server, and measure time. The result of this part shows in table 4.4.

After this part, raw data of second part is mean difference used time to get images and HTML pages and convert image quality, those shown in Appendix III, after use statistical theory, the results shown in Appendix III.

From the table 4.4, mean used time of Testpage1.htm include 2 JPEG images is (1000.67 + 1374 + 1589.33) = 3964 milliseconds. Mean used time of Testpage2.htm include 7 JPEG images is (1025.00 + 1228.33 + 1257.00 + 1513.33 + 1453.33 + 1407.33 + 1426.00 + 1423.33) = 10733.65 milliseconds.

4.3.2.3 Third part

In third part difference from second part, because in second part we get images and HTML pages from web server or another proxy server, but in third part we get them from cache of Scale down proxy server.

OBJECT NAME	SIZE (Byte)	Mean Time (Milliseconds)
Dog1.jpg	34K	1233.67
Dog2.jpg	4K	1113.33
300E.jpg	12K	1154.00
E320w.jpg	31K	1255.67
Group.jpg	341K	5123.33
Testpage1.htm	1K	*1000.67* **[Total 3964]**
GG2.jpg	43K	1374.00
BkFlower3.jpg	8K	1589.33
Testpage2.htm	4K	*1025.00* **[Total 10733.65]**
Docs.jpg	3K	1228.33
Tools.jpg	3K	1257.00
H_logo.jpg	6K	1513.33
Backgrnd.jpg	3K	1453.33
H_browse.jpg	3K	1407.33
H_samp.jpg	5K	1426.00
Powered.jpg	3K	1423.33

Table 4.4 The result of second part

We use JPEG images and HTML pages in table 4.1 and 4.2 for testing in this part. We will do that and measure time. The result of this part shows in table 4.5.

OBJECT NAME	SIZE (Byte)	Mean Time (Milliseconds)
Dog1.jpg	34K	587.33
Dog2.jpg	4K	588.67
300E.jpg	12K	572.00
E320w.jpg	31K	595.33
Group.jpg	341K	1263.67
Testpage1.htm	1K	*1038.33* **[Total 2441.99]**
GG2.jpg	43K	662.33
BkFlower3.jpg	8K	741.33
Testpage2.htm	4K	*1063.33* **[Total 5933.99]**
Docs.jpg	3K	669.33
Tools.jpg	3K	690.00
H_logo.jpg	6K	722.33
Backgrnd.jpg	3K	750.00
H_browse.jpg	3K	682.00
H_samp.jpg	5K	677.00
Powered.jpg	3K	680.00

Table 4.5 The result of third part

After this part, raw data of second part is mean difference used time to get images and HTML pages from cache of Scale down proxy serve, those shown in Appendix III, after use statistical theory, the results shown in Appendix III.

From the table 4.5, mean pused time of Testpage1.htm include 2 JPEG images is (1038.33 + 662.33 + 741.33) = 2441.99 milliseconds. Mean used time of Testpage2.htm include 7 JPEG images is (1063.33 + 669.33 + 690.00 + 722.33 + 750.00 + 682.00 + 677.00 + 680.00) = 5933.99 milliseconds.

4.3.3 Analysis the result

This section shows result of analysis after testing three parts (in section 4.3.2)

4.3.3.1 Analysis the first part.

From table 4.3, it explains getting a small image uses times less than a big image, and getting a HTML page that contain a few images uses times less than a HTML page that contain many images.

4.3.3.2 Analysis the second part.

From table 4.4, it explains the first getting converted images and HTML pages use times more than getting original images and HTML pages. Example, mean time of Group.jpg in table 4.3 equals 2141.67 milliseconds, and mean time of Group.jpg in table 4.4 equals 5123.33 milliseconds. For getting HTML page, total mean time of Testpage2.htm and images, those contained in it in table 4.3 equals 9264.35 milliseconds, and mean difference time of Group.jpg in table 4.4 equals 10733.65 milliseconds. So, the result is getting original object use time more than the first getting converted object.

4.3.3.3 Analysis the third part.

From table 4.5, it explains the getting converted images from cache and use times less than getting original images. Example, mean time of Group.jpg in table 4.3 equals 2141.67 milliseconds, and mean time of Group.jpg in table 4.5 equals 1263.67 milliseconds. For getting HTML page, total mean time of Testpage2.htm and images, those contained in it in table 4.3 equals 9264.35 milliseconds, and mean time of Group.jpg in table 4.5 equals 5933.99 milliseconds. So, the result is getting converted object from cache use time less than the getting original object.

4.3.3.4 Calculate percentage of reduced time.

From below formula, we use it to calculate percentage of reduced time from the getting original object and the getting converted object from cache.

$$percentage\_of\_reduced\_time = \frac{(Present\_value - New\_value)}{Present\_value} \times 100$$

The percentage of reduced time : Group.jpg : 40.996 %

The percentage of reduced time : Testpage2.htm : 35.948 %

From above results, Scale down proxy server can reduce many times to getting objects.

4.4 Compare JPEG image scaling methods

Reducing JPEG image size has 3 methods reducing image resolution, reducing image quality, transform image format.

4.4.1 Reducing image resolution

Reducing image resolution can reduce JPEG image size but after processing image is reduced width and height too. Therefore, HTML page is changed positioning objects. This method isn't appropriate methods for our program.

4.4.2 Reducing image quality

Reducing image quality reduces color of image. Therefore image is dimmed, but width and height of image isn't reduced too. This method can reduce quality by level (1-100). Level 100 is best color and level 1 is worst color. After we try to reduce image quality, we choose level 10, because this level can reduce image size too much and we can understand image. We choose this methods to work with our program.

4.4.3 Transform image format

Image has many formats, transform JPEG image format has 2 ways, the first way is transform JPEG image format to other image formats, the second way is transform JPEG image format to other JPEG image format.

In the first way, JPEG image format can transform to other image formats, such as BMP, GIF, M-JPEG$_8$ (or MPEG), etc. But for this way, JPEG image format is smaller than other image formats, therefore we must not transform JPEG image to other image format.

In the second way, JPEG has two JPEG-based file formats :

- JFIF$_8$ (JPEG File Interchange Format), a "low-end" format that transports pixels and not much else.

- TIFF/JPEG$_8$, aka TIFF$_8$ 6.0, an extension of the Aldus TIFF format$_8$. TIFF$_8$ is a "high-end" format that will let you record just about everything ever wanted to know about an image, and a lot more besides.

JFIF$_8$ has emerged as the de-facto$_8$ standard on Internet, and is what is most commonly meant by "a JPEG file". Most JFIF$_8$ readers are also capable of handling some not-quite-JFIF-legal variant formats$_8$.

The TIFF$_8$ 6.0 spec for incorporating JPEG is not widely implemented, partly because it has some serious design flaws. A revised TIFF/JPEG$_8$ design is described by TIFF Technical$_8$, this design will be the one used in TIFF$_8$ 7.0. New implementations of TIFF$_8$ should use the Tech Note's design for embedding JPEG, not the TIFF$_8$ 6.0 design. TIFF/JPEG$_8$.) Even when TIFF/JPEG$_8$ is stable, it will never be as widely used as JFIF$_8$. TIFF$_8$ is far more complex than JFIF$_8$, and is generally less transportable because different vendors often implement slightly different, non-overlapping subsets of TIFF$_8$. Adding JPEG to the mix hasn't helped any.

Apple's Macintosh QuickTime software uses a JFIF-compatible datastream$_8$ wrapped inside the Mac-specific PICT format$_8$. Conversion between JFIF$_8$ and PICT/JPEG$_8$ is pretty straightforward, and several Mac programs are available to do it. If you have an editor that handles binary files, you can even strip a PICT/JPEG$_8$ file down to JFIF$_8$ by hand.

From above reasons, we use JPEG-based file format JFIF$_8$, therefore transform JPEG-based file format isn't chose for this program.

CHAPTER V

DISCUSSION

In searching information on WWW, we found many images embedded on HTML pages. It 's the problem of searching speed. If those images are big image, searching speed will slow down. However, if we can reduce image size, searching speed will fast. From this problem, we get idea about proxy, which can reduce image and store it to cache, for next request. That proxy is Scale down proxy server. Web server usually has a proxy server for cache data. In operation of Scale down proxy server, Scale down proxy server is secondary proxy server for used to view HTML page, that has many images. So, Scale down proxy server is used or not used during operation, user can determine. Although, Scale down proxy server can solve problem about searching big image information, however it has merits and demerits yet.

5.1 Benefits of Scale down proxy server

After use Scale down proxy server, we can generate benefits of Scale down proxy server, as follows

5.1.1 Speed of searching.

Users can quick view HTML pages those embedded many images.

5.1.2 Reduce used time.

Users can get wanted image information quickly.

5.1.3 Reduce spaces of storage device for store low quality image.

Scale proxy server can convert low quality image from original during requesting.

5.1.4 Easy to use Scale down proxy server.

User can view HTML page and image by usually operation.

5.1.5 Flexible of Scale down proxy server.

System has no impact, if we remove Scale down proxy server out of system. And we can choose to use or not.

5.2 Limitation of Scale down proxy server

Scale down proxy server can useful for quick search image information, but it has limitations as follows.

5.2.1 Reduce JPEG image only

Scale down proxy server reduce JPEG image only, but if we would like to reduce other image formats, we will change external program for convert quality image or include convert image format function. To convert other image to JPEG image and reduce quality image, then cache it and send it to client.

5.2.2 Cache JPEG image only

In normal proxy server, it keeps all objects in cache, but scale down proxy server cache JPEG image only. We solve this limitation by connect to normal proxy server, so scale down proxy server can get all objects from cache of normal proxy server.

5.2.3 Recommend work with normal proxy server

From section 5.2.1, 5.2.2 scale down proxy server will work with normal proxy server, because it have no full functions of standard proxy server, but it works with normal proxy server for solve this limitation.

5.2.4 Limitation of connection

Scale down proxy server gets client connection less than or equal 500 connection. This limitation of connection is estimated from experiment. In real situation, scale down proxy server don't get all client connections, it get client connections, those set IP address of scale down proxy server in browser, when get image information (JPEG) only. So, it has work rate less than normal proxy server.

5.3 Further development of Scale down proxy server.

From section 5.1 although, Scale down proxy server have benefits, but it has weak points yet. In the future, we can develop performance of Scale down proxy server as follows.

5.3.1 Integrate image quality convert function Scale down proxy server

Scale down proxy server can improve speed of operation by integrate image quality convert function and include function to scale many images in the same time. Scale down proxy server can reduce quality image of JPEG format only. If it can reduce quality image of all image formats, user can search many image formats.

5.3.2 To add function to reduce other objects such as audio, animation

If Scale down proxy server gets image, audio, animation and converts quality of them, user has the convenient way to travel on WWW and reduce time to travel on WWW.

5.3.3 Include other all function of standard proxy server

If Scale down proxy server has all function of standard proxy server, we can use Scale down proxy server to replace other proxy server.

5.3.4 Improve stability of client connection

If Scale down proxy server has stability of client connection, Scale down proxy server can work without limitation of client connection.

5.4 Browsing flexibility

User can use normal operation to browse information, but if we would like to browse original image, we just increase "noconvert." in front of URL. Using Scale down proxy server has a little rule.

Example :

Normal URL :

"http://www.mahidol.ac.th" for this URL, we can get converted images, and HTML pages.

Increase Command extension on Normal URL :

"http://noconvert.www.mahidol.ac.th" for this URL, we can get original images and HTML pages.

5.5 Using Scale down proxy in real situation

Scale down proxy server will used on Web server or another proxy server. Scale down proxy is a proxy, so it should used on same place as normal proxy server. But scale down proxy server can keep JPEG image only, so in real situation it should worked with another proxy server, because we can use performance of normal proxy server and include performance of scale proxy server too. Scale proxy server will connect to normal proxy server for get objects, if those objects can found in cache of normal proxy server, they will sent from cache of normal proxy server. But if those objects can't found in cache of normal proxy server, normal proxy server will get those objects from target host, and send them to scale down proxy server. Figure of using scale down proxy server in real situation shows in Appendix I.

CHAPTER VI

CONCLUSION

Scale down proxy server is the result of this study. Scale down proxy has good operation, it can reduce image size and reduce time to get it in the next request time. Although, at first time of getting image, it use time more than usually operation but at next times, we use a little time to get that image. If there are many users get cached image, it will improve performance of Scale down proxy server. In this chapter, we will explain conclusion, performance of Scale down proxy server.

6.1 Performance of Scale down proxy server.

Scale down proxy server is a first model, it must have some weak points, but it has some prominent points too.

Scale down proxy server is a system flexibility, because it is created from Java language, and it easy change image quality. We will explain this section to each topics as follows

6.1.1 Small program.

Scale down proxy server is a small program, it use a little space for store it on storage device. Use a little memory. Therefore, it won't reduce performance of Web server, that Scale down proxy server run on.

6.1.2 No impact to HTML page, that contain converted image.

For reducing image size, we can do it by many ways. Those ways can reduce image size, but some ways have impact to HTML pages, that contain converted image. If we reduce image size by reduce resolution, width, height of image, HTML page, that contain converted image will receive impact of those ways. For Scale down proxy server, use reduce quality of image because this way don't change width and height of image, so it isn't impact to HTML page, that contain converted image.

6.1.6 Flexible to use

User can choose to use Scale down proxy server or not, user will set on browser application in setting proxy server. If we won't like to use Scale down proxy server, we will change setting proxy server in browser application to another proxy server or not set anything. But if we would like to use Scale down proxy server, we will set IP address of Web server, that Scale down proxy run on, and port number of Scale down proxy server. Scale down proxy server has independent platform and easy to change image quality level from initial file and it can change quality convert program, because it is a external program.

6.2 Recommendation of this research

Scale down proxy server has some weak points, those are standard function of proxy server, Scale down proxy server have no all standard function of proxy server, example, Scale down proxy server doesn't store HTML page and original image into cache. Scale down proxy doesn't keep frequency of used cache entry for decision to

remove cache entry or update cache entry. Speed of Scale down proxy server can improved, because it is created from Java language. Although it has many benefits from Java language, but Java language has speed less than C language.

And, Scale down proxy server uses external program to convert image quality, it makes system flexible but it makes speed slow down. From above sentence, in the future, if there are new languages or improved language, we will use this system model to make new proxy server, that has more performance than Scale down proxy server.

However, Scale down proxy server has limitation of client connection, in the future it we can solve stability of client connection problem.

REFERENCES

1. A. Fox and E.A. Brewer, Reducing WWW latency and bandwidth requirements by real-time distillation, 5th International World Wide Web Conference, May 1996, Paris, France, Computer Networks and ISDN Systems, Vol.28(7-11), p. 1145, http://www5conf.inriafr/fich_html/papers/P48/Overview.html.
2. C. Brooks et al., Application-specific proxy servers as HTTP stream transducers, 4th International World Wide Web Conference, Dec, 1995, Boston, Massachusetts, http://www.w3.org/pub/Conferences/WWW4/Papers/56/.
3. L. Delgrossi et al., Media scaling in a multimedia communication system, Multimedia Systems 1994, 2 : 172-180, Springer-Verlag.
4. T. Berners-Lee et al., Hypertext Transfer Protocol - HTTP/1.0, Informational RFC 1945, http://ds.internic.net/rfc/rfc1945.txt. 1996.
5. T. Berners-Lee et al., Hypertext Transfer Protocol - HTTP/1.1, Informational RFC 2068, http://ds.internic.net/rfc/rfc2068.txt. 1997.
6. T. Berners-Lee et al., Uniform Resource Locators (URL), RFC 1738. http://www.w3.org/pub/WWW/Addressing/rfc1738.txt. 1995.
7. T.Berners-Lee and D.Connolly, HTML2.0, RFC1866, ftp://ds.internic.net/rfc/Rfc1866.txt, Nov, 1995.
8. Tom Lane, JPEG FAQ. http://www.faqs.org/faqs/jpeg-faq/, 28, March, 1999.

9. T.Bray, Measuring the Web, 5th International World Wide Web Conference, May 1996, Paris, France, Computer Networks and ISDN Systems. Vol. 28 (7-11). P.993, http://www5conf.inria.fr/fich_html/papers/P9/ Overview.html.

10. D.M. Kristol and L. Montulli, HTTP state management mechanism, HTTP Working Group Internet Draft. ftp://ftp.ietf.org/internet-drafts/fraft-ietf-http-state-mgmt-05.txt, 1996.

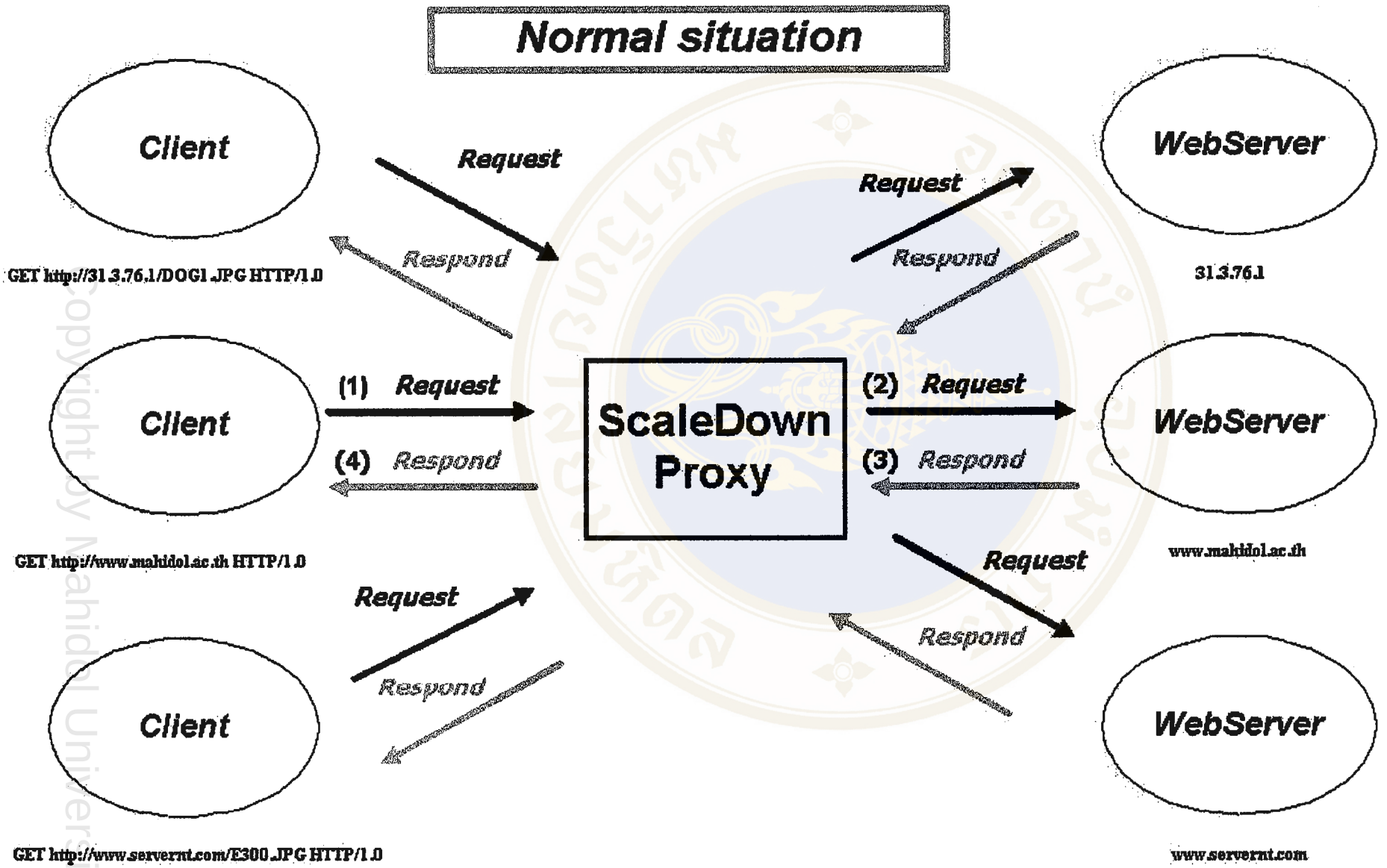
11. Kirsten L.Jones, NIF-T-NAV:A hierarchical navigator for WWW pages, 5th ed., International World Wide Web Conference, May 1996, Paris, France, Computer Networks and ISDN Systems, Vol, 28(7-11), p. 1345, http://www5conf.inria.ft/fich_html/papers/P39/Overview.html.

12. Gary Cornell and Cay S. Horstmann, Core java., United States of America: SunSoft Press A Prentice Hall Title; 1996.

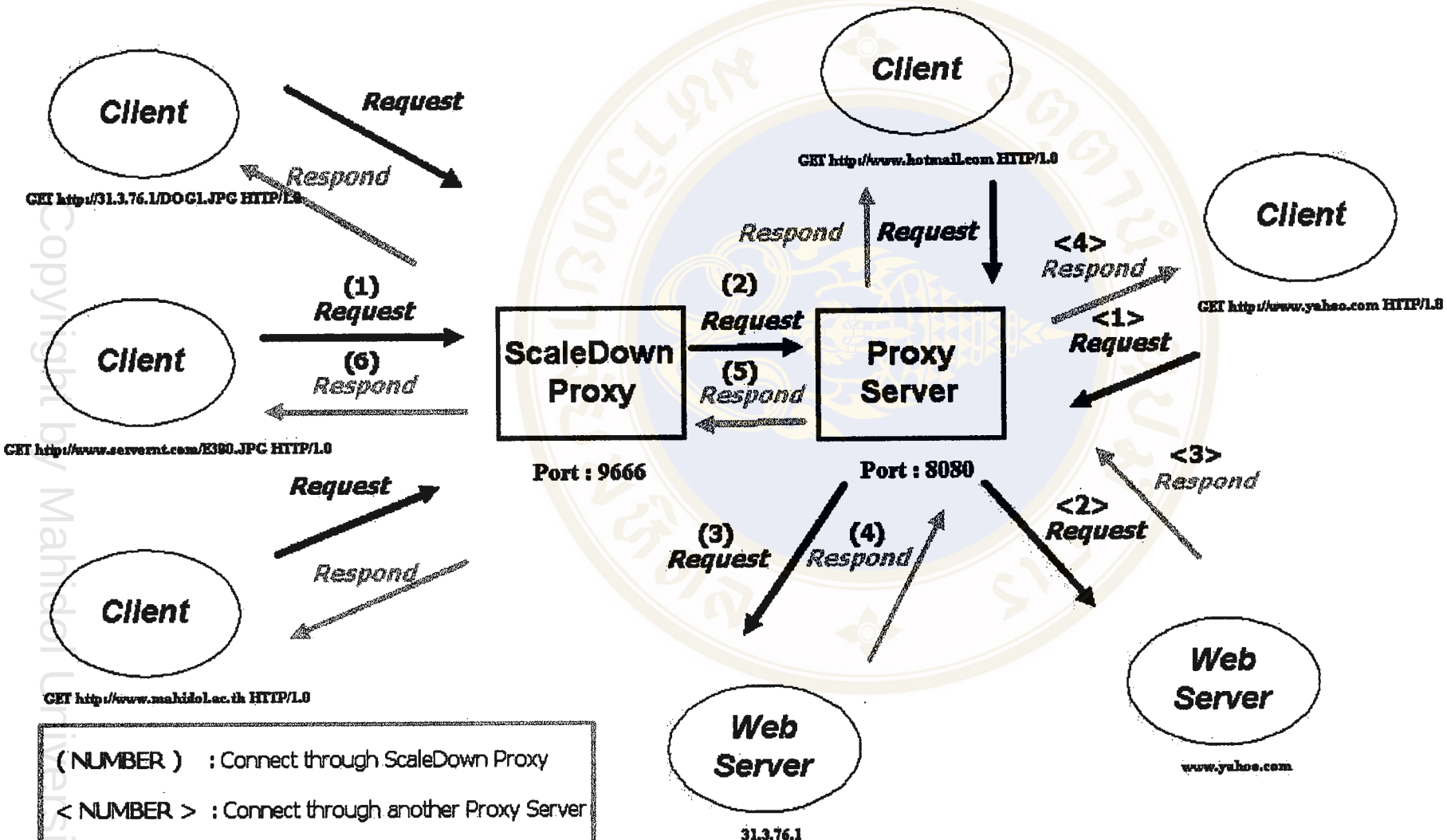
13. Faculty of science Chulalongkorn unvcrivsity, Probability and Statistic 7th ed., 1996.

APPENDIX I

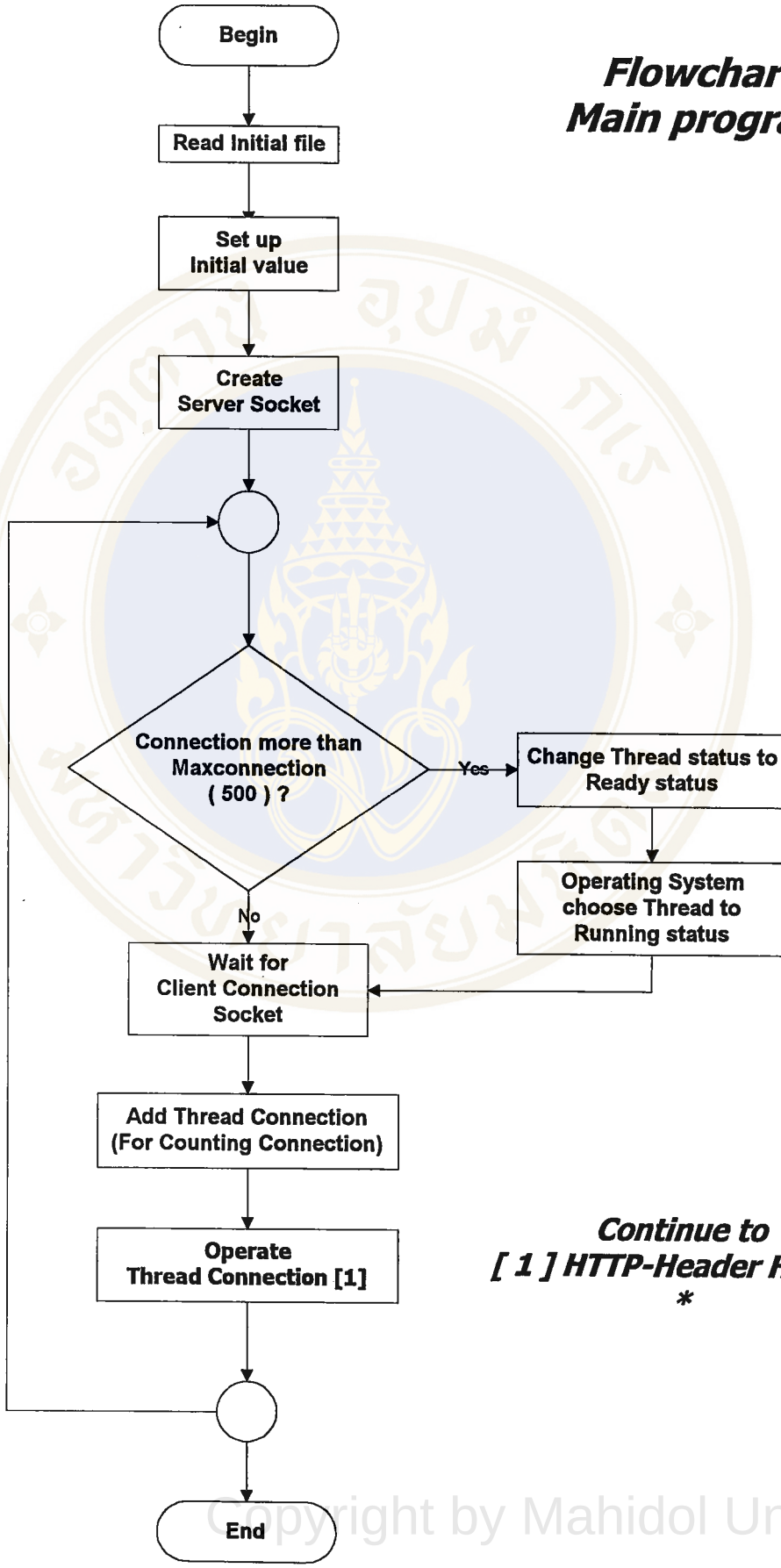
FLOWCHART



Connected to another Proxy server situation

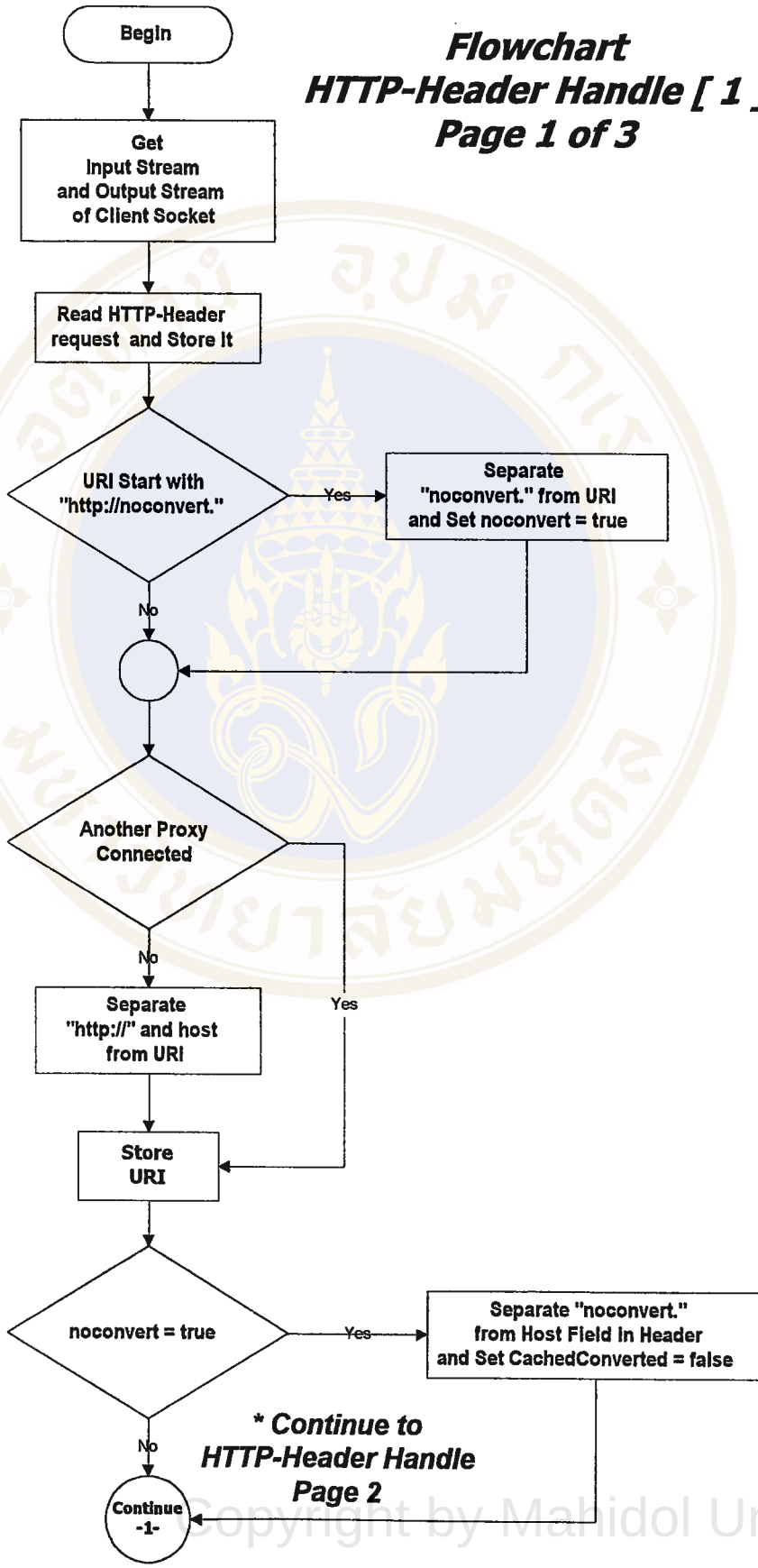


Operation of Scale down proxy server with cache control

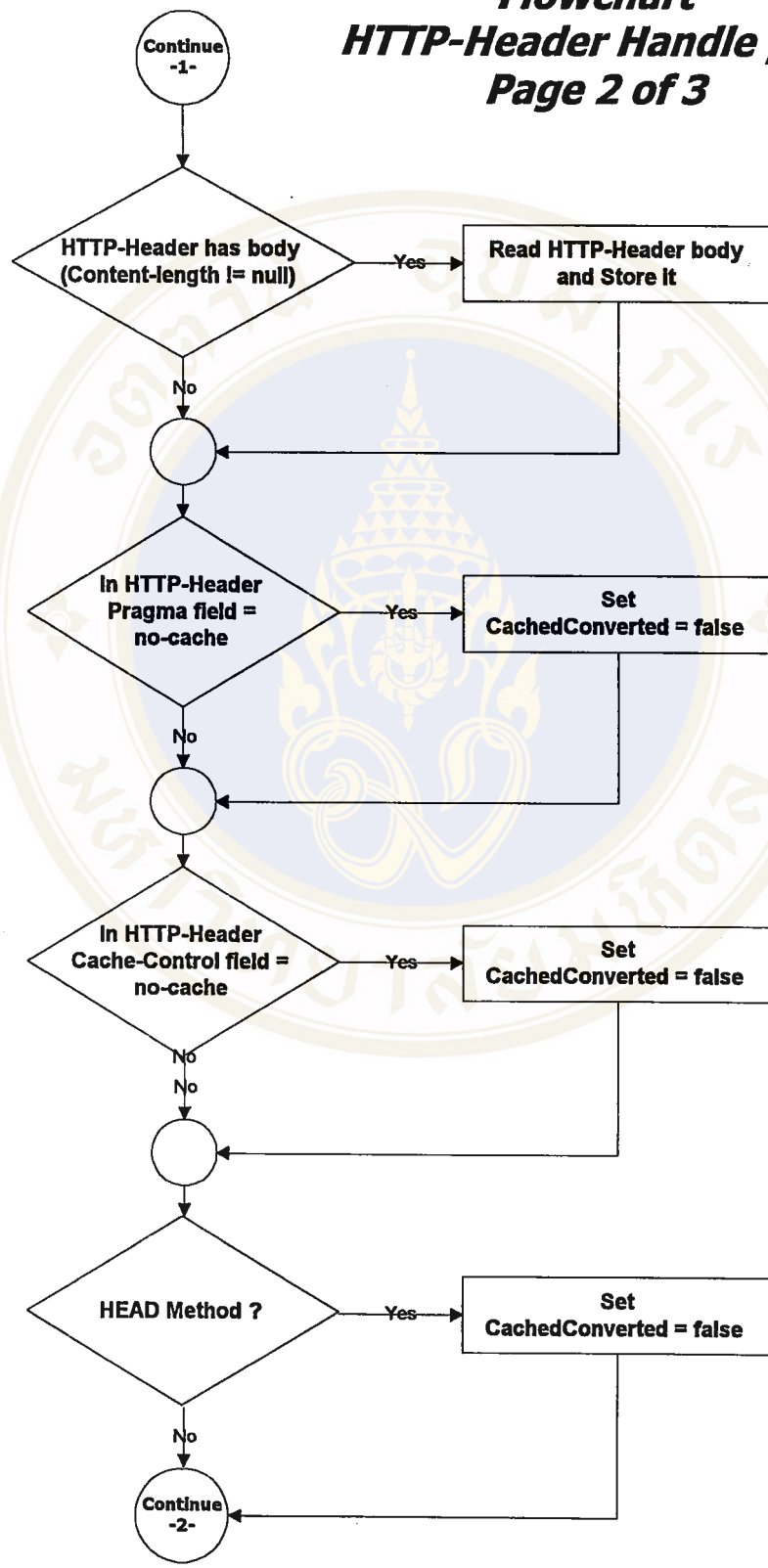


Flowchart Main program

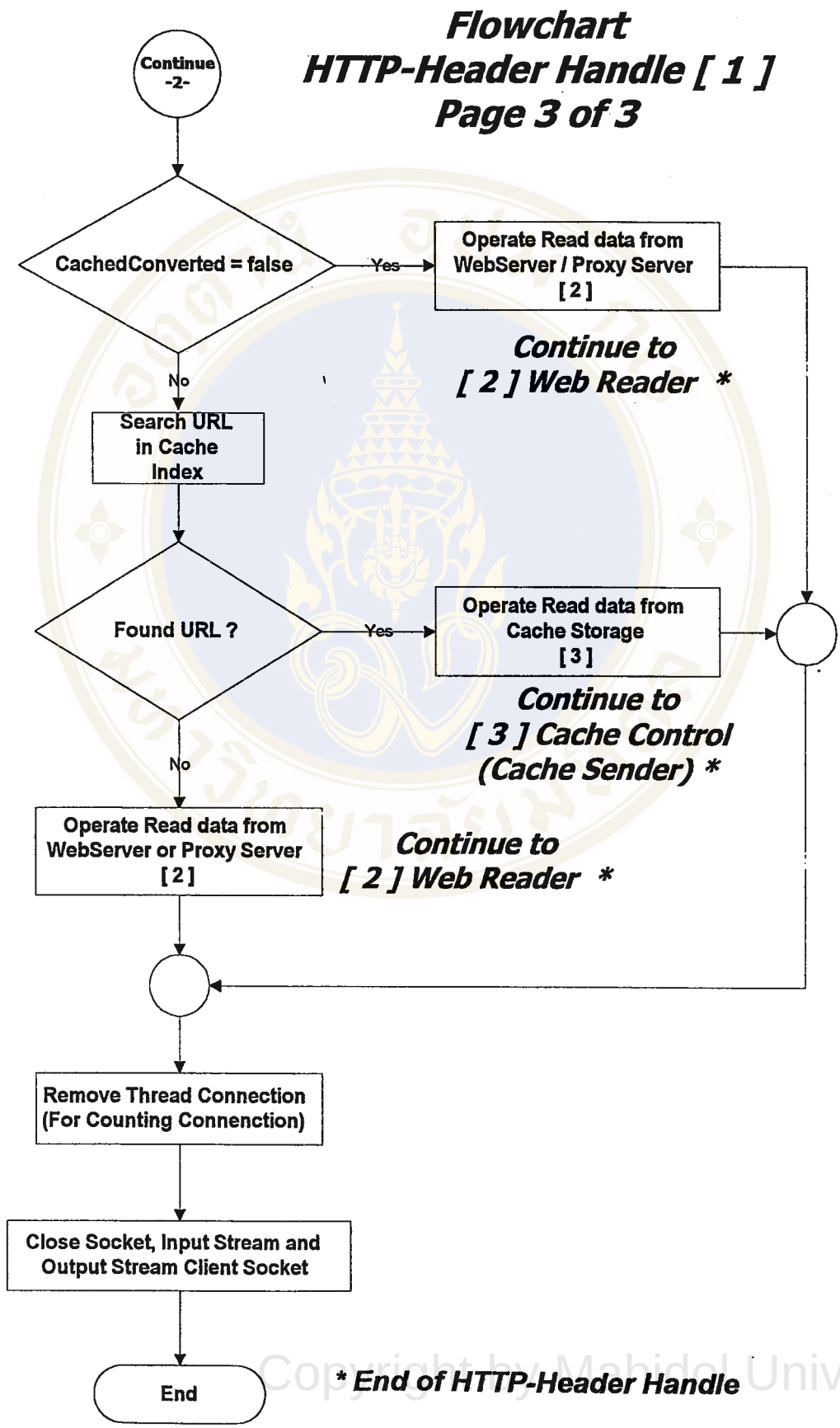
Continue to [1] HTTP-Header Handle *

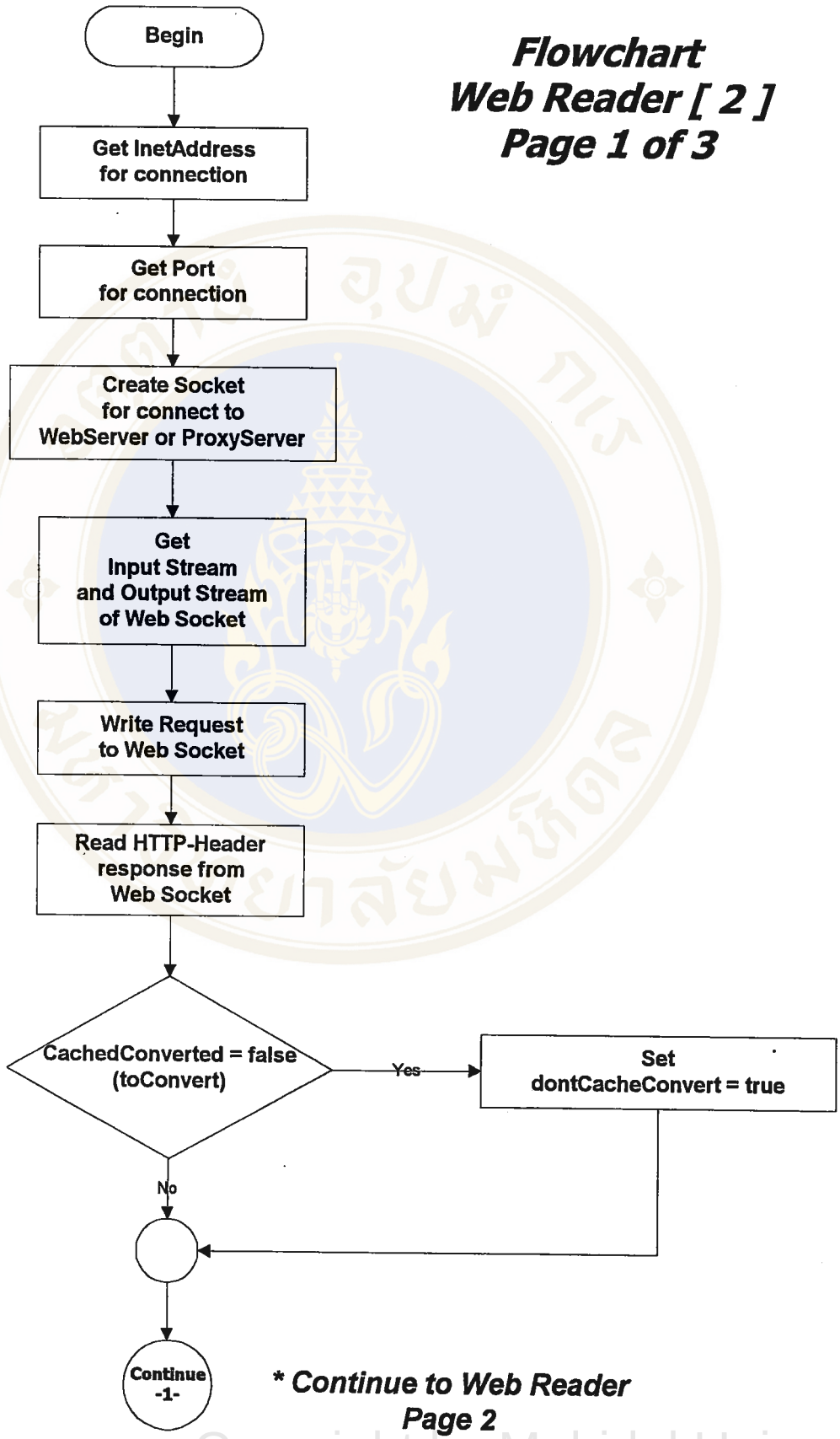


Flowchart
HTTP-Header Handle [1]
Page 2 of 3



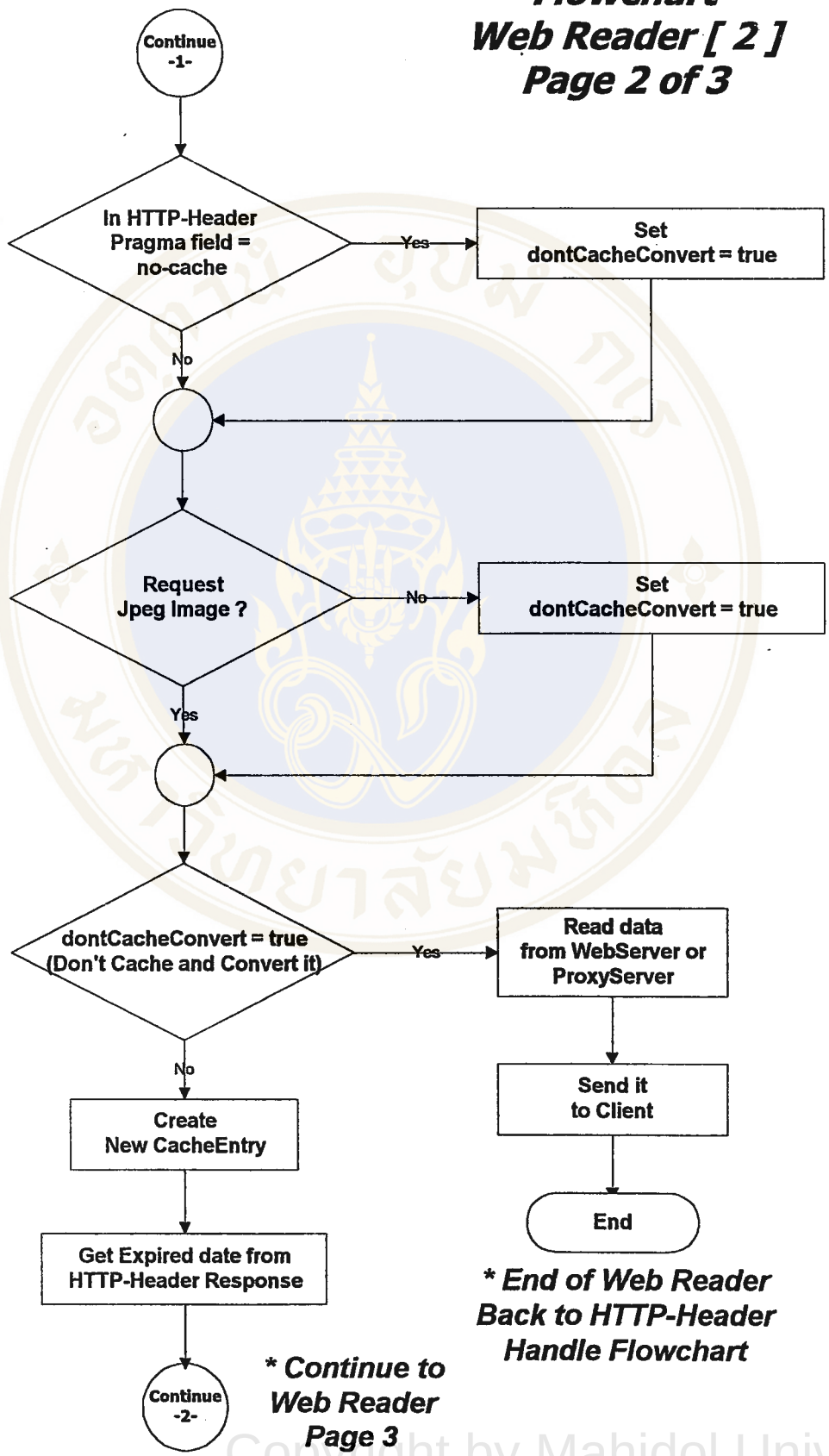
** Continue to HTTP-Header Handle Page 3*



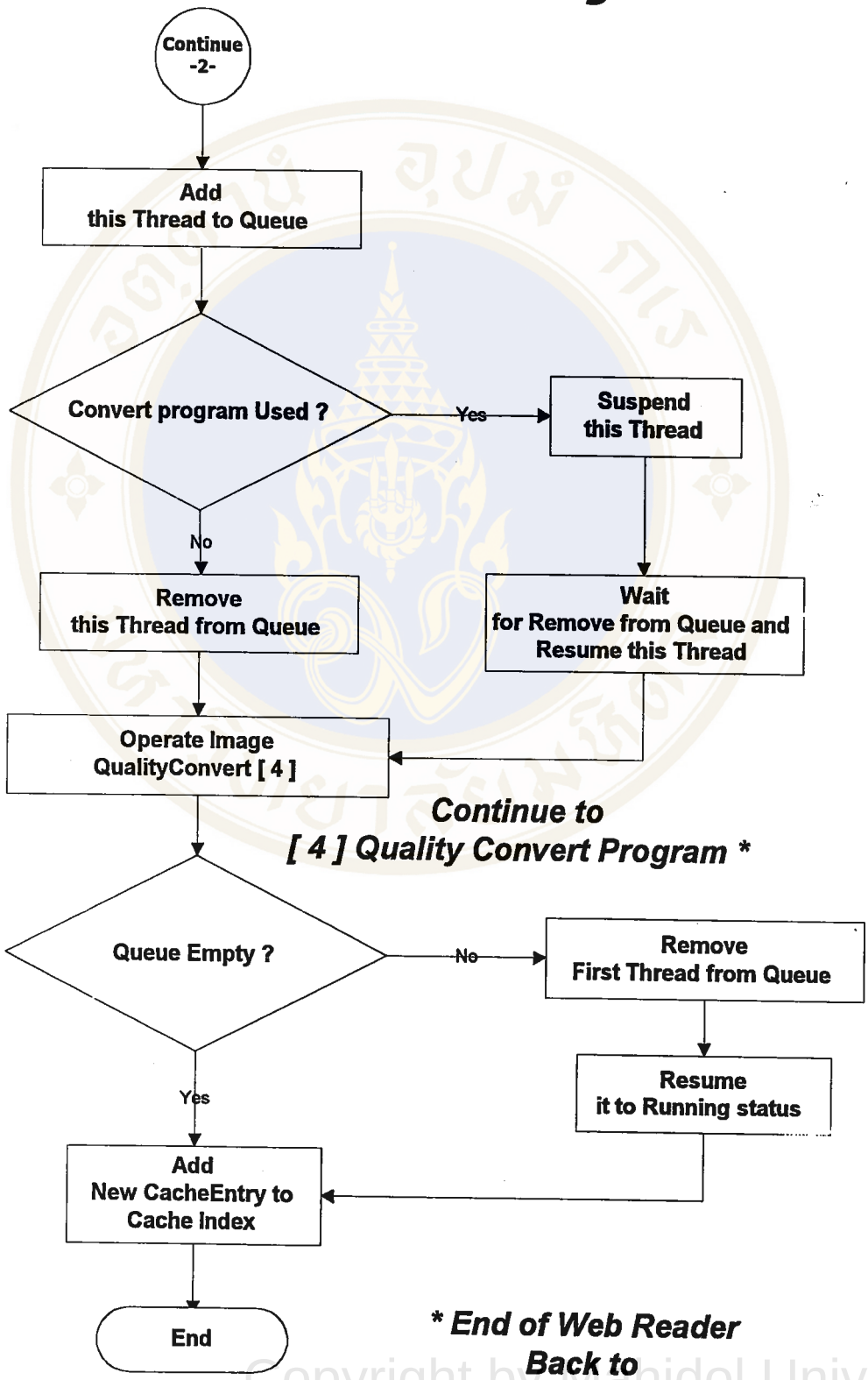


Flowchart
Web Reader [2]
Page 1 of 3

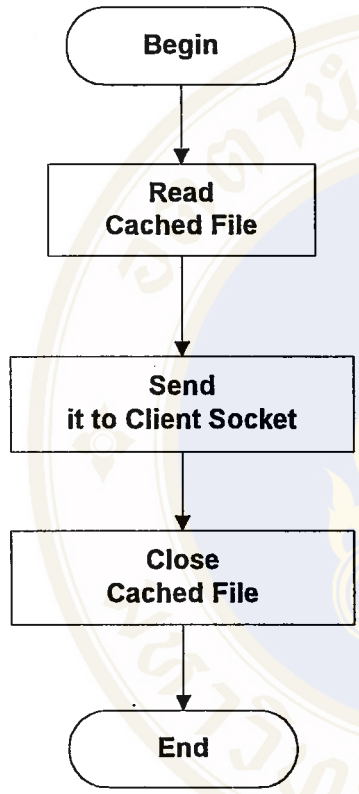
**** Continue to Web Reader Page 2***



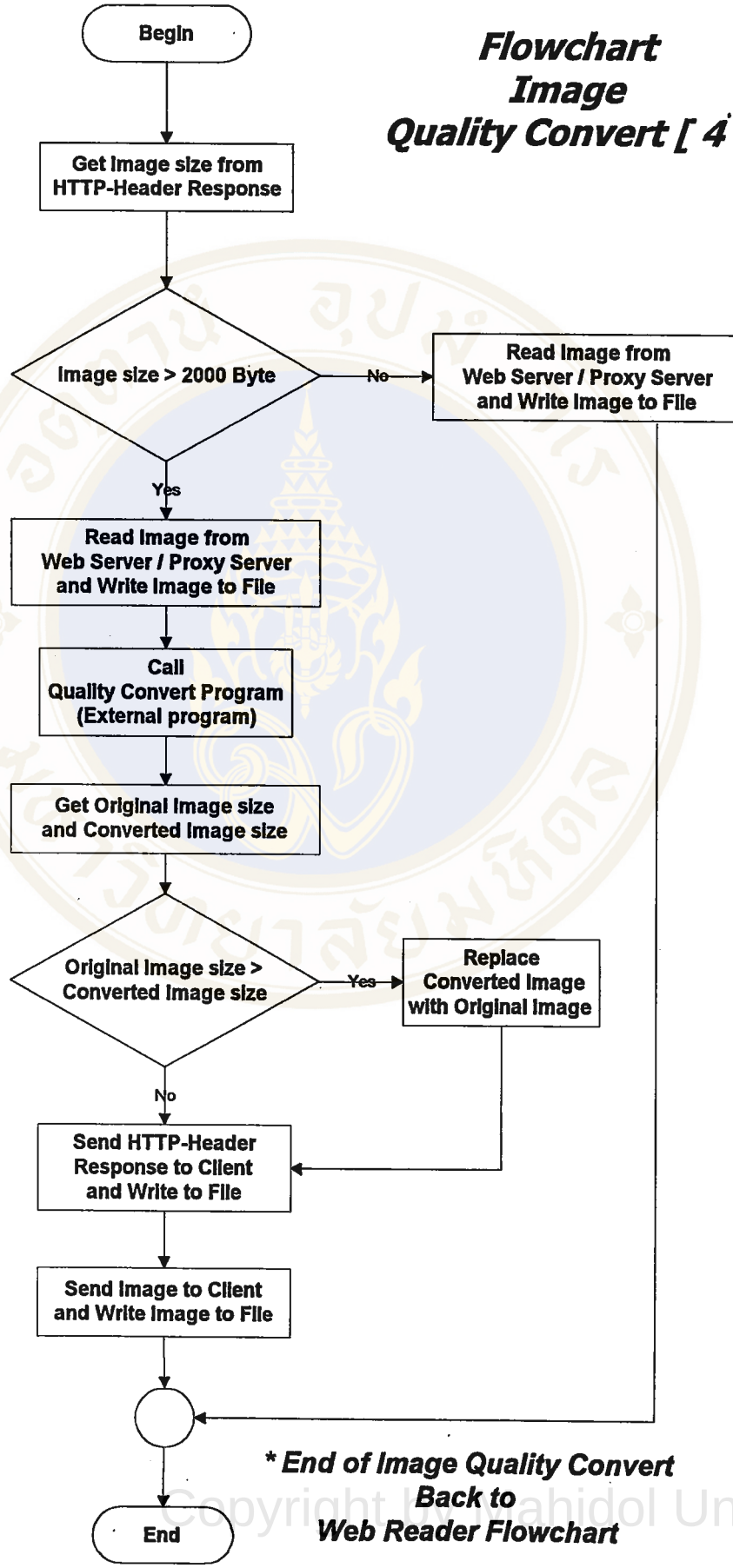
Flowchart
Web Reader [2]
Page 3 of 3

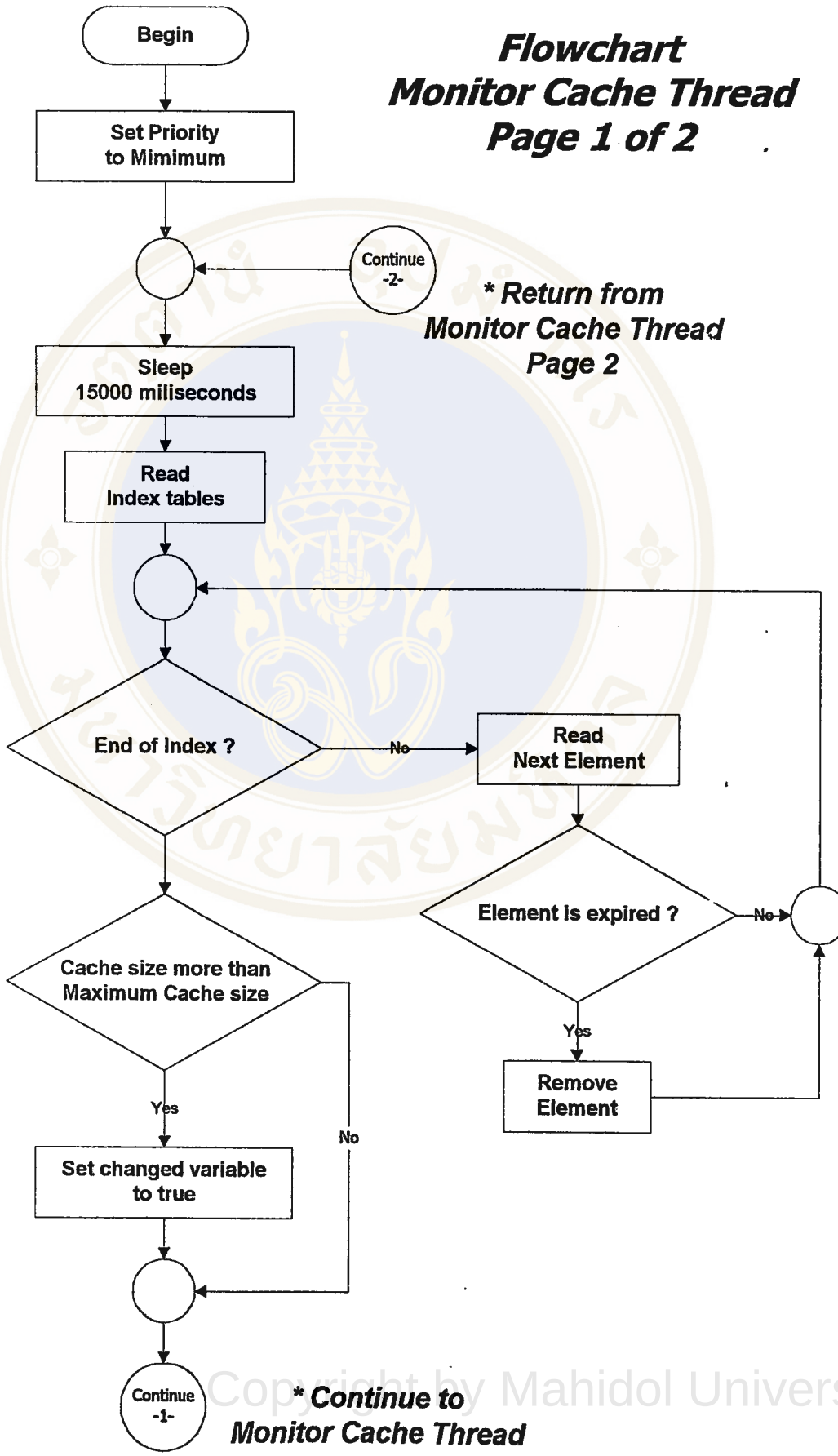


Flowchart
Cache Control [3]
(Cache Sender)

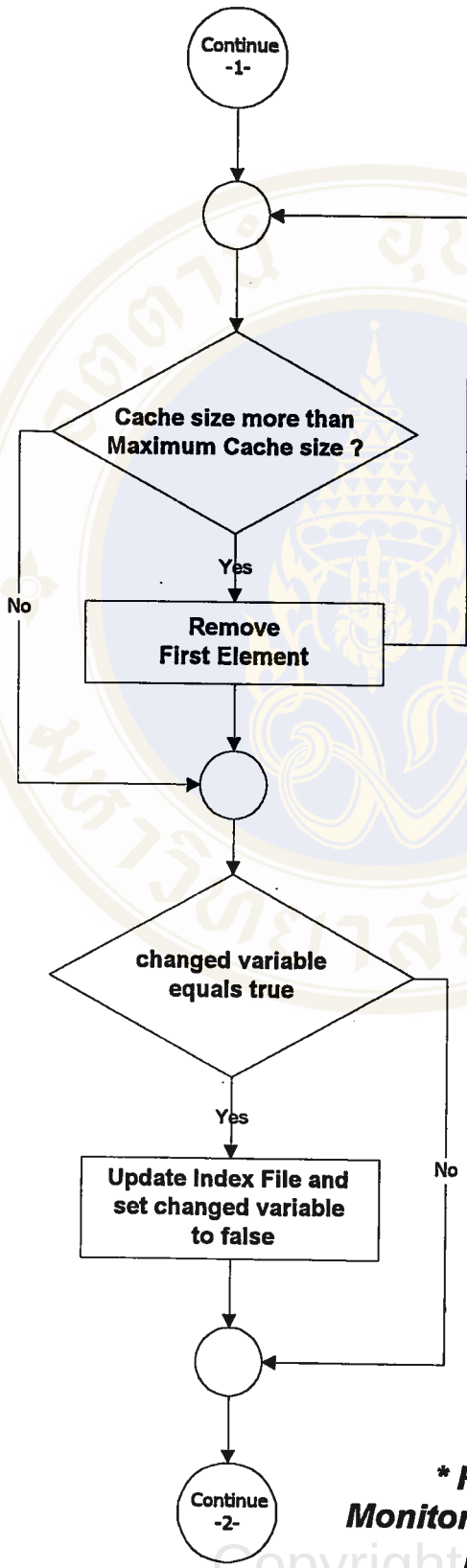


**** End of Cache Sender***
Back to
HTTP-Header Handle Flowchart

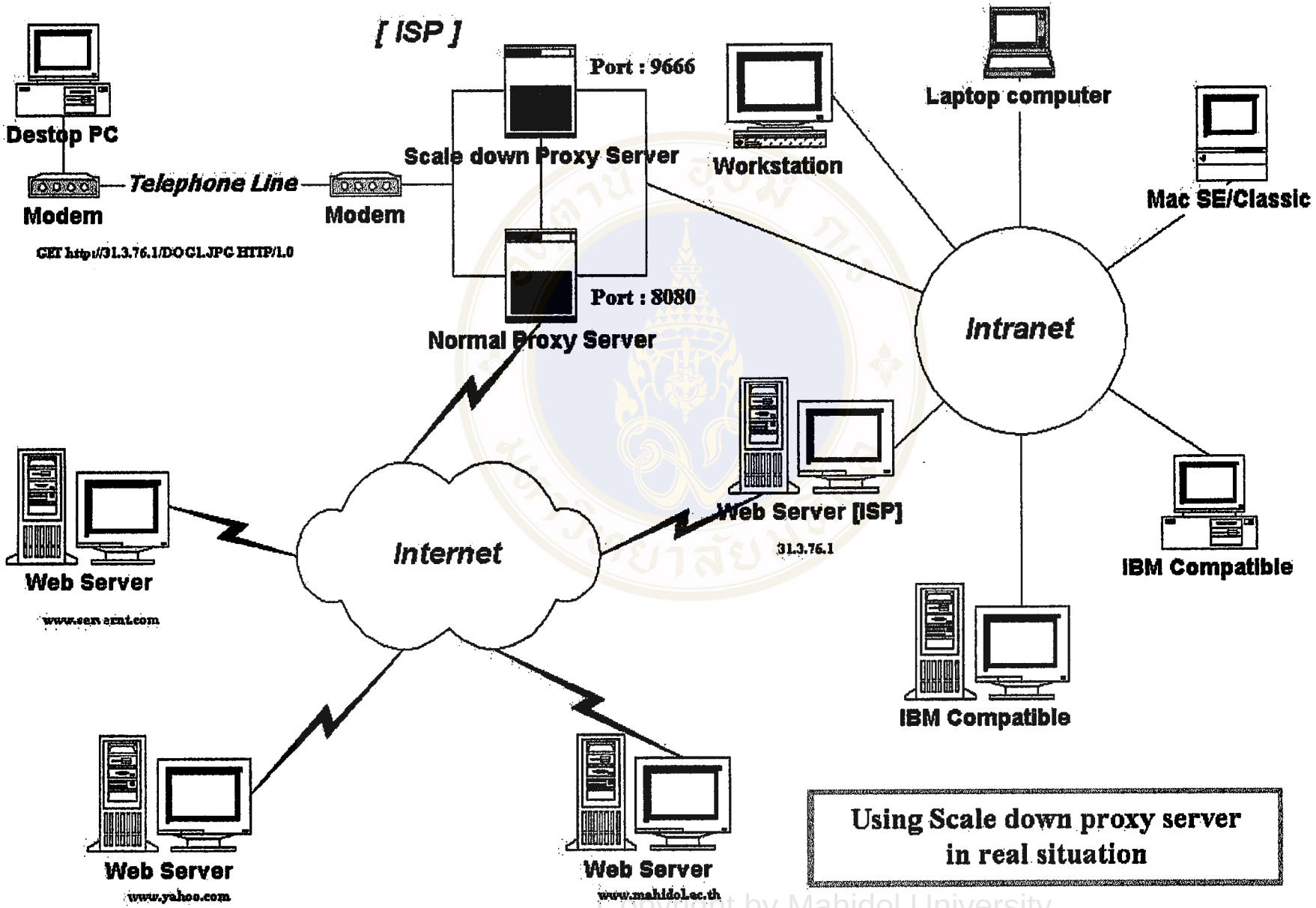




Flowchart
Monitor Cache Thread
Page 2 of 2



* Return to
Monitor Cache Thread
Page 1



Using Scale down proxy server in real situation

APPENDIX II

EXAMPLE FILES

PROXY.CONF

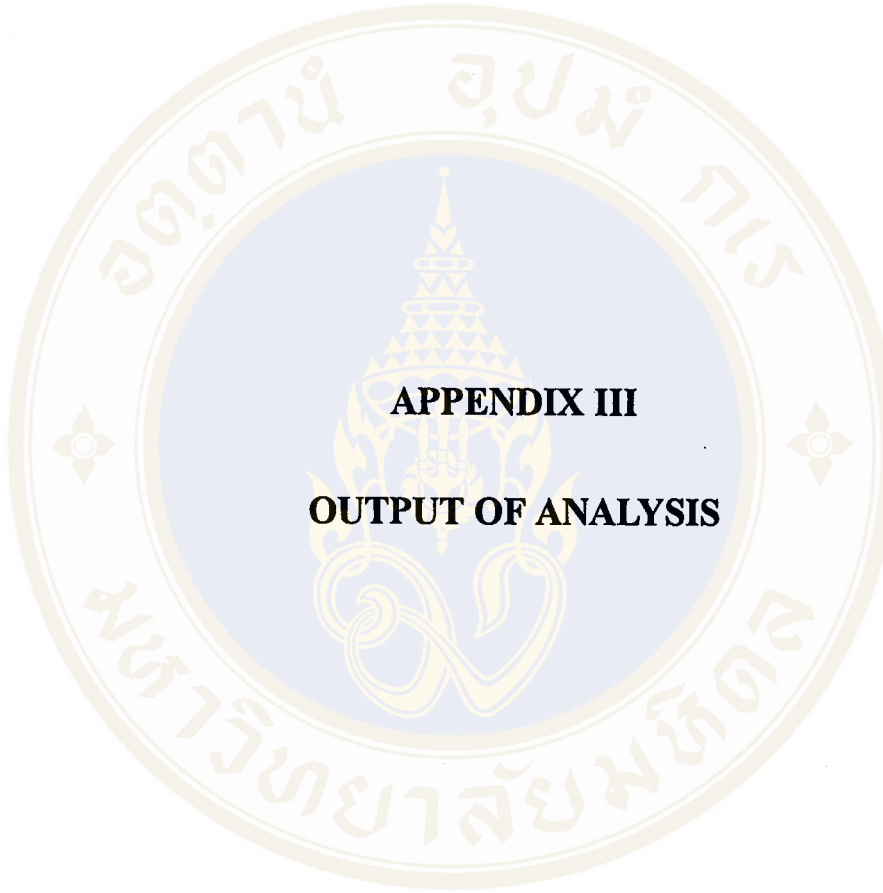
```
# Initial file for ScaleDown Proxy.
#
# Blank lines are ignored and what follows a '#' is a  comment.

# The name of this proxy.
name=ScaleDown Proxy

# The port we listen on.
port=9666

# Connected to another proxy.
proxyhost=proxy.mahidol.ac.th
proxyport=8080

# The log accesses.
accesslog=log/access.log

# The log errors.
errorlog=log/error.log

# The cachedirectory
cachedir=cache

# maximum size of cache in MB.
cachemaxsize=100

# time to cache (in hours).
cachetime=24

# The quality convert program.
convert=convert

# The Maximum concurrent connections.
maxconnections=500

#The image quality level
qualityImage=10
```

ACCESS.LOG

```
31.3.76.2 - 29/Apr/2000:17:53:45 PDT "GET http://31.3.76.1/testpage2.htm
HTTP/1.0" 200 -
31.3.76.2 - 29/Apr/2000:17:53:46 PDT "GET http://31.3.76.1/docs.jpg HTTP/1.0"
200 - used cache
31.3.76.2 - 29/Apr/2000:17:53:46 PDT "GET http://31.3.76.1/tools.jpg HTTP/1.0"
200 - used cache
31.3.76.2 - 29/Apr/2000:17:53:46 PDT "GET http://31.3.76.1/h_logo.jpg HTTP/1.0"
200 - used cache
31.3.76.2 - 29/Apr/2000:17:53:46 PDT "GET http://31.3.76.1/backgrnd.jpg
HTTP/1.0" 200 - used cache
31.3.76.2 - 29/Apr/2000:17:53:47 PDT "GET http://31.3.76.1/h_browse.jpg
HTTP/1.0" 200 - used cache
31.3.76.2 - 29/Apr/2000:17:53:47 PDT "GET http://31.3.76.1/h_samp.jpg HTTP/1.0"
200 - used cache
31.3.76.2 - 29/Apr/2000:17:55:49 PDT "GET http://31.3.76.1/dog1.jpg HTTP/1.0"
200 - ImageRatio:9786/34335=0.28501529
31.3.76.2 - 29/Apr/2000:17:55:52 PDT "GET http://31.3.76.1/dog2.jpg HTTP/1.0"
200 - ImageRatio:1275/3075=0.41463414
31.3.76.2 - 29/Apr/2000:17:55:59 PDT "GET http://31.3.76.1/e320w.jpg HTTP/1.0"
200 - ImageRatio:8686/30839=0.28165635
31.3.76.2 - 29/Apr/2000:17:56:11 PDT "GET http://31.3.76.1/testpage1.htm
HTTP/1.0" 200 -
31.3.76.2 - 29/Apr/2000:17:56:12 PDT "GET http://31.3.76.1/gg2.jpg HTTP/1.0" 200
- ImageRatio:5141/43853=0.117232576
31.3.76.2 - 29/Apr/2000:17:56:13 PDT "GET http://31.3.76.1/BkFlower3.jpg
HTTP/1.0" 200 - ImageRatio:1398/7592=0.1841412
```

ERROR.LOG

[04/Jun/2000:23:00:04 PDT][convert (convert) not found,
is your path correct?]
[04/Jun/2000:23:00:04 PDT][Started]
[04/Jun/2000:23:00:05 PDT][You must specify both proxyhost and proxyport]
[04/Jun/2000:23:03:51 PDT][Unknown host (www.hotmail.com)]
[04/Jun/2000:23:05:17 PDT][Unknown host (convert.31.3.76.1)]
[04/Jun/2000:23:06:30 PDT][Can't remove node, because Queue is Empty Now]
[04/Jun/2000:23:10:11 PDT][Error while reading image]
[04/Jun/2000:24:20:13 PDT][File not cacheable, wont convert!]



APPENDIX III

OUTPUT OF ANALYSIS

List of used time to get original images and HTML pages (Part 1)

(Miliseconds)

DOG1	DOG2	E300	E320W	GROUP	TEST PAGE1	GG2	BKG FLOWER
1270	1040	1040	1090	2090	1050	1310	1150
1100	1040	1100	1040	1980	1050	1260	1150
1100	1050	1050	1150	2150	1040	1260	1150
1150	1040	1100	1150	2090	1040	1270	1160
1100	1040	1100	1150	2090	1040	1260	1150
1210	1040	1100	1100	2030	1090	1270	1160
1320	1040	1100	1100	2090	1100	1320	1150
1150	1050	1100	1100	2140	1040	1260	1150
1150	1050	1100	1100	2090	1100	1210	1270
1100	1040	1090	1100	2030	1050	1260	1150
1100	1090	1100	1150	2090	9900	1260	1150
1160	1040	1050	1100	2200	1100	1260	1260
1210	1040	1100	1090	2080	1040	1260	1150
1100	1050	1040	1100	2140	1100	1260	1150
1040	1040	1050	1100	2200	1050	1260	1200
1100	1100	1100	1100	2090	1100	1260	1150
1100	1050	1100	1040	2190	1040	1320	1150
1100	1050	1100	1100	3250	1430	1320	1210
1150	1050	1040	1050	2090	1050	1260	1260
1100	1040	1100	1100	2090	1090	1270	1100
1100	1050	1210	1380	2150	1040	1370	1650
1150	1100	1100	1310	2080	1160	1320	1160
1150	1040	1100	1100	2190	1040	1260	1210
1090	1050	1040	1040	2090	1040	1260	1150
1100	1040	1050	1090	2030	1040	1260	1320
1100	1040	1050	1100	2090	1050	1270	1210
1100	1040	1050	1090	2200	1160	1260	1150
1160	1040	1090	1100	2040	1150	1270	1270
1050	1040	1100	1100	2090	1210	1260	1150
1100	1040	1100	1090	2090	1150	1210	1150

List of used time to get original images and HTML pages (Continue)

(Miliseconds)

TEST PAGE2	DOCS	TOOLS	H_LOGO	BACK GROUND	H_BROWSE	H_SAMP	POWER
1430	1100	1160	1160	1210	1150	1210	1150
1050	1090	1150	1150	1200	1150	1150	1100
1040	1160	1160	1210	1160	1150	1210	1150
1040	1150	1200	1150	1200	1210	1210	1160
1100	1200	1260	1210	1260	1200	1150	1150
1050	1260	1160	1260	1260	1150	1150	1150
1050	1270	1210	1210	1270	1210	1150	1210
1050	1100	1100	1120	1150	1150	1100	1150
1040	1160	1160	1210	1210	1150	1150	1150
1040	1100	1100	1210	1160	1150	1150	1160
1100	1150	1150	1200	1200	1160	1160	1160
1100	1150	1150	1150	1210	1160	1210	1210
1050	1100	1150	1150	1210	1150	1150	1150
1100	1150	1210	1150	1210	1160	1160	1150
1040	1210	1160	1160	1210	1160	1160	1160
1040	1160	1160	1160	1210	1150	1100	1160
1100	1160	1160	1210	1210	1150	1100	1150
1100	1200	1150	1210	1210	1210	1150	1210
1040	1160	1160	1160	1210	1160	1160	1160
1150	1150	1200	1150	1150	1160	1150	1160
1150	1210	1480	1260	1540	1210	1210	1160
1100	1160	1210	1210	1160	1260	1160	1150
1040	1160	1160	1210	1210	1210	1150	1160
1210	1160	1160	1160	1160	1150	1100	1150
1050	1210	1100	1150	1200	1160	1100	1100
1040	1160	1100	1210	1210	1200	1150	1150
1050	1160	1160	1210	1210	1090	1150	1100
1040	1210	1100	1160	1160	1160	1100	1160
1150	1150	1150	1210	1210	1210	1150	1150
1150	1100	1160	1150	1210	1210	1150	1210

Identify Mean and Std. Deviation of used time to get original image

	N	Mean	Std. Deviation
DOG1	30	1130.33	59.39
DOG2	30	1048.67	16.97
E300	30	1085.00	34.72
E320W	30	1113.67	69.65
GROUP	30	2141.67	216.53
TESTPAG1	30	1384.67	1610.24
GG2	30	1271.67	32.39
ELCFLOWE	30	1194.67	99.68
TESTPAG2	30	1089.67	79.07
DOCS	30	1162.00	44.98
TOOLS	30	1169.67	69.21
H_LOGO	30	1184.00	35.58
BACKGROU	30	1212.67	68.83
H_BROWSE	30	1171.67	33.54
H_SAMP	30	1151.67	33.95
POWER	30	1123.00	194.85

Identify interval time of used time to get original image

			95% Confidence Interval of the Difference	
	df	Mean Difference	Lower	Upper
DOG1	29	1130.33	1108.16	1152.51
DOG2	29	1048.67	1042.33	1055.00
E300	29	1085.00	1072.04	1097.96
E320W	29	1113.67	1087.66	1139.68
GROUP	29	2141.67	2060.81	2222.52
TESTPAG1	29	1384.67	783.39	1985.94
GG2	29	1271.67	1259.57	1283.76
ELCFLOWE	29	1194.67	1157.45	1231.89
TESTPAG2	29	1089.67	1060.14	1119.19
DOCS	29	1162.00	1145.20	1178.80
TOOLS	29	1169.67	1143.82	1195.51
H_LOGO	29	1184.00	1170.71	1197.29
BACKGROU	29	1212.67	1186.97	1238.37
H_BROWSE	29	1171.67	1159.14	1184.19
H_SAMP	29	1151.67	1138.99	1164.34
POWER	29	1123.00	1050.24	1195.76

List of used time to get converted images and HTML pages (Part 2)

(Miliseconds)

DOG1	DOG2	E300	E320W	GROUP	TEST PAGE1	GG2	BKG FLOWER
1210	1100	1210	1380	5170	1150	1420	1590
1320	1150	1150	1210	5000	1160	1260	1590
1320	1150	1160	1270	5330	1100	1210	1540
1260	1090	1150	1370	5220	1100	1260	1590
1260	1100	1150	1270	5220	990	1370	1590
1210	1100	1160	1260	5220	990	1370	1650
1210	1150	1150	1260	5060	990	1490	1600
1210	1100	1150	1200	5050	990	1380	1540
1210	1100	1150	1260	5220	1050	1370	1590
1270	1100	1160	1260	5000	1040	1430	1650
1210	1100	1150	1210	5000	1050	1370	1590
1210	1150	1150	1200	5050	1050	1370	1590
1260	1100	1150	1210	5110	990	1320	1540
1260	1100	1150	1210	5000	990	1370	1540
1210	1100	1150	1210	5050	990	1420	1590
1210	1150	1150	1320	5000	1040	1430	1650
1210	1100	1150	1270	5160	1050	1370	1590
1210	1100	1150	1270	4990	990	1370	1590
1200	1100	1150	1260	5050	1040	1380	1600
1200	1150	1150	1260	5220	1040	1370	1590
1210	1100	1150	1260	5000	1050	1420	1590
1260	1100	1150	1260	5270	1040	1370	1590
1260	1150	1210	1210	5220	990	1370	1590
1200	1100	1150	1260	5220	1050	1370	1590
1270	1100	1150	1260	5220	990	1370	1540
1210	1100	1150	1270	5050	1050	1370	1590
1260	1100	1100	1260	5050	990	1380	1600
1210	1160	1160	1210	5170	990	1380	1600
1210	1100	1160	1260	5170	1040	1430	1590
1260	1100	1150	1260	5210	1040	1430	1600

List of used time to get converted images and HTML pages (Continue)

(Miliseconds)

TEST PAGE2	DOCS	TOOLS	H_LOGO	BACK GROUND	H_BROWSE	H_SAMP	POWER
1100	1320	1210	1590	1430	1420	1420	1370
1040	1320	1150	1600	1430	1370	1370	1370
1040	1260	1150	1490	1600	1420	1420	1420
1040	1210	1320	1430	1540	1370	1430	1430
990	1260	1150	1430	1540	1420	1420	1420
990	1160	1430	1590	1370	1370	1430	1430
1050	1320	1210	1430	1590	1420	1420	1430
1040	1210	1590	1480	1320	1430	1380	1430
1040	1150	1260	1370	1540	1370	1430	1380
1040	1150	1310	1480	1590	1430	1430	1490
1040	1210	1320	1490	1600	1420	1420	1420
980	1320	1210	1590	1430	1430	1430	1430
990	1320	1210	1540	1430	1420	1480	1480
990	1150	1260	1540	1370	1430	1430	1370
1040	1150	1260	1530	1420	1430	1430	1430
990	1150	1320	1540	1430	1420	1420	1420
1040	1150	1260	1370	1540	1430	1430	1430
1050	1150	1310	1420	1530	1430	1430	1430
1040	1210	1320	1540	1430	1430	1430	1430
1050	1320	1210	1530	1430	1370	1420	1420
1050	1270	1160	1540	1430	1420	1420	1430
1040	1210	1270	1590	1430	1480	1480	1430
990	1160	1270	1540	1430	1380	1430	1430
990	1430	1150	1540	1320	1370	1420	1420
990	1150	1260	1590	1420	1370	1430	1430
990	1320	1210	1540	1430	1420	1420	1420
1040	1100	1260	1480	1370	1380	1430	1430
980	1260	1150	1540	1370	1370	1420	1420
1050	1310	1210	1530	1420	1420	1430	1430
1050	1150	1310	1530	1420	1380	1430	1430

Identify Mean and Std. Deviation of used time to get converted image

	N	Mean	Std. Deviation
DOG1	30	1233.67	34.69
DOG2	30	1113.33	23.39
E300	30	1154.00	18.31
E320W	30	1255.67	43.45
GROUP	30	5123.33	101.69
TESTPAG1	30	1000.67	163.79
GG2	30	1374.00	55.25
ELCFLOWE	30	1589.33	28.64
TESTPAG2	30	1025.00	30.60
DOCS	30	1228.33	81.71
TOOLS	30	1257.00	92.41
H_LOGO	30	1513.33	64.18
BACKGROU	30	1453.33	80.96
H_BROWSE	30	1407.33	29.00
H_SAMP	30	1426.00	20.10
POWER	30	1423.33	25.51

Identify interval time of used time to get converted image

			95% Confidence Interval of the Difference	
	df	Mean Difference	Lower	Upper
DOG1	29	1233.67	1220.71	1246.62
DOG2	29	1113.33	1104.60	1122.07
E300	29	1154.00	1147.16	1160.84
E320W	29	1255.67	1239.44	1271.89
GROUP	29	5123.33	5085.36	5161.30
TESTPAG1	29	1000.67	939.51	1061.83
GG2	29	1374.00	1353.37	1394.63
ELCFLOWE	29	1589.33	1578.64	1600.03
TESTPAG2	29	1025.00	1013.57	1036.43
DOCS	29	1228.33	1197.82	1258.84
TOOLS	29	1257.00	1222.49	1291.51
H_LOGO	29	1513.33	1489.37	1537.30
BACKGROU	29	1453.33	1423.10	1483.56
H_BROWSE	29	1407.33	1396.51	1418.16
H_SAMP	29	1426.00	1418.49	1433.51
POWER	29	1423.33	1413.81	1432.86

List of used time to get cached images and HTML pages (Part 3)

(Miliseconds)

DOG1	DOG2	E300	E320W	GROUP	TEST PAGE1	GG2	BKG FLOWER
550	600	550	610	1210	990	650	760
600	600	550	550	1200	1040	660	710
600	550	600	600	1430	1040	660	770
550	600	600	600	1320	1100	660	770
550	600	550	610	1270	1100	660	710
550	550	550	660	1490	990	660	770
660	600	550	550	1320	1100	660	710
550	600	550	600	1540	1100	660	770
610	550	610	600	1320	1150	660	720
610	610	550	600	1150	1160	660	770
550	550	550	600	1270	990	660	770
600	600	540	540	1590	1040	660	720
600	550	610	650	1310	990	660	770
600	600	610	600	1150	1040	660	770
600	710	550	540	1150	990	720	720
610	610	610	550	1160	990	600	710
550	600	610	600	1150	990	660	770
600	550	550	660	1270	990	660	770
550	550	600	600	1210	1040	600	720
550	600	550	660	1210	990	710	770
660	610	550	660	1210	1150	660	770
550	610	550	550	1210	990	660	720
600	550	550	600	1210	1040	660	710
610	550	550	610	1270	1040	660	770
600	600	550	550	1210	1050	710	710
550	600	550	660	1370	1040	660	710
600	610	610	550	1210	1050	660	770
600	550	550	610	1150	990	650	710
550	600	600	540	1150	990	710	710
660	600	660	550	1200	990	660	710

List of used time to get cached images and HTML pages (Continue)

(Miliseconds)

TEST PAGE2	DOCS	TOOLS	H_LOGO	BACK GROUND	H_BROWSE	H_SAMP	POWER
1040	710	660	710	770	660	660	660
1050	660	710	710	770	660	660	660
990	610	660	710	770	660	660	660
1160	660	710	710	770	660	660	660
990	660	710	660	770	660	660	720
1040	760	760	880	880	720	720	710
990	660	660	710	710	710	660	720
1050	710	710	760	760	710	710	720
1050	660	660	710	770	660	660	660
990	660	710	720	770	660	710	660
1160	660	660	710	710	710	660	720
1100	660	720	660	720	720	660	710
1150	720	710	720	770	660	710	660
980	660	710	710	710	710	660	660
1150	600	660	720	770	650	650	660
1100	660	710	710	770	660	660	660
1040	770	720	720	720	660	710	660
1050	660	660	710	710	660	660	660
990	660	720	770	770	660	710	660
1210	610	720	710	710	660	660	720
1040	710	660	770	770	660	720	710
1050	650	660	710	760	660	660	660
1050	650	650	710	710	720	660	660
1210	660	720	720	660	710	650	710
1040	660	720	720	770	720	660	710
990	710	720	770	820	720	710	710
1100	660	660	660	720	660	660	660
990	660	660	720	720	720	720	660
990	660	660	710	710	660	710	660
1160	650	650	760	760	720	660	660

Identify Mean and Std. Deviation of used time to get cached image

	N	Mean	Std. Deviation
DOG1	30	587.33	35.42
DOG2	30	588.67	34.11
E300	30	572.00	31.78
E320W	30	595.33	40.91
GROUP	30	1263.67	118.07
TESTPAG1	30	1038.33	54.34
GG2	30	662.33	25.15
ELCFLOWE	30	741.33	28.74
TESTPAG2	30	1063.33	70.43
DOCS	30	669.33	38.05
TOOLS	30	690.00	31.29
H_LOGO	30	722.33	40.83
BACKGROU	30	750.00	41.61
H_BROWSE	30	682.00	28.33
H_SAMP	30	677.00	26.15
POWER	30	680.00	26.91

Identify interval time of used time to get cached images

			95% Confidence Interval of the Difference	
	df	Mean Difference	Lower	Upper
DOG1	29	587.33	574.11	600.56
DOG2	29	588.67	575.93	601.40
E300	29	572.00	560.13	583.87
E320W	29	595.33	580.06	610.61
GROUP	29	1263.67	1219.58	1307.76
TESTPAG1	29	1038.33	1018.04	1058.62
GG2	29	662.33	652.94	671.72
ELCFLOWE	29	741.33	730.60	752.06
TESTPAG2	29	1063.33	1037.03	1089.63
DOCS	29	669.33	655.13	683.54
TOOLS	29	690.00	678.31	701.69
H_LOGO	29	722.33	707.09	737.58
BACKGROU	29	750.00	734.46	765.54
H_BROWSE	29	682.00	671.42	692.58
H_SAMP	29	677.00	667.24	686.76
POWER	29	680.00	669.95	690.05

Percentage of reduce time

Filename	Used time (Part 1) (Miliseconds)	Used time (Part 3) (Miliseconds)	% of Reduce time
DOG1	1130.33	587.33	48.04
DOG2	1042.33	588.67	43.52
E300	1072.04	572	46.64
E320W	1087.66	595.33	45.27
GROUP	2060.81	1263.67	38.68
TESTPAGE1	3851.01	2441.99	36.59
TESTPAGE2	9264.35	5933.99	35.95

APPENDIX IV

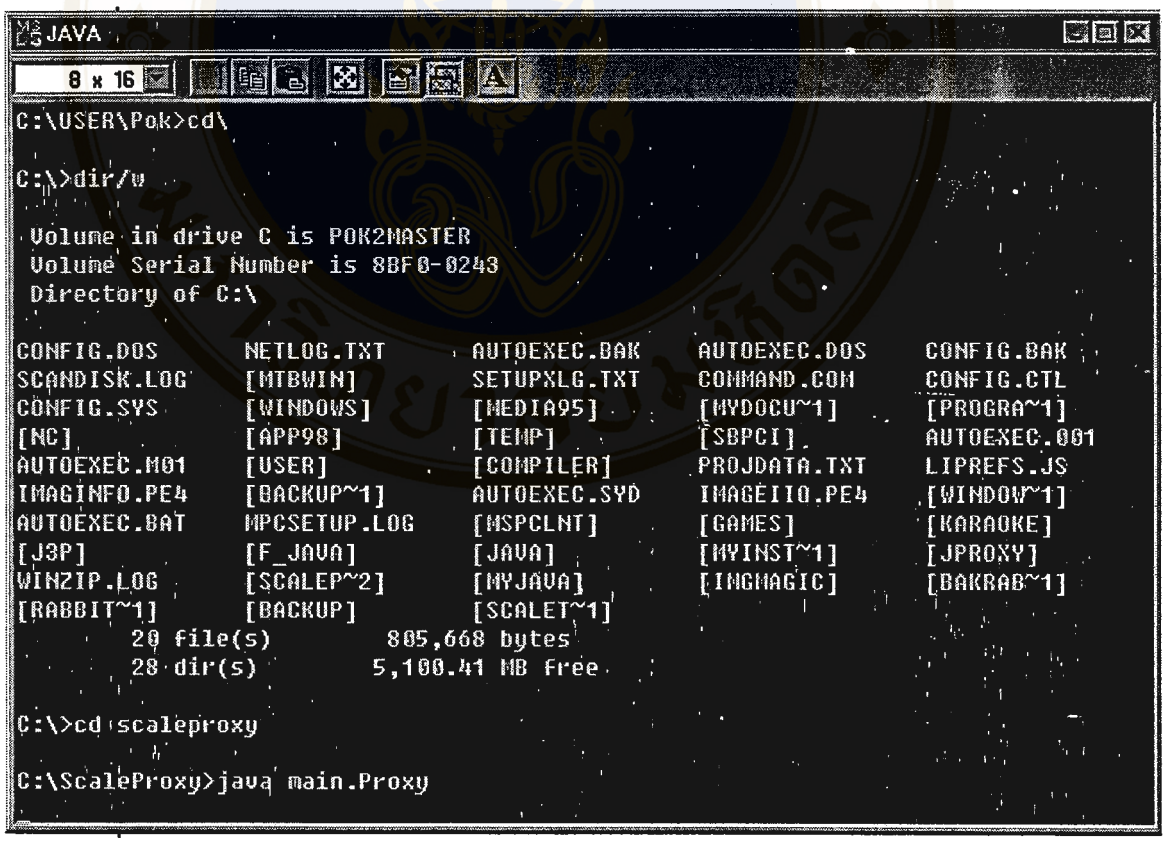
MANUAL

Manual of Scale down proxy server

1. Installation

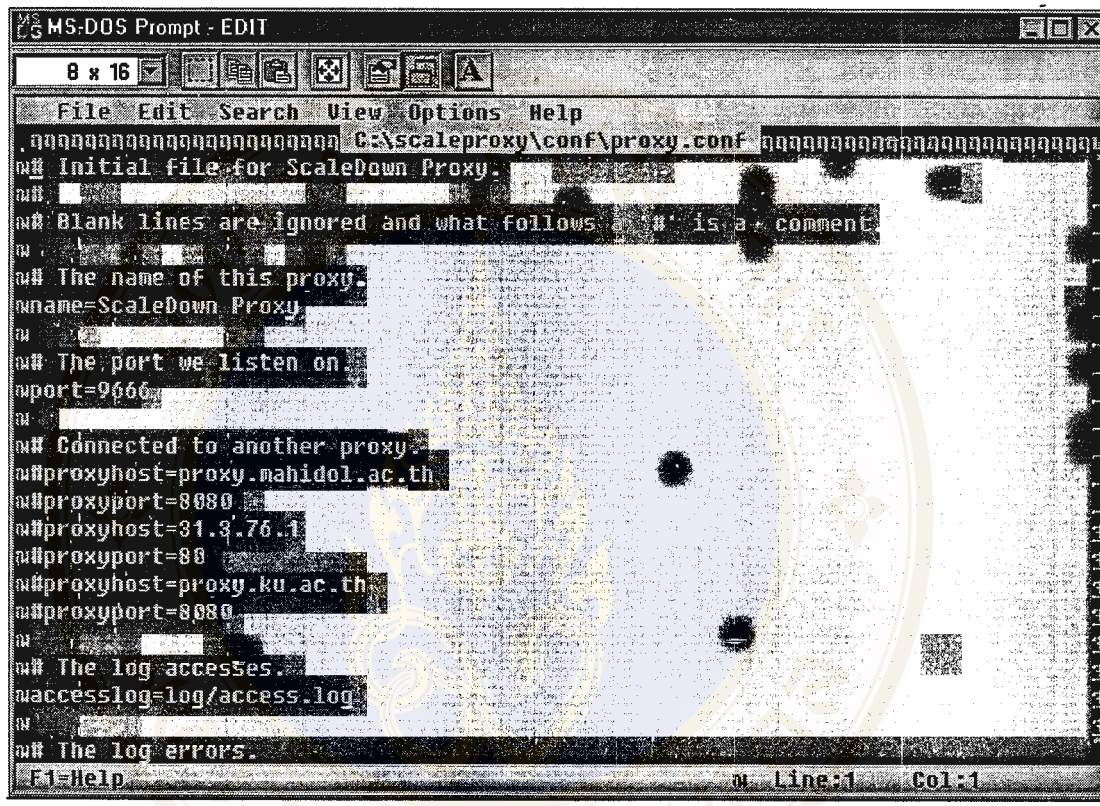
- Install Java Compiler in HardDisk
- Copy Scale down proxy server program in Directory ScaleProxy

2. Run program



- Change directory to ScaleProxy directory
- Type [java main.Proxy]

3. Change initial file

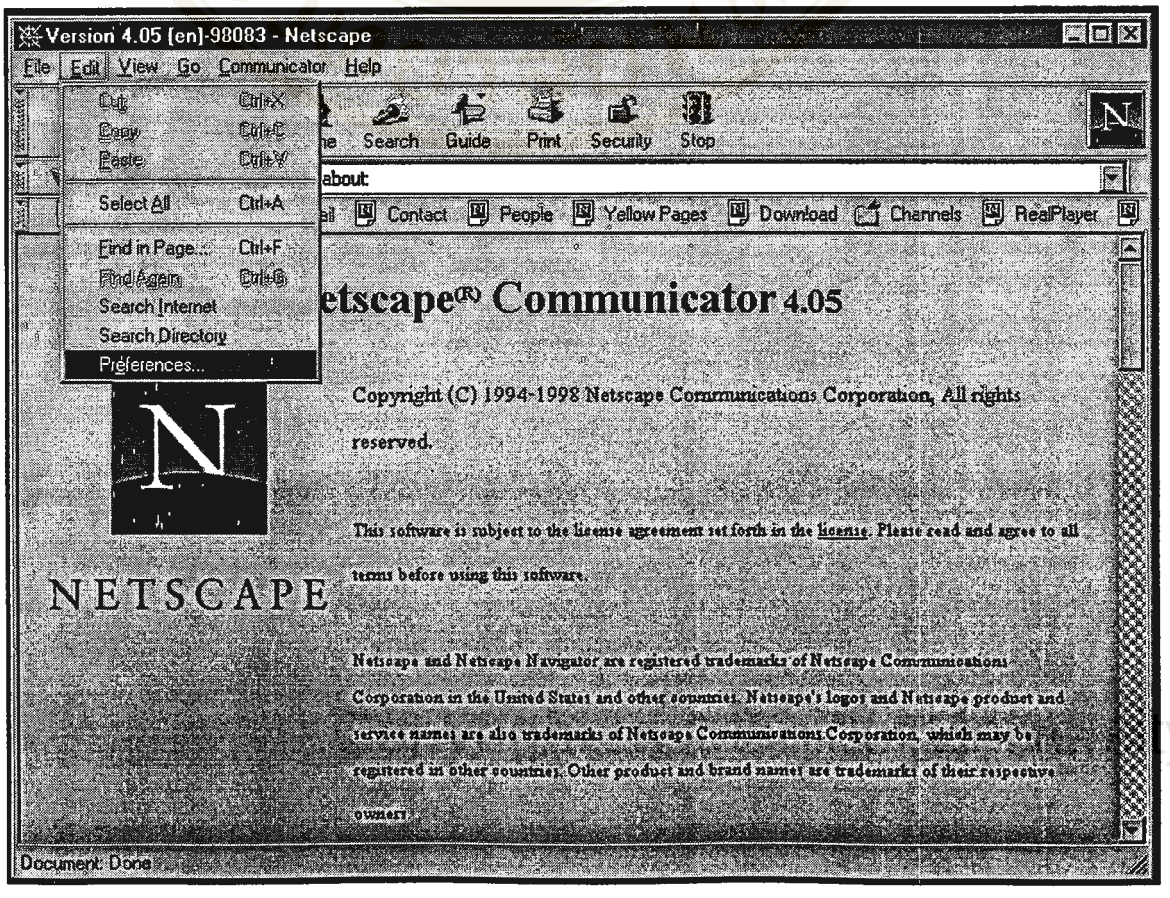


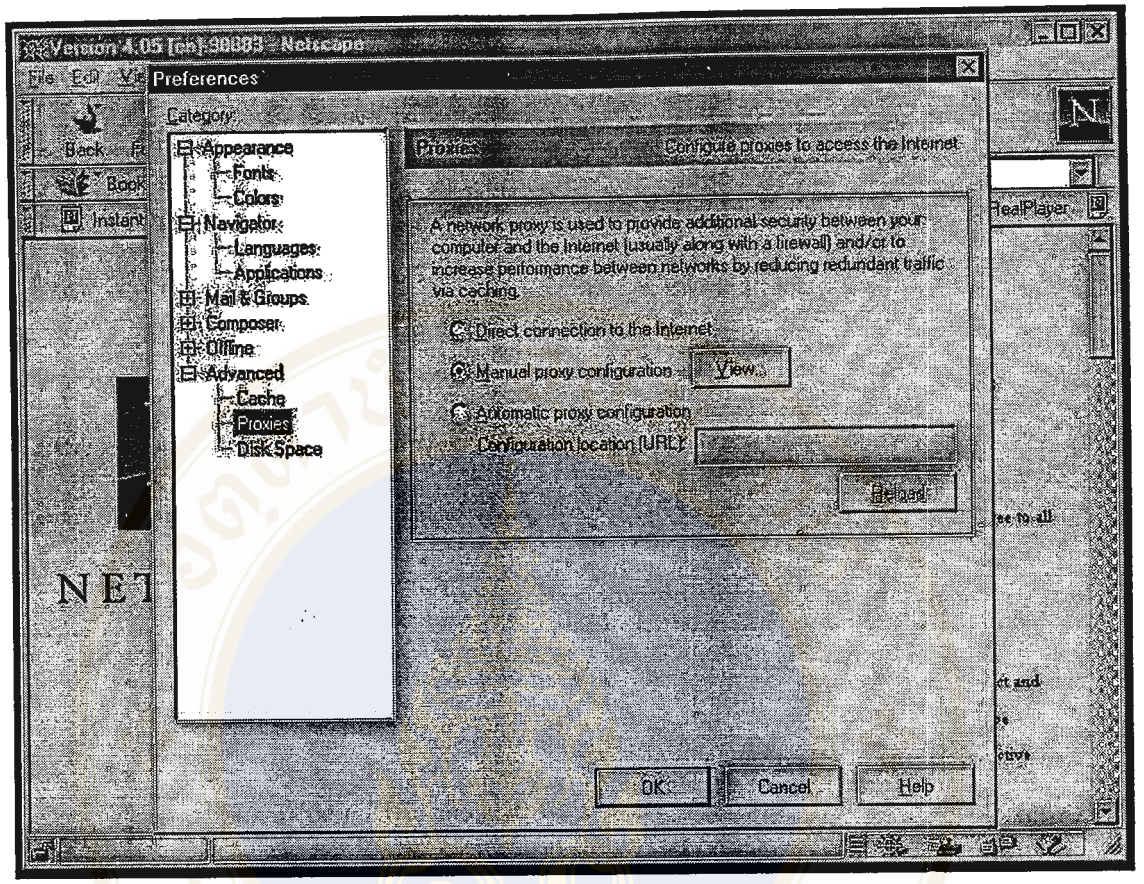
- Before change initial file, ScaleProxy won't run.
- Change directory to conf directory
- Use text editor to change value in initial file

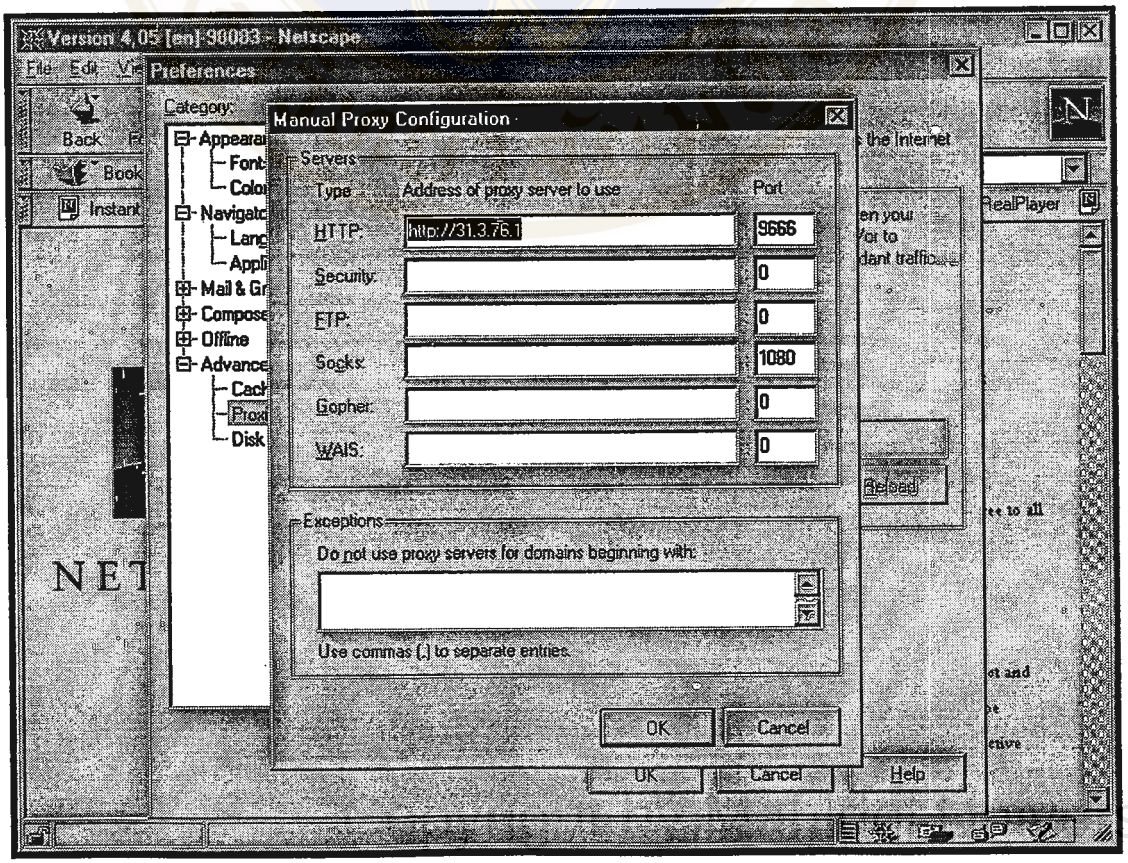
4. Set Scale down proxy server in browser

- Run Netscape Communicator 4.05
- Click Menu <EDIT>
- Click <Preferences>

- Select Category <Advanced>



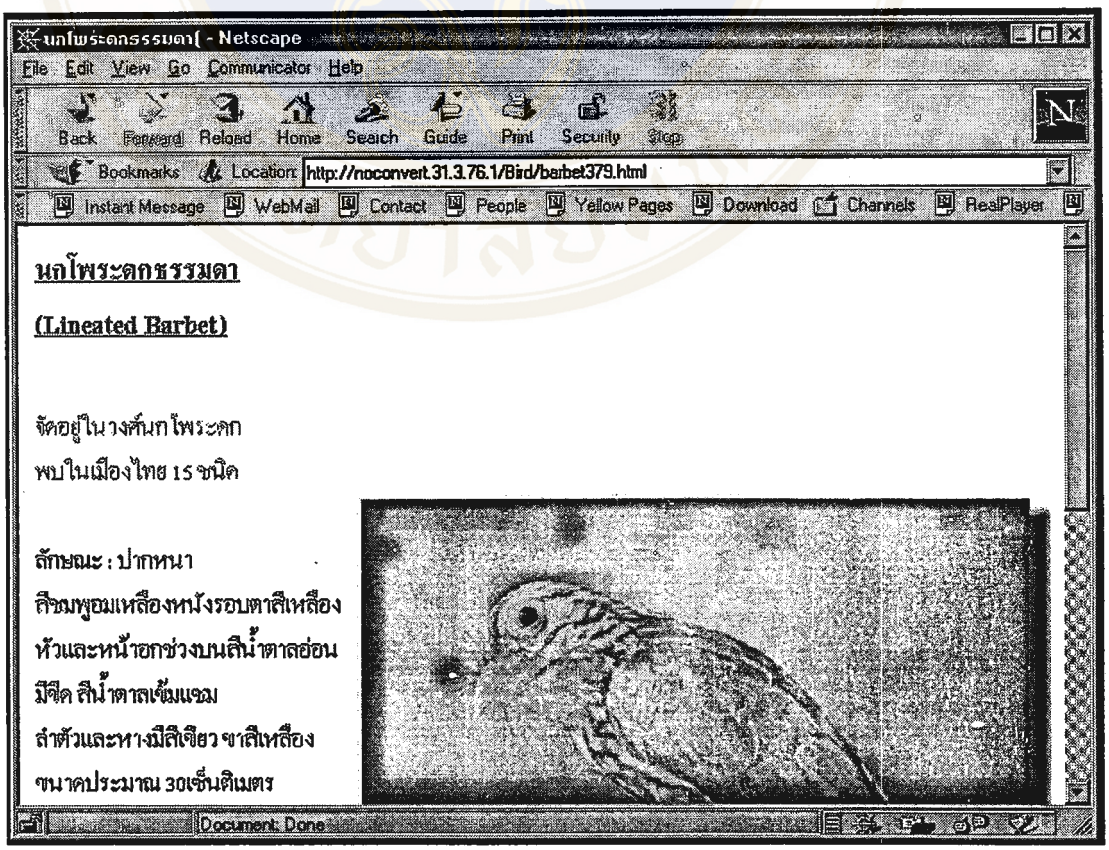

Preferences
Category
Appearance
Fonts
Colors
Navigator
Languages
Applications
Mail & Groups
Composer
Offline
Advanced
Cache
Proxies
Disk Space
Proxies
Configure proxies to access the Internet
A network proxy is used to provide additional security between your computer and the Internet (usually along with a firewall) and/or to increase performance between networks by reducing redundant traffic via caching.
Direct connection to the Internet
Manual proxy configuration
View...
Automatic proxy configuration
Configuration location (URL):
Reload
OK
Cancel
Help


Preferences
Manual Proxy Configuration
Servers
Type
Address of proxy server to use
Port
HTTP:
http://31.3.76.1
9666
Security:
0
FTP:
0
Socks:
1080
Gopher:
0
WAIS:
0
Exceptions
Do not use proxy servers for domains beginning with:
Use commas (,) to separate entries.
OK
Cancel
OK
Cancel
Help

- Select <Proxies>
- Select <Manual proxy configuration>
- Click <View> button
- Type hostname or IP address of Scale down proxy server in <HTTP address> and port of Scale down proxy server in <HTTP port>
- Click <OK> button

5. Using command extension "noconvert."

- Type command extension "noconvert" in <Location> in front of URL after that we will receive original image



BIOGRAPHY

NAME	Mr. Suparerk Pisuchpen
DATE OF BIRTH	31 March 1976
PLACE OT BIRTH	Bangkok, Thailand
INSTITUTIONS ATTENDED	King Mongkut's Institute of Technology North Bangkok, 1993-1997: Bachelor of Science (Applied Computer Science)
	Mahidol University, 1997-2000: Master of Science (Technology of Information system management)