Efficient Information Retrieval From
The World Wide Web

by

Bing He

A thesis submitted in conformity with the requirements
for the Degree of Master of Science
Graduate Department of Computer Science
University of Toronto

© Copyright by Bing He 1998
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

0-612-33938-6
To my grandparents, Liu, Wei-Bang and Lu, Lan-Ying
To my parents, He, Zhi-Liang and Liu, Xi-Fang
Efficient Information Retrieval From The World Wide Web

Bing He

Master of Science 1998
Graduate Department of Computer Science
University of Toronto

Abstract

With the explosion of information that is currently available through the World Wide Web (the Web), searching for and retrieving anything has become an increasingly daunting task. For this reason, many different search engines such as AltaVista, Excite, HotBot, InfoSeek, Lycos, Yahoo, etc, have been built to help Internet users. These search engines employ a program called Web Robot which searches and retrieves any new or updated documents from many Web sites, then passes the retrieved documents to the index server. The Web Robot is also used for other purposes such as statistical analysis and mirroring Web sites. The wide popularity of using the Web Robot resulted in increased usage of the limited Internet resources, especially the network bandwidth. Because of this, the results that are delivered by the Web Robot are far from satisfactory.

In this thesis, we first discuss in detail the pros and cons of using the Web Robot. Then we introduce the Distributed Information Gatherer (DIG), a system that provides an efficient and effective method of continually gathering information from the Web, as a way to improve the efficiency in utilizing the limited Internet resources. Using the DIG specifications, we present the results of an experiment to analyze the utilization that could be achieved through the deployment of the DIG over the Internet. Finally, we generalize the ideas behind the system and discuss the deployment issues of this system.
Acknowledgment

I would like to thank Professor Alberto O. Mendelzon, my supervisor, for his continuous support, advice and feedback. Without his guidance, this thesis could not have been finished. I would also like to thank Professor Charles Clarke, my second reader, for his valuable recommendations with regard to this thesis.

I would also like to express my gratitude to all the members in the Databases group and Information System group, because I have benefited tremendously from my associations with them.

The nature of this undertaking imposed a tremendous burden on my family that often went unrecognized. I greatly appreciate their contribution of patience and understanding to my work and study at University of Toronto.

During my years of study, I missed the chances to visit my greatest mentors, my grandfather Liu, Wei-Bang and my grandmother Lu, Lan-Ying before they left me forever. I dedicate this thesis to their memory.
# Contents

1 Introduction
   1.1 Motivation .......................................................... 1
   1.2 Thesis Organization ............................................... 2

2 Review
   2.1 An Overview of Key Terms ......................................... 4
      2.1.1 The Internet and The Web .................................. 4
      2.1.2 HTTP .......................................................... 5
      2.1.3 URL ........................................................... 6
      2.1.4 HTML .......................................................... 7
      2.1.5 Java ........................................................... 7
   2.2 Related Work ....................................................... 7
      2.2.1 ALIWEB ....................................................... 8
      2.2.2 Harvest ...................................................... 8
      2.2.3 Archie ....................................................... 9

3 Web Robot
   3.1 Definition .......................................................... 10
      3.1.1 Exclusion or Inclusion .................................... 11
      3.1.2 Changed Documents ....................................... 12
      3.1.3 Notification ................................................. 14
   3.2 The Cost of Using Web Robots .................................. 14
# Distributed Information Gatherer

## 4.1 Distributed Information Gatherer

- Local Robot ........................ 18
- Intermediate Results .................. 19
- Notification method ................... 20
- Passive Web Robot .................... 21

## 4.2 Potential Benefits of Distributed Information Gatherer

## 4.3 Distributed Information Gatherer vs. Harvest

## 4.4 Design and Implementation Details

- The Intermediate Results .......... 25
- Master Site Index Configuration File .... 28
- The index Directory .................. 30
- Notification Method ................. 31
- The Retrieval Procedure ............ 31

# Experimental Analysis

## 5.1 Building the Test Application

- Restrictions and Limitations ........ 34
- Test Environment ..................... 34
- The Selection of Web Sites .......... 35
- The Programming Language and Compression Algorithm .... 36
- The Overhead of Using ZIP Archives .... 37

## 5.2 Running the Test Application

- The Test Robot ....................... 38
- The Analysis Program ................. 40

## 5.3 Full-Index Set

- The Size of the Full-Index Set ........ 40
- Document Sizes ...................... 43
- 404 Not Found ....................... 46
- Depth ................................ 47
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>robots.txt file from Web site of Sun Microsystems, Inc.</td>
<td>13</td>
</tr>
<tr>
<td>4.1</td>
<td>siteidx.txt sample file for the first local index run</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>siteidx.txt sample file for the second local index run</td>
<td>30</td>
</tr>
<tr>
<td>4.3</td>
<td>siteidx.txt sample file with split data files</td>
<td>30</td>
</tr>
<tr>
<td>4.4</td>
<td>The URL of sample files in the index directory</td>
<td>31</td>
</tr>
<tr>
<td>4.5</td>
<td>Retrieving algorithm for Web Robot</td>
<td>32</td>
</tr>
<tr>
<td>5.1</td>
<td>The document size distribution</td>
<td>45</td>
</tr>
<tr>
<td>5.2</td>
<td>Correlation of the Depth to the Number of Documents</td>
<td>48</td>
</tr>
<tr>
<td>5.3</td>
<td>The graphs of Number of Documents versus Depth at the visited Web sites</td>
<td>50</td>
</tr>
<tr>
<td>5.4</td>
<td>The graphs of MB of Uncompressed Data versus Depth at the visited Web sites</td>
<td>51</td>
</tr>
<tr>
<td>5.5</td>
<td>Document and data distribution over the entire collection of information</td>
<td>52</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Search Engine Comparison ........................................ 11

4.1 Record structure for the Entire Site Index File ............... 26
4.2 Status Flag of documents that have changed .................. 27

5.1 Server configurations .............................................. 34
5.2 A list of Web sites that are selected for visit .................. 36
5.3 Summary of the information about Full-Index Set .............. 41
5.4 Data reduction at each Web sites by using the Full-Index Set .... 42
5.5 Summary of the information on retrieved documents ............ 43
5.6 The distribution of document size ............................... 44
5.7 The document depth information ................................... 47
5.8 home.netscape.com Web site ...................................... 54
5.9 www.w3.org Web site .............................................. 55

A.1 List of File Extensions ............................................. 61
Chapter 1

Introduction

1.1 Motivation

Tim Berners-Lee [1] started the current World Wide Web (WWW or the Web) revolution, by seeking to build a distributed hypermedia system. There is no central storage site that holds all the available documents on the Web. Instead, a vast collection of interconnected documents is stored on different Web sites all over the world, and the Web sites themselves are interconnected by a common shared network. Virtually every piece of information available on the Web can be found through browsing and traversing the hyperlinks. Most documents on the Web are written in Hypertext Mark-up Language (HTML)(See Section 2.1.4). This language allows the author to specify certain formatting styles, to embed images and sounds and to specify links to other documents.

In a walk of the Web, performed by AltaVista [2] in April, 1996, 30 million HTML and text documents were collected. It took about 150 GB to store all the collected data. However, this was only a snapshot of the Web. New documents are published on the Web everyday. Meanwhile, the existing information is continuously changing. The Web, as an entity, is constantly evolving and growing at an exponential rate. There are an estimated half million to one million Web servers currently running worldwide.

Finding information on the Web is a daunting task. Users can start from any...
CHAPTER 1. INTRODUCTION

HTML document, manually browsing and traversing the hyperlinks. This is very time-consuming for the users. Alternatively, users can query the search engines, which depend on the information that is collected by a computer program called the Web Robot. With the vast amount of information that is available on the Web, more and more users are depending on the search engines in order to find it.

In order for the search engines to respond to the user queries with any relevant information, the Web Robots that are collecting data for the search engines have to constantly roam the Internet to discover and retrieve the information for indexing. However, many users have experienced that the many so-called relevant hyperlinks that are returned by the search engines are 'NOT FOUND'. A lot of new information is not returned by the search engines because the Web Robots are not able to get to it on time. Despite the millions and millions of documents that are discovered and retrieved by the Web Robots, they cannot keep up with the demand.

The continuous search and retrieval by Web Robots place a huge demand on the Internet bandwidth. The network bandwidth of the Internet is a limited resource, which has not been able to keep up with the growth of the Internet. It becomes very important therefore to be able to reduce the network traffic that is generated by the Web Robots.

For search engines to serve the Internet community better, the Web Robots have to be able to discover and collect the information in an efficient and effective manner, and with much less demand on the Internet resources. This is what motivated the formulation of this thesis.

1.2 Thesis Organization

This thesis is organized as follows:

- Chapter 2, Review, is an overview of the key terms that are used in this thesis and of some related works in the area of distributed information retrieval.
• Chapter 3, **Web Robot**, is an analysis of the current *Web Robot* in detail, focusing on its pros and cons.

• Chapter 4, **Distributed Information Gatherer**, is an introduction to the *Distributed Information Gatherer (DIG)*. We compare the DIG with the *Web Robot* and *Harvest* system [19], and analyze the potential benefits of implementing the DIG. In the second part of this chapter, we present the design and implementation details of the DIG.

• Chapter 5, **Experimental Analysis**, presents an experiment designed according to the DIG specifications. We analyze and discuss the results from the experiment.

• Chapter 6, **Conclusions**, draws conclusions and suggests future developments.
Chapter 2

Review

In the first half of this chapter, we give an overview of the key terms that are used in this thesis. In the second half, we present several related works in the area of distributed information retrieval.

2.1 An Overview of Key Terms

2.1.1 The Internet and The Web

The Internet is a network of networks. The term Internet comes from the wide-area network (WAN) concept of internetworking, which connects many heterogeneous networks together. The hardware used to build the Internet has been upgraded several times since its initial development in the 1970’s, each upgrade providing more capacities and faster processing. However, it still lags behind the growth of the Web.

By using hypertext and hypermedia [3], the Web makes the exchange of information much easier. The underlying idea is that documents can be linked at various points according to natural relationships so that the reader can jump from one document to another, following the appropriate path for that reader’s needs.
2.1.2 HTTP

HyperText Transfer Protocol (HTTP) [4, 5] is specially designed for the rapid and efficient delivery of hypertext materials over the Internet. It is an application-level protocol for distributed, collaborative and hypermedia information systems. HTTP is a generic, stateless, object-oriented protocol [5]. HTTP allows the systems to be built independently of the data being transferred. HTTP communication currently uses Transmission Control Protocol / Internet Protocol (TCP/IP) connections. However, other protocols or networks can also be used to implement the HTTP.

HTTP has been in use since 1990. The first version of HTTP, referred to as HTTP/0.9, was a simple protocol for raw data transfer across the Internet. A improved version HTTP/1.0 [4] is widely use at this time. One of the new features is that it allows the Multipurpose Internet Mail Extensions (MIME) [6] messages to be used. One of the shortcomings for these two versions of HTTP is that a separate TCP connection has to be established to handle each request. This increases the load on the HTTP servers and causes congestion on the Internet. Several analyses of HTTP performance problems [7, 8, 9, 10] are listed in the bibliography.

The round trip time (RTT) is the time taken to send a packet from the user end of the connection to the Web server and back. The time required by HTTP/1.0 to transfer a file is calculated as follows: 1 RTT is used to open the TCP communication channel between the client and the server; 0.5 RTT is required for the client to send the request to the server; and 0.5 RTT is required for the first packet to be sent back from the server to the client. Additional time is required to transmit the rest of the file if the file is larger than one packet. HTTP/1.0 requires at least two RTT for each call [9]. For smaller HTML documents, the RTT becomes a significant overhead for each HTTP call.

The next version of the HTTP protocol, referred to as HTTP/1.1 [5], includes more stringent requirements than HTTP/1.0 in order to insure reliable implementation of its features. One of the new features is the proposed persistent connections. The persistent connections allow a single TCP connection to serve multiple requests, and
they also will reduce the RTT required overall [9].

2.1.3 URL

URL stands for *Uniform Resource Locator* [11]. It is a scheme for locating Internet resources by using a string of printable ASCII characters. In general, URLs are written as:

\[ < \text{scheme}> : < \text{scheme-specific-part} > \]

Many different schemes, such as *ftp, mailto, news, telnet, wais* and *file*, just to name a few, can be expressed by the URL. In this thesis, we focus on the URL scheme for the HTTP protocol. The formal syntax of the URL scheme for HTTP is:

\[ < \text{http}> : / / < \text{Host} > : < \text{Port} > / < \text{Path} > / < \text{Filename} > [ < \text{options} > ] \]

*Host* is the fully qualified Internet domain name [12, 13] of the network host site, or its IP address\(^1\) as a set of four decimal digit groups separated by the period. For example, the fully qualified domain name for the Web server of the *Department of Computer Science, University of Toronto* is *www.cs.utoronto.ca*, and its IP address is *128.100.2.99*.

*Port* is the port number of the server to be connected. If this is omitted, a commonly understood default value of *80* is assumed, and the colon is omitted as well.

*Path* is the location of the resource in the hierarchical structure of the Web server.

*File Name* is the specified file name on the host.

*Options* is used to specify optional attributes or values.

\(^1\)Internet Protocol Address is a 32-bit address defined by the Internet Protocol in RFC 791 [14].
CHAPTER 2. REVIEW

2.1.4 HTML

*HyperText Markup Language (HTML)* [15, 16] is a simple markup language, that uses instructions embedded within a document to describe how the document should be formatted for viewing by browser software. HTML has important extensions that allow for hypertext links from one document to another. Especially, *Anchor Tag* (<A>) is used to specify the URL of another document. These links create the Web of documents. HTML documents are stored on the Web server as text files. They can be read and edited by any text file editors.

The original HTML specification was written by Tim Berners-Lee while he was at the European Laboratory for Particle Physics (CERN). Since then, three HTML versions have been published - version 2.0 in November 1995, version 3.2 in January 1997 and version 4.0 in December 1997. Many additional features were added into these versions.

HTML is continually evolving. Many new features and enhancements are being included into the next HTML release.

2.1.5 Java

The *Java™* [17] programming language (Java) is a platform-independent object-oriented programming language, developed by Sun Microsystems, Inc. It has inherited many of the best features of object-oriented languages. Java comes with a rich library of predefined packages of classes, which make Internet applications much easier to develop. Java is used in this thesis to develop an application to test the proposed standard.

2.2 Related Work

Many different distributed index tools have become available throughout the Internet. There are two methods that are employed by these tools to collect information. The first method is to use the standard Internet protocols to traverse the Internet...
sites with no assumption about data types and formats. The widely used *Web Robot* employs this method. The *Web Robot* is analyzed in detail in Chapter 3. The second method is to collect pre-summarized local index information in some pre-defined format. In this section, we discuss several tools that use one or both of these two methods.

### 2.2.1 ALIWEB

*ALIWEB* [18] proposes that each Web site produce a site description file (*site.idx*) to contain information about the server, organization, documents, available services and users at the site. An index server retrieves this description file from every Web site, and then combines them into a searchable database.

However, this description file is manually produced. To construct this description file for a Web site that has thousands of documents and users is very time consuming. Secondly, this is only a description file, and the data still has to be retrieved by the *Web Robot*. Thirdly, *ALIWEB* does not have any process to manage the task imposed by constant changing of the data.

### 2.2.2 Harvest

The goal of the *Harvest* system [19] is to provide an integrated set of customizable tools for gathering information, building topic-specific content indexes, replicating indexes, and caching objects as they are retrieved across the internet. The motivations for building the *Harvest* system are the same as the motivations of this thesis, such as reducing network traffic, reducing the server load at Web sites and providing support for the incremental updates.

At the heart of the *Harvest* system is the *Gatherer* sub-system and the *Broker* sub-system. The *Gatherer* collects and extracts indexing information from one or more Web sites. The *Broker* retrieves the information from one or more Gatherers or other Brokers, and updates the many different indexes. The *Broker* also provides a query interface to the gathered information. The *Gatherer* can reside in the provider
host, and pass the pre-collected summarized information to the Broker. The Harvest system becomes another peripheral index/search site.

One of the reasons that the Harvest system can reduce the network traffic substantially, is that it uses its own selection techniques to extract content summaries from the sources, and provides the content summaries to others to be used for indexing. The content summaries hold a much smaller set of data than the original data. One of the test cases that is used to demonstrate the significant data reduction of content summaries is that 111 MB worth of content summaries is extracted from 2.44 GB of data distributed around the Internet, totaling a 95.6% reduction.

The information that the Harvest system provides to others is found in the form of summarized, selected and extracted indexes. Many other index/search engines require unscreened original data from each Web site. They do not want to limit themselves to any specific topics or community, because they want a wide area of data from the Internet. They also have their own methods and techniques of selecting, extracting and creating master indexes.

2.2.3 Archie

Many FTP\textsuperscript{2} site administrators have prepared a preprocessed site directory listing, usually stored in a file named ls-lR, ls-lR.gz or ls-lR.Z. The format of this listing is usually similar to the result of using dir -R command under ftp. At most FTP sites, this site listing is kept in its root directory. The FTP site listing is the cornerstone of the success for the Archie [21] index system. The Archie depends heavily on the individual FTP site administrator to produce a listing in order to drastically reduce the amount of fetch operations that Archive must perform.

There is no incremental index method. If the information at any FTP site is changed, the entire file of ls-lR is retrieved. However, because the Archie index system is only doing file name indexing, the size of the site listing is not very large.

\textsuperscript{2}File Transfer Protocol.
Chapter 3

Web Robot

In this chapter, we present a detailed analysis of the Web Robot, especially its pros and cons.

3.1 Definition

The task of collecting information from the Web is currently done by the Web Robot. A Web Robot is a computer program that uses standard Web protocols to traverse the Web's hypertext structure automatically, starting from some known documents or Web sites, then recursively retrieving all documents that are referenced. The goal of the Web Robot is to discover what is out there on the Web and to determine if it should be retrieved and passed on to some other programs for further processing.

Since the Web Robot program can be easily implemented, it has become a primary tool for the data collection and resource discovery over the Web. By the last count [25], over 150 different Web Robot programs are roaming the Web. The collected data is being used for statistical analysis, maintaining broken hyperlinks, mirroring Web sites, resource discoveries and building Web index/search facilities.

Many popular Web search engines, such as AltaVista [29], Excite [30], HotBot [31], InfoSeek [32], Lycos [33], NorthernLight [34], WebCrawler [35], to name a few, are using the Web Robots to collect data from the Web. Table 3.1 compares some key
features of these search engines as of November 1997 [23].

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>Indexed Pages</th>
<th>Daily Retrieved Pages</th>
<th>Freshness</th>
</tr>
</thead>
<tbody>
<tr>
<td>AltaVista</td>
<td>100 Million</td>
<td>Up to 10 Million</td>
<td>1 day to 3 months</td>
</tr>
<tr>
<td>Excite</td>
<td>55 Million</td>
<td>3 Million</td>
<td>1 to 3 weeks</td>
</tr>
<tr>
<td>HotBot</td>
<td>80 Million</td>
<td>Up to 10 Million</td>
<td>1 day to 2 weeks</td>
</tr>
<tr>
<td>InfoSeek</td>
<td>30 Million</td>
<td>N/A</td>
<td>Minutes to 2 months</td>
</tr>
<tr>
<td>Lycos</td>
<td>30 Million</td>
<td>6 to 10 Million</td>
<td>1 to 2 weeks</td>
</tr>
<tr>
<td>NorthernLight</td>
<td>30 to 50 Million</td>
<td>N/A</td>
<td>2 weeks</td>
</tr>
<tr>
<td>WebCrawler</td>
<td>2 Million</td>
<td>N/A</td>
<td>Updated weekly</td>
</tr>
</tbody>
</table>

Table 3.1: Search Engine Comparison

The goal of the Web Robot is to retrieve all the documents from a Web site on the first visit. On the follow up visit, the Web Robot is only interested in the changed documents between two visits. The questions that a robot needs to have answered by the individual Web site are:

1. Which documents should be included in the indexing, and where are they?

2. Which documents were added/removed/changed since my last visit, and where are they?

3. Could the Web Robot be informed about any change that takes place?

### 3.1.1 Exclusion or Inclusion

An unofficial Robot Exclusion Standard [24] currently exists. It contains information about parts of the Web server that should not be accessed, i.e., which pieces of information should be excluded. This unofficial standard is used as a mechanism to keep visiting robots out of restricted areas of the Web server. A robots.txt file is placed in the root of the Web server for retrieval by the visiting Web Robot. Figure 3.1 shows an example of the robots.txt file. A detailed list of what should be included in the robots.txt file is available [26]. The Robot Exclusion Standard does not provide any straightforward answers to the first question.
The visiting Web Robot has to (i) retrieve the Home Page\textsuperscript{1} document from the Web site, (ii) parse it to find out all the local and foreign hyperlinks, (iii) compare these local hyperlinks with the disallowed area in the Robots Exclusion file, and then (iv) repeat the above process until no more local hyperlinks are reached or a certain depth is reached. The Web Robot not only has to retrieve the files, but also it has to read and parse the file in order to find out additional local hyperlinks. In a sense, it has to dig out the information from the Web site. Not all HTML files conform to the HTML standard. Parsing them will produce errors that will cost the Web Robot to miss some local and foreign hyperlinks, and to miss some documents that should be included in the indexing.

From the point of view of the Web site, it knows which files should be included for indexing if it knows what should be excluded. In addition, it knows where the files are stored. It is relatively painless for the Web site to produce a listing, containing all the files that should be indexed, and placed in a standard location for the visiting Web Robot to pick up. However, the question arises: Is there enough incentive for the Web site to do it this way?

3.1.2 Changed Documents

On the repeat visit, the Web Robot does not want to retrieve the entire Web site again if it can be avoided. The only thing it wants to know is which files were added/deleted/changed since the last visit. Currently, there is no easy way of getting this information, because there is no single request that can determine if a document is added, removed, or changed.

The HTTP protocol does provide the If-Modified-Since mechanism, whereby the user-agent can specify the modification time-stamp of a cached document along with a request for the document. The Web server will then only transfer the contents if the document has been modified since that time. Otherwise, a header is send back with the information that there is no change to the requested file. This method does reduce

\textsuperscript{1}The Home Page is the first page of a Web site.
# /robots.txt for www.sun.com

# Aug 22 11:10:08 PDT 1997, Fred.Elliott@eng
# o Added path to new download site.
#
# Thu Jan 30 16:58:19 PST 1997, Fred.Elliott@eng
# o Created this file to prevent indexing of one
# SME directory.

User-agent: *

Disallow: /sparc/SPARCengineUltraAX/oem/
Disallow: /microelectronics/SPARCengineUltraAX/oem/
Disallow: /javachip/SPARCengineUltraAX/oem/
Disallow: /javachips/SPARCengineUltraAX/oem/

Disallow: /sparc/SPARCengineUltraAX/download/
Disallow: /microelectronics/SPARCengineUltraAX/download/
Disallow: /javachip/SPARCengineUltraAX/download/
Disallow: /javachips/SPARCengineUltraAX/download/

Figure 3.1: robots.txt file from Web site of Sun Microsystems, Inc.
to some degree the amount of data that the individual Web site has to send back to the Web Robot, but it does nothing to the total number of requests that have to be handled by the Web site. However, not all Web Robots are using the If-Modified-Since field, and also not all Web servers can handle this mechanism either.

The knowledge about the changed documents is available at the Web site, because someone at the site has to physically perform the tasks, i.e. add the file to the Web server, change the file at the Web server, or delete the file at the Web server.

It is relatively painless for the Web site to produce a listing, containing the information about what was changed since some previous date, and to place it in a standard location for the visiting robot to pick up. However, the question arises: Is there enough incentive for the Web site to do it this way?

### 3.1.3 Notification

There is no standard mechanism to inform the index server regarding when a change has occurred at the individual Web site. The Web Robot does not know and has no reliable way of predicting when the information on the Web site is going to change. Thus, it has to periodically re-visit these Web sites to check. However, the individual Web site does know when the content at its Web site has been changed. Currently, there is a URL link to the manual submission form at many search engine sites, for people to manually submit information about changed pages or new sites. There even exists a Web site, http://www.submit-it.com/, to help users to make the submission to all the search engines. Manual submission is not a very effective method, because it cannot keep up with the constant evolution of the Internet. What is needed is an automatic notification method that is agreed upon by both parties.

### 3.2 The Cost of Using Web Robots

Despite the wonderful work that Web Robots can do for us, there are large costs that are associated with using robots.
CHAPTER 3. WEB ROBOT

Number of HTTP Requests

HTTP is a stateless connection. For every Web Robot request, a separate process is created at the individual Web server to handle the newly created TCP connection. It searches through the file system for the requested object, transmits the object, then finally closes the connection and terminates the process. The Web Robot will generate, at minimum, the same number of HTTP requests as the number of available files in the Web site on every visit. In most cases, there are many unsuccessful HTTP requests due to broken hyperlinks, server errors and network errors.

Demand On The Web Servers

The huge numbers of HTTP requests from the Web Robots create a considerable load on the individual Web server. Depending on the frequency with which the Web Robot requests documents from the individual Web server, this can result in a lower level of service for other Web users accessing the same Web server. This problem repeats with every robot visit.

Demand On The Internet Bandwidth

A large network bandwidth is required to serve just a few robots, as shown in Table 3.1. Suppose that 5 KB is the average document size [22], then 1 million pages is close to 5 GB of data. With the exponential growth of the Web, robots have to retrieve increasingly more data everyday just to try to keep up with the growth of the Web.

Redundant Work

A Web Robot program cannot avoid redundant retrieval of the same documents, because it has no way of knowing if a document is changed or not. The safest way to determine if this document needs to be re-indexed is to retrieve the document and compare it with the previous cached document. This is not a problem if the robot is visiting a new Web site. However, it is a big problem if the robot is re-visiting a
previously indexed Web site. Large amounts of data are being retrieved through the Internet and then discarded because they did not change.

Due to an increasing number of documents being stored at the individual Web site, most of the Web sites have to provide a site index and a search facility to the user. Many different index techniques are used throughout the Internet. They all have different index structures. If the remote search engines can use the individual site index master file(s), it will greatly reduce the workload on both sides. However, the main problem is that no standards exist to define what can be shared.

**Hardware Requirements**

A powerful computer is required to run the robot program. For example, AltaVista [29] uses separate computers to host the Web Robot program, index server and the query engine. The Web Robot program runs on a DEC4100 5/300 AlphaStation with 1.5GB of RAM and a 30GB RAID array [2], using 100 megabits per second Internet access. However, with the exponential increase of the amount of data available on the Web, the advance in hardware technology may not be able to match its growth.

**Out of Date Query Results**

Millions of documents are retrieved from many Web sites everyday, but there is no guarantee of any degree, regarding how current the indexed information is from any of the search engine sites. Many queries that are returned by the search engines contain out-of-date information because the Web Robots are not able to re-index many Web sites on time. This points to a need for an effective method of discovering and gathering information.
Chapter 4

Distributed Information Gatherer

In the first part of this chapter, we introduce the Distributed Information Gatherer (DIG), then we compare the DIG to the Web Robot and Harvest system, and analyze the potential benefits of implementing the DIG. In the second part of this chapter, we present the detail design and implementation of the DIG.

4.1 Distributed Information Gatherer

The Distributed Information Gatherer is a system that provides an efficient and effective method of continually gathering information from Web sites. DIG does not propose any special network-level protocols or platform-specific solutions. Instead, standard Internet protocols are used. The DIG system contains two parts, a client-side sub-system that runs at individual Web sites, a server-side sub-system that runs at the Web Robot sites.

The client-side DIG sub-system contains two programs, a Local Robot program and an analysis program. The Local Robot is used to traverse the local Web sites and produce a Full-Index Set. The analysis program performs three tasks. The first task is to find the difference between two Full-Index Sets, and to produce a Change Set that contains the difference. The second task is to produce a Master Site Index Configuration File that contains control information about the Full-Index Set, the Change Set(s) and other site-specific information. The third task is to use
the notification method to broadcast the availability of the result sets, so that the server-side of the DIG sub-system can come to pick up these intermediate results.

The server-side DIG sub-system has one program, a Passive Web Robot (section 4.1.4), with which it performs three tasks: the monitoring of a communication channel for the client-side broadcast, the sending of the requests to retrieve and parse the configuration file, and the sending of the requests to retrieve the intermediate result sets.

4.1.1 Local Robot

The Local Robot is a localized version of Web Robot. It only gathers information from a selected few local Web servers. Four tasks from the Web Robot are assigned to the Local Robot at the individual Web site. They are:

1. Find out what is available at each Web site.
2. Decide which documents should be included.
3. Detect which document has changed since the last search by the Robot.
4. Decide when the Web Robot should re-visit this Web site.

The Local Robot can do these four tasks more easily and more efficiently than the Web Robot because it has two advantages over the latter. First, the Local Robot generally has faster access through the local area network, or even has a direct access to the file system in which the documents reside, other than the Internet connections that are used by the Web Robot. Second, the webmaster of the Web site has direct control over the Local Robot, as to when, where, and how it is run and how it is configured.
4.1.2 Intermediate Results

The intermediate results are the specially prepared data files for the visiting Web Robot to use.

Master Site Index Configuration File

The Master Site Index Configuration File is a text file that contains the control information regarding the Full-Index Set, the Change Set(s) and other site-specific information. The intent of this configuration file is to be the first file that a Web Robot requests from a Web site.

Full-Index Set

The Full-Index Set is produced by the Local Robot. It contains two compressed files, an Entire Site Index File and an Entire Site Data File. During the Local Robot runs, any document that is allowed to be indexed by the Web Robot is stored in the Full-index Set as follows: an index record is formed with the several properties of the document and is stored in the Entire Site Index File; the content of the document without any modification is compressed and packed into the Entire Site Data File.

The size of the Full-Index Set depends on two factors: the type of the data that is packed into the Entire Site Data File and the compression algorithm that is used to pack the data. The assumption is that the size of the Full-Index Set is much smaller than the original size of the data. As demonstrated in Chapter 5 of Experimental Analysis, the size of Full-Index Set is about one-third the size of the original data (Only HTML and text file format are considered and the ZIP compression algorithm is used).

The goal is to have the visiting Web Robot retrieve and use the smaller Full-Index Set rather than to do the recursive document retrieving if the Web Robot wants to do a full index of this Web site.
Chapter 4

DISTRIBUTED INFORMATION GATHERER

Change Set

A Change Set is produced by the analysis program. It contains two compressed files, a Changed Index File and a Changed Data File. The key aim of the analysis program is to find out the differences between two successive local robot runs. The different index records are stored in the Changed Index File and the newly added and/or updated documents are compressed and packed in the Changed Data File.

If the final size of the Change Set is larger then the size of the Full-Index Set, the Change Set will not be kept. Although the size of the Change Set depends mainly on how much data were changed between two successive local robot runs, the Change Set is much smaller than Full-Index Set in most cases.

The Change Set is used as the data set for an incremental update of a Web site by the Web Robot and to reduce the amount of data that has to be retrieved in order to re-index this Web site. In order to re-index a Web site, all that Web Robot has to do is to retrieve the Change Set from the Web site if the set is available.

4.1.3 Notification method

After the intermediate result sets are generated by the Local Robot, the web site needs to broadcast the availability of the results. We face the problem of whom should we notify? The solution we propose is to use the existing infrastructure of the newsgroup. A new newsgroup by the name of www.robot.announce should be created. An email with a subject line that contains the starting URL of the web site is the only information posted to this newsgroup to announce the availability of the new result sets.

By posting an email to the www.robot.announce newsgroup, we achieve our goal of announcing to everyone that the Web site has new data. This is exactly what the Web Robot wants to know, so that it can send the requests to retrieve the new data. The Passive Web Robot does not have to send out requests to any other Web site that does not make any announcement.
4.1.4 Passive Web Robot

The Web Robot is actively sending out the requests all over the Internet to seek out and to retrieve documents. A more precise name for this type of robot is Active Web Robot. In contrast, the Passive Web Robot does not actively send out requests all over the Internet. Most of the time, the Passive Web Robot sits back and monitors the broadcast channel for any new broadcast. It sends out retrieval requests to an individual Web site only if there is a broadcast from that Web site. The tasks of the Passive Web Robot are simpler than those of the Active Web Robot. The new tasks are:

- Monitoring the newsgroup of www.robotannounce and parsing the subject line of posted email for the URL of the Web site.

- Retrieving and parsing the Master Site Index Configuration File for information about the Full-Index Set and Change Set.

- Deciding which set of data to retrieve and sending the request to retrieve it.

The retrieved data set is passed on to other program for further processing.

4.2 Potential Benefits of Distributed Information Gatherer

In this section, we compare the DIG with the Web Robot, and present the potential benefits of implementing the DIG and using the New Web Robot.

Number of HTTP Requests

By using the DIG, only a few HTTP requests are required by the Passive Web Robot to index or re-index a Web site. The actual number of requests depends on how the intermediate results are kept at the Web site (See Section 4.4.1), but the number is usually a single digit. The bulk of recursive retrieval document task is shifted from the Web Robot to the Local Robot.
Demand On The Web Servers

The limited number of HTTP requests from the Passive Web Robot will create no load problems for the Web server. Under the DIG, the Local Robot has the task of indexing the local Web site. Because the Local Robot is under the control of local Webmaster, the load problem that generated by the Local Robot will be better managed than the current implementation of Web Robot.

Demand On The Internet Bandwidth

The DIG and the Passive Web Robot use the Internet bandwidth more efficiently than the Active Web Robot. The Passive Web Robot retrieves less data in order to index/re-index a Web site than the Active Web Robot, because the Full-Index Set and Change Set are smaller than the original data. The DIG uses an incremental update method, so that only the data that was changed is requested and sent rather than the data for the entire site during the re-indexing. The Passive Web Robot does not waste any Internet bandwidth to search for new or changed documents over the Internet. By using the Notification Method of the DIG, the Passive Web Robot waits for the information to come.

Redundant Work

The Passive Web Robot does not have to send retrieval requests to any Web site if that Web site does not post an announcement beforehand. Every data set that is retrieved by the Passive Web Robot is useful, nothing is thrown away. If the data set cannot be used, the Passive Web Robot does not have to retrieve it. This decision is made using the information supplied by the DIG in the Master Site Index Configuration File.

Hardware Requirements

With the reduction of the HTTP requests and amount of data required to index a Web site, the Passive Web Robot requires less in term of memory, Internet bandwidth,
processing power and storage space. The retrieved and processed data sets do not have to be cached at the Passive Web Robot site, because the Passive Web Robot does not need these data sets as bases for change comparison.

There is additional storage requirement of the individual Web site in order to store the data sets, the actual amount of disk space varies from one Web site to another. The magnitude of the required disk space by the DIG implementation is in hundreds of megabytes.

**Out of Date Query Results**

The DIG can increase the accuracy of the user query result. This is achieved by using the notification method to keep the Passive Web Robot informed of the change, so that the robot can send requests to retrieve the data. Because the Passive Web Robot can collect the information from the Web more effectively than the Active Web Robot, the index server in turn can keep much more up-to-date index information. The resulting queries that are returned by the search engine have less out of date information.

### 4.3 Distributed Information Gatherer vs. Harvest

The key focus of the Distributed Information Gatherer is to provide an efficient and effective mechanism for collecting and retrieving information from distributed sources. It is different from the Harvest system in that it provides an integrated set of customizable tools for gathering information. In this section, we present three fundamental differences between these two systems.

**Output Data**

Both systems intend to provide the gathered information to be used by other systems, and both systems preprocess the gathered data before passing it on to other systems. The DIG system uses a lossless compression on the gathered data and packs the compressed data into archive files. The original gathered data is not altered in
any other way. By applying unpack and decompress operations, other systems can obtain the original gathered data from the DIG output. The Harvest system uses the Essence sub-system [20] to extract content summaries from the gathered data, and to provide the content summaries as input to other systems. One of the reasons that the Harvest system fails to “make the Web Robot decreasingly useful over time” [19] is because many other systems that use the Web Robot want the original unfiltered data, so that they can implement their own filtering and indexing methods. The DIG system is designed to provide original unfiltered data.

Notification Method

The Harvest system does not provide any kind of notification mechanism for the Gatherer sub-system to notify the Broker sub-system about the information at the individual site. The remote Gatherer does much the same thing as the Active Web Robot, sending out requests throughout the Web to gather information from existing and new Web sites. In contrast, the DIG system uses much less resources to bridge the communication between the client-side and server-side of DIG sub-system. The Passive Web Robot of the DIG does not send requests throughout the Web unless there is broadcast from an individual Web site about newly available data sets. Any Web site that has the client-side DIG sub-system running can send such a broadcast. The notification method of the DIG system provides a public available communication channel so that any interested parties can monitor it.

Incremental Update

Without providing any notification method between Gatherer and Broker sub-systems of the Harvest system, the Broker uses an active agent, called Collector, that periodically sends update requests to each Gatherer or Broker specified in its configuration file. The Gatherer replies with a list of object summaries to be created, removed, or updated, based on a timestamp passed from the requesting Collector. The Gatherer has to be constantly running in order to provide feedback to the incoming requests from Collector. In contrast, the Change Set of the DIG is generated once to
provide feedback to all incoming update requests. The DIG does not need any special programs to handle the incoming update requests.

4.4 Design and Implementation Details

In this section, we present the details concerning the design and implementation of the Distributed Information Gatherer.

4.4.1 The Intermediate Results

Four intermediate result files are produced by the Local Robot. They are divided into two different sets, a Full-Index Set and a Change Set. The Full-Index Set, which contains the Entire Site Index File and the Entire Site Data File, is produced during the traversal of the Web site by the local robot. The Change Set, which contains the Changed Index File and Changed Data File, is a result of the comparing between the current and previous Full-Index Set.

The Entire Site Index File

The Entire Site Index File is a compressed text file. Any document that is allowed to be indexed has a corresponding record in this file. Table 4.1 contains a description of the record fields. Only the HTML and the text file formats are considered at this time, because any browsers can display them directly. In the future, other types of documents, such as postscript files and Portable Document Format (PDF) files should also be considered. A blank space is used to separate the different fields.

The depth of a Web document is defined as the minimum number of hyperlink clicks required for a user to navigate to it starting from the home page. The depth of the home page is defined as zero, any other pages that can be directly reached through a single click of the hyperlinks on the home page have depth of one, and so on. The depth of any Web pages that cannot be reached by traversing the hyperlink is defined as -1. The Depth Level is used in three areas: as an ascending sort order to group the records that have the same depth, as a processing option to the Passive
Field Description | Type | Format
--- | --- | ---
The Depth of the document | Integer | -
Last modified timestamp of the document | Long Integer | In milliseconds
The Size of the document | Long Integer | In number of bytes
The URL of the document | String | URL
Optional file name entries in the archive file | String | -

Table 4.1: Record structure for the *Entire Site Index File*

*Web Robot* so that it can use the data to only selected depth, and as an indicator to divide the *Entire Site Data File*.

The timestamp of each document is in the number of milliseconds since January 1, 1970, 00:00:00 GMT.

**The Entire Site Data File**

The *Entire Site Data File* is a multi-file-packed archive file. Every record in the *Entire Site Index File* contains a URL that is pointing to a file. This file is compressed and packed into the *Entire Site Data File*. There is an overhead associated with each compressed file when it is stored in the archive file. The amount of overhead depends on the compression algorithm that is used, and that depends on how the files in the archive are named.

One of the methods is to label the files in the archive sequentially as 0, 1, 2, 3 etc, corresponding to the index record position in the *entire site index file*, so that the optional file name field can be left out. The amount of overhead is also much smaller than using the file name part of URL as a stored name in the archive file.

**The Changed Index File**

The *Changed Index File* is a compressed text file that contains the difference between two successive *Entire Site Index Files*. On top of the fields in Table 4.1, a one-byte change flag field is added to the beginning of each record in this file.

Table 4.2 contains the description of this field. The previous *Entire Site Index File*
is used as the base for comparison. The rest of the record fields are taken from the current \textit{Entire Site Index File} if the documents are newly added or changed. The deleted file records are taken from the previous \textit{Entire Site Index File}.

A document is \textit{new} if it appears in the current \textit{Entire Site Index File} but was not there previously. A document is \textit{deleted} if it appears in the previous \textit{Entire Site Index Set} but is not there currently.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>This document is newly added to be indexed</td>
</tr>
<tr>
<td>-</td>
<td>This document is deleted from index list</td>
</tr>
<tr>
<td>*</td>
<td>The content in this document has changed</td>
</tr>
</tbody>
</table>

Table 4.2: Status Flag of documents that have changed

To decide if a document is changed is a bit more complicated. If the URLs of two documents are the same, but the \textit{last modified timestamp} and/or the \textit{size} of the document are different, then this document is considered as changed. However, if the document is moved to another part of file system, the \textit{path} part of its URL is changed. In this case, the document with the changed URL is counted as a \textit{new} file, and the document with the old URL is counted as \textit{deleted}.

**The Changed Data File**

The \textit{Changed Data File} is a multi-file-packed archive file. Only documents that are newly added or changed are compressed and packed into this file. The data of each compressed document is taken from the current \textit{Entire Site Data File} without any modification.

**Other Requirements**

The stored file names of \textit{Entire Site Index File}, \textit{Entire Site Data File}, \textit{Changed Index File} and \textit{Changed Data File} can be chosen by the Web site administrator. These file names do not have to conform to any conversion, as long as they are distinct, and can be retrieved and processed by the visiting \textit{Web Robot}. These file names, together
with several other attributes, are stored in the *Master Site Index Configuration File* (See Section 4.4.2).

Lossless compression software must be used for compressing the data, in order to prevent loss of data during the compressing and uncompressing processes. For our experiments, ZIP/UNZIP software was selected. See section 5.1.4.

### 4.4.2 Master Site Index Configuration File

A *Master Site Index Configuration File*, *siteidx.txt*, is created to be the first file that a *Web Robot* is going to request from a Web site. This file is in text format, and can be read by any text editor. It contains information about intermediate results and other site-specific information.

**The Current-Index-Set Record**

The *Current-Index-Set Record* contains information about the *Full-Index Set*. This record has a minimum of three fields. The first field is the timestamp, represented by a positive long integer that is the number of milliseconds from January 1, 1970, 00:00:00 GMT to the time at which the Full-Index Set was created. The second field is the file name of the *Entire Site Index File*. The third field is the file name of the *Entire Site Data File*. If the *Entire Site Data File* is too large, it can be split into multiple parts with different names. The multiple names will appear as the third field and the fourth field, and so on. Figure 4.3 shows how the *siteidx.txt* file is configured in this situation.

Only one copy of the *Full-Index Set* should be kept because each set takes up a large amount of disk space. The previous *Full-Index Set* may be used as a backup and stored on other media, such as tape.

**The Change-Set Record**

The *Change-Set Record* contains information about the *Change Set*. This record has three fields. The first field is the timestamp of the previous full index run. It
has the same format as the timestamp field in the *Current-Index-Set Record*. The
second field is the file name of the *Changed Index File*. The third field is the file
name of the *Changed Data File*. This record can appear many times, depending on
how many *Change Sets* the individual Webmaster decides to keep, but at least one
should be retained. The *Change-Set Record* will not appear in the *siteidx.txt* file, if
it is the first index generation. In the case that the size of the *Change Set* is larger
than the size of *Full-Index Set*, not only will this *Change Set* not be kept, but its
*Change-Set Record* will not be presented in the *Master Site Index Configuration File*,
and all the previous *Change-Set Records* will also be removed from the configuration
file. Figure 4.1 demonstrates this situation.

<table>
<thead>
<tr>
<th>Current-Index-Set</th>
<th>872522888000 FULLIDX.ZIP FULLDATA.ZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit-Rate</td>
<td>30</td>
</tr>
</tbody>
</table>

Figure 4.1: *siteidx.txt* sample file for the first local index run

**The Hit-Rate**

The *Hit-Rate* is used to indicate to the visiting *Web Robot*, how long it should
wait in number of seconds before sending another request to the server. The *Hit-Rate*
of each Web site is determined by the Webmaster of the Web site.

**The Update Process**

When the new set of intermediate results is produced, the *Current-Index-Set
Record* in the *siteidx.txt* file is updated. The timestamp of that record is kept and
becomes the timestamp of the *Change-Set Record* for the *Change Set*, as seen in the
Figure 4.2.

**Limitations**

The *Master Site Index Configuration File* is not limited to the above few records;
it can also store other information.
4.4.3 The index Directory

An index directory is added to the root of each Web server. The main purpose of introducing this directory is to group all indexing related information in one place on the Web server. Using the directory structure of the file system is the simplest and the easiest method to manage the index data, because it provides additional options to the site administrator, such as the following:

- This directory can be mapped to a different physical hard drive.

- A request for the data in this directory can be handled by a secondary Web server for relieving the workload on the primary Web server.

The URLs of the intermediate results in the siteidx.txt Figure 4.3 are displayed in Figure 4.4.
4.4.4 Notification Method

The DIG proposes the creation of a new newsgroup with name www.robot.announce. The Local Robot and the Passive Web Robot use this newsgroup as a communication link. An email is posted to the www.robot.announce newsgroup by the Local Robot containing a subject line that has the starting URL of the Web site. The Passive Web Robot will constantly monitor this newsgroup and parse the subject line of any posted email for the URL. By following the subject line URL, the Passive Web Robot sends a request to retrieve and parse the Master Site Index Configuration File for additional information.

4.4.5 The Retrieval Procedure

Figure 4.5 is the simple algorithm for a Passive Web Robot to use during the retrieving process. The big decision the visiting Passive Web Robot has to make is which set of data it should retrieve. On the re-visit by the same Passive Web Robot, the Change Set should be used if possible, because in most cases, the Change Set is much smaller than the Full-Index Set.
- Monitor the www.robot.announce newsgroup
- Parse the subject line of posted email for the starting URL
- Retrieve the master site index configuration file at
  URL/index/siteidx.txt

- Parse the siteidx.txt file

If this is first time visit by this Web Robot then

  Retrieve the Full-Index Set
  Store the timestamp and the URL of this visit in a Database

Else

  If the timestamp of Change Set matches the timestamp of last visit then

    Summarize the total size of the Change Sets that have a timestamp
    which is greater or equal to the timestamp of last visit

    If the summarized size is greater than the Full-Index Set then

      Retrieve the full index set

    Else

      Retrieve all the Change Sets that have the timestamp which
      is greater or equal to the timestamp of last visit

    End if

  Else

    Since no Change Set that has a matching timestamp is available,
    retrieve the Full-Index Set

  End If

End if

Figure 4.5: Retrieving algorithm for Web Robot
Chapter 5

Experimental Analysis

In the previous chapter, the Distributed Information Gatherer is introduced in detail. A key aim of this new system is to reduce the network traffic over the Internet. In this chapter, we will focus on the quantitative measure of the reduction.

To measure the network traffic reduction that would directly result from implementing this Distributed Information Gatherer, we look at how much data has to be sent through the Internet by using this standard and compare it to the current method.

5.1 Building the Test Application

The main objective of building the test application is to find out what kind of savings we could expect from implementing the Distributed Information Gatherer, or if there would be any saving at all. The test application contains two parts, a test robot to retrieve the data from the Web server and to build the Full-Index Set, and an analysis program to contrast the results between two index runs to produce the Change Set.
5.1.1 Restrictions and Limitations

Ideally, the test robot should be run locally at the individual Web site. Preferably it should even run on the same computer on which the Web server is hosted, in order to reduce the distance that the data has to travel. Since we do not have direct access to any Web server or Web site, the test application has to be run remotely through the Internet.

We could not measure the server load created by the local robot, nor the server load reduction resulting from implementation of DIG system. However, these comparisons are less concern to us. The test robot runs mostly on weeknights and weekends, to avoid generating too much network traffic, which is what this proposed DIG system aims to reduce. Secondly, the contents at the individual Web site are less likely to change outside of working hours. No time-of-the-day restriction is placed on running the analysis program because the analysis program requires very little computing resources and the Internet connection is not required.

5.1.2 Test Environment

The test application is run on the three general-purpose compute servers of the Computer Science Lab (CSLab), in the Department of Computer Science, at the University of Toronto. The configurations of the servers are listed in the Table 5.1. The elapsed times of the completion are taken for the test application. This information is used to decide which Web site should be selected for revisit and the interval of the revisit, i.e. daily or weekly.

<table>
<thead>
<tr>
<th>Server</th>
<th>Model</th>
<th>CPU</th>
<th>Memory(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sun SPARCstation Ultra 2/2170</td>
<td>2</td>
<td>256</td>
</tr>
<tr>
<td>2</td>
<td>Sun SPARCstation 20/712</td>
<td>2</td>
<td>256</td>
</tr>
<tr>
<td>3</td>
<td>Sun SPARCstation 20/612</td>
<td>2</td>
<td>160</td>
</tr>
</tbody>
</table>

Table 5.1: Server configurations

The CSLab currently has three 10Mbit Ethernet connections to the University
100Mbit FDDI backbone, which in turn is presently attached to the University's regional network provider (ONET) by an 8Mbit ATM connection.

5.1.3 The Selection of Web Sites

During the initial Web sites selection, we encounter the problem of how to uniquely identify a Web site.

Some Web servers run identical URL spaces on several different machines with different IP addresses for each of them, but only one domain name, such as Webcrawler's Web site\(^1\), which has a minimum of 16 different IP addresses. In this case, using the domain name can uniquely identify the Web site.

On the other hand, some sites may have multiple different domain names supported by one IP address. For example, the domain names of www.cs.utoronto.ca and www.db.utoronto.ca are different, but both are configured to the same Web server, the Web server for Department of Computer Science, University of Toronto, and use the same IP address '128.100.2.99'. Using the domain name cannot uniquely identify the Web site in this case, but if the domain names are converted into IP addresses and then compared, then we could determine the Web site uniquely.

In the case of the Netscape Web site, accessing the domain name of www.netscape.com with the IP address of 207.200.73.73, and the domain name of home.netscape.com with the IP address of 207.200.73.34 will lead to the same Netscape home page. In this case, we can not use domain name or ID address to uniquely identify this Web site.

The test robot will use the domain name to identify a Web server. The host part of any URL that is flagged as pointing to another Web site is double checked by using its IP address equivalent to make sure it does not refer to the current Web server. In the above case concerning Netscape Web Site, if the test robot starts with home.netscape.com, and there is a local absolute URL with the host name of www.netscape.com, then the algorithm will flag it as a non-local URL. According to the log from the test run against the Netscape Web site, there is no URL with the

\(^{1}\)http://www.webcrawler.com/
host name of www.netscape.com. I believe that the main reason for this is that the local URLs in the retrieved HTML documents are ALL relative URLs. This will not always be the case. However, even if some documents are missed and not retrieved by the test robot, the results presented in this chapter will not be affected greatly.

Table 5.2 shows the list of selected Web sites. Each Web site is assigned a reference ID for later use.

<table>
<thead>
<tr>
<th>ID</th>
<th>Web Sites</th>
<th>The Name of the Origination</th>
</tr>
</thead>
<tbody>
<tr>
<td>aol</td>
<td><a href="http://www.aol.com">www.aol.com</a></td>
<td>America Online, Inc.</td>
</tr>
<tr>
<td>can</td>
<td><a href="http://www.can.ibm.com">www.can.ibm.com</a></td>
<td>IBM Canada Ltd.</td>
</tr>
<tr>
<td>hp</td>
<td><a href="http://www.hp.com">www.hp.com</a></td>
<td>Hewlett-Packard Company</td>
</tr>
<tr>
<td>ibm</td>
<td><a href="http://www.ibm.com">www.ibm.com</a></td>
<td>International Business Machines Corporation</td>
</tr>
<tr>
<td>mot</td>
<td><a href="http://www.mot.com">www.mot.com</a></td>
<td>Motorola, Inc.</td>
</tr>
<tr>
<td>net</td>
<td>home.netscape.com</td>
<td>Netscape Communication Corporation</td>
</tr>
<tr>
<td>pbs</td>
<td><a href="http://www.pbs.org">www.pbs.org</a></td>
<td>PBS Online</td>
</tr>
<tr>
<td>star</td>
<td><a href="http://www.thestar.com">www.thestar.com</a></td>
<td>The Toronto Star (Newspaper)</td>
</tr>
<tr>
<td>sun</td>
<td><a href="http://www.sun.com">www.sun.com</a></td>
<td>Sun Microsystems, Inc.</td>
</tr>
<tr>
<td>ut</td>
<td><a href="http://www.utoronto.ca">www.utoronto.ca</a></td>
<td>University of Toronto</td>
</tr>
<tr>
<td>w3</td>
<td><a href="http://www.w3.org">www.w3.org</a></td>
<td>World Wide Web Consortium</td>
</tr>
</tbody>
</table>

Table 5.2: A list of Web sites that are selected for visit

In order to study the behavior of the Change Set on a daily basis, the test robot has to complete the visit of a Web site overnight. Two Web sites are selected, the home.netscape.com Web site and the www.w3.org Web site, in which both of them have fewer documents than other selected Web sites except the America Online Web site (See Table 5.5).

5.1.4 The Programming Language and Compression Algorithm

The test robot and the analysis program are implemented entirely in the Java programming language. There are two reasons for selecting Java. First, Java has the built-in network package (java.net) and the ZIP package (java.util.zip) in Java
CHAPTER 5. EXPERIMENTAL ANALYSIS

Development Kit (JDK) 1.1.3. The java.net package contains the URL and the URL Connection classes. These classes implement the HTTP calls, such as opening and terminating the HTTP connection, retrieving header information of the resources and retrieving the data. The java.util.zip package contains the implementation of the Lempel-Ziv [27] based compression algorithms. The implementation consists of two classes, the ZIP class and the GZIP class.

Although the ZIP compression algorithm may not be the best in terms of the highest compression ratio or the fastest decompression time, the ZIP file format is widely used. Many commercial and free ZIP tools are available on most operating platforms: UNIX, VMS, MSDOS, OS/2, Windows NT, Minix, Atari and Macintosh. One other important feature is the grouping of multiple compressed files into one archive. For these reasons, the ZIP class is used instead of the GZIP class.

Secondly, the Java programming language is platform independent. An application written in Java can be compiled and run on many different operating platforms without changing one line of the source code.

Many different factors can affect the compression ratio. The compression ratio is different from file to file. There does not exist an official compression ratio for the ZIP compression algorithm. The usual reduction achieved by the ZIP algorithm on the text file is somewhere from 59.3% and up [28].

5.1.5 The Overhead of Using ZIP Archives

Different ZIP implementations produce slightly different numbers of bytes of overhead. The byte overhead for the java.util.zip package that is associated with packing each compressed file into a zip archive comes from three areas. First, there are 22 bytes of overhead per zip archive file. Second, each stored file in the archive requires 92 bytes excluding the file name part. Third, each character of the stored file name requires 2 bytes to be stored in the archive.

In the implementation of the test application, the file name of each stored document is replaced with a base 36 counter, which is corresponding to the row record number in the Entire Site Index File. Printable characters from 0 to 9 and a to z
are used to represent the counter from 0 to 35. The three characters per file name encoding method can represent up to 46,656 files. This is sufficient for the experiment.

The encoded name of each file is not stored in the index file, because the row number of each record in the index file corresponds to the stored file name in the archive after applying the conversion from the base 10 number to the base 36 number. No information is lost due to the encoding of the file name, because the full URL of the retrieved document is stored in the index file.

The overhead per compressed and stored document in the zip archive is 98 bytes, which is composed of 92 bytes of the fixed overhead and 6 bytes for the document file name.

## 5.2 Running the Test Application

### 5.2.1 The Test Robot

The test robot starts from the root of each Web site, and recursively retrieves all the referred local documents, until no more documents can be retrieved from that Web site. Any foreign URLs are not followed up at all. The test robot employs the breadth first algorithm in its search process. It attempts to retrieve all the documents that have the same depth level first before going to the next level. The Robot Exclusion Standard is followed by the test robot, because that is the current unofficial standard.

Two lists of hyperlinks are maintained by the test robot: a successfully retrieved hyperlink list (a good list) and an unsuccessfully retrieved hyperlink list (a bad list), so that the same URL will not be attempted more than once. The information about these two lists is reported in the Section 5.3.3.

Each retrieved document is processed in four ways. First, the properties of the document are extracted and stored in the *Entire Site Index File* (Table 4.1). Second, the document itself is read and all the embedded hyperlinks are extracted. Only the hyperlinks that refer to the documents on the same Web site are kept for further filtering. The hyperlinks that pass all the filtering are checked against the success-
fully retrieved hyperlink list and an unsuccessfully retrieved hyperlink list. Only the hyperlinks that have not been attempted yet are added to the list of the next depth level. Third, the retrieved document is compressed individually. The size information of the compressed document is saved for later use by the analysis program. For example, the overhead associated with packing each compressed document into the archive is not measured but computed. Finally, the retrieved document is erased from the computer memory, and its compressed version is deleted from the hard drive to save disk space.

Only the HTML and text format documents are considered at this time. To decide which document is of HTML or text format is a bit more complicated. First, although most HTML documents have a file extension in some form of html, such as .html, .HTML, .htm and .HTM, not every HTML document has that kind of extension. Second, to know which documents are in text format is very difficult by just looking at the file extensions. For example, .doc file extension could mean a Microsoft Word document which is in a binary format, or a text document.

The HTTP protocol has an accept field, to be used by the sender to specify what kind of file formats are acceptable, such as the "text/*" format. However, the sender still must make the HTTP calls for any hyperlinks in order to get a yes/no reply from the Web server.

To avoid limiting the test robot in any way, we have to start from what we know. The files that have the extension in the Table A.1 are the least likely to be confused with the HTML and the text file formats, and therefore are filtered out, so that the total number of HTTP calls to the Web server are reduced and the retrieval process is speeded up.

The workload of the test robot is greatly reduced by not following up on the CGI scripts, so that it can avoid being trapped in the infinite URL spaces and avoid dynamically generated pages. These two areas are also being avoided by the current Web Robots[26]. This will not compromise the test results.
5.2.2 The Analysis Program

The key aim of the analysis program is to find out the differences between the two index runs, such as the number of newly added, changed or deleted documents, the uncompressed and compressed size of these documents, and the size of the Change Set. The analysis program takes two Entire Site Index files that are generated from the same Web site but are produced on different days as inputs, and analyzes the difference. The older index file is used as the contrasting base. Despite the fact that the actual documents of both index runs are not kept, the information about the size of each uncompressed and compressed document is available from the test robot run.

5.3 Full-Index Set

The selected Web sites that are listed in Table 5.2 were visited between November 7\textsuperscript{th}, 1997 and November 9\textsuperscript{th}, 1997 for the study of Full-Index Set. We present the results in this section.

5.3.1 The Size of the Full-Index Set

The final size of the Full-Index Set consists of three different parts: the compressed size of the entire site index file, the compressed size of the entire site data file without overhead, and the overhead that is associated with packing each compressed document into the archive of the Entire Site Data File.

Table 5.3 contains a detailed break down of the constituents of the Full-Index Set. The Entire Site Index File and the overhead contribute less than 0.5\% and 4\% respectively to the overall size of the Full-Index Set. The overhead ratio is very much dependent on the size of the compressed document, because the overhead is fixed at 98 bytes per compressed document. The overhead ratio will be larger if a Web site has many smaller documents, as does the \texttt{www.utoronto.ca} Web site. In contrast, the overhead ratio will be smaller if a Web site has many larger documents as in the case of \texttt{www.thestar.com} Web site. The pure compressed data part of the Full-Index Set
is a little more than 95% of the set on average over the selected Web sites.

<table>
<thead>
<tr>
<th>Web Sites</th>
<th>Index File² (MB)</th>
<th>ZIP Only³ (MB)</th>
<th>Overhead⁴ (MB)</th>
<th>Data File⁵ (MB)</th>
<th>Full-Index Set⁶ (MB)</th>
<th>IDX %⁷</th>
<th>OH %⁸</th>
</tr>
</thead>
<tbody>
<tr>
<td>aol</td>
<td>0.012</td>
<td>2.494</td>
<td>0.081</td>
<td>2.575</td>
<td>2.583</td>
<td>0.46%</td>
<td>3.12%</td>
</tr>
<tr>
<td>can</td>
<td>0.115</td>
<td>17.626</td>
<td>1.114</td>
<td>18.740</td>
<td>18.854</td>
<td>0.61%</td>
<td>5.91%</td>
</tr>
<tr>
<td>cnn</td>
<td>0.132</td>
<td>43.538</td>
<td>1.118</td>
<td>44.656</td>
<td>44.788</td>
<td>0.29%</td>
<td>2.50%</td>
</tr>
<tr>
<td>hp</td>
<td>0.371</td>
<td>93.698</td>
<td>3.032</td>
<td>96.729</td>
<td>97.100</td>
<td>0.38%</td>
<td>3.12%</td>
</tr>
<tr>
<td>ibm</td>
<td>0.188</td>
<td>18.645</td>
<td>0.957</td>
<td>19.603</td>
<td>19.721</td>
<td>0.60%</td>
<td>4.85%</td>
</tr>
<tr>
<td>mot</td>
<td>0.160</td>
<td>21.780</td>
<td>1.191</td>
<td>22.970</td>
<td>23.128</td>
<td>0.68%</td>
<td>5.15%</td>
</tr>
<tr>
<td>net</td>
<td>0.063</td>
<td>20.548</td>
<td>0.673</td>
<td>21.221</td>
<td>21.284</td>
<td>0.30%</td>
<td>3.16%</td>
</tr>
<tr>
<td>pbs</td>
<td>0.310</td>
<td>62.719</td>
<td>2.321</td>
<td>65.041</td>
<td>65.351</td>
<td>0.84%</td>
<td>3.55%</td>
</tr>
<tr>
<td>star</td>
<td>0.118</td>
<td>44.365</td>
<td>1.133</td>
<td>45.498</td>
<td>45.616</td>
<td>0.26%</td>
<td>2.48%</td>
</tr>
<tr>
<td>sun</td>
<td>0.225</td>
<td>49.979</td>
<td>1.652</td>
<td>51.631</td>
<td>51.856</td>
<td>0.43%</td>
<td>3.19%</td>
</tr>
<tr>
<td>ut</td>
<td>0.175</td>
<td>18.836</td>
<td>1.600</td>
<td>20.437</td>
<td>20.612</td>
<td>0.85%</td>
<td>7.76%</td>
</tr>
<tr>
<td>w3</td>
<td>0.115</td>
<td>20.055</td>
<td>0.813</td>
<td>20.868</td>
<td>20.983</td>
<td>0.55%</td>
<td>3.88%</td>
</tr>
<tr>
<td>SUM</td>
<td>1.912</td>
<td>414.293</td>
<td>15.686</td>
<td>429.969</td>
<td>431.881</td>
<td>0.44%</td>
<td>3.63%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Over Above Selected Sites</th>
<th>Mean</th>
<th>Standard Deviation( STD)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.49%</td>
<td>4.06%</td>
</tr>
</tbody>
</table>

Table 5.3: Summary of the information about Full-Index Set

The most important ratio in this experiment is the percentage of reduction that can be achieved by retrieving the Full-Index Set. Table 5.4 summarizes the reduction ratios for test runs against the selected Web sites in Table 5.2. The average data reduction per site is anywhere from as low as 63.84% at www.cnn.com Web site, to a high of 74.30% at www.hp.com Web site. If all the collected data are summarized into one Full-Index Set, the reduction stands at 68.54%. The mean and the standard deviation of the reduction ratio over the entire array of selected Web sites are 67.27%

²Index File = The compressed size of the Entire Site Index File.
³ZIP Only = The compressed size of the Entire Site Data File without archive overhead.
⁴See Section 5.1.5. Overhead = (22 bytes + (98 bytes per compressed document × Total Number of Retrieved Documents (See Table 5.5)) ÷ 1 MB).
⁵Data File = The compressed size of the Entire Site Data File with archive overhead = ZIP Only + Overhead.
⁶Full-Index Set = The size of Full-Index Set = Index File + Data File.
⁷IDX% = Index File ÷ Set × 100%.
⁸OH% = Overhead ÷ Set × 100%.
and 3.00% respectively.

If the Web Robots retrieve the Full-Index Set instead of one-file-at-a-time, then the Web Robots have 68% less data to be retrieved and are thus at least 68% faster in retrieval. The round trip time (RTT) per HTTP call becomes insignificant compared to the transmission time of the Full-Index Set, because only a few HTTP calls to the individual Web server are required by the Distributed Information Gatherer. Even under the persistent connection feature of HTTP/1.1, additional time is saved at each end of the connection, because thousands of URL requests are no longer needed to be sent by the Web Robots from one end of the connections and processed by the Web server at the other end.

<table>
<thead>
<tr>
<th>Web Sites</th>
<th>Total Retrieved Data(^9) (MB)</th>
<th>Full-Index Set(^{10}) (MB)</th>
<th>Reduction Ratio(^{11})</th>
</tr>
</thead>
<tbody>
<tr>
<td>aol</td>
<td>7.851</td>
<td>2.583</td>
<td>67.05%</td>
</tr>
<tr>
<td>can</td>
<td>59.626</td>
<td>18.854</td>
<td>68.38%</td>
</tr>
<tr>
<td>cnn</td>
<td>123.851</td>
<td>44.788</td>
<td>63.84%</td>
</tr>
<tr>
<td>hp</td>
<td>377.885</td>
<td>97.100</td>
<td>74.30%</td>
</tr>
<tr>
<td>ibm</td>
<td>64.026</td>
<td>19.721</td>
<td>69.20%</td>
</tr>
<tr>
<td>mot</td>
<td>72.813</td>
<td>23.128</td>
<td>68.24%</td>
</tr>
<tr>
<td>net</td>
<td>62.227</td>
<td>21.284</td>
<td>65.80%</td>
</tr>
<tr>
<td>pbs</td>
<td>180.860</td>
<td>65.351</td>
<td>63.87%</td>
</tr>
<tr>
<td>star</td>
<td>141.154</td>
<td>45.616</td>
<td>67.68%</td>
</tr>
<tr>
<td>sun</td>
<td>160.903</td>
<td>51.856</td>
<td>67.77%</td>
</tr>
<tr>
<td>ut</td>
<td>55.893</td>
<td>20.612</td>
<td>63.12%</td>
</tr>
<tr>
<td>w3</td>
<td>65.586</td>
<td>20.983</td>
<td>68.01%</td>
</tr>
<tr>
<td>SUM</td>
<td>1372.674</td>
<td>431.881</td>
<td>68.54%</td>
</tr>
</tbody>
</table>

Table 5.4: Data reduction at each Web sites by using the Full-Index Set

\(^9\)Uncompressed Data.
\(^{10}\)From Table 5.3.
\(^{11}\)Reduction Ratio = (Total Retrieved Data - Full-Index Set) ÷ Total Retrieved Data × 100%. 
5.3.2 Document Sizes

The average retrieved document size over all the visited Web sites is close to 8.4KB. This is 68% larger than the average of 5KB per document obtained by the Web Robot walk performed in April of 1996 [22]. I believe this is partly the result of the adoption of new HTML version 2.0 and 3.2 and the Java Programming Language by the Internet community. Features such as META data information and Java Applet appear in many retrieved documents. With additional features are proposed and added to the HTML specification, the average document size will continue to grow.

<table>
<thead>
<tr>
<th>Web Sites</th>
<th>Number of Docs</th>
<th>Doc Size(KB)</th>
<th>Compressed(^{12})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Retrieved</td>
<td>Mean(^{15})</td>
<td>STD</td>
</tr>
<tr>
<td></td>
<td>404(^{13})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aol</td>
<td>863</td>
<td>9.316</td>
<td>4.172</td>
</tr>
<tr>
<td>can</td>
<td>11917</td>
<td>5.124</td>
<td>6.754</td>
</tr>
<tr>
<td>cnn</td>
<td>11963</td>
<td>10.601</td>
<td>24.487</td>
</tr>
<tr>
<td>hp</td>
<td>32438</td>
<td>11.929</td>
<td>22.140</td>
</tr>
<tr>
<td>ibm</td>
<td>10244</td>
<td>6.400</td>
<td>17.369</td>
</tr>
<tr>
<td>mot</td>
<td>12739</td>
<td>5.853</td>
<td>14.355</td>
</tr>
<tr>
<td>net</td>
<td>7203</td>
<td>8.846</td>
<td>16.472</td>
</tr>
<tr>
<td>pbs</td>
<td>24838</td>
<td>7.456</td>
<td>9.519</td>
</tr>
<tr>
<td>star</td>
<td>12126</td>
<td>11.920</td>
<td>5.176</td>
</tr>
<tr>
<td>sun</td>
<td>17681</td>
<td>9.319</td>
<td>17.592</td>
</tr>
<tr>
<td>ut</td>
<td>17122</td>
<td>3.343</td>
<td>8.170</td>
</tr>
<tr>
<td>w3</td>
<td>8700</td>
<td>7.720</td>
<td>43.415</td>
</tr>
<tr>
<td>Sum</td>
<td>167834</td>
<td>8.375</td>
<td>18.627</td>
</tr>
</tbody>
</table>

Table 5.5: Summary of the information on retrieved documents

The overall standard deviations for uncompressed and compressed documents are 2.2 times and 1.9 times their means respectively. That is, the variations to either side

\(^{12}\) Without archive overhead. Numbers are in KB.

\(^{13}\) 404 error code is referring to the file not found error in which is returned by the HTTP call.

\(^{14}\) % = The Percentage of 404 Not Found = 404 \(\div\) Retrieved \(\times\) 100%.

\(^{15}\) Mean of Document Size = Total Retrieved Data (From Table 5.4) \(\div\) Number of Documents Retrieved.

\(^{16}\) Mean of Compressed Document Size = ZIP Only (From Table 5.3) \(\div\) Number of Documents Retrieved.
of the mean are much greater than the value of the mean. In order to understand how the documents are distributed over different sizes, the entire retrieved documents and the corresponding compressed documents are grouped into 1KB intervals. The results are displayed in Table 5.6 and graphed in Figure 5.1.

From Table 5.6, we see that 49% of retrieved documents are less than 5KB in size and less than 26% of retrieved documents are over 10KB in size. The majority of retrieved documents are not large. Almost 92% of compressed documents are under 5KB.

Neither graph presented in Figure 5.1 has a bell-shaped curve. Rather, they appear to have a geometric distribution.

<table>
<thead>
<tr>
<th>Document Size</th>
<th>Uncompressed</th>
<th>Compressed(^{17})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Documents</td>
<td>% of Total</td>
</tr>
<tr>
<td>(0 KB, 1KB)(^{18})</td>
<td>25836</td>
<td>15.39%</td>
</tr>
<tr>
<td>(1 KB, 2KB)</td>
<td>19412</td>
<td>11.57%</td>
</tr>
<tr>
<td>(2 KB, 3KB)</td>
<td>13678</td>
<td>8.15%</td>
</tr>
<tr>
<td>(3 KB, 4KB)</td>
<td>12541</td>
<td>7.47%</td>
</tr>
<tr>
<td>(4 KB, 5KB)</td>
<td>11009</td>
<td>6.56%</td>
</tr>
<tr>
<td>(5 KB, 6KB)</td>
<td>9690</td>
<td>5.77%</td>
</tr>
<tr>
<td>(6 KB, 7KB)</td>
<td>9469</td>
<td>5.64%</td>
</tr>
<tr>
<td>(7 KB, 8KB)</td>
<td>10398</td>
<td>6.20%</td>
</tr>
<tr>
<td>(8 KB, 9KB)</td>
<td>6957</td>
<td>4.15%</td>
</tr>
<tr>
<td>(9 KB, 10KB)</td>
<td>5846</td>
<td>3.48%</td>
</tr>
<tr>
<td>(10KB,∞)</td>
<td>42998</td>
<td>25.62%</td>
</tr>
<tr>
<td>(0 KB, 5KB)</td>
<td>82476</td>
<td>49.16%</td>
</tr>
<tr>
<td>(0 KB, 8KB)</td>
<td>112058</td>
<td>66.77%</td>
</tr>
</tbody>
</table>

Table 5.6: The distribution of document size

\(^{17}\text{Without archive overhead.}\)

\(^{18}\text{[low, upper] represents low < Document Size <= Upper.}\)
Figure 5.1: The document size distribution
5.3.3 404 Not Found

The number of 404 Not Found documents presented in the Table 5.5 refer to local broken hyperlinks only - no hyperlinks in which point to other Web sites are attempted. The percentage of local broken hyperlinks can be used as an indicator to show how well the individual Web site is maintained, and it indirectly shows how much change has taken place at that Web site over a period of time.

The hyperlinks that are embedded in the document are created to point to an existing resource. Over time, the resource and the document that contains the hyperlink could be altered. Three actions can affect the resources. A resource can be moved to another part of the file system, deleted from the Web site, or changed. Similarly, the document that contains the relative URLs can also be moved, deleted or changed. Even typos in the document can also result in hyperlinks being broken.

One can expect a higher percentage of broken local hyperlinks at news media Web sites, such as CNN and Toronto Star Web sites, because many documents are created or moved every day in order to respond to the news. The percentage of broken local hyperlinks for these two Web sites are 9.70% and 12.32%, which is higher than the average. The high percentage of broken hyperlinks at the W3 Consortium Web site may be the result of poor maintenance.

The percentage of broken local hyperlinks of these selected Web sites are higher than I expected, even though the percentages do not include any broken hyperlinks that are pointing to other Web sites. There are no automated tools to help to discover and fix the broken local hyperlinks, and it is very difficult to fix this problem manually. One of the good side effects of running Local Robot is that it could be used to discover the broken local hyperlinks.

Any tools that are used to fix broken hyperlinks are only trying to solve the problem after the hyperlink is broken. We need a way to prevent the hyperlinks from being broken in the first place, especially the local hyperlinks in which the Web site administrators have more control.
CHAPTER 5. EXPERIMENTAL ANALYSIS

5.3.4 Depth

The depth of a document is defined as the minimum number of hyperlink clicks required for a user to navigate to it, starting from the home page. There are two reasons for studying the depth. First, the greater the depth of a document, the harder it is for the user to navigate to that document without using any kind of searching facility. By analyzing the depth information, we can design a better Web site. Secondly, the depth information can be used to determine how to split the large size of Full-Index Set into several smaller size documents. Table 5.7 displays some information related to the depth.

<table>
<thead>
<tr>
<th>Web Sites</th>
<th>Retrieved Documents</th>
<th>Depth</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Maximum</td>
<td>Mean</td>
<td>STD</td>
</tr>
<tr>
<td>aol</td>
<td>863</td>
<td>7</td>
<td>3.629</td>
<td>1.134</td>
</tr>
<tr>
<td>can</td>
<td>11917</td>
<td>46</td>
<td>5.481</td>
<td>3.337</td>
</tr>
<tr>
<td>cnn</td>
<td>11963</td>
<td>27</td>
<td>6.010</td>
<td>2.944</td>
</tr>
<tr>
<td>hp</td>
<td>32438</td>
<td>52</td>
<td>6.048</td>
<td>2.358</td>
</tr>
<tr>
<td>ibm</td>
<td>10244</td>
<td>32</td>
<td>7.581</td>
<td>2.985</td>
</tr>
<tr>
<td>mot</td>
<td>12739</td>
<td>16</td>
<td>7.242</td>
<td>2.450</td>
</tr>
<tr>
<td>net</td>
<td>7203</td>
<td>14</td>
<td>6.143</td>
<td>2.866</td>
</tr>
<tr>
<td>pbs</td>
<td>24838</td>
<td>74</td>
<td>6.381</td>
<td>4.651</td>
</tr>
<tr>
<td>star</td>
<td>12126</td>
<td>13</td>
<td>6.229</td>
<td>2.290</td>
</tr>
<tr>
<td>sun</td>
<td>17681</td>
<td>53</td>
<td>5.979</td>
<td>3.024</td>
</tr>
<tr>
<td>ut</td>
<td>17122</td>
<td>55</td>
<td>12.391</td>
<td>3.742</td>
</tr>
<tr>
<td>w3</td>
<td>8700</td>
<td>27</td>
<td>5.037</td>
<td>2.159</td>
</tr>
<tr>
<td>Over Entire Collected Data</td>
<td>6.831</td>
<td>3.731</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Over Above Selected Sites</td>
<td>6.513</td>
<td>2.106</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.7: The document depth information

The maximum depth over the selected Web sites varies greatly, from a low of 7 at www.aol.com Web site to a high of 74 at www.pbs.org Web site. It seems that the more documents that are available at a Web site, the higher is the maximum depth. On the other hand, the mean depth changes very little between different Web sites.

The correlation between the maximum depth and the number of documents, and between the mean depth and the number of documents are plotted in Figure 5.2. The
least-squares method is used to plot the two lines in the figure. As we can see from the figure, the maximum depth increases with the number of documents at the Web site, but it does not affect greatly the mean of the document depth. The slope of the mean depth is very close to zero. One of the conclusions we can draw from the graph is that the maximum depth does not have much effect on the mean depth.

Figure 5.2: Correlation of the Depth to the Number of Documents

In addition, if we take out the smaller www.aol.com Web site from the graph, the slope of the mean depth is almost zero and becomes a horizontal line. Thus, the mean depth is very close to being invariant to the number of documents at the Web site. This shows that when a Web site reaches a certain number of documents, the mean depth of these documents does not change much as more documents are added to that Web site.

The standard deviation for each of the mean depths in Table 5.7 is approximately
half of the mean, which means that the most documents are distributed around the mean depth. Figure 5.3 and Figure 5.4 show how the documents and MB of data, respectively, are distributed at different depth level for each of the visited Web sites. Figure 5.5 shows the document and data distribution over the entire collected information. All of these graphs produce a bell-shaped curve, and the data are heavily concentrated around the mean.

5.4 Change Set

The goals of using the Change Set are to provide a method for an incremental update and to reduce the amount of data that have to be retrieved in order to re-index a Web site.

5.4.1 Setup of the Case Study

Two Web sites, home.netscape.com and www.w3.org, were selected for a period of eight days of study of the Change Set. The test robot was run each night against these two Web sites, from Sunday November 9th, 1997 to Sunday, November 16th, 1997. The data that were retrieved on the first Sunday were used as the base. The retrieval on the subsequent day was compared with that of the previous day. The results are presented in the Table 5.8 for the Netscape Communication Web site and in the Table 5.9 for the WWW Consortium Web site. Along with the information for the Change Set, three other areas of information are also presented in these tables. They are the number of documents, retrieved data before compression, and the size of Full-Index Set.
Figure 5.3: The graphs of Number of Documents versus Depth at the visited Web sites
Figure 5.4: The graphs of MB of Uncompressed Data versus Depth at the visited Web sites
Figure 5.5: Document and data distribution over the entire collection of information.
5.4.2 The Size of the Change Set

The amount of data (without applying compression) that was changed during the period of study for both Web sites was relatively small compared to the total amount of data (without applying compression) that were available from each of these Web sites. The daily percentage of changed data ranged from 0% at the www.w3.org Web site to a high of 7.44% at the home.netscape.com Web site. After compression, the size of Change Set ranged from 0% to 2.36% of the total size of data, and ranged from 0% to 6.90% of the size of Full-Index Set. When possible, retrieving Change Set rather than Full-Index Set will result in additional savings in terms of amount of data that is required to re-index a Web site.

The vast majority of data that was available at each of these two Web sites was not changed during the period of study. However, there were always documents that were newly added to the Web sites, deleted from the Web sites, or changed daily, except one Sunday when nothing changed at all at the www.w3.org Web site. By retrieving the Change Set after initial retrieval of the Full-Index Set, the Passive Web Robot can reasonably assess if the retrieved information is relatively up-to-date.

5.4.3 404 Not Found

The number of broken local hyperlinks that were encountered during the study period did not vary much from day to day. This suggests that no software program was run to sniff out and correct this problem at these two Web sites.

\[ \text{Sub-Tot} = \text{New} + \text{changed} + \text{Deleted}. \]
\[ \text{Percentage of documents that are different between this day and previous day.} \quad \% = \text{Sub-Tot} \times 100\% / \text{Retrieved}. \]
\[ \text{Ratio of uncompressed Change Set data (New + Changed) vs uncompressed total retrieved data.} \]
\[ \text{The Changed Index File.} \]
\[ \text{The compressed size of the Changed Data File without archive overhead.} \]
\[ \text{The compressed size of the Changed Data File with archive overhead.} \]
\[ \text{The Change Set} = \text{Idx File} + \text{Data File.} \]
\[ \text{\% of Data} = \text{chg set} \times 100\% / \text{Total Retrieved Data.} \]
\[ \text{\% of Full} = \text{chg set} \times 100\% / \text{Full-Index Set.} \]
### Chapter 5. Experimental Analysis

<table>
<thead>
<tr>
<th>Date</th>
<th>9th</th>
<th>10th</th>
<th>11th</th>
<th>12th</th>
<th>13th</th>
<th>14th</th>
<th>15th</th>
<th>16th</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997-Nov</td>
<td>7205</td>
<td>7205</td>
<td>7207</td>
<td>7212</td>
<td>7223</td>
<td>7209</td>
<td>7221</td>
<td>7221</td>
</tr>
</tbody>
</table>

#### Number of documents retrieved

<table>
<thead>
<tr>
<th>Type</th>
<th>1997-Nov</th>
<th>9th</th>
<th>10th</th>
<th>11th</th>
<th>12th</th>
<th>13th</th>
<th>14th</th>
<th>15th</th>
<th>16th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>7205</td>
<td>7205</td>
<td>7207</td>
<td>7212</td>
<td>7223</td>
<td>7209</td>
<td>7221</td>
<td>7221</td>
<td></td>
</tr>
<tr>
<td>New</td>
<td>0</td>
<td>2</td>
<td>13</td>
<td>11</td>
<td>8</td>
<td>13</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changed</td>
<td>84</td>
<td>88</td>
<td>173</td>
<td>387</td>
<td>249</td>
<td>131</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deleted</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>22</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-Tot</td>
<td>84</td>
<td>90</td>
<td>194</td>
<td>398</td>
<td>279</td>
<td>145</td>
<td>51</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Amount of data retrieved (IN KB)

<table>
<thead>
<tr>
<th>Type</th>
<th>1997-Nov</th>
<th>9th</th>
<th>10th</th>
<th>11th</th>
<th>12th</th>
<th>13th</th>
<th>14th</th>
<th>15th</th>
<th>16th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>63789</td>
<td>63802</td>
<td>63843</td>
<td>63518</td>
<td>63795</td>
<td>63789</td>
<td>63900</td>
<td>63901</td>
<td></td>
</tr>
<tr>
<td>New</td>
<td>0</td>
<td>31</td>
<td>136</td>
<td>112</td>
<td>88</td>
<td>150</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changed</td>
<td>1086</td>
<td>1256</td>
<td>2495</td>
<td>4635</td>
<td>2833</td>
<td>1848</td>
<td>704</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deleted</td>
<td>0</td>
<td>0</td>
<td>510</td>
<td>0</td>
<td>139</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New+Chg</td>
<td>1086</td>
<td>1287</td>
<td>2632</td>
<td>4747</td>
<td>2992</td>
<td>1997</td>
<td>704</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Full-Index Set (IN KB)

<table>
<thead>
<tr>
<th>Type</th>
<th>1997-Nov</th>
<th>9th</th>
<th>10th</th>
<th>11th</th>
<th>12th</th>
<th>13th</th>
<th>14th</th>
<th>15th</th>
<th>16th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idx File</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>Data File</td>
<td>21745</td>
<td>21748</td>
<td>21760</td>
<td>21669</td>
<td>21734</td>
<td>21732</td>
<td>21773</td>
<td>21772</td>
<td></td>
</tr>
<tr>
<td>Set</td>
<td>21810</td>
<td>21813</td>
<td>21825</td>
<td>21734</td>
<td>21799</td>
<td>21797</td>
<td>21838</td>
<td>21838</td>
<td></td>
</tr>
<tr>
<td>%</td>
<td>65.8%</td>
<td>65.8%</td>
<td>65.8%</td>
<td>65.8%</td>
<td>65.8%</td>
<td>65.8%</td>
<td>65.8%</td>
<td>65.8%</td>
<td></td>
</tr>
</tbody>
</table>

#### Change Set (IN KB)

<table>
<thead>
<tr>
<th>Type</th>
<th>1997-Nov</th>
<th>9th</th>
<th>10th</th>
<th>11th</th>
<th>12th</th>
<th>13th</th>
<th>14th</th>
<th>15th</th>
<th>16th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idx File</td>
<td>0.855</td>
<td>0.969</td>
<td>2.017</td>
<td>3.866</td>
<td>2.406</td>
<td>1.411</td>
<td>0.544</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP Only</td>
<td>351</td>
<td>148</td>
<td>1459</td>
<td>146</td>
<td>4400</td>
<td>594</td>
<td>6705</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overhead</td>
<td>8.06</td>
<td>8.63</td>
<td>17.82</td>
<td>38.11</td>
<td>24.62</td>
<td>13.80</td>
<td>4.90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data File</td>
<td>359</td>
<td>420</td>
<td>793</td>
<td>1501</td>
<td>837</td>
<td>608</td>
<td>231</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chg Set</td>
<td>360</td>
<td>421</td>
<td>795</td>
<td>1504</td>
<td>839</td>
<td>610</td>
<td>232</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idx %</td>
<td>0.24%</td>
<td>0.23%</td>
<td>0.25%</td>
<td>0.26%</td>
<td>0.29%</td>
<td>0.23%</td>
<td>0.23%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OH %</td>
<td>2.24%</td>
<td>2.05%</td>
<td>2.24%</td>
<td>2.53%</td>
<td>2.93%</td>
<td>2.26%</td>
<td>2.12%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>% of Data</td>
<td>0.56%</td>
<td>0.66%</td>
<td>1.25%</td>
<td>2.36%</td>
<td>1.32%</td>
<td>0.95%</td>
<td>0.36%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>% of Full</td>
<td>1.65%</td>
<td>1.93%</td>
<td>3.66%</td>
<td>6.90%</td>
<td>3.85%</td>
<td>2.79%</td>
<td>1.09%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Other Interesting Results

<table>
<thead>
<tr>
<th>Type</th>
<th>1997-Nov</th>
<th>9th</th>
<th>10th</th>
<th>11th</th>
<th>12th</th>
<th>13th</th>
<th>14th</th>
<th>15th</th>
<th>16th</th>
</tr>
</thead>
<tbody>
<tr>
<td>40413</td>
<td>362</td>
<td>362</td>
<td>361</td>
<td>361</td>
<td>361</td>
<td>361</td>
<td>364</td>
<td></td>
<td></td>
</tr>
<tr>
<td>%</td>
<td>5.02%</td>
<td>5.01%</td>
<td>5.01%</td>
<td>5.00%</td>
<td>5.01%</td>
<td>5.04%</td>
<td>5.04%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.8: home.netscape.com Web site
<table>
<thead>
<tr>
<th>Number of documents retrieved</th>
<th>Total</th>
<th>New</th>
<th>Changed</th>
<th>Deleted</th>
<th>Sub-Tot</th>
<th>%20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997-Nov 9th</td>
<td>8701</td>
<td>8791</td>
<td>8797</td>
<td>8794</td>
<td>8792</td>
<td>8795</td>
</tr>
<tr>
<td>10th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8795</td>
</tr>
<tr>
<td>11th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8795</td>
</tr>
<tr>
<td>12th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8795</td>
</tr>
<tr>
<td>13th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8795</td>
</tr>
<tr>
<td>14th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8795</td>
</tr>
<tr>
<td>15th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8795</td>
</tr>
<tr>
<td>16th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8795</td>
</tr>
<tr>
<td>% of New</td>
<td>1.74%</td>
<td>0.38%</td>
<td>0.09%</td>
<td>0.43%</td>
<td>0.15%</td>
<td>0%</td>
</tr>
<tr>
<td>Amount of data retrieved (IN KB)</td>
<td>Total</td>
<td>New</td>
<td>Changed</td>
<td>Deleted</td>
<td>New-Chg</td>
<td>%21</td>
</tr>
<tr>
<td>1997-Nov 9th</td>
<td>67170</td>
<td>4245</td>
<td>261</td>
<td>128</td>
<td>4505</td>
<td>6.32%</td>
</tr>
<tr>
<td>10th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.72%</td>
</tr>
<tr>
<td>11th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.06%</td>
</tr>
<tr>
<td>12th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.64%</td>
</tr>
<tr>
<td>13th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.34%</td>
</tr>
<tr>
<td>14th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0%</td>
</tr>
<tr>
<td>15th</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.03%</td>
</tr>
<tr>
<td>% of New</td>
<td>68.0%</td>
<td>68.5%</td>
<td>68.4%</td>
<td>68.4%</td>
<td>68.4%</td>
<td>68.4%</td>
</tr>
<tr>
<td>Full-Index Set (IN KB)</td>
<td>Idx File</td>
<td>Data File</td>
<td>Total</td>
<td>New</td>
<td>Changed</td>
<td>Deleted</td>
</tr>
<tr>
<td>1997-Nov 9th</td>
<td>118</td>
<td>21372</td>
<td>71256</td>
<td>71123</td>
<td>71095</td>
<td>70692</td>
</tr>
<tr>
<td>10th</td>
<td>119</td>
<td>22334</td>
<td>71256</td>
<td>71123</td>
<td>71095</td>
<td>70692</td>
</tr>
<tr>
<td>11th</td>
<td>119</td>
<td>22330</td>
<td>71256</td>
<td>71123</td>
<td>71095</td>
<td>70692</td>
</tr>
<tr>
<td>12th</td>
<td>119</td>
<td>22320</td>
<td>71256</td>
<td>71123</td>
<td>71095</td>
<td>70692</td>
</tr>
<tr>
<td>13th</td>
<td>119</td>
<td>22198</td>
<td>71256</td>
<td>71123</td>
<td>71095</td>
<td>70692</td>
</tr>
<tr>
<td>14th</td>
<td>119</td>
<td>22210</td>
<td>71256</td>
<td>71123</td>
<td>71095</td>
<td>70692</td>
</tr>
<tr>
<td>15th</td>
<td>119</td>
<td>22210</td>
<td>71256</td>
<td>71123</td>
<td>71095</td>
<td>70692</td>
</tr>
<tr>
<td>16th</td>
<td>119</td>
<td>22210</td>
<td>71256</td>
<td>71123</td>
<td>71095</td>
<td>70692</td>
</tr>
<tr>
<td>% of New</td>
<td>68.0%</td>
<td>68.5%</td>
<td>68.4%</td>
<td>68.4%</td>
<td>68.4%</td>
<td>68.4%</td>
</tr>
<tr>
<td>Change Set (IN KB)</td>
<td>Idx File</td>
<td>ZIP Only</td>
<td>Overhead</td>
<td>Data File</td>
<td>Chg Set</td>
<td>Idx %7</td>
</tr>
<tr>
<td>1997-Nov 9th</td>
<td>1.703</td>
<td>1093</td>
<td>1.323</td>
<td>1107</td>
<td>1108</td>
<td>0.15%</td>
</tr>
<tr>
<td>10th</td>
<td>1.548</td>
<td>151</td>
<td>2.223</td>
<td>153</td>
<td>154</td>
<td>0.36%</td>
</tr>
<tr>
<td>11th</td>
<td>0.211</td>
<td>16.6</td>
<td>0.500</td>
<td>17.1</td>
<td>17.3</td>
<td>1.22%</td>
</tr>
<tr>
<td>12th</td>
<td>0.616</td>
<td>137</td>
<td>2.318</td>
<td>139</td>
<td>140</td>
<td>0.44%</td>
</tr>
<tr>
<td>13th</td>
<td>0.295</td>
<td>76.0</td>
<td>1.266</td>
<td>77.3</td>
<td>77.6</td>
<td>0.38%</td>
</tr>
<tr>
<td>14th</td>
<td>0</td>
<td>0.091</td>
<td>0.21%</td>
<td>9.54</td>
<td>9.75</td>
<td>0%</td>
</tr>
<tr>
<td>% of Data</td>
<td>1.56%</td>
<td>0.22%</td>
<td>0.02%</td>
<td>0.20%</td>
<td>0.11%</td>
<td>0%</td>
</tr>
<tr>
<td>% of Full</td>
<td>4.93%</td>
<td>0.69%</td>
<td>0.07%</td>
<td>0.35%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Other Interesting Results</td>
<td>404</td>
<td>2814</td>
<td>2921</td>
<td>2925</td>
<td>2923</td>
<td>2917</td>
</tr>
<tr>
<td>%</td>
<td>32.3%</td>
<td>33.2%</td>
<td>33.3%</td>
<td>33.2%</td>
<td>33.2%</td>
<td>33.2%</td>
</tr>
</tbody>
</table>

Table 5.9: www.w3.org Web site
5.5 Summary

From the experimental results presented in this chapter, a number of conclusions can be drawn:

- By retrieving the Full-Index Set, the amount of data that is required to fully index a Web site by the visiting Passive Web Robots is reduced by 60% to 75%. The number of HTTP requests for the Web documents is reduced by almost 100%, down to a single digit in most cases, because the visiting Passive Web Robots only need to request the few documents that make up the Full-Index Set.

- By retrieving the Change Set in a timely fashion, the Passive Web Robots require significantly less data from the individual Web sites in order to re-index the same Web sites.
Chapter 6

Conclusions and Future Work

6.1 Conclusions

6.1.1 The Distributed Information Gatherer

In this thesis, we introduce the Distributed Information Gatherer, a system to provide an efficient and effective method of continually gathering information from the Web as a way to improve the efficiency in utilizing the limited Internet resources. The purpose of the DIG is to change the way the information is retrieved by the Web Robot.

Experimental results showed a two-third reduction in the amount of data required to retrieve an entire Web site by using ZIP compression algorithm, and an even higher reduction ratio for the incremental update. The amount of HTTP requests that are required to index/re-index a Web site is less than 1% of what is required by the Web Robot. With better communication between the individual Web site and the Passive Web Robot, the Passive Web Robot is able to visit more Web sites and keeps the retrieved data more up-to-date than the current Active Web Robot.
6.1.2 Store HTML Documents as compressed Files

One of the reasons that the Distributed Information Gatherer is successful is because HTML document is stored as text file. The average size of the HTML documents is getting larger and larger. Because HTML is the format of Web documents, millions and millions of HTML documents are requested and transmitted daily, and consume a considerable amount of Internet bandwidth. Any efficiency that is achieved by changing the way that HTML document is stored on the Web sites, goes a long way to reduce the congestion of the Internet.

HTML documents should be stored as compressed files at the Web sites rather than in the current text file format. The compression process should be done during the step when the HTML document is stored on the disk. This will take very little time and CPU cycles. The compressed HTML documents would then be published to the Web sites in the usual way. The Web server does not have to do anything different from what it is currently doing now.

At the other end, an extra process is required to decompress the compressed HTML document by the Web browser, before formatting the document for display. This process will take a fraction of second every time the browser encounters a compressed HTML document.

The requirements for the compression/decompression algorithm are that:

- The compression ratio should be as high as possible in order to reduce the stored size of the HTML documents.

- The decompression cycle should be the shortest, because it has to be performed by the Web browser many times. No time requirement is placed on the compression cycle, because it is only done once for each HTML document.

To increase the compression ratio, the HTML tags should be either all upper case or all lower case.

The Experimental Analysis chapter, (especially the data that is presented in Table 5.5 and Table 5.6 and graphed in Figure 5.1), shows the kind of size reduction that can be achieved by using the ZIP compression algorithm.
6.2 Deployment Issues

To deploy the Distributed Information Gatherer requires four carefully planned phases: initial phase, infrastructure phase, deployment phase and acceptance phase.

In the initial phase, we would follow the Request for Comments (RFC) process, by submitting an Internet Draft document about the DIG to the Internet Engineering Task Force (IETF), so that it may be read and commented on. The next step is to develop a document to be approved for publication as a RFC by the IETF.

We focus on two key areas in the infrastructure phase. The first focus is to develop two sets of multi-platform DIG systems. The client-side of the DIG system runs at local Web sites that contain the functions and the features from Local Robot and the analysis program. The server-side of DIG system runs at the Web Robot sites. The second focus is to market the DIG benefits and the programs of the DIG system to the group of Web Robot owners. The Web Robot owner group is a much smaller group than the general individual web site owner group. The www.robot.announce newsgroup is also created during this phase.

During the deployment phase, we start the distribution of the client-side of the DIG system through the Internet, and turn on the server-side of the DIG system. The Passive Web Robot and the Web Robot will be in use at the same time. The key goal in this phase is to market the DIG benefits to the individual Web site owners, and have the Web site to start running the client-side of the DIG system.

During the acceptance phase, we would have the client-side of the DIG system incorporated into the new release of the Web server software. As more and more data come through the Passive Web Robot channel, the computing resources will be shifted from the Active Web Robot to the Passive Web Robot.
6.3 Future Work

During the experimental analysis, the mean depth level of the visited Web sites is found to be close to seven (See Table 5.7). That is, it requires close to seven hyperlink clicks in order to reach the bulk of the data. More study is required in order to determine what is the best mean depth level of a Web site for the average users to navigate through the sites.

At the current stage, the Distributed Information Gatherer only packs the raw data, because different index servers use different methods to create the master index files. Many Web sites have local search engines in which a local index file must be created. Additional tasks can be distributed to the individual Web sites, if a common index file format can be agreed upon, so that the local index file can be used by the remote index server.

The current development of the Web site is very much a mirror of the initial development of the database. Both started by using a file sub-system of the operating system to manage the data. As in the case of database, it evolved into a database management system (DBMS), which consists of a collection of interrelated data and programs to access that data [36]. The HTML documents correspond to the data in the DBMS, the CGI scripts and Java programs correspond to a set of programs to access that data. The studies of query language for the Web, the indexing of the Web, transaction processing for the Web, etc., all have their roots in the DBMS. We can give a new name for them, webbase management system (WBMS). More studies of WBMS are needed.
Appendix A

List of File Extensions

<table>
<thead>
<tr>
<th>File Type</th>
<th>File Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive Files</td>
<td>.arj .bz .gz .hqx .hla .lzh .sea .shar .tar .taz .tgz .uu .uue .z .zip</td>
</tr>
<tr>
<td>Audio Files</td>
<td>.aif .aiff .au .mid .mod .ra .ram .snd .wav</td>
</tr>
<tr>
<td>Backup Files</td>
<td>.bak .old .tmp</td>
</tr>
<tr>
<td>Binary Files</td>
<td>.a .bin .cab .cmd .com .dll .drv .exe .ini .lib .o .obj .pyc .shs .l .so .sys</td>
</tr>
<tr>
<td>CGI Files</td>
<td>.bat .cgi</td>
</tr>
<tr>
<td>Image Files</td>
<td>.bmp .cgm .eps .gif .fts .ico .img .jpeg .jpg .msp .pbm .pcx .pgm .pic .pict .pm .png .ppm .ras .rgb .rle .scr .shg .tif .tga .tif .wmf .xbm .xpm</td>
</tr>
<tr>
<td>Java, JavaScript Style sheet Files</td>
<td>.class .css .jar .java .js</td>
</tr>
<tr>
<td>Log Files</td>
<td>.log</td>
</tr>
<tr>
<td>Video Files</td>
<td>.avi .mov .mpg .mpeg .qt</td>
</tr>
</tbody>
</table>

Table A.1: List of File Extensions
Bibliography

http://www.w3.org/pub/WWW/History/1989/proposal.html


Available at http://ds.internic.net/rfc/rfc1945.txt


BIBLIOGRAPHY

'94: Mosaic and the Web, October 1994. Available at
http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/mogul/HTTPLatency.html


[10] Henrik Frystyk Nielsen, Jimes Gettys, Anselm Baird-Smith, Eric
http://www.w3.org/Protocols/HTTP/Performance/Pipeline.html


http://ds.internic.net/rfc/rfc791.txt


http://web.nexor.co.uk/public/doc/introduction.html


[28] Jeff Gilchrist, Archive Comparison Test, December 1997. Available at
   http://www.geocities.com/SiliconValley/Park/4264/act.html


[33] Lycos Search Engine. At http://www.lycos.com/


[36] Henry F. Korth and Abraham Silberschatz, Database System Concepts, 2ed,