Traffic Tracking and Analysis for E-commerce

by

Sheng-hong Lu

A Thesis
Submitted to the Faulty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree

Master of Science

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada

Traffic Tracking and Analysis for E-commerce

BY

Sheng-hong Lu

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

Abstract

The increasing popularity of the Internet and e-commerce makes online merchants to constantly seek tools that would permit them to attract new and retain old customers. Traffic tracking and analysis tools can help businesses know more about their customers. These tools track visitors' behaviors on Web sites. The information obtained from Web traffic tracking and analysis can help online merchants target specific audiences with customized products and services. Most commonly used approaches include Web log file, packet monitors, and single-pixel image approach. Each of these approaches has some drawbacks, which limits the types of data it can track or the user environment.

In this thesis, a tracking and analysis approach, which has fewer limitations and more advantages than the existing approaches, is proposed. Three different approaches (i.e., improved single-pixel image, JavaScript tracking and HTTP proxy server), which work together to track a user's activities are discussed. In analyzing the tracked data, a model for path analysis is implemented. Path analysis is pivotal for Web site management and marketing in e-commerce. In addition, page content and user clustering analyses are also performed. In modeling the tracking and analysis approach, a formal technique is used for quality assurance.

Acknowledgements

I am very grateful to my advisor and friend, Dr. Sylvanus Ehikioya, for his invaluable guidance and advice, efforts, encouragements, patience, and time spent in guiding me through this thesis work. Every time, even if he was very busy, he always managed time to welcome me and discuss my work.

I would like to thank Dr. Helen Cameron and Dr. Peter C. J. Graham for their time and advice in writing my proposal for this thesis; I would also like to thank Dr. Vojislav Misic and Dr. Mary Brabston for their evaluation of my thesis and helpful suggestions.

I am grateful to Dr. Dereck Meek for his kindness and financial aid.

Also, I would like to thank Dr. Ruppa (Tulsi) Thulasiram for his frequent care and Mr. Sajid Hussain for his suggestions about proxy design.

Many thanks go to my grandparents and my parents, especially to my parents, who have been encouraging and supporting all their children to pursue as much education as possible even in financially hard times. I would also like to thank my brothers and sisters, they make their efforts to contribute to the family, and support and help one another. I am fortunate to be one in such a harmonious and happy family.

My love goes to Qin Xu, who makes my life more beautiful with her love and understanding. Qin always brings happiness to my days and cheers me up during my moody days.

Finally, I would like to thank all members of the Department of Computer Science. They have been a great source of support and help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The growth of the Web is phenomenal. The number of Web users and the number of Web sites are now measured in the millions. They are growing so quickly, no one knows the actual figures about the growth statistics exactly.

Regardless of the actual figures, an immense base of users, together with high speed Internet technologies, gives rise to the emergence and development of e-commerce, which allows a business of any shape and size to reduce costs, improve communications and ensure that there are no geographical constraints on products and services.

Customers are the base of all businesses. When businesses lack knowledge of their customers, businesses cannot develop their marketing activities efficiently. It is a general misconception that all a business has to do is to invest in a Web site and then wait for its payback to shoot through the roof.

1.1 Motivation

The increasing popularity of the Internet and e-commerce causes online merchants to constantly seek tools that would permit them to attract new and retain old customers. To achieve these two goals, a business should know as more as possible about its customers.

However, e-commerce is fundamentally different from other traditional businesses, because a traditional business can easily construct a demographical and psychographic target audience profile. In e-commerce, every customer's activity occurs on the Web. This means a business can not know much about the visitors to its Web site, because most visitors are anonymous, they come and go without much trace. This lack of trail imposes a heavy burden on businesses that want to get more information about their customers beyond sales data.

Another feature of the Web is its interactivity, which means it not only transfers information to the visitors, but also be able to get some information from the visitors. This feature makes it possible for e-commerce Web sites to obtain their visitors' data while the visitors are browsing. This is one major reason why Web analytic market is growing so fast today.

Traffic tracking and analysis tools can help businesses know more about their customers. These tools track visitors' behavior on Web sites. When the information obtained from Web traffic tracking and analysis is combined with traditional demographic and psychographic business information, such valuable information as what people like, how they feel, how they respond, and so on, can be obtained. Such information is important to any business because, based on this information, a company can target specific audiences with customized products and services.

To achieve obtaining data from online visitors, Web log analysis is the most commonly used approach that is based on the data from Web server logs. The analysis process, however, is complicated due to Web log file characteristics, such as large size and different formats for different servers. Other analyses are based on TCP/IP packet monitoring. This approach may lower the server performance if monitoring is performed on the server. Monitoring by packet sniffing may put an extra load on the server. Single-pixel analysis gathers its data from each page that a visitor browses. Because Single-pixel analysis is simply based on HTTP image loading and JavaScript technologies, it has some limitations since JavaScript may not be completely supported or can be disabled by the visitor.

Such situation motivates the need to find a different tracking tool which can track as much data and with less limitations as the above methods or tools.

1.2 Problem Definition

An online merchant would be interested in a variety of issues regarding both the general performance and effectiveness of its Web site in order to shape its marketing strategies. Some basic ones are:

- Which browsers and operating systems are the most common?
- How many visitors visit the Web site?
- Which page is most frequently requested?
- From what sites are visitors coming?

Some advanced analysis may include:

- Path analysis, which allows the merchant to know how visitors come to its site.
- Grouping, which groups the visitors by their interest and other attributes.

In addition, the following more advanced questions attract the interests of online merchants.

- How many people made purchases among the visitors?
- What is User A's interest?
- Which referrer results in the most purchases, not just hits?

Capturing the data necessary to provide answers to these basic and advanced questions is the thrust of this thesis. While the answers to the basic questions can easily be derived from traditional web logs, answers to the more advanced questions can be possible by integrating customer profiles and other specific data such as the contents of page views that are not normally available in traditional web logs. The first step in doing so is to capture the necessary data, which will then be analyzed to derive the relevant information that accurately addresses those questions above. Thus, it is necessary to develop a way to track online visitors' behaviors and analyze Web traffic for an e-commerce system. The methods for achieving these two tasks, the tracking and analysis of Web traffic, are discussed in this thesis.

1.3 Preview of Contributions

This thesis contributes to the following aspects of research and applications in e-commerce systems:

- The end product of this thesis is a tool that can be used to track and analyze Web traffic. This tool can capture the activities of visitors on a given site. It enables answers to questions such as which page is frequently visited? How many visitors visit the given site every day? How many visitors make a purchase? Besides, this tool can track as much data as possible and the tracked data can correctly reflect the visitors' activity on the given site.
- This tool uses a method different from the most commonly used methods, such as Web log files, packet monitoring, and fundamental Single-pixel analysis. Based on the method here, this tool can capture the Web page content. This feature has not been addressed by those commonly used methods.
- This tool captures main keyboard and mouse events, and any form input on the page which a user browses. This feature is unique to this tool. This tool can capture a user' data even if the user's browser does not support JavaScript, or the user disables the browser's JavaScript function.
- This thesis introduces a different way to represent Web site navigation path, a model to analyze path, and implementation of this model.

1.4 Organization

The rest of this thesis is organized as follows: Chapter 2 provides background information and related literature reviews. Chapter 3 presents the requirements analysis and design for the tracking and analysis of online visitors. Chapter 4 describes challenges of developing such a tool, quality assurance of the tool development, and the implementation of the tool. Finally, Chapter 5 presents conclusion and suggests some directions for future work.

Chapter 2

Background and Related Work

Traffic analysis is a relatively new research field originating from incredible growth of the Web over the last decade. Related research work on both traffic tracking and analysis is presented below.

2.1 E-commerce and Traffic Analysis

"E-commerce generally refers to all forms of transactions relating to commercial activities including both organizations and individuals, that are based upon the processing and transmission of digitized data, including text, sound, and visual images" [1]. E-commerce offers many business advantages, such as lower purchasing costs, more accurate inventory, immediate international sales presence, and shopping 24 hours/day, 365 days/year. Some e-commerce applications are developed and deployed as Web-based applications, thus leveraging the vast ubiquitous Internet infrastructure. Users use the Web browser (flexible and easy to use) tool generally available on almost every computer system to access these e-commerce applications.

Generally, every time a visitor browses the Web, the visitor merely makes an Internet connection to a server (a remote computer) and requests some data via HTTP over the TCP/IP protocol. The data requested is split into packets by this protocol and sent to the visitor's computer over the Internet. These packets are then reassembled by the visitor's computer and displayed by the browser. The packets that pass through a node on the Internet constitute the "traffic" of that node. Web traffic tracking is the process of tracking or capturing Web site visitors' traffic, while Web traffic analysis is the process of analyzing Web site visitors' online behavior.

Web traffic tracking and analysis mainly serves network management and marketing campaign development by analyzing Web site visitors' online behavior. The tracking and analysis of a Web site's traffic is important because of the business values of such data. Based on the data tracked on the site and other customer data, which may be tracked (implicitly or explicitly), enterprises can analyze visitors' behavior using structured reports, online analytical processing software, visualization tools, or other tools.

2.2 An Overview of Traffic Analysis Process

Traffic analysis process includes data tracking process and data analysis process, which are shown in Figure 2.1. The traffic analysis process includes four stages: Capture, Store, Analyze, and Report. The first stage, Capture, is to track data about online visitors. Since these data are the base for later different analyses, therefore, they need to be stored. This is achieved by the second stage, Store. To make raw data useful, we need to analyze the raw data in the third stage, Analyze; while the last stage, Report, is to use the results of the analysis to generate reports.

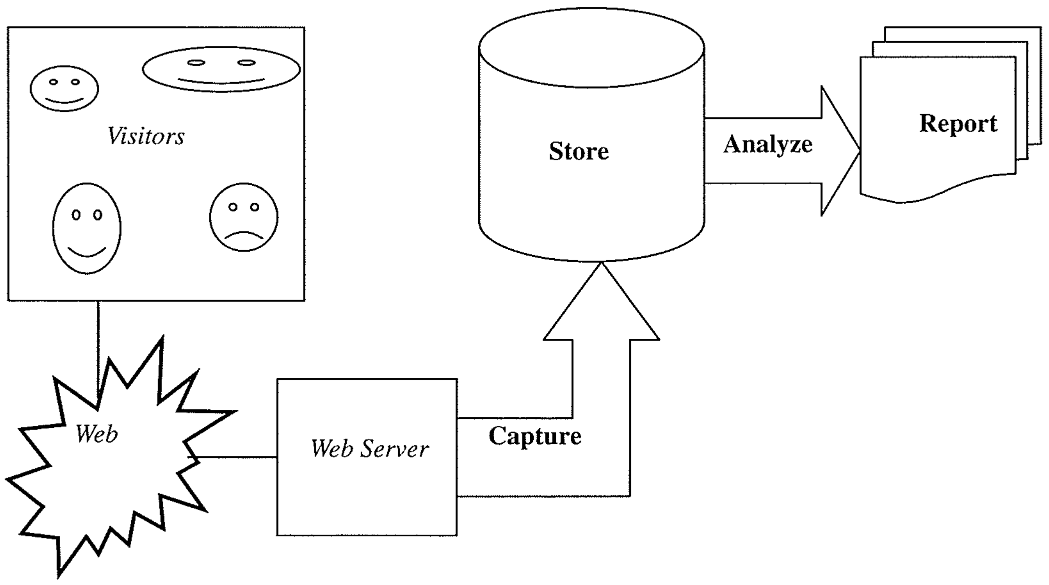


Figure 2.1: Web Traffic Analysis Process

Obviously, the first stage "Capture" is critical for all the following three stages, because the data tracked in the first stage are the data sources for all later three steps. The final report will be incorrect or weak if the data obtained in the first stage are incorrect or inadequate.

2.3 Main Traffic Analysis Approaches

Several traffic tracking and analysis tools for performance measurement are described in [17]. Most of these tools are based on well-defined metrics such as delay, packet loss, flow capacity, and availability. Although some tools that focus on Internet traffic flow measurement are being developed, these tools are mainly for improving network performance, which does not directly contribute to marketing in e-commerce. Many commercial Web site analysis tools are also available for both site management and marketing purposes for e-commerce [3]. However, most of these tools are inadequate because of the approaches they adopt in tracking data. The tracking approach determines how the data and what data can be tracked. Obviously, a tool that can work in more situations and obtain more data is more useful.

There are three primary approaches in Web traffic analysis market. They are user-centric, site-centric, and ad-centric. Site-centric and ad-centric technologies are based on Web logs, HTML tags, and packet sniffers, while user-centric technology is based on polling [41]. As for the tools for doing analysis, the traditionally and widely used approach to capture data from Web site visitors is the HTTP server log files and packet monitors [2, 3, 4, 5]. Single-pixel technology is another, but a newer approach [2, 4]. Other technologies such as HTML forms, URL rewriting, HTML hidden fields, cookies and sessions [39] are also used to capture data from visitors.

2.3.1 HTTP Web Server Log File Analysis

When visiting a Web site, the user connects to a Web server, which serves the files the user requests. The Web server creates a text file (one entry for each file request) to record this activity, each text file consists of the Web log file for that Web server.

The HTTP Web server log analysis technology is the oldest technology developed in 1995 [40].

This analysis is used primarily to obtain a general understanding of what happened on a Web site. Webmasters and system administrators often need information typically for Web site management purposes: how much traffic they got, how many requests failed, and what kinds of errors were generated. Web log file analysis tools have become the most frequently and widely used approach by Web marketers, with e-commerce becoming an important platform for business.

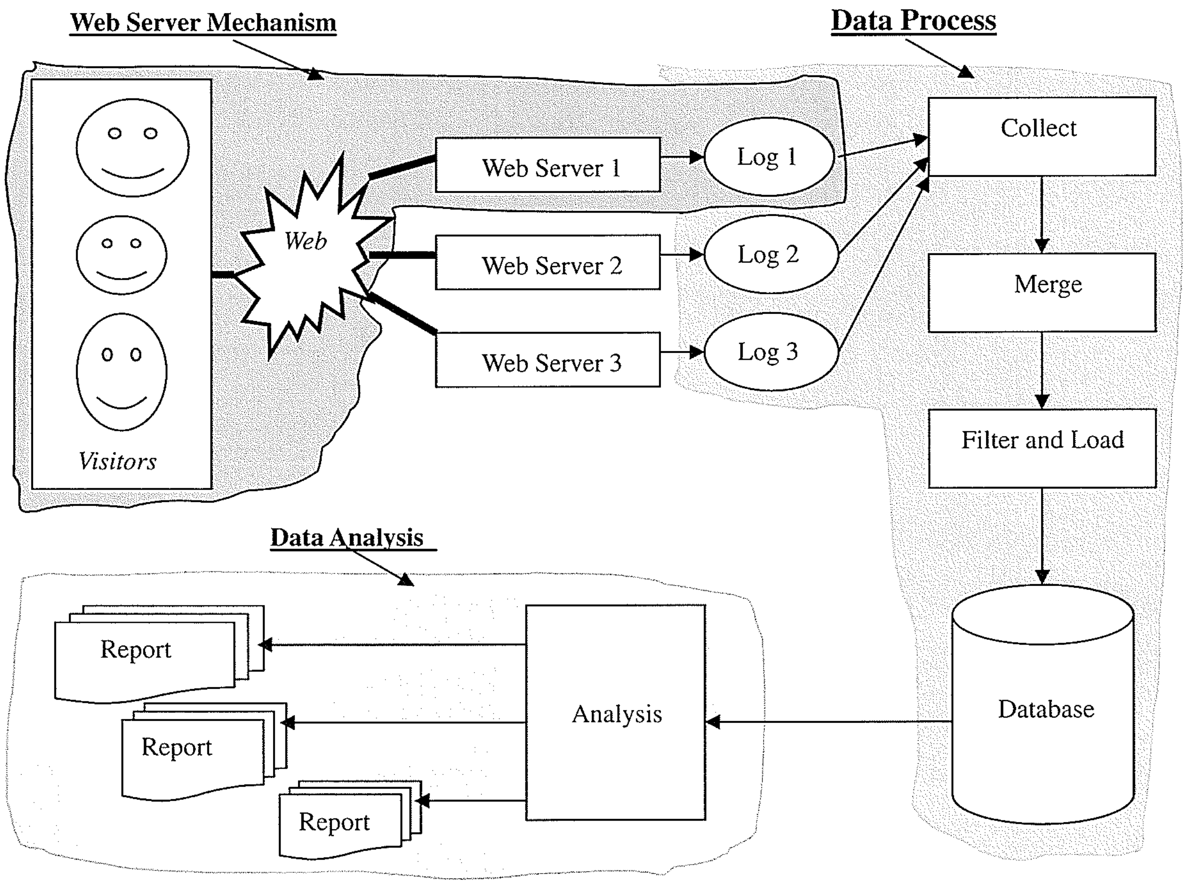


Figure 2.2: Web Log File Analysis

Figure 2.2 shows how the analysis is done at a higher abstract level. It includes the following three parts (with three different shadow areas): Web Server Mechanism, Data Process, and Data Analysis.

2.3.1.1 Web Server Mechanism and Log File Formats

A Web server records its HTTP traffic continuously into its log file. Normally, each Web server has one log file – a text file containing all HTTP requests and responses from the Web server. The information in a log file generally depends on which Web server generates them.

All Web servers such as Apache, Microsoft Internet Information Server, Netscape, Domino GO, etc. provide logging capabilities, configuration options for logging such as enabling or disabling logging, and specifying the format of the log file. The common formats include the following formats [2]:

- The National Center for Supercomputing Applications (NCSA) Common Log Format (access log)
- NCSA Separate Log Format (3-log Format)
- NCSA Combined Log Format
- W3C Extended Log Format

All the three NCSA log formats are based on NCSA httpd (a Web server), and widely accepted as standard by HTTP server vendors. NCSA Common Log Format is the simplest. It logs basic HTTP access information without the referral, user agent, or cookie information, which can be optionally set up in the NCSA Combined Log Format. The information logged by NCSA Common Log Format is in a single file. NCSA Separate Log Format logs information into the following three separate files commonly known as common log or access log, referral log, and agent log. This information includes what NCSA Common Log Format has, but lacks cookies information. The NCSA Combined Log Format extends the NCSA Common Log Format and logs the information in one file too. The additional typical information in this format includes referrer and user-agent.

The W3C Extended Log Format logs one entry for each HTTP request. Compared with NCSA formats, the W3C Extended Log Format has a different syntax, and has more flexible and useful features. In addition, the W3C Extended Log Format allows the user to add or remove multiple optional fields independently. Another feature of W3C Extended Log Format is the use of special directives, which are added to a file and contain information about the file format. With this feature, it is possible to change the log format at any time. In addition to the information logged by any NCSA format, the W3C Extended Log Format supports other types such as referrer and user-agents.

Typically, almost all log files include the following six basic fields:

- IP address or DNS hostname to identify the remote browser.
- The RFC931 remote log name identification field, mostly it is a dash.
- The date and time field for the request in Greenwich Mean Time.
- The request or transaction field, which usually includes the request method (Get or Post), the requested file, and the protocol used, for example, HTTP/1.0.
- The HTTP status code returned to the visitor by the Web server.
- The byte size of the document transferred.

Most Web sites today use W3C Extended Common Log Format, which adds the following three parts in a log file:

- The referrer field, which identifies from where the visitor comes to this site.
- The agent field, which identifies what kind of browser the visitor is using and on what platform.
- The cookie field, which can be used to identify whether the visitor is a new one or a returning one.

2.3.1.2 Log Data Process

A single raw log file contributes very less readable information to the management and marketing staff of the given site to perform site management or market development. The file is a long text string, which includes some data fields depending on the log file format. These raw data log files should be processed to derive useful information. The first stage is to collect all single entry data from different Web servers (if more than one Web severs are used) of the given Web site. The collect stage achieves the goal of bringing all the log files from all the Web severs of the given site to a centralized location for later processing and analysis.

In any e-commerce site, every visitor's activities occur on the Web, using HTTP, a stateless protocol. The stateless nature of the HTTP protocol means that no trail will be left after the Web server responds to a visitor's request. Therefore, each Web log file entry is a simple record of every visit. To know more about the visitor's visits to the site, we should know all the activities of this visitor during the visit to the site. To achieve this goal, we need to merge all the data related to this visitor in the same session. The merge stage groups individual Web log entries and relates log files data to each visitor.

Depending on the purposes of the Web log file analysis, some information not needed should be omitted at this stage. The filter and load stage allows all necessary information for later analysis to go to the data store device, and removes all unnecessary data.

2.3.1.3 Web Log Data Analysis

Web Log files provide valuable insight into Web site usage. Based on the data from the server log file, the basic usability analysis includes:

- Number of requests made ("hit").
- Total files successfully served.
- Number of requests by type of files, such as HTML page views.
- Distinct IP addresses served and the number of requests each made.
- Number of requests by HTTP status codes (successful, failed, redirected, informational).
- Referring pages to the given site.
- Browsers and versions which made the requests.

Some advanced analysis data may be obtained from Web log files. This analysis should answer the following questions [40]:

- Who visited the given site? Session data can be used to identify whether a unique visitor is returning to the given site or not.
- Which path took the visitors to the site? With the knowledge of each page a visitor viewed and the order of viewing the pages, the trends in how the visitors navigated through the pages can be determined. And which HTML element (link, icon, and image) a visitor

clicked on each page to get to the next page also yields important information.

- How long did visitors spend on each page? The length of time most visitors' stay on a page can be used to deduce whether the page is interesting or not.
- Where did visitors leave the site? The last page a visitor left your site is the logical place to end the visit, it may be the page the visitor did not enjoy and thus wanted to leave the site.
- What was the success rate of user's experience at the site? To compute this value, we can use measures such as how many purchases occurred and how many downloads were finished.

2.3.1.4 Advantages of Web Log File Analysis

A Web server captures actual usage data in its working environment in a natural way. A Web log file reflects the activity of most visitors to a given site, over a potentially long period of time. Therefore, it is extremely valuable for Web design engineers to do usability assessments.

Although Web log files began as a way for IT administrators to ensure adequate bandwidth and server capacity on their organizations' Web sites [42], this approach has been advanced considerably in recent years by e-commerce companies. These companies are mining log files for details about visitor profiles and buying activity, through learning about the usability of their Web sites, (i.e., how successfully they meet their marketing goals, and the visitors' satisfaction.).

2.3.1.5 Main Problems with Web Log File Analysis

Web server log files contain a lot of information [14]. A one-megabyte (1 MB) log file typically contains about 4000 to 5000 page requests. Web site analysis tools typically import the log file data into a built-in database, which in turn transforms the data into readable, graphical reports for different purposes. This process is difficult and time-consuming because the log data are abbreviated and cryptic.

The original purpose of log files is to produce site-level performance statistics. Thus, it is not a surprise that they can not provide accurate and effective usability information on a given site.

Two main problems make log file analysis as usage indicator difficult. The first one is that the tracked data in a log file are insufficient, which may be caused by the lack of certain types of usage data that should have been logged. Another problem is that extraneous data can go into a log file. These problems give rise to insufficient, unsound, and misleading information, which results in weak answers to most of the questions required for advanced analysis (see 2.3.1.3). The following are the main shortcomings of a Web log file:

- A log file cannot track the visitor's ID even if the site has a login page. This inability to track a visitor's ID makes it hard to identify a unique visitor. Although session data and IP addresses are available, it may still be difficult to identify a visitor because a lot of visitors use dynamic IP addresses offered by their Internet service providers.
- When the site owner, or the Internet service provider, has a central cache for Web pages, or when the visitor uses the browser's cache, the pages obtained by the visitor by pressing the back button during the subsequent visits in the same session are typically from those caches and thus can not be logged (or recorded) in a log file. This problem results in an incorrect path analysis. In addition, page caching may give wrong information to know where the visitor left. For example, if the last page was from a cache, the log file cannot reflect it.
- The Web server would record nothing in the log file when a visitor arrived at a page by typing the URL in the address box, using a bookmark, or following an email link [43]. This problem will result in wrong information about the referrers.
- A log file only records the time when data transmission was initiated, not the time when the data transmission completes. Therefore, the time spent on a page by this visitor can only be roughly estimated based on comparing the timestamps of the current request and the next request.

Additional details of the challenges associated with Web log file analysis are available in [15, 16].

2.3.2 Packet Monitors

Packet monitors collect traffic data directly from the TCP/IP packets sent to and from a given Web server. They use packet sniffing technology, which is the equivalent of telephone tapping

[4]. When a Web browser connects to a Web server, it communicates with the server by sending requests and receiving responses. All requests and responses are transferred as data, which in fact are split into TCP/IP packets by HTTP, along with other Internet application protocols. It is possible to "sniff" these packets when they move across the network to collect Web site traffic data. Two types of packet monitors exist: server monitor and network monitor. Each type is based on where to collect the traffic data.

Typically, a server monitor runs as a plug-in to the Web server. A server monitor gets information about each event happening on the Web server it monitors through an application programming interface (API). But what events and data are visible to the server monitors depend on the Web server [1]. Normally, a server monitor can get a visitor's ID, referrer pages and some other information about the events on the server.

A network monitor is a packet sniffer, which can capture all data packets passing through a given network. A network monitor can be set up to capture the packets to a certain machine or all packets traversing a given network. Network monitors are more frequently used than server monitors in tracking traffic. A network monitor can see almost everything between the visitor and the given Web server. The tracked information include the requests, the server responses, cookies, and transferred HTML files. In addition, a network monitor can also track stop events issued by the browser, which makes it possible for the site owner to know those pages that are taking too long to generate and display by the client's browser. In addition, it can record the Web server's response time to each request. Some network monitors can capture HTML "form data" transmitted via a POST method when the visitor clicks a submit button [1].

2.3.2.1 Advantages of Packet Monitors

One obvious advantage of packet sniffing over Web log files is that data can be captured and analyzed in real time. Another advantage is that it can track almost all information found in a Web log file, and the information at the network level such as "stop" events, which are not present in log files. Besides, packet sniffing can be compatible with almost any custom Web server, because it is independent of log file format and underlying operating system. In addition, organizations with distributed Web servers can easily and automatically collect their traffic information in a centralized data warehouse for later analysis.

2.3.2.2 Problems with Packet Monitors

Server packet monitoring can introduce unexpected problems into the server itself (e.g., can cause the server to shut down when the packet monitor crashes) since a monitor acts as a plug-in. In addition, the kinds of data a server packet monitor can track depend on the Web server. A network monitor can be installed on each Web server although such configuration is difficult to manage if the Web servers are in different geographic locations. Although a network monitor can track more data than Web log files, it consumes a lot of processor time and, therefore, causes a heavy load on Web servers.

Another problem is that packet sniffing captures data in real-time and the data are not logged, therefore, if something goes wrong with the packet sniffer, the data will be lost.

2.3.3 Single-pixel Technique

A single-pixel is known by various names such as 1-by-1 GIFs, invisible GIFs, beacon GIFs, and Web bugs. A single-pixel method inserts an IMG tag on a Web page. This tag encloses some HTML and JavaScript codes, which are interpreted by the user's browser when the user gets such a page from the site's server and thus, captures relevant data from that page. The single-pixel technique is commonly used with cookies and JavaScript, and is based on the image downloading mechanism. When a page is downloaded, the browser not only creates a connection to the Web server where the page resides, but also creates a connection to the Web server where the image resides to request the image.

Information tracked using the single-pixel technique include:

- The IP address of the computer making the request.
- The type of browser that sent the request.
- The URL of the page on which the image is located.
- The time the image was downloaded.

- The URL of the image.
- A previously set cookies value.

2.3.3.1 Advantages of Single-pixel Technique

Web log files record a user's activity by "hits". This means if a page has three images on a page, the log file would create at least four separate records for the request for this page. These four records constitute the results of the request for this page, and one for each image. Unlike Web log files, single-pixel gathers information by page views, i.e., one record for each page. Detailed information is collected when the page is fully loaded. One obvious benefit of single-pixel technique is that the information can be easily collected and put into a database, while Web log analysis normally needs data from multiple Web servers that may be located in different places. That means, for example, if a page has five IMG tags, and these five images are located on five servers, a request to this page will lead to a record in each log file of these five servers. To analyze the data, we have to collect these records from those five servers, but with single-pixel analysis, it simply creates one record because it is associated with one image. Once the image is downloaded, the tracking program runs and records all necessary data as one data entry.

When working with client-side JavaScript code, which can capture more client-side data (such as screen resolution), the single-pixel can capture more data than those tools working on the server side like Web log files and packet monitors, and these data are not obtainable by these server-side tools.

2.3.3.2 Problems with Single-pixel Technique

The single-pixel approach which uses JavaScript can fail to capture the required data when the visitor's browser disables JavaScript or when the browser does not support JavaScript [2, 4].

Another problem is that, like packet sniffing, single-pixel captures data in real-time. If something affects the downloading of the image, the data will be lost.

An additional problem with the single-pixel technique is some current concern over the use of this technology that it is surreptitious because, unlike cookies, Web browsers do not offer preference settings to warn users that pages contain Web bugs [45]. Some consumers may be unhappy with the awareness that they are being monitored and tracked [46, 47]. Privacy issues are always complex and important. Any technology that tracks individual online users' navigational behaviors and personal information usually causes privacy concerns in ethical and legal aspects.

- Privacy issues in ethical aspect

  Ethical issues mainly include whether the tracking without the users' consent can be done or not? Is it appropriate for an online business to use the data tracked? What kinds of personal information or behavior data can be tracked? Who can use the data tracked and what can these data be used for?

- Privacy issues in legal aspect

  Legal aspect in privacy issues involves clear definitions of privacy and how to protect privacy legally. Although the issues in legal aspect are similar to those in ethical aspect, legal penalties are enforced if any violation of the laws occurs. Generally, to protect the privacy of online users two principles are proposed to govern the data collection: 1) "Data-gatherers should inform consumers what information they are collecting, and how they intend to use such data"; and 2) "Data-gatherers should provide consumers with a meaningful way to limit use and re-use of personal information" [82]. In addition, different countries have different laws or regulations for privacy protection [83]. The W3C (World Wide Web Consortium) has also proposed the P3P (Platform for Privacy Preferences) to guide user profiling companies obtain data from online users while protecting users' privacy in electronic commerce. The details of the P3P guidelines are available in [84].

- Privacy laws in Canada

  In Canada, both the federal and most provincial governments have laws for protecting privacy, although the provincial legislations (e.g., Freedom of Information and Protection of Privacy Act (FIPPA) [87] in Manitoba) for privacy protection vary. These laws regulate how to collect, use, and disclose personal information. While the Federal Privacy Act [88] is adapted to the federal government departments or agencies, the Personal Information Protection and Electronic Document Act (PIPED Act) [89] is adapted to private sector organizations and effective in January 2004. The details of more privacy acts can be found on the government's site [85].

The tool proposed in this thesis is not currently implementable in Canada due to privacy issues. Additional details are provided in the future work section.

2.3.4 Other Approaches

In addition to those tracking approaches discussed above, some other approaches are also used to collect online visitors' data. Some approaches work explicitly, such as HTML forms, while others work at the background.

- **HTML Forms**

The most direct method of gathering data from Web visitors is by using HTML forms. This method allows a site to capture information about its visitors and what they want. Once a visitor fills out a form and submits it, the information contained in the form is sent to a server side program, which parses and saves the data into a database for later analyses.

The main advantage of this approach is that the Web site can collect data of interest. By using pre-designed forms, the tracked data are easy to save and analyze later. However, this approach is unattractive because most Internet users are unwilling to spend a few minutes in completing a form if it is optional.

- **HTML Hidden Fields**

HTML hidden fields capture a visitor's session data about the visitor. An HTML hidden field looks like the following: <INPUT TYPE = "HIDDEN" NAME = "session" VALUE = "...">. When the form is submitted, the specified name and value are included in the GET or POST data.

HTML hidden fields are easy to use, but they only work if every page is dynamically created.

- **HTTP Cookies**

HTTP cookies are widely used to collect data from visitors. A cookie contains small bits of textual information that a Web server sends to a Web browser the first time that browser connects to the Web site. On subsequent visits, the Web browser sends the same ID back to the Web server, telling the Web site that a specific user has returned.

Web site developers can easily identify individual visitors using cookies, which results in a better understanding of how the site is used; for example, visitors on most order sites do not need to re-enter session information already saved or some personal data every time on subsequent visits.

The major disadvantage of this technology is the fact: "Browsers generally accept only 20 cookies per site and 300 cookies in total, and each cookie can be limited to 30 kilobytes" [39]. Therefore, cookies cannot be used to keep a lot of information about each visitor.

- **URL-Rewriting**

URL-Rewriting is another session control technique whereby a browser appends extra data that identifies the session to the end of each URL, and the Web server associates the session identifier with data it has stored about the session. For example, in the following URL, http://some/site/file.html;someId=123, the session information is attached as someId=123.

This approach works even when the browser does not support cookies or when the visitor disables the cookies. A disadvantage of this technique is that it adds tedious processing tasks to the Web server. In addition, if a visitor leaves a session and comes back via a bookmark or link, the session information can be lost.

2.4 Advanced Analysis of Web Traffic: Data Mining

Mining data from the Web is a new hot area in data mining. Research in this area integrates the data obtained from the Web with some other data sources, such as customer databases. Advanced relationship between the Web data and external data sources is used when creating reports.

Many data miners, especially those who mine data to develop new marketing strategies, new product development, and enhance customers' services, pay high attention to Web traffic data. To them, Web traffic analyses are gold. Most analyses are based on data from log files, or from packet monitoring.

Using data obtained from packet monitoring, Morris and Lin [17] show that Web traffic aggregates in a Poisson-like manner in certain situations. Mah [18] obtains statistics and distributions for higher-level quantities, such as the size of HTTP files, the number of hits per "Web page", and visitor browsing behavior. Feldmann et al. [19] discuss a packet tracing system using packet level data to extract Web data.

Web log file analysis has attracted a lot of attention in recent years. Lamm and Reed [20] examine visitor WWW access patterns by extending a virtual reality system, which is designed to analyze and display real-time performance data and applied to the analysis of WWW traffic. They found "that a geographic display metaphor has provided new insights into the dynamic of traffic patterns and provides a model for development of a WWW server control center, similar to that in network operations". Crovella and Bestavros [21] show the mechanisms that give rise to self-similar network traffic using data collected on over thirty WWW sites. Schechter et al. [22] conclude that path profiles can be efficiently created by processing session data and HTTP server log files based on the concept of path profile, which "is a set of pairs, each contains a path and the number of times that path occurs over the period of the profile". Using path profiles, some novel Web applications, such as pushing a Web page to a visitor before it is requested, can be developed. Cunha et al. [23] identify a number of trends and reference patterns in WWW use and show that many characteristics of WWW use can be mathematically modeled by power-law distributions. Masseglia et al. [24] develop a novel data structure, called a Web access pattern tree, to mine access patterns from Web logs efficiently. Zaiane et al. [25] present the design of a knowledge discovery tool, WebLogMiner, to mine data from Web logs. Among the research on Web log files, some [26, 27, 28, 29, 44] analyze visitors' behavior by access patterns, while others focus on recommendation applications [30, 31, 32].

Contents of WWW pages are rich in information but cannot be queried and manipulated in a general way, because the information is semi-structured and is also dynamic (and therefore, not easily regenerated). Several research focus on approaches to mine data from Web pages. For example, Myllymaki and Jackson [35] convert HTML into XML because the data are easy to parse from an XML file. Hammer et al. [36] describe a "configurable tool" for extracting semistructured data from a set of HTML pages and for converting the extracted information into database objects. Bergholz and Freytag [37] present "a graph-based three-layer model" for querying semistructured data while Webfoot, a preprocessor that parses Web pages into "logically coherent segments" based on page layout cues, is presented in [38].

Chapter 3

The Traffic Analysis Tool Design

The design of the traffic analysis tool includes the tracking part and the analysis part. The tracking part handles how to obtain as much data as possible from a given Web site; the analysis part involves generating new information from the tracked data. The main focus of this thesis is on the former, while a few analyses are done to assess the success of the tracking functionality. The end product of this thesis is a tool that can be used to track and analyze Web traffic for e-commerce systems. The traffic tracking and analysis tool has the following features:

- **Usability**

This tool is usable. It captures activities of the users on any given site. Generally, it enables the user (i.e., management) to answer questions such as which page is frequently visited? How many users visit our site every day? How many users make a purchase? For a unique user, what (which page) is the user interested in?

- **Security**

E-commerce involves business transactions and customer's private information, such as credit card data, shipping address, and other customer personal information. Any e-commerce site and associated systems must protect the interests of both the buyer and the seller by providing adequate security. This tool does not cause any security problem to the e-commerce system.

- **Correctness**

This tool is used mainly for marketing development, and for improving efficiency of an e-commerce system, including its Web site. By following software engineering procedures to develop this tool, we ensure the tool will give correct information about users to the site.

- **Performance**

This tool should track as much data as possible. Since all the tracking programs are separated from the Web site components, these programs would not cause any functional effect on those components except some insignificant delays could exist as a result of running the tracking programs in the background. The proxy tracking works as a normal proxy server, only a negligible delay exists when a user requests a page; single-pixel simply adds one 1-by-1 pixel image to each page, no visible delay to the user exists; since the JavaScript tracking code is downloaded at the beginning and works on the client side, and the tracked data is sent when the user unloads the page, consequently, the user does not feel any delay.

3.1 Data Tracking and Storage

Traffic tracking is achieved in three ways: Proxy Sever, Improved Single-pixel, and JavaScript tracking. Proxy server tracks all requests and responses between a user and the Web server. Improved single-pixel uses a single-pixel image to track the user, while JavaScript tracking resides on the client-side and mainly tracks a user's events on a page. We briefly discuss these techniques below.

3.1.1 Proxy Sever

A proxy is an application-specific server program, which sits between a user's browser and the requested Web server. When a user visits the Web server, the user connects to the proxy server program first, and then the program sets up relay connections to the Web server. After receiving and checking the request, the Web server then returns the Web page or an error page to the proxy, which directs the page to the user. Thus, two connections are set up - one between the user's client on the source network and the proxy, and another between the proxy and the destination Web server. The proxy server transfers packets from one connection to the other throughout the lifetime of the connection. In the meantime, when the proxy server returns the page to the user, it saves the page into the data warehouse. In the later analysis, the content of the page is parsed by an HTML Web page parser, which is developed using mainly Java string manipulation technology. The parsed content of the page includes the summary of important elements such as links and images. Other session data of the user are captured directly from the proxy. This process is illustrated in pseudo code shown in Figure 3.1. Figure 3.2 shows the high-level architecture of the proxy-based data tracking mechanism.

The address of the target Web server is obtained by specifying the predetermined Web server. In this case, the relay connection can be set up automatically. Proxy function is supported by most current Web browsers.

```
10   Wait for a connection from a client host
20   Accept and open a connection to the client host
30   Establish the address of the target Web server
40   IF the connection is allowed
50        THEN BEGIN
60             Open a connection to the target Web server
70             WHILE connections are open and idle time has not expired
80                  DO BEGIN
90                       IF the data from Web server are available
100                           THEN BEGIN
110                                Read the data from the Web server
120                                Write the data to the other connection
130                                Parse and save the data into the data warehouse
140                  END IF
150         END WHILE
160  END IF
170  END
```

Figure 3.1: Simplified Application Proxy Server Algorithm

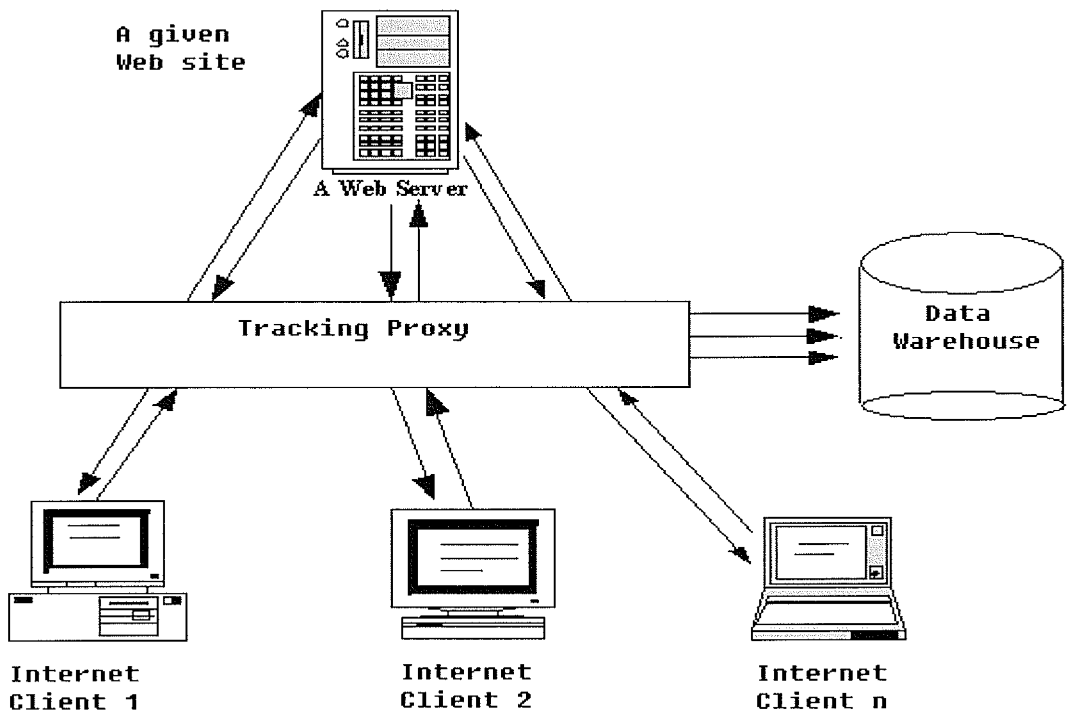


Figure 3.2: A Proxy for Tracking Traffic

3.1.2 Improved Single-pixel Tracking Approach

Another method for tracking online users in this thesis is based on single-pixel image. This method uses HTML and Java servlet technologies. The tracking process is based on Java technology and the mechanism of image downloading. An image tag is inserted on each of those pages on which we need to collect users' data. Every time the page is requested and loaded, the image is loaded from the server. This image loading event triggers the tracking process by execution of the code attached or associated with the invisible IMG tag on the server. The data tracked by this approach include the access time, IP addresses, user-agent, information about the request such as request paths, request methods (get or post), and session data. The triggered program, i.e., the tracking servlet, which is a program written in Java and runs on the server-side, collects the above information from the client through the HTTP protocol, and saves the tracked data into a data warehouse. Figure 3.3 shows the mechanism of this process.

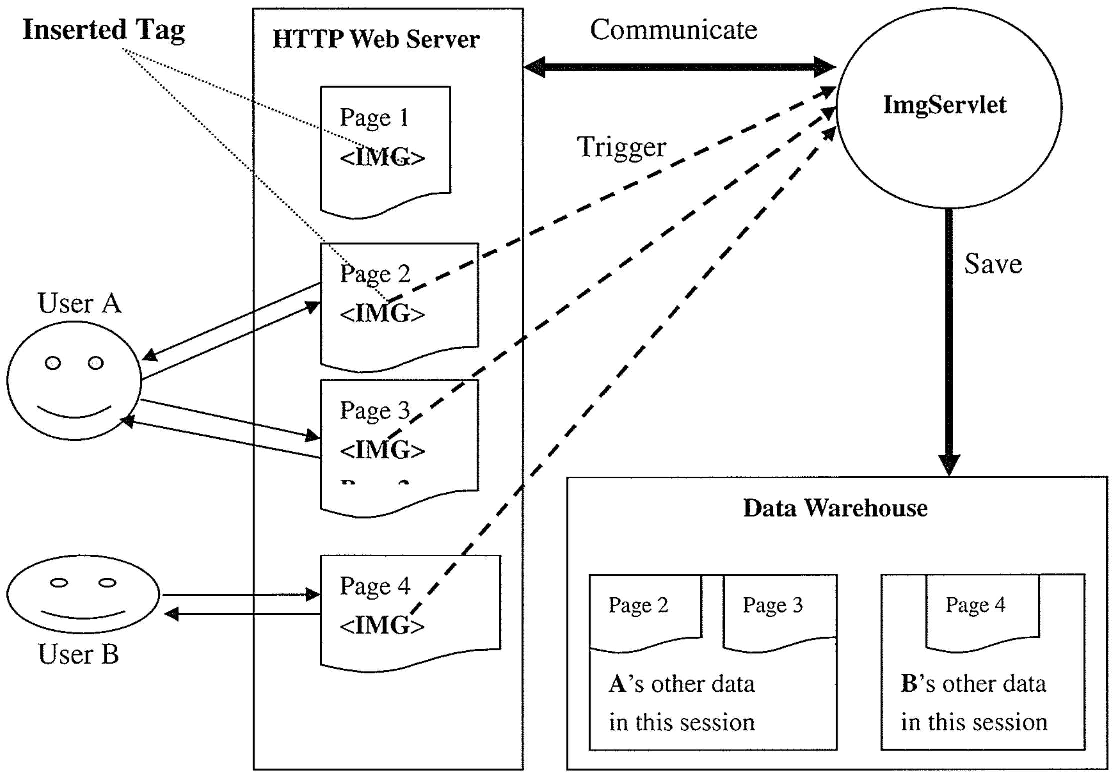


Figure 3.3: Improved Single-Pixel Tracking Approach

The mechanism used by this traffic tracking approach is similar to that of the commonly used Single-pixel technique, which is based on HTML, JavaScript technologies, and some CGI programs. However, compared with the general Single-pixel technique, the advantages of my improved single-pixel approach are:

- Unlike those available tools which need the user's browser to support JavaScript, my approach can work even if a user's browser does not support JavaScript or the user turns off JavaScript functionality on the browser. This is one of the most distinctive features which make my approach different from those available tools.
- Java Servlets are extension of Java technology. They are designed to fulfill the "Write Once, Run Anywhere" promise. Therefore, they are compatible with many browsers on different

platforms, while JavaScript and most CGI programs have some limitations on some platforms with some browsers.

- Java Servlet technology offers a more powerful control on the users' sessions than most current available methods, such as cookies. Although the session technique is based on cookie technology, the Java technology adds more powerful functions for session control and thus makes e-commerce sites to identify users' sessions and identify the unique users more effectively.
- With Java Servlet technology, more data, such as all the Web pages that a user browsed in each session, can be tracked. This is important for later analysis, since Web page content is a valuable source of data for Web data mining.

With the development of computer technologies, a lot of pages are dynamically created in response to a user's request to the Web server. To insert tracking image into those dynamic pages cannot be done as in Figure 3.3. However, the tracking image can be embedded in the dynamic code for outputting the page. In this way, the inserted image can trigger the tracking servlet when the page is requested and the image is downloaded.

3.1.3 JavaScript Tracking Approach

JavaScript enhances Web pages and servers performance. It has become the most widely used scripting language. JavaScript functions can be called, and often executed by keyboard strokes, mouse functions, buttons, or other actions from the user. Through JavaScript's full control of a user's Web browsers, the user's behavior on the pages can be captured. JavaScript is used to track a user's keyboard and mouse movements on a page.

To achieve JavaScript tracking, a few lines of JavaScript code are inserted on each of those pages where we need to collect users' data. For the JavaScript enabled browsers, when the page is downloaded, the inserted code on the page is also downloaded and runs locally to collect the user's data. When the user unloads the page, the tracked data are sent to the server program, which writes the data into data warehouse. Since JavaScript code works on the client-side, it can track almost all of a user's events, which include mouse movements and keyboard events, form input, etc. In addition, the server-side program, which is triggered by the client-side events, can track session data about the user. Figure 3.4 shows the process of JavaScript tracking.

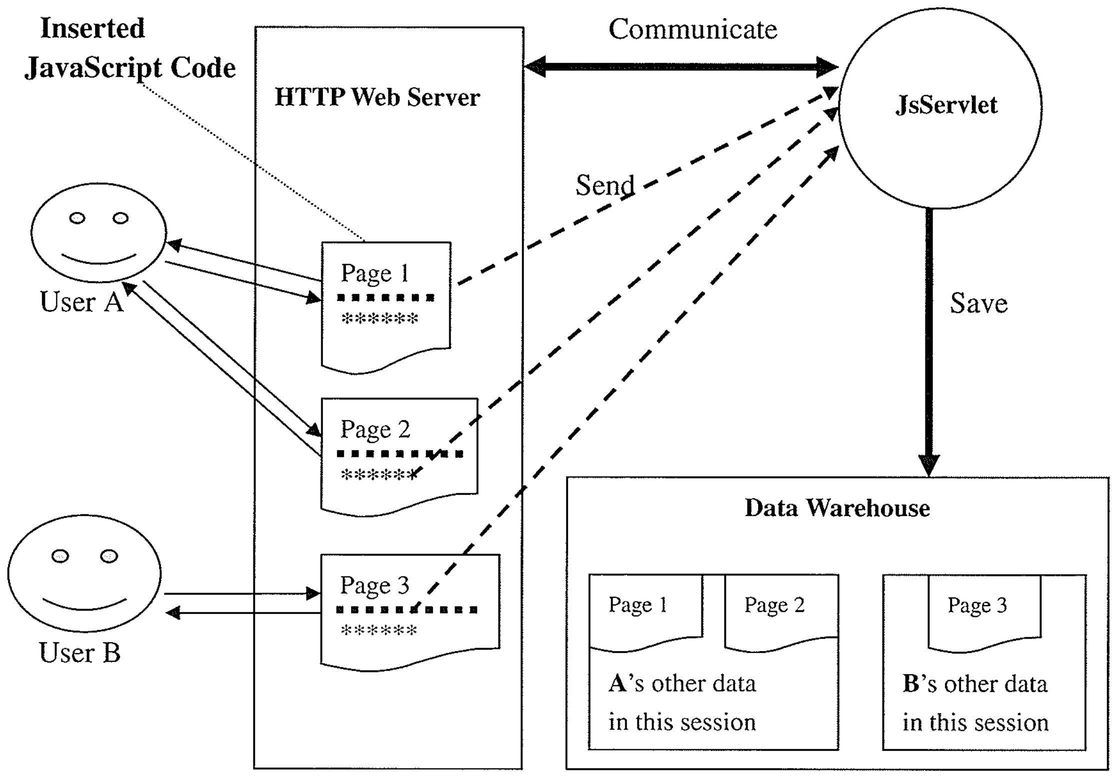


Figure 3.4: JavaScript Tracking Approach

Figure 3.3 and Figure 3.4 look similar. One difference between them is that the tracking task is achieved by tracking servlet, ImgServlet, which is triggered by the inserted image tag; while in JavaScript tracking approach, the main tracking task is implemented by the inserted JavaScript code. During navigation on a page, some user's events will trigger the JavaScript code to send the tracked data to the server-side program, JsServlet, which mainly works for the purposes of session control and basic data processing.

Besides the advantages of using Servlet over CGI programs as in Single-pixel tracking, the other major advantages of my JavaScript tracking approach are:

- It is easier to add the tracking code on any Web page, since all the JavaScript codes are grouped in several external files.

- It works on both static and dynamic Web pages.
- It can capture the user's keyboard events and mouse movements. A unique feature of the JavaScript tracking technique that distinguishes it from the other tracking approaches is that it can capture any form input including text-area, text-field, selection, etc.

3.1.4 Data Storage

Another important aspect of the tracking system is the data storage. Any popular Web site has hundreds and thousands of online users. To keep all data of the users for such a Web site, a huge storage space is needed. Since the data include not only the Web usage data, which a lot of log files can capture, but also the users' keyboard movements and page content, which involve images and links, these data need much more storage than textural information.

A data warehouse is an ideal storage model for the large sets of data. In this thesis, the proxy tracks page and session data. The proxy saves all data into the data warehouse when the page is requested by the users. The data captured using the JavaScript code on each page are saved into strings with a pre-defined format and sent to the data warehouse finally by the servlet program. The data captured using the single-pixel approach are saved to the data warehouse when the image is downloaded.

Figure 3.5 shows how the tracking code is inserted into a static Web page. To insert IMG tag and JavaScript code into a dynamic Web page, simply embed them on the response page. When the page arrives at the client-side, the client's browser will download the image and the JavaScript code. As a result, the single-pixel tracking and JavaScript tracking will run.

```
<HTML>
< HEAD><TITLE>User Behavior Tracking Code</TITLE>
<SCRIPT> JavaScript Tracking Code</SCRIPT>
<IMG SRC= "http://ImgTrackingServlets"BORDER=1 HEIGHT=1></IMG>
</HEAD>
<BODY >
<SCRIPT>
  Call JavaScript functions in the above code
  Send tracked data to the server side.
</SCRIPT>
                              Original Page Content
</BODY >
</HTML>
```

Figure 3.5: Insert the Tracking Code into a Web Page

3.2 Data Analysis

In e-commerce, users' navigational behavior indicates their steps through the buying process [54]. Therefore, analyzing tracked navigational data is rapidly becoming one of the most important activities for any business on the Web. On the other hand, from the analysis of what the online users do, the business knows how its site is being used.

The anonymity of online users means that the tracked data about the users' navigational behavior are of little use or meaning if the following three major problems cannot be solved. First, any data should be related to a unique user; second, all the data related to a unique user should be sorted in sessions; and third, determine how the users navigate around this site. The analytical results of these three issues are the common focus of all businesses for the purpose of marketing or site management. In Web data mining, these three problems areas called User Identification, Session Identification, and Path Analysis, respectively.

3.2.1 User Identification

To identify a unique online user is a hard task. The existence of local caching, corporate firewalls and proxy servers contributes to the complexity of this task.

Normally, local caching (e.g., the user's browser's caching), allows a user to get a requested page from its browser's caches if the page already exists in the cache. If the local caches exist, when the user clicks "back" to return to a visited page, this request is not passed to the Web server, and the proxy cannot log this event. But with the improved single-pixel method, this event can still be tracked and, thus, this does not represent a significant problem.

IP address is the most commonly used method to identify an online user, but proxy servers and firewalls raise some challenges to this method. A lot of online users come from corporate networks. Most corporate networks today are behind firewalls or use proxy servers. In this context, a proxy server is a computer which is set up and stands between an organization's Intranet and the Internet. Only the proxy server's IP address, which hides all computers' real IP addresses on the intranet, can be seen publicly by computers outside the organization. When the computers inside the intranet are started, the proxy server manages and dynamically assigns the organization's IP addresses to them. Also, the proxy server manages an organization's intranet's connectivity. With a proxy server, a computer outside the proxy server sees the requests or responses from any computer on the Intranet as originating from the proxy server's IP address and uses that IP address to communicate; on the other hand, the proxy records the dynamically assigned IP addresses of all the computers on the Intranet, and routes the requests or responses from the outside computers to the inside destination computers [73].

A firewall is a collection of components placed between the company's intranet and the Internet. It may be part of a proxy server or a router. The purpose of a firewall is to prevent unwanted users from accessing computers within an organization's intranet. At the same time it allows inside users to access the Internet. A firewall can be configured to allow or disallow specific external IP addresses to access some or all of the organization's resources, such as the organization's HTTP Web servers, FTP servers, telnet servers, etc. [73]. Like a proxy server, the firewall can prevent making public individual computers' IP addresses inside the firewall to computers outside the firewall. Instead, all requests and responses from the computers inside the firewall would appear to the computers outside the firewall to come from the same IP address.

Even with the improved single-pixel approach, to uniquely identify a user simply by the IP address is impossible or at least is a very inaccurate way. Some other methods should be used to deal with this problem.

Cookie technology and dynamic URL with an embedded session identifier have been widely used by Web sites to identify unique users [39], though they both are not as powerful and effective in keeping users' information as session control technologies. However, cookie technology is the basis of session technology, and we still track cookie value as a supportive approach to identify unique users if such values are available, while using user Ids tracked in JavaScript code as the major measure to identify the unique users.

Since we have three sets of tracking procedures, we have more session data. When one approach does not work well, we still have the approach to collect data. Based on the rich data, we can identify a user in different ways once some measures are less effective. Moreover, our tool records data by page views instead of hits. This approach reduces the amount of work required to purge those unnecessary data in the process of identifying unique users.

Figure 3.6 shows the algorithm to identify whether two tracked records in the data warehouse are from the same user. The steps to identify whether A and B are from the same user are as follows:

1) Check whether A and B are associated with some userid values. If the values are available and the same, A and B are from the same user. Otherwise, go to next step.
2) Check whether A and B are associated with cookie values. If yes, that means the browsers for A and B were cookie enabled. If either or both of A and B are from a browser with cookie disabled, go to step (3).

3) Check the values of the cookie. If same, it means A and B were from the same user. Otherwise, they were from two different users.
4) Check whether A and B are from the same IP address. If not, we regard A and B as from different users. Otherwise they are from the same IP address, and more checks need to be done.
5) Check whether the browsers and operation systems for A and B are the same or not. If not the same, we assume they are from different users. Otherwise, go to next step.
6) Now, A and B are from the same IP address and are associated with the same browser on the same operating system. So far, we still cannot determine whether A and B are from the same user. Some other supportive work needs to be done to make this decision. First, we check whether the referrers for A and B follow some logical relation or not based on the site topology. For example, to find out whether B's referrer page could be reachable from page A's link (suppose A's access time is the earlier one here). If not, we think A and B were from two users. Another check is done on the access time. If the interval of the access time for A and B are more than 30 minutes, we think A and B were from two different users under this situation where after all other checks have been done and no obvious evidence supports A and B are from the same user.

Due to the characteristics of the Web environment, it is impossible for any algorithm to 100% precisely identify unique online users. This algorithm does not either. In some situations, for example, if a user browses a site with two different browsers on one computer at the same time and no cookies enabled, the tracked records A and B from the two browsers are regarded as from two users [49]. Another example is that if a user leaves a site for a couple of hours and comes back. If the user's browser is cookie disabled and no userid is tracked, it is impossible to differentiate whether it is from two different users or one same user, though A and B show that they are from the same IP address, the same browser on the same operating system. However, this algorithm regards these two records as originating from two different users. This algorithm, however, based on most Web users' regular behavior, can offer fairly accurate and valuable statistics of the traffic of the site over time.

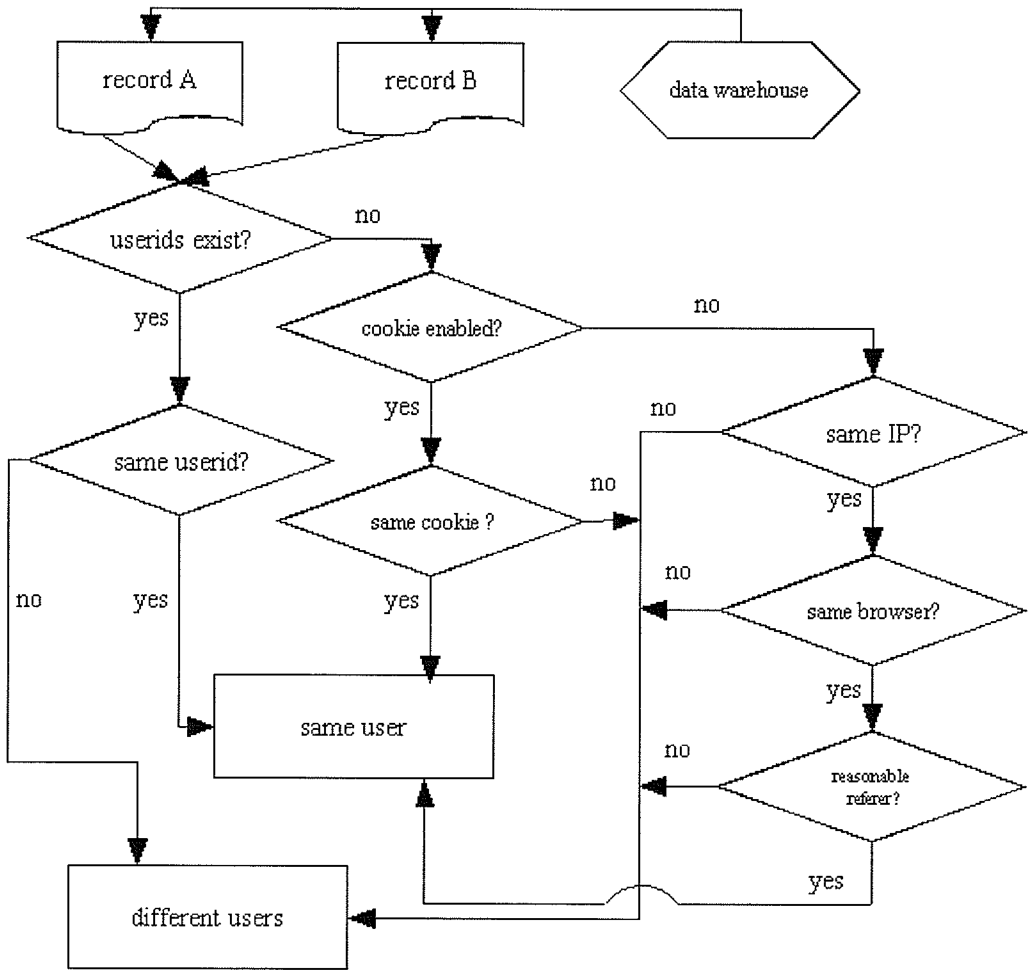


Figure 3.6: Identify Unique Users

3.2.2 Session Identification

A session is the span of the time that a user stays with a site during one visit. To any Web site, users can be new or returning ones. The data warehouse already has the data about all previous users. For a frequent user, the data are huge in size. Less meanings could be found from user's data like these: page (A) 3 times, Page (B) 4 times, Page (C) 5 times during three visits. But if we divide these data into three sessions, {A, B, C}, {A, B, B, C, C}, and {A, B, C, C}, we can know exactly what the user browsed during each of the past three visits. The purpose of the session identification process is to group all activities of a user based on each visit of the user.

Because every time a user spends different lengths of time in each visit, this raises the difficulty to define the length of a session. Generally, most commercial Web products use 30 minutes [48] as a default timeout or 25.5 minutes based on empirical data according to [49]. Using these values as a guide offers a way to session division, though, sometimes, the actual length of the timeout can vary with the actual usage of different sites. In this thesis, the length of a standard session is 30 minutes.

Figure 3-7 shows the algorithm for identifying sessions. In this algorithm, we use a list structure to keep each session data, although other data structures could also be used to keep session data. Each session is a node on the list. Each session consists of a vector which includes one or more records. After completion of identifying sessions, the original data become nodes of the list which is linked and sorted by the access time. This data organization technique facilitates the later search and analysis process.

Through user and session identification processes, all the original tracked data are now grouped according to different users and sessions. From these processed data, we can derive some simple facts about the users, such as which pages a user browsed, when, etc. To know more information about the users, we must know how they navigate through the site, where they come, and exit the site.

3.2.3 Path Analysis

Path analysis is a technique, based on the analysis of the users' navigational patterns, to understand how the users navigate through the site. It also sheds light into the site structure and helps locate trouble spots of the site [50]. Unlike a regular Web site where most users navigate following random patterns, e-commerce sites share a lot of common features in their site structures so that shoppers can easily view product catalogue, add or remove products, and make payment. For example, most sites have catalogue pages (for showing products), shopping-cart or basket operation pages (for adding or removing products), check-out page (for payment), and payment confirmation page (for payment confirmation). Therefore, for most e-commerce sites, each of them has at least one expected typical path which leads its users to navigate through the site and achieve a purchasing process with a high level of effectiveness, which means ease and ability to provide users with impulse purchase opportunities during browsing and attract and retain more customers.

A practical business goal of path analysis is to understand why some shoppers abandoned the buying process and leave before making a purchase. Focusing on the entry-URL and exit-URL, we can find or assume the answers to the above question. Normally, the expected entry-URL is the first page of the site such as index.html, home.html, etc.; the exit-URL is the payment confirmation page. A single shopper's data can not reflect something very meaningful. But if a lot of shoppers leave before going to make payment, this may suggest they do not like what they put into their shopping carts, or perhaps some pages on their way to make payment are running too slowly [50].

Basic path analysis is based on checking both referrers and site topology to support user identification [49]. Advanced path analysis can be used to achieve more complex tasks such as serving as a basis of personalization [55] or recommendation systems. One of such examples is to predict HTTP requests [51], which is based on path profiles and recommends an URL with a high probability to the user before the user makes such a request. A lot of analysis has applied sequential pattern discovery techniques [52, 53] to discover frequent path patterns. In this section, we develop a data structure and an algorithm to find the frequent navigational paths.

```
(10)  // create a list for all sessions
(20)  list sessions = null
(30)  // define a variable to keep each session id
(40)  session = null
(50)  // outside loop to input a record, I is the control variable
(60)  I = 0
(70)    WHILE ( input record( I ) ≠null )
(80)        // check whether this record is already on the list
(90)        FOR each record on the list ( stepping through using a sequenceIndex)
(100            set a flag to true if record( I ).sessionId is found on the list
(110)       END FOR
(120)       IF the flag is true
(130)           GOTO (320)
(140)       ELSE
(150)           session1 = record( I )
(160)           //inside loop starts at J.
(170)            J = I + 1
(180)           WHILE (input record( J ) ≠null)
(190)               IF record( I ).sessionId = = record( J ).sessionId
(200)           //record( J ) and record( I ) in the same session, add record( J ) to session
(210)                   session.add(record( J ).sessionId)
(220)                   //check next data
(230)                   J = J + 1
(240)                END IF
(250)               // else if the two session ids are not same, input the next record
(260)               ELSE    J = J + 1
(270)             END WHILE
(280)               //now session includes all data for this session, add it to the list
(290)             sessions.add(session)
(300)         END ELSE
(310)         // next outside while loop at (70)
(320)         I = I + 1
(330)     END WHILE
(340) END
```

Figure 3.7: Algorithm for Session Identification

3.2.3.1 Definitions

Before presenting more details of path analysis, we define some terms as in [51].

Definition: Path

A path P is a sequence of URLs accessed by a single user, which are ordered by access time. In the sequence, same URLs can appear more than once. For example, (1, 1, 3) and (1, 3) are two different paths.

Definition: Path Length

The path length |P| is the length of the path P. |P| equals to the number of pages accessed by a single user in one session. For example, if a user visits the following pages (1, 3, 5, 6, 5, 7) in one session, |P| is 6. A session S for a user denotes all URLs ordered by access time and accessed by the user in one visit or during a predefined length of time, for example, 30 minutes.

Observation: In a given session $S = (S_1, S_2, \cdots, S_n)$, $S_1, S_2 \cdots, S_n$ are all paths of length one, (S_1, S_2), (S_2, S_3), and so on are all paths of length two, etc. Therefore, there are |S| paths of length one, |S|-1 paths of length two, |S|-2 paths of length three, and finally, only one path of length |S|, the session path. In the session S, the total paths P in S are:

$$(|S|-0) + (|S|-1) + \cdots [|S|-(|S|-1)] = \frac{|S|*(|S|+1)}{2} \approx \frac{|S|^2}{2}$$

Some mathematical analysis in [51] show that the average length of the path in session S is $\frac{|S|}{3}$, thus the total number of URLs that would be needed to store every path can be calculated as follows:

$$\frac{|S|^2}{2} * \frac{|S|}{3} = \frac{|S|^3}{6}$$

Obviously, it makes sense if the above number is an integer and more than 1. To satisfy these two requirements, we have $\frac{|S|^3}{6} \geq 1$, which leads to $|S| \geq 1.82 \approx 2$. This means in any session with only one page is requested, the path analysis based on one page is of little meaning and, thus, can be overlooked. This finding allows us to consider those important paths in analysis. This helps decrease the requirement for storage and memory in the process of analysis.

Definition: Prefix and Maximal Prefix of a path

A path Q is a prefix of path P if and only if the following two conditions hold:

1). $|Q| < |P|$, and

2). All the elements of Q are the first $|Q|$ elements of P and in the same sequence.

If Q is a prefix of path P and $|P| - |Q| = 1$, that means all URLs in Q are included sequentially in P, and P has one more URL (the last one in the sequence), then Q is the maximal prefix of P.

Definition: Suffix of a path

A path R is said to be the suffix of path P if and only if the following two conditions hold:

1). $|R| < |P|$, and

2). All the elements of R are the last $|R|$ elements of P and in the same sequence.

3.2.3.2 Path Tree Node Data Structure

For in the later analysis, we must know which paths exist and how many times a path occurred. To efficiently reflect these two requirements for a path, we use the record data structure, which is similar to that in [51] and shown in Figure 3.8, to represent a node of the path tree.

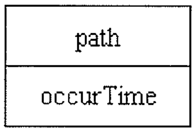


Figure 3.8: Data Structure for A Node in the Path Tree

In this data structure,

1) "occurTime" is an integer, which means the number of occurrences of this path. Its default value is 0.
2) "path" denotes a path which is already in the tree.

3.2.3.3 Data Structure for URLs

In order to obtain all paths in each session of a user, we should know all the pages the user accessed in that session. This leads to a problem: whether the URLs we are using to form paths should contain any parameter field or only the name of the requested file.

The current version of HTTP allows a Web client to pass almost everything in the parameter field when using the GET method and the attached information forms an essential part of the URLs. The search engine "Google" is one obvious example. When searching for "hello" using "Google", after the search, the URL on the result page of the search becomes "http://www.google.ca/search?hl=en&ie=UTF-8&oe=UTF-8&q=hello&meta=". We find some parameters are attached to the original URL "http://www.google.ca". However, on the Web, most of the attached parameters to URLs serve no purpose other than tracking the users' sessions [56]. They are of less use for analyzing paths. Since in the path analysis, for example, we are interested only in knowing whether a client used "Google", not what the client searched. Though no agreement about whether or not keeping parameters to URLs in data mining has been reached, in this thesis, all parameters are ignored.

After processing the parameters in all URLs, we use the following data structure to represent a URL to facilitate comparison and store process of URLs. Suppose a user has browsed the following 5 URLs in a session:

http://www.yahoo.com	(p1)
http://autos.yahoo.com	(p2)
http://www.yahoo.com	(p1)
http://auctions.shopping.yahoo.com	(p3)
http://user.auctions.shopping.yahoo.com/showcase/fathersday03	(p4)

We assign each URL a same letter and a unique integer (for example: p1, p2, p3, p4) to represent a URL string. If a URL appears twice (for example, p1) in a session, that string (p1) appears twice too. In this way, the above session can be represented as a string array {p1, p2, p1, p3, p4} which in fact represents all the URLs the user browsed in that session. One benefit of this kind of data structure is that we can easily apply string operations to differentiate and show paths in a straightforward way. For example, if we know a path is from page 1 to page 2, we can use p1p2 to represent it; on the other hand, a path of p1p3p4 can be straightforwardly interpreted as the path starts from page1 to page 3, and then to page 4.

3.2.3.4 An Algorithm to Create a Path Tree

```
10   INPUT an array A // all the Web pages a user browsed in a session
20   set LENGTH = the length of the array A
30   create an empty path node T as root node, T.path = null and T.occurTime = 0
40   set outer loop control variable I = 0
50   WHILE (I < LENGTH)
60     set T as currtentNode
70     set inner loop control variable J = I +1
80     set N as the minimum path length required in the path tree
90     create a temp node TEMP, TEMP.path = A[ I ], TEMP.occurTime = 0
100    WHILE (J < LENGTH)
110        set TEMP.path = TEMP.path + A[ J ] // string operation
120        IF the length of TEMP.path > N
130            IF there exists a node Y with the same path as TEMP
140               set Y.occurTime = Y.occurTime + 1
150               set Y as current node
160            ELSE
170               add TEMP as a child of the root node T
180               set TEMP.occurTime = 1
190               set TEMP as current node
200        NEXT J
210    END WHILE
220  NEXT I
230  END WHILE
240  END
```

Figure 3.9: An Algorithm to Build a Path Tree

The tree structure is an efficient and straightforward way to store non-linear data like paths. We show an example of such a tree based on the data from one user's session. Figure 3.9 shows the pseudo-code of the algorithm, while Figure 3.11 shows the diagram of the tree built using this algorithm.

To use this algorithm, the first task is to determine what data source should be used. Some authors (e.g., [54]) think a session with at least five page views offers a usable data source. Though this choice can be made subjectively, in this thesis, we think a session with at least two page views can be used.

The number of Web users is large, running into several millions. Considering data storage and memory constraints, we can only record those "important paths" into the path tree. No uniform opinion about what paths are "important" for data analysis exists. Some people prefer at least two URLs, while others would like more [51]. This algorithm provides such scalability and flexibility by allowing the user to choose the minimal path length to record into the path tree.

An empty node, shown in Figure 3.10, is created every time a node is needed.

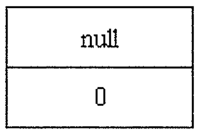


Figure 3.10: Initial Values for the Empty Node

We follow the algorithm to record all the important paths for all users' sessions into the tree. The following steps show how this algorithm works. Suppose a user visited the following five pages in one session:

Page 1: http://www.yahoo.com

Page 2: http://photos.yahoo.com

Page 3: http://www.yahoo.com

Page 4: http://photos.yahoo.com

Page 5: http://maps.yahoo.com

According to how we represent URLs as strings, the above five URLs in this session can be represented as a string array: (p1, p2, p1, p2, p3).

At the beginning of the execution of the algorithm, we know the length of the input array is five. Also, we create and initialize an empty node, as Figure 3.10, to be the root node of the tree. To control the outer WHILE loop, a control variable I is set. The range of I, in this example, is from 0 to 4 (less than 5).

The outer WHILE loop starts with setting the root node as current node and setting another index J for controlling the inner WHILE loop.

In the first outer WHILE loop, I equals to 0. The current node is the root node. The value of the inner loop control variable J equals to 1 (J = I + 1). A temporary node TEMP is created with the path value as A [0] (i.e., p1). In this example, we set N, the minimum length of a path string which we record in the path tree to 2. The first inner WHILE loop starts with J = 1. The path of node TEMP becomes p1p2. The length of this path is 2, which is not less than 2 (i.e., the value of N). This path string satisfies the length requirement for the path string in the path tree. The next step is to check whether there exists a node with the same path string as p1p2 in the tree or not. If such a node exists, we simply increase the occurrence times of that path one more time and set that node as a current node; otherwise, add this node TEMP as a child node of the current node, set the occurrence time of node as 1, and make this node as the current node. Since no such node in the tree with the path value of p1p2 exists, in this example, the node TEMP is added to the current node, i.e. root node. The new node has the path value of p1p2 and occurTime value of 1. This new node becomes the current node. Figure 3.11(a) shows this result. After adding this node, the inner loop runs again with J = 2. In this loop, since the variable TEMP keeps its old values (i.e., TEMP.path = p1p2 and TEMP.occurTime = 1), after a new string A[2] (i.e., p3) is added to the current path string. The new path value of TEMP becomes p1p2p3. Since there is no such node with the same path value of p1p2p3 existing in the tree, this node is added to the tree as a child of the current node, which we just created in the first inner loop. Also, the new set value of occurTime of this node overrides the value saved from the previous loop. This ensures that every time a new node is created, the occurTime of the new node is 1. Figure 3.11(b) shows the result. Following the same procedures two more times (J = 3 and 4), we arrive at figure (c) and (d) of Figure 3.11. So far, the first outer loop (I = 0) ends, and the current node goes back to the root node.

In the second outer loop, I equals to 1. The inner loop starts with J = 2. Just like in the first outer loop, new paths p2p1, p2p1p2, and p2p1p2p3 are added to the tree as shown in Figure 3.11(e) after 3 inner loops with J = 2, 3, and 4.

In the third outer loop, I is 2. The inner loop starts with J = 3. In this first inner loop, there exists a node, which is created at the beginning and whose path is p1p2, same as that of TEMP.

According to the algorithm, we cannot add TEMP to the tree as a child of the current node. Instead, we update the occurTime of that node in the tree by increasing it by 1 and set that node as the current node. This is shown in the Figure 3.11(f). The occurTime of the node whose path equals to p1p2 is 2. The third outer loop ends with another new node with the path value of p1p2p3 added to the tree.

In the fourth outer loop, I equals to 3. The inner loop starts with J = 4, which is the range limit of the J. Therefore, the inner loop can run only once and a new node is added as a result. Figure 3-11(g) shows this. Although the outer loop can run one more time with I = 4, since J exceeds its range limit, therefore, Figure 3-11(g) is the final path tree for this user's session.

Results of Each Step	(a)	(b)	(c)	(d)	(e)	(f)	(g)
Outer Loop: I	0	0	0	0	1	2	3
Inner Loop: J	1	2	3	4	3	2	1

Table 3-1: Results of Each Step to Create the Path Tree in Figure 3.11

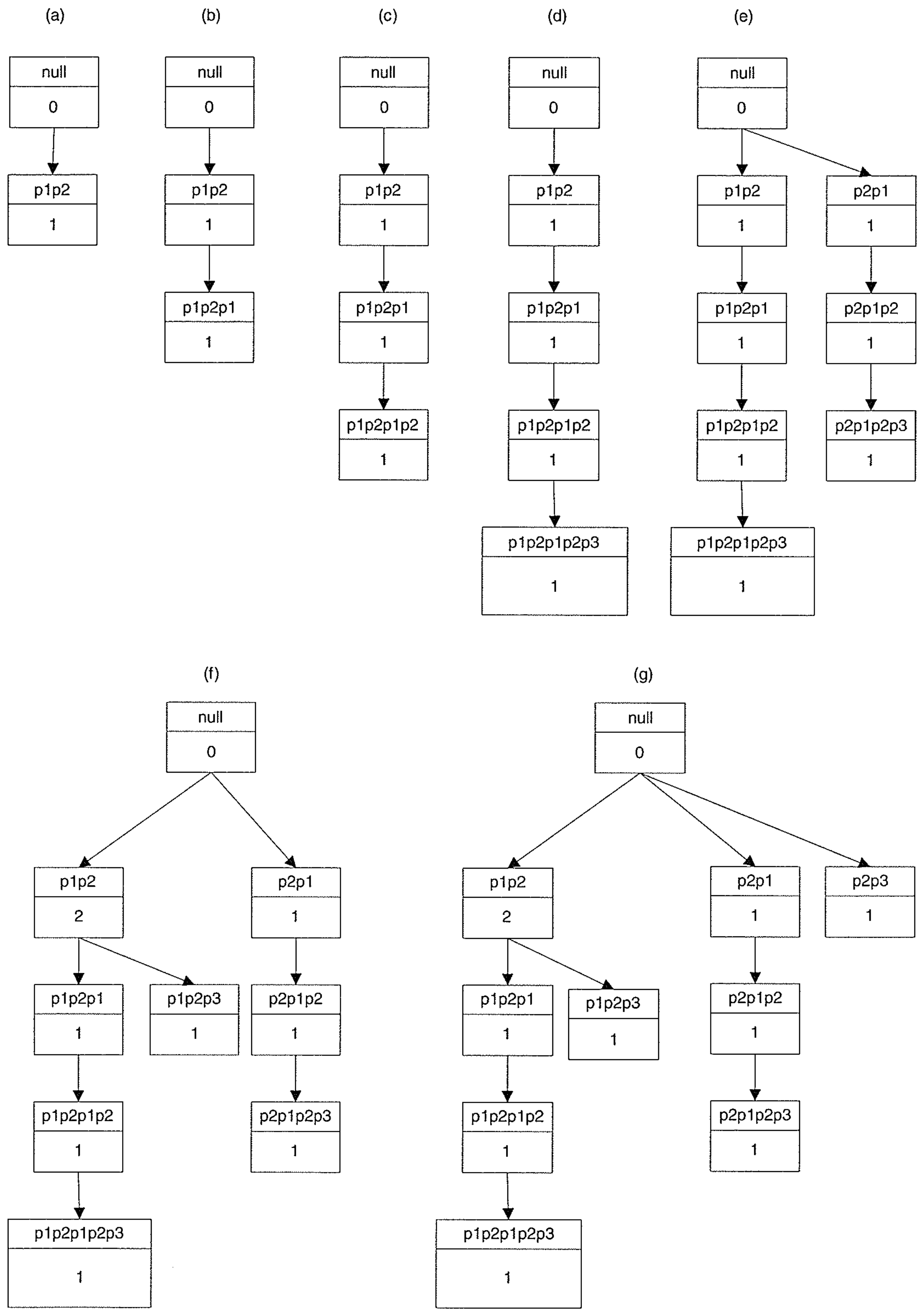


Figure 3.11: The Steps to Build a Path Tree

3.2.3.5 Path Analysis and Evaluation

To analyze and evaluate our path tree, we can apply the statistical concepts of support and confidence [58], which are usually used to evaluate association rules. Association rule [59 60] is another popular approach in data mining.

Definition: Association Rule [31]

Let P = $\{p_1, p_2, \ldots, p_m\}$ denotes a collection of m items, $T=\{t_1, t_2, \ldots, t_n\}$denotes another collection of n items whose elements always occur together, and $T \subseteq P$. An association rule between two sets of items X and Y, such that X, $Y \subseteq P$ and $X \cap Y = \varnothing$, means that the presence of items in the set X in T indicates a strong probability that items from the set Y are also present in T. Such an association can be denoted as $X \Rightarrow Y$.

Two key values, support and confidence, are commonly used to evaluate the quality of association rules. The support (s) measures the occurrence frequency of the pattern in the rule while the confidence (c) is the measure of the strength of that rule. For a rule $X \Rightarrow Y$, the support s is defined as:

$$s\% = \frac{\text{number of considered sets which containing } X \cup Y}{\text{number of considered sets}}$$

In other words, support s means that s% of considered sets contain $X \cup Y$. For the rule $X \Rightarrow Y$, the confidence (c) indicates that c% of considered sets that contain X also contains Y.

$$c\% = \frac{\text{number of considered sets containing } X \cup Y}{\text{number of considered sets containing } X}$$

A high confidence value means the predication is with a high accuracy, while a high support means the considered rules occur more frequently. Therefore, when using this approach, high confidence and support values are preferred.

To apply association rules to our analysis, we can think of our previous paths in the following way: Two paths P_a = p1p2p3 and P_b = p1p2p3p4p5 can be considered as p1p2→p3 and p1p2p3→p4p5, respectively.

Now, consider the following path array {p1p2p3, p1p2p3p4, p1p4, p2p4, p2p1p2p6, p1p2p3p4, p1p2p3p6p7, p1p2p5, p4p1p2, p1p2p3}, which contains 10 paths. Following the model, we set X = p1p2, Y = P3. We can obtain the support (s) = 50% (5 out of 10), and the confidence (c) = 62.5% (5 out of 8). We can translate these results as: 50% users who go to the path p1p2 will go to p3; 62.5% users who have been to p1p2 have been to p3.

Path analysis plays a key role in traffic analysis of e-commerce sites. One obvious benefit is that we can find which paths are navigated more often so that we can put our key products there. Also, we can find why some path are visited less by checking the site topology. Moreover, path analysis can lead to a basis for personalization and recommendation systems. For example, if we know a path P_c is followed by another path P_d with a high support and confidence, we can recommend the pages matching the path P_d to those users who are navigating the path P_c.

3.2.4 Other Statistical and Knowledge Discovery Techniques

Recommendation systems and personalization are two related and popular research areas in Web data mining. They both apply statistical and knowledge discovery techniques to achieve serving/selling of more products, thereby making more money for e-commerce sites [58].

Collaborative filtering [61, 62, 63, 64] is one of the earliest and most commonly used approaches in data analysis for developing the basis of recommendation systems and personalization. This approach works by matching a new user against a pre-built database, which stores consumers' preferences for products. If some neighbors, who are customers already in the database and have the same taste as the new user, are found, products favored by those neighbors are recommended to the new user. Although this approach is most suitable for homogeneous, simple products [57], it has the following three major drawbacks: the requirement of large base of customers, the reliance on mainly human inputs which are subjective and prone to biases, and easily aging user profiles [56, 57].

Another most commonly used data mining approach in e-commerce is association rules. One purpose of this approach is to find association rules between a set of co-purchased products. In other words, it is to discover "association between two sets of products such that the presence of some products in a set is also present in the same transaction" [58]. Some authors [49, 54 65, 66, 67] have proposed to extract Web usage patterns from Web logs for marketing development purpose, while others [68, 69, 70,71] have applied clustering of user sessions to judge user behavior [56].

In this thesis, we do not intend to develop such an application as recommendation systems or personalization; instead, we present another statistical and knowledge discovery technique to help our analysis.

3.2.4.1 Clustering User Transactions

Clustering is the process of grouping similar objects. As a result, each cluster is a set of similar objects [72]. The basic data in a clustering operation consists of a number of similarity judgments on a set of objects. This mechanism helps us apply clustering to mine information from Web traffic data.

To apply clustering method to our data analysis, we need to define a user session in another way as in most Web usage mining literature, such as [56].

Definition: A User Session File and a User Session

A user session file includes all sessions of a user. A user session $P = \{p_1, p_2,\ldots,p_n\}$ can be thought as one transaction of n page views (i.e., n pages can be represented in one transaction P), or n transactions of single page view (i.e., one transaction includes n pages), where p_1, p_2, … p_n are URLs browsed in that session.

Any Web page can be categorized as a content page, navigational page which leads a user to the content page, or mixed. Based on a page's category, a user session which includes one or more pages can be divided into one or more small transaction sets [49]. To simplify our analysis, we assume that each user session is viewed as a single transaction and those session files with low support references (for example, not more than two URLs) are filtered and discarded. This can remove the noise from the data and improve space and speed of later analysis. [56].

After identifying all users and all session files for a given Web site, we can obtain the following set U, which contains all unique URLs and a set T which contains m user transactions.

$$U = \{url_1, url_{2,\ldots,}url_m\}$$

$$T = \{t_1, t_2 \ldots, t_n\}$$

where $t_i = \{url_1, url_{2,} \ldots url_r\} \in T$ is a non-empty subset of U.

In this way, all the user transactions can be mapped into an $n \times m$ matrix of URLs. Each transaction is a row in this matrix. We use binary weights to represent whether or not a URL is included in a transaction $t_{i.}$ The following example shows one 3×4 dimensional matrix which represents a set of 3 transactions, $T = \{t_1, t_2, t_3\}$ and a set of 4 unique URLs, $U=\{url_1, url_2, url_{3,} url_4\}$.

$$\begin{array}{c|cccc|} & url_1 & url_2 & url_3 & url_4 \\ t_1 & 1 & 0 & 0 & 1 \\ t_2 & 1 & 1 & 1 & 0 \\ t_3 & 0 & 1 & 0 & 1 \end{array}$$

To find similarity between two transactions, we have to find the distance between them. Since each transaction can be regarded as a vector in the above matrix, it is a straightforward way to find the distance between two transactions by measuring the cosine value of the angle between two edges – two vectors. The cosine value is zero if two transactions have no URLs in common and one if the two transactions are identical. In other situations, the cosine value is an intermediate value. If the value is closer is to one, the two transactions are more relevant. The cosine formula is shown as follows.

$$\cos(\vec{a},\vec{b}) = \frac{\vec{a} * \vec{b}}{\| \vec{a} \| \times \| \vec{b} \|}$$

where $\vec{a}$ and $\vec{b}$ are two transactions, the * means the inner product, while the vertical bars denotes the 2-norm.

To decrease the dimension of our transaction matrix, like what we presented in path analysis, we do not consider those "unimportant" transaction files.

After calculating all the distances, we obtain a new $(n-1) \times (n-1)$ dimensional matrix which consists of all distances. Each element of this matrix is a float value between 0 and 1 inclusive. This matrix forms the basis of our clustering.

Many clustering algorithms are available for clustering operations, such as the k-means algorithm [72]. Although we use simple distance clustering in our later implementation, we show the approach to apply k-means algorithms on our data source – the distance matrix. To achieve this, we can create a set S of transaction clusters. $S = \{s_1, s_2, \ldots, s_r\}$. Each element of S is a set of user transactions and represents a group of users with similar navigational patterns.

Instead of creating a mean vector in each cluster as in [56], we go back to each transaction file and compute the percentage of each URLs occurrence times in each cluster. This way is straightforward and much simpler. To decrease the dimension of each cluster, we can set a threshold percentage to keep only those important transactions which are of high percentages.

These clusters can be used to analyze users' interest. For example, suppose a user has browsed the following pages: url_1, url_3, url_5. If there exists a cluster $\{url_1, url_3, url_5, url_9, url_2\}$, we can assume that, like other users in the cluster, this user may also be interested in url_9 and url_2. Moreover, if the percentage of url_9 is higher than url_2 in the cluster, then url_9 can be recommended to this user in a simple Web recommendation system, though most recommendation systems do more analysis before such a recommendation is made.

3.3 Design

Figure 3.12 and Figure 3.13 show the class diagrams of the tracking part and analysis part for this traffic analysis system.

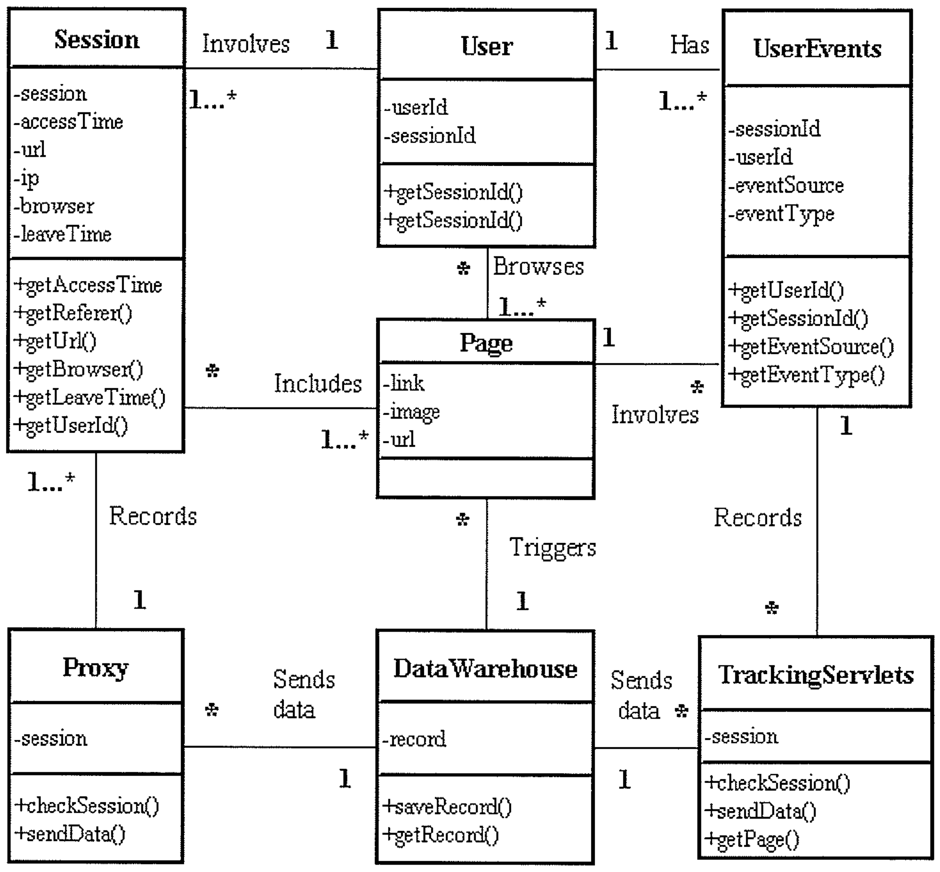


Figure 3.12: Class Diagram for the Tracking Part

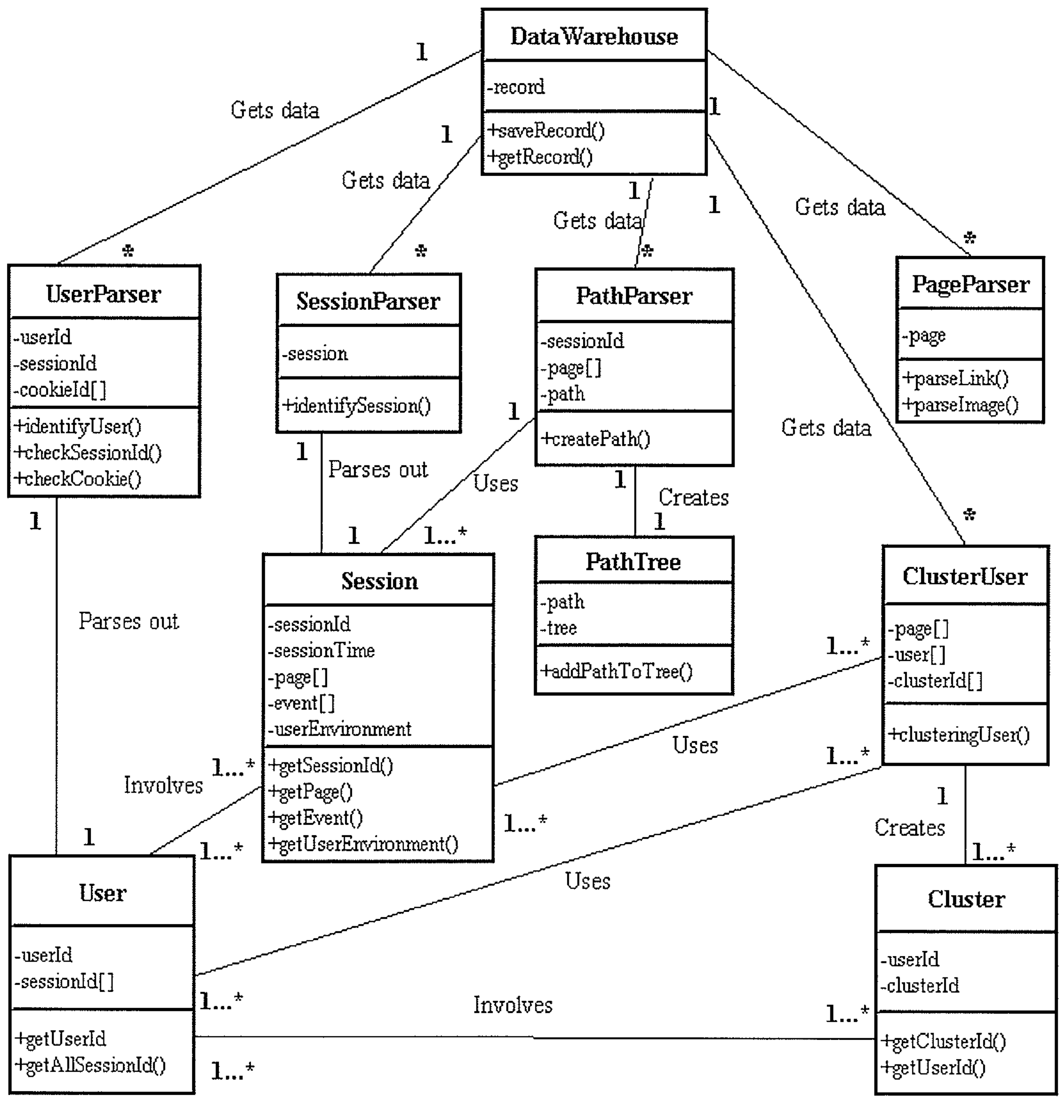


Figure 3.13: Class Diagram for the Analysis Part

Chapter 4

Implementation

4.1 Introduction

The traffic tracking and analysis tool runs at the back end of an E-commerce Web site mainly for the tracking and analyzing of the users' data on that site. Due to anonymity of online users and statelessness of the HTTP protocol, tracking on the server side as the traditional Web log files is not only necessary but indispensable to keep session data, and thus help identify unique session or unique user. An HTTP proxy and several Java servlets are used to implement the server-side tracking.

However, simply doing tracking on the server side alone does not offer much relevant data. The HTTP protocol does not send any data back to the server after a page is rendered, unless the user makes another explicit request. That means, just by server-side tracking, a lot of data are missing while the user is surfing on a page. To retrieve those data, tracking on the client-side makes much sense. To achieve this goal and also make this tool distinctive from most tracking tools which track users on the server-side, client-side tracking technology is used to obtain user data more directly and naturally. JavaScript, and HTML image technologies are used for the client-side tracking task.

Single-pixel image is used as one major part of client-side tracking in this tool. The tracking program can be triggered by any browser (with image downloading function) running on almost all kinds of operating systems on the client side. It runs when the page is downloaded. The other tracking program sits on the server-side. The user data is sent to the server-side servlet once the program starts. Therefore, the user has no knowledge of this program running at the background.

4.2 Major Challenges of Client-side Tracking and Solutions

Diversities of browsers and operating systems raise challenges to the development of client-side tracking: the issue of incompatibility of different products. The inconsistency mainly involves using JavaScript on the client side.

4.2.1 Major Challenges and Solutions of Using JavaScript

Issues involved with using JavaScript include a browser does not support JavaScript, or a user manually disables it. The incompatibility issues for using JavaScript exist between different types of browsers, different versions of JavaScript, and different operating systems.

One well-known example of incompatibility issue of JavaScript is the Microsoft Internet Explorer (MSIE) and Netscape Navigator (NS). They both support JavaScript, but in different ways and to a different degree. MSIE browsers, especially those later versions, fully support JavaScript, though sometimes, they display a different behavior than their main competitor, NS. In addition, MSIE supports VBscript and Jscript as well, while NS does not. Different versions of the same browsers also have incompatibility issues too. For example, the NS 4.x shows very different behaviors than later versions like NS 6.x when running same pieces of JavaScript code.

JavaScript is originally from Netscape to enhance Web pages and servers performance. As the most widely used scripting language, JavaScript functions can be called, and often executed by keyboard strokes, mouse functions, buttons, or other actions from the user. Through JavaScript's fully control of a user's Web browsers, the user's behavior on the page can be captured. JavaScript is used to track a user's keyboard and mouse movements on a page.

A browser does not support JavaScript if it is in one of the following cases: the user explicitly disables JavaScript; the browser is some special types like Lynx (a multi-platform text browser) or Amaya (an editor with browsing capabilities); or that the browser is a very old one, such as

MSIE 3.0 or earlier, NS 2.0 or earlier, Opera 2.1 or earlier. But according to the latest statistics of browser usage made by counter.com, [74] in May, 2003, 94% Internet users used MSIE with 59% of version 6.x, 34% of version 5.x, and 1% of version 4.x; while NS occupied 3% with 2% of version 5.x and 1% of version 4.x. Another statistics from Zeitgeist [75] based on users of Google in September 2003 shows the trend of the users who use the MSIE 6.x is rapidly increasing.

Although a user can disable JavaScript explicitly in the browser, statistics still shows that 86% [74] are using JavaScript version 1.2 or above, while 13% browsers are JavaScript non-supported or disabled. The operating system is another important factor affecting JavaScript running (e.g., some JavaScript codes do not run on Mac). But 93% [74], or 91% [75] of Internet users are known to use MS windows operating systems, while Mac occupies 2% to 3%.

From the above facts and analysis, we can conclude that most Internet users are using or will use browsers and operating systems, which support JavaScript. This conclusion forms the basis of designing the client-side tracking tool using client-side JavaScript. The JavaScript tracking tool mainly targets those users using MSIE 4.x or later, with MS windows operating systems, and JavaScript enabled.

4.2.2 Major Challenges and Solutions of Using Java Applets

A Java applet is a small application program written in Java language, which can be embedded in an HTML document and run on the World Wide Web by the user's browser. Client-side JavaScript cannot access to a database server directly. To transfer the tracked data to a server side program or a database, we need a program to talk to the server. Since java applets can run on the client-side too, and they easily talk to a server program written in Java, therefore, Java applets are a good bridge connecting the client-side and the server-side.

One major problem with using Java applets on the Internet is that the user can disable Java functions in the browser. If this occurs, it will cut off the communication between the client-side and server-side programs. Fortunately, 84% of Internet users' browsers are Java-enabled, while Java-unknown 14%, and 0% browsers are Java-disabled [74]. This means that except for some users using browsers with Java non-supported, few users purposely disable the Java function in their browsers. Recall most Internet users are using IE 5.x and IE 6.x browsers, both support Java functions. This high number of IE browser users offers a basis to conclude that most Internet users use a Java supported browser. These users should have no problem running applets in their browsers.

4.2.3 Challenges of Sending JavaScript Data to Server-side

Everyone hates being disturbed by pop-up windows when surfing a site. We do not wish our site users to complain. That is a reason to make this tracking tool to run at the background without the user's knowledge. This approach raises another issue: we cannot explicitly send the tracked data just like sending a form on a page back to the server-side (i.e., a database or server-side program). Instead, we need an implicit method.

Tracking mouse and keyboard movements results in large amounts of tracked data on each page. Also, some data may be the user's personal data. The amount of the data and the security of sending the data across the Internet limit that we cannot use the "GET" method to append these data to an URL to send them to the server programs. Although cookies are another method frequently used to save client-side data, its limitations as addressed in previous chapters limit their use for this purpose. HTML hidden form looks like an optimal measure to save and send user data, but the problem is that one cannot send a form secretly. What really happens is: when sending the form automatically, the user's next URL will be always automatically redirected to the form action page (i.e., the hidden server-side program). This method not only interrupts the user's navigation, but also causes the user to notice the running program behind the scene. Java applet can be used to resolve this issue for its flexibility on both the client-side and server-side.

4.3 Implementation Environment

This tracking and analysis tool tracks users on e-commerce sites, which specify the Web environment. In the development and testing of this tool, a mini-Web environment has been developed. This mini-Web environment includes the following components: local host, a Web server, a database server, a Web application server, Web clients, and a testing Web site.

- **Local Host**

To make the development and testing of this tool simple and easy to control, all programs are running on the local host. Table 4-1 shows the major components of this local host.

Component	Specification
Operation System 1	Microsoft Windows XP Professional, Version 2002, service pack 1
Operation System 2	Microsoft Windows 2000 Professional
CPU	Celeron® 1.80GHz
RAM	256 MB
Internet Access	Cable modem

Table 4-1: Local host Information

- **Web Server**

The Web server hosts all Web pages for the testing site. These pages include both static and dynamic pages. The Web server handles all HTTP requests and responses.

Component	Specification
Web Server	Tomcat (version 4.1.27) built-in Web server
Host	24.76.6.241
Port	8080
Running Environment	Standalone

Table 4-2: Components of the Web server

- **Web Application Server**

The Web application server is responsible for hosting all server-side application programs, i.e., all servlets and the HTTP proxy server. Tracking servlets are automatically triggered by client-side programs, such as java applets and single-pixel image. Analyzing servlet is triggered by the user of this tool. Besides, all communications with database are achieved through this application server. Details about this server are shown in Table 4-3.

Component		Specification
Web Application Server		Tomcat (version 4.1.24) built-in server
Host		24.76.6.241
Port		8080
Running Environment		Standalone
Implemented Servlet Specification		2.3
Hosting Servlets	JsServlet	Process Data from JavaScript Code
	ImgServlet	Process Data from Single-pixel Image
	ProxyServlet	Proxy Server
	DataAnalysis	Analyze Tracked Data

Table 4-3: Web Application Server

- **Database Server**

Although all programs run on the local host, all tracked data are sent by those Web application programs (i.e., Java servlet programs) to a database, which resides on the database server. This process simulates the case in the real world, where the database could possibly reside somewhere on the Internet. Table 4-4 lists the components of the database server.

Component	Specification
Database Server Software	Microsoft SQL Server 2000
Running Host	24.76.6.241
Driver for Java Applications	Microsoft SQL Server 2000 Driver for JDBC

Table 4-4: Database Server

- **Web Clients**

All Internet users use their browsers to access the Web. To a Web server, any browser is a Web client. Table 4-5 shows two Web clients used in testing this tool.

Component	Specification
Client 1 / host	Microsoft Internet Explorer 6.0 / 24.76.6.241
Client 2 / host	Netscape Navigator 7.0 / 24.76.6.241

Table 4-5: Web Clients

- **Testing Web Site**

To simulate online users' behavior, a simple testing Web site is built. This site has nine pages, which include six static HTML pages and three dynamic Web pages. The six static Web pages are normal HTML pages, made up of some HTML elements, such as links, images, or forms. Three dynamic Web pages are created by the Java servlet programs. The details of the testing site are shown in Table 4-6.

Page / File ***	Type*	Link	Image	Form
P1 / test1.html**	S	P2, P3, P4, P5, P6, P8		
P2 / test2.html	S	P1, P3, P5	G1.gif	
P3 / test3.html	S	P1, P2, P6	G2.gif, G3.gif	
P4 / test4.html	S	P1, P6	G4.gif	F1 (sends data to P7)
P5 / test5.html	S	P1, P3, P8		
P6 / test6.html	S	P1, P2	G5.gif, G6.gif	F2 (sends data to P7) F3 (sends data to P9)
P7 / FormServlet	D	P1, P5		
P8 /Page8Servlet	D	P1, P3, P5	G7.gif	
P9/TFormServlet	D	P1, P2		
Note:	*S—Static; D—Dynamic. **Page 1 is index page of this testing site. ***P1~ P9 denote Page1 ~ Page 9, respectively.			

Table 4-6: Components of the Testing Site

4.4 Implementation Strategies

Implementation strategies include how to build the Web environment and mini Web site, how to implement different tracking methods such as single-pixel image tracking, JavaScript tracking, HTTP proxy server tracking, and data analysis.

4.4.1 Strategies for Development and Test Environment

This tool tracks online users' activities on a given site. Therefore, a Web environment is needed to implement this tool.

- **Constructing the Web Environment on the Local Machine**

As presented above, the built-in Web server of Tomcat 4.1.27 is used here. Tomcat is an open source project from Apache Foundation [76]. Although considered as an application server, it can be run as a standalone Web server. Tomcat does not need much setup, but the host should be connected to on the Internet. The local host is both a Web server and a Web client after Tomcat starts. As a Web server, all requests and responses to and from this Web server start with http://localhost:port. The default port is 8080, which can be changed by modifying the file "server.xml". Also, like other Web servers, Tomcat specifies different directories for different files of the Web site.

- **Constructing the Mini -Web Site**

To simulate a real Web environment, all pages of the testing Web site are put in the special directories as required by the Tomcat Web server. To make things simple, a directory called "hk" is created to hold all files. Under "hk", the six static html pages are put in the server-required directory "servlets", where all public HTML files are stored. The three dynamic pages are stored in the server-required directory "classes", a place to keep all servlet classes. Before we can start the Web server and send requests, we must register all those servlets in the file "web.xml" as required by the Web server.

We can start the Tomcat server in a DOS window to run it as a standalone server. After it starts successfully, we can browse all pages on this mini-Web site.

4.4.2 Strategies for Implementing Tracking

Strategies for implementation mainly include how to track the data and save the data effectively.

4.4.2.1 JavaScript Tracking

Strategies for implementing JavaScript tracking involve how to insert JavaScript code into a current page, which include both static and dynamic page; what events of a user should be captured, and how to send data back to a database.

- **Use External JavaScript Files**

JavaScript is an interpreted event-driven language, it can be put anywhere on an HTML page. Usually, people put the code in the HTML <head> section. In this tool, JavaScript code is mainly used to track users' mouse and keyboard movements, and users' input as well. The entire code to achieve these goals is several hundreds of lines in length. If we put them in <head> section of each page as usual, it would result in the source code of each page to be too long and hard to read and maintain. Instead, external JavaScript files are used here.

Based on the different functions, the JavaScript code is separated into three JS files. For example, one is for tracking fixed data such as browser's type, operation systems; one is for tracking varying data, such as mouse movements and form input on the page; the other is for saving and sending those tracked data to server-side. We put these three JS files in the same directory as those HTML pages on which these files are added.

For static HTML pages, simply add these three files in the <head> section. Figure 4.1 shows the method to achieve this goal.

For dynamic pages, if the dynamic page is a response of the Web server to the user, we add those three JS files in the <head> section of the response. After the response gets at the client side, the JavaScript works the same way as in the static pages. As for those dynamic pages without any response to the user, we do not need to track anything.

Once those three JS files are added to each page, a few lines of JavaScript code added in the <body> section of the page call relative functions in the JS files based on the user's action. In this way, the user's behavior is captured and sent to the server-side program. Figure 4.1 shows how to call functions defined in the external JavaScript files.

```
<HTML>
<HEAD>
<SCRIPT SRC="BasicTracking.js"></SCRIPT>
<SCRIPT SRC="EventTracking.js"></SCRIPT>
<SCRIPT SRC="FormTracking.js"></SCRIPT>
<IMG SRC = "imgTrackingServlet" Width = 1 height = 1></IMG>
</HEAD>
<BODY>
This is an example for JavaScript Tracking
<CENTER><IMG SRC="p1.gif"name ="image1" width=150></CENTER>
<FORM NAME ='f1' METHOD = "post" >
<BR><TEXTAREA   NAME="comments" ROWS=6
 COLS=55></TEXTAREA>
<input type="submit"></input>
</FORM>
<SCRIPT>
        callTrackingMethods( );
        sendData( );
</SCRIPT>
</BODY>
```

Figure 4.1: Insert JavaScript Code into a Static Web page

- **Capture Only Meaningful Events of a User**

A user can have many keyboard and mouse movements on a page. If we track all these movements, the data will be huge. For example, if we select to track key-press event, when a user types a resume in a text-area, each key movement can be tracked. If the resume has 3000 characters, we will have 3000 records. To transfer so much data to server is a huge task if we have many users at the same time. As a matter of fact, this tool can capture almost all of a user's movements, though some movements, such as moving a mouse over a form, are of no importance for us to do analysis. As a result, non-important events are ignored. The most important user events of interest (for this thesis) are: mouse move-over and click for links and images; field change event, mouse move-out, and keyboard-pressed event for form input box; field change event, mouse move-out for other form input such as text-area, selection, etc.

- **Handle Java Applet in JavaScript**

Sending much data from client-side JavaScript to a server program or database is not that easy as sending data from an application program to a database sitting on the same machine. Fortunately, JavaScript can call public methods of Java applets directly, though the reverse process is impossible without some extra software package. To implement using Java applets to send data to the server from the client-side, on each page, the JavaScript code calls Java applets by sending data to them when any registered event of the user occurs.

- **Use Java Applet / Servlet Pair to Ease Sending Data**

Java applets act as bridges between the client-side and server-side programs. When the applet on a page gets its parameters, which, in fact, are tracked data, sent by the JavaScript code, the applet calls a server-side program and sends those data to that servlet program. Both written in Java, Java applets can easily talk to servlet programs

- **Use Servlet to Connect to Database**

Java servlets have many advantages over CGI programs, especially when used as an application program connecting to a database. A servlet can maintain open JDBC database connections. When a new request arrives, instead of making a new connection for each request, a servlet can re-use an existing database connection. This persistent connection avoids 1-2 seconds of latency by opening a new connection and thus enhances the performance of data transfer [76].

4.4.2.2 Strategies for Implementing Single-pixel Image Tracking

Single-pixel image tracking fits all browsers with or without JavaScript enabled. Strategies for implementing this technology include where to put such an image, and how to put an image on dynamic pages, and how to prevent the image being cached.

- **Put the Image in HTML <head> Section**

Most generally, images are put in the <body> of a HTML page. When the body of a page is downloaded, images in the body are downloaded. We follow this format to put the single-pixel image at any position in the body of the tracked page. As in the JavaScript tracking by putting the external JavaScript files in the <head> section to ease the management of both the tracking code and the HTML page, the single-pixel image is put in <head> section of the HTML page, as shown in Figure 4.1. Since the image is a 1-by-1 pixel size, and the code in <head> section is executed earlier than that in <body> section, these two factors allow the tracking image to download before the body of the Web page is finished.

- **Insert the Image into Dynamic Pages**

To put the single-pixel image on a dynamic page is to embed the single-pixel image into the response page, which is created by the Web server. When the browser renders the response page and downloads the single-pixel image, the tracking program is triggered. Embedding a single-pixel image is similar to put JavaScript code on dynamic pages.

- **Prevent Image from being Cached**

The single-pixel image is in fact a servlet program, not a real picture, which is updated every time it runs. Also, this servlet is designed as no-cache by setting values of these headers (the expires header, the cache control header, and HTTP-EQUIV META Tags) of the response page. For example, we set the expire date as a past date so that a client machine always has an outdated cached page and needs to request the page from the server every time. In addition, to ensure no-cache to work with different versions and brands of most current browsers, the above three headers are set. As a result, unlike usual images on the Web, this servlet is always requested from the server instead of from the cache.

4.4.2.3 Implement HTTP Proxy Server Tracking

HTTP proxy server tracking serves as a traditional Web log files tracking. Strategies for implementing this involve how to setup listening address (i.e., the Web server's address) and port in order that it can work correctly; and how to send captured data to the database.

- **Setup Listening Address and Port**

This proxy listens to the Tomcat built-in Web Server. The address is local host. Since the Web server listens on port 8080, that means when a user sends a request to port 8080, this request should go to the proxy first, then the proxy redirect this request to the real port of the Web server. In order to achieve this, run the HTTP proxy server and configure it to listen on port 80, which is standard port for HTTP requests and responses.

After running the HTTP proxy server on port 80, all requests sent to port 80 are redirected to the Web server‘s port 8080. For example, when a user sends a normal HTTP request to request page1 of the testing Web site, the user should go to: http://localhost/hk/servlets/test1.html. Since the port number of this request is omitted, therefore, the request goes to the default standard port for HTTP request, i.e., 80, where the HTTP proxy server is listening on. After getting this request, the HTTP proxy server redirects the request to the Web server, which is listening on port 8080. Thus, it seems the HTTP proxy server itself sends the following request: http://localhost:8080/hk/servlets/test1.html. This request goes to the Web server's 8080 port. After receiving this request, the Web server sends a response back to the HTTP proxy server, and similarly, the HTTP proxy server redirect the response back to the client. Every time a request or response transferred by the HTTP proxy server, the HTTP proxy server logs it and sends it to the database.

4.4.2.4 Use Database Connection Pooling

Data tracked from all three sources, JavaScript, Single-pixel image, and HTTP proxy server are sent to a database. However, opening a connection to database is a time-consuming process. Sometimes, it takes much longer to open a connection than to perform a real database operation. Therefore, reusing a connection object makes much sense. To make this work, connection-pooling technology is used. When a database is initially connected, some database connections are pre-allocated, when an operation finishes with a connection, this connection is recycled instead of being closed as normal.

Database connection pooling can enhance connection performance. In this design, each of the two servlet programs and the HTTP proxy server has a separate connection pool.

4.4.2.5 Handle Multiple-Threads Environment

The Web environment is a multiple–user environment. This environment requires the tracking tool to be able to handle multiple users at the same time. There is no thread safety problem with the client side JavaScript and single-pixel image tracking, since each user gets a separate copy of those files from the Web server, which can handle multiple users. Servlets are intrinsically multithreaded, which means that a single instance can be accessed by more than one thread. This allows those two servlet programs, which send tracked data to the database, to handle multiple users. Also, the HTTP proxy server is explicitly designed to be multi-threaded.

4.4.2.6 Session Control

All the tracked data are for analysis purpose. Session identification is a critical step for advanced analyses. For a popular site, several hundreds or thousands of the users may be online concurrently. The data about each of these users are tracked and written into the database almost concurrently. Therefore, without any session control, these data are totally unable to be unidentifiable and of less use.

To solve this session control problem, the following strategies are adopted:

- Like most sites, we explicitly request a user to turn on cookie function to be able to access the Web site.
- JavaScript tracking program uses cookie on the client-side, while its server-side servlet program creates session for each user. When a user comes, the JavaScript code creates a cookie, and sends this cookie to the servlet program. On the server-side, the servlet checks if there is an existing session ID, and creates one for that client if none exists. This session ID is sent back in the response to the client-side as another cookie if the user's browser is cookie enabled. Since we use session control in the tracking servlet, even if the user turns off the cookie function of the browser, the user can be still tracked by JavaScript code, and the tracked data sent to the database are still associated with a session ID which is created on the server-side.

- Single-pixel tracking program simply uses server-side sessions to control users. Since these servlet programs run on the same domain, they can share the same session ID. That means, if any of these servlet programs finds out a session ID exists for a user, that session ID is used and no new session ID will be created. Also, for later analysis, we create another cookie in this approach.
- The HTTP proxy server tracking tool can log all session data from the client-side. These session data include those cookie and session IDs created by the other tracking programs.

4.4.3 Strategies for Data Analysis

Strategies for data analysis mainly include how to identify user and sessions, do path analysis and clustering analysis.

- **User Identification**

In almost every e-commerce site, a user usually completes some forms. Since the JavaScript code on each page can capture the form data, this offer a way to identify the user. For those users without any form data, cookie is another way to identify users, although it is not very effective, as users can delete cookies manually. Although an IP address and operation environment can be considered as an alternative way to identify a user, they lack accuracy. We do not use this approach. In this thesis, we identify users by session ID created by the serve-side tracking programs and userId tracked by JavaScript approach. Only in the following two cases, any two records are regarded as from the same user: the two records with the same user ID; the two records have the same session ID but two different user IDs, they are regarded as from the same user with two user IDs.

- **Session Identification**

For a returning user that has more than one session, the tracked time on the server side is used to identify a session. We based the session identification on the assumption: all session IDs are unique. In this way, a session is easy to identify.

- **Path Analysis**

Path analysis is based on session data. Although we have session data from each tracking approach, single-pixel image tracks users by page view, that means, one page leads to one data entry in the database. This would greatly simplify the data preparation process.

- **Cluster Analysis**

Implementation of cluster analysis, in this thesis, is based on the following assumption: If a user is interested in some pages, in a later visit, the user would visit these pages again. This assumption allows us to consider only the session with most unique pages visited by the user. In other words, a user may have more than one session data, but we only consider one session for this user. In that session, the user may visit more unique pages than in other sessions.

4.5 Deployment Diagram

Deployment of this tool follows a three-tier architecture, as shown in Figure 4.2. "The three-tier architecture is a versatile and modular infrastructure intended to improve usability, flexibility, interoperability and scalability" [77]. In this architecture, the first tier, known as presentation tier, mainly the Web browser, includes client-side control such as user input validation; the second tier, known as the application server, provides the business processes and data operations; the third tier stores the data.

Some advantages of a three-tier architecture include [77, 78]: (1) modifying or replacing any tier component is easy without affecting the other tiers, (2) better load balancing because of separation of the application and database functionality; (3) more security policies can be enforced on the server tiers because of the separation of the client and server.

4.6 Quality Assurance

Quality assurance deals with the overall management of quality, while quality control identifies defects and corrects them so that a defect-free product is produced [79]. Although such a difference exists [80], the following section focuses on more technical aspects, verification and validation process, an integral part of quality control than overall management. Notice that quality control is a subcomponent of the overall quality assurance task in software development.

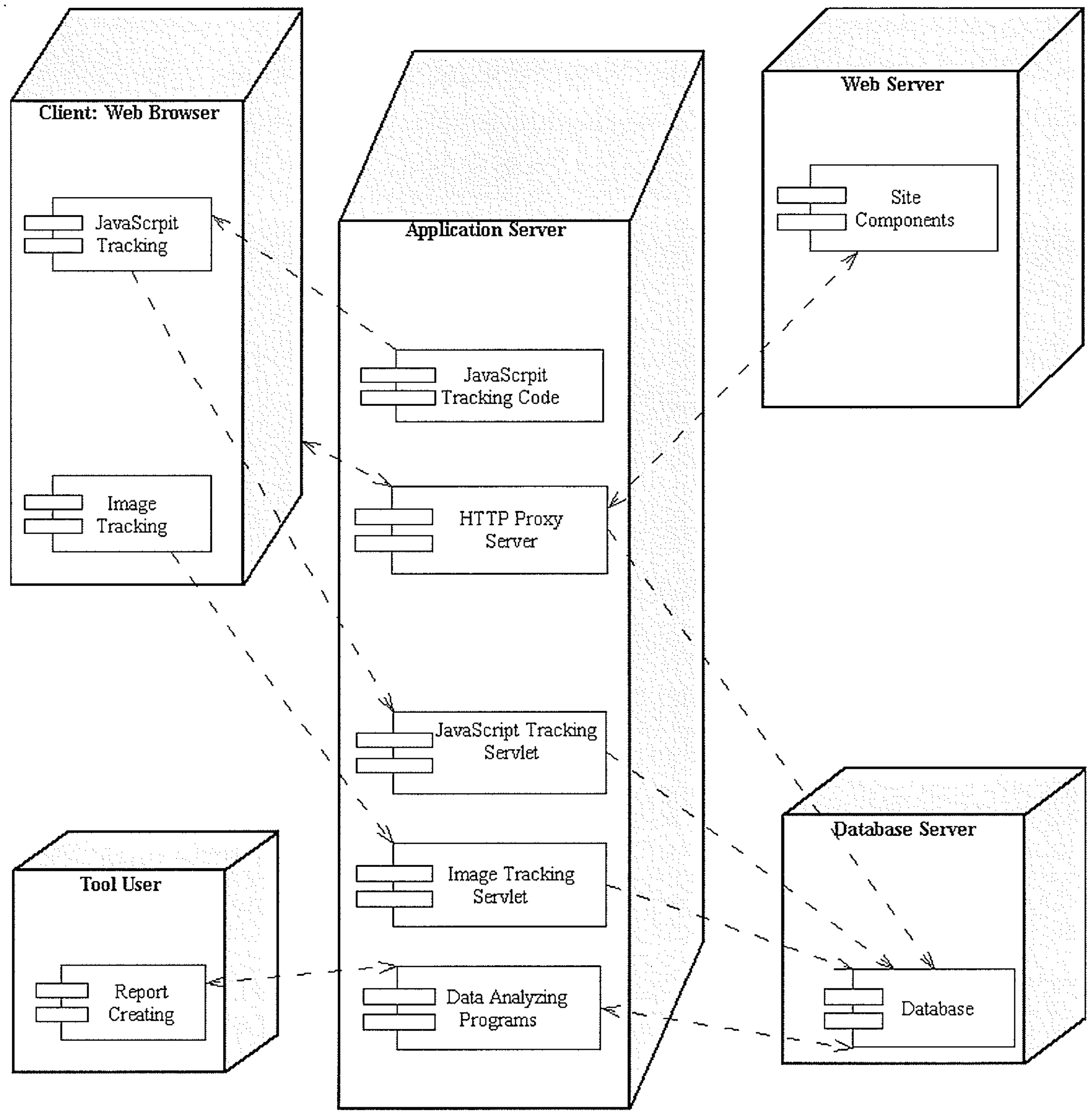


Figure 4.2: Deployment Diagram

4.6.1 Formal Specification

The formal specification presented in this thesis does not use any particular specification languages; rather, mathematical theory, especially discrete mathematical theory, and notations are used. Set theoretical notations are used to described structural components, while predicate logic is used to describe requirements.

- **Basic Data Types**

The following basic data types in table 4-7, together with some basic operations, have their corresponding data types in almost all programming languages. Therefore, instead of re-defining them, we use them as a basis for our specification.

Symbol			**Denotation**	**Basic Operation**
Z			Integer data type	
R			Real data type	
S			String data type	Comparison: s1==s2 ? Concatenation: s1=s2 +s3
B	F	False	Boolean data type	
	T	True		
N			NULL	

Table 4-7: Basis Data Types

- **Composite Types**

Simple data types, as in Table 4.7, are insufficient to describe the structure of this tool. More composite types are defined in the following table for this specification.

Sign		**Denotation**
sequence of X		A sequence of type X, which can be abstracted as an array
sizeof(s)		Current size of the sequence of s, always zero or a positive
structure of … end		To define a structure, like the structure in C language
TypeName = TypeExpression		To define a new type
VariableName: TypeName		To declare a variable
requirement	pre-condition	Must be true before any operation for this requirement is invoked
	post-condition	Must be true after any operation for this requirement is invoked

Table 4-8: Composite Types Definition Template

- **User Defined Types**

Cookie and Session are used to identify unique session and user. They both have IDs, names and values. Table 4-9 and Table 4-10 show the structure of a cookie and session, respectively.

```
Cookie = structure of
          cookieId : S
          cookieName : S
          cookieValue : S
        end
```

Table 4-9: User Defined Data Type: Cookie

```
Session = structure of
          sessionId : S
          sessionName : S
          sessionValue : S
        end
```

Table 4-10: User Defined Data Type: Session

Also, to facilitate data analysis later, each page is assigned a unique ID. The ID is a string, which consists of a character “P” and an integer, for example, a page http://localhost/hk/servlet/index.html can be denoted as “P1”.

PageId : S

Table 4-11: User Defined Data Type: PageId

- **Global Variables**

The following global variables are defined, see Table 4-12. JS_TABLE, IMG_TABLE, PROXY_TABLE USER_TABLE, PAGE_TABLE, and PATH_TABLE are tables in the database. The first three tables keep tracked data from the three tracking programs: JavaScript code, single-pixel image, and HTTP proxy separately. The rest are for analysis purpose. The right column variables in the table are maximum size for each table. In addition, JsDataEntry, ImgDataEntry, ProxyDataEntry, UserEntry, Page, PathEntry, and ClusterEntry are defined later in the specification as required,

JS_TABLE : sequence of JsDataEntry	MAX_JS_ENTRIES : Z
IMG_TABLE : sequence of ImgDataEntry	MAX_IMG_ENTRIES : Z
PROXY_TABLE : sequence of ProxyDataEntry	MAX_PROXY_ENTRIES : Z
USER_TABLE: sequence of UserEntry	MAX_USER_ENTRIES : Z
PAGE_TABLE : sequence of Page	MAX_PAGE_ENTRIES : Z
PATH_TABLE : sequence of PathEntry	MAX_PATH_ENTRIES : Z
CLUSTER_TABLE : sequence of ClusterEntry	MAX_CLUSTER_ENTRIES : Z

Table 4-12: Global Variables

- **Specification Statement**

Predicate logic is used to describe pre- and post-conditions for any requirement as shown in Table 4-8. A simple predicate, which usually has one or more arguments, is of the form P(x), where x is an argument that is used in the predicate P. A general form of a quantified statement can be written in one of the following forms shown in Table 4-13:

Statement 1:	<quantifier><declaration(s)>●<predicate>
Statement 2:	<quantifier><declaration(s)>|<constraint>●<predicate>

Table 4-13: Forms of Specification Statements

Two frequently used kinds of quantifiers are universal ($\forall$) and existential ($\exists$). To create compound predicates, statements can be nested and combined together using one or more logical connectives, such as $\wedge$, $\vee$, $\neg$, $\Rightarrow$, and $\Leftrightarrow$. Truth tables for the logical connectives are available in [11, 81]. The formal specification of a requirement in this thesis follows this format:

<name><input parameter (s)><output parameter (s)>
pre-condition
post-condition

Table 4-14: Format Used in This Specification

Both the pre-conditions and post-conditions are given as predicates. In any operation, we use p' to indicate the variable p has been changed after that operation

4.6.1.1 Specification of Traffic Tracking

All user-tracking processes can be regarded as writing entries into a database table. Any database table can be abstracted as an array of entries. Since each tracking method captures different kinds of data about a user almost concurrently, therefore, entries and tables are different. As a result, they should be defined separately based on each tracking method.

4.6.1.1.1 JavaScript Tracking

JavaScript tracking focuses on tracking a user's operation environment, mouse and keyboard movements, and form inputs. Also, session and cookie are captured if available.

- **Data Types Defined for JavaScript Tracking**

Variable JsFixedData (see Table 4-15) keeps a user's all non-varying data in the session. Variable JsVaryData (see Table 4-17) keeps a user's real time data, such as keyboard and mouse movements or form inputs. JsVaryData is made up of two string arrays: jsForm and jsEvent. While jsForm keeps all a users input on a page, JsEvent keeps a user's current events on a page. Both JsForm and JsEvent are string data types.

```
JsFixedData = structure of
              jsJavaScriptEnabled : B
              jsJavaEnabled : B
              jsCookieEnabled : B
              jsCookies : JCookie
              jsBrowserType : S
              jsScreenSize : S
              jsClientTimeIn : S
          end
```

Table 4-15: User Defined Data Type: JsFixedData

JCookie in a JsFixedData object is made up of two parts, as shown in Table 4-16.

```
JCookie = structure of
              jsCookie : Cookie
              jsOtherCookie : S
          end
```

Table 4-16: User Defined Data Type: JCookie

```
JsVaryData = structure of
              jsEvent : JsEvent
              jsForm : S
          end
```

Table 4-17: User Defined Data Type: JsVaryData

The information in JsEvent include the source and the type of the event. Besides, all fields of a user's inputs can be put into a big string. Table 4-18 shows the structure of JsEvent.

```
JsEvent = structure of
              srcOfEvent : S
              typeOfEvent : S
          end
```

Table 4-18: User Defined Data Type: JsEvent

```
JsDataEntry = structure of
                jsFixedData : JsFixedData
                jsVaryData : JsVaryData
                jsServerTime : S
                jsSession : Session
                jsIpAddr : S
          end
```

Table 4-19: User Defined Data Type: JsDataEntry

JsDataEntry consists of all data tracked from JavaScript code. That includes two parts, JsFixedData and JsVaryData. Also, it records when this tracking program starts and the IP address of the client. The data is sent to JS_TABLE table in the database.

- **Requirements Specification**

In the following section, each requirement for JavaScript tracking is specified using the pre-defined format.

Name		initializeFixedData
input parameter (s)		None
output parameter (s)		jsFixedData(i)
pre-condition	description	None
	specification	None
post-condition	description	Ensure all variables in jsFixedData(i) are initialized.
	specification	$\forall$ i : Z | i ≥1 • jsFixedData(i).jsJavaScriptEnabled = F $\wedge$ jsFixedData(i).jsJavaEnabled = F $\wedge$ jsFixedData(i).jsCookieEnabled = F $\wedge$ jsFixedData(i).jsCookies = N $\wedge$ jsFixedData(i).jsBrowserType = N $\wedge$ jsFixedData(i).jsScreenSize = N $\wedge$ jsFixedData(i).jsClientTimeIn = N

Table 4-20: Initialize Fixed Data Variables

name		trackFixedData
input parameter (s)		None
output parameter (s)		jsFixedData(i)′
pre-condition	description	None
	specification	None
post-condition	description	A JsFixedData object jsFixedData(i)′, whose values are not null, exists.
	specification	$\forall$ i : Z | i ≥1 • jsFixedData(i)′.jsJavaScriptEnabled ≠ N ∧ jsFixedData(i)′.jsJavaEnabled ≠ N ∧ jsFixedData(i)′.jsCookieEnabled ≠ N ∧ (jsFixedData(i)′.jsCookies == N ∨ jsFixedData(i)′.jsCookies ≠ N) ∧ jsFixedData(i)′.jsBrowserType ≠ N ∧ jsFixedData(i)′.jsScreenSize ≠ N ∧ jsFixedData(i)′.jsClientTimeIn ≠ N

Table 4-21: Tracking Fixed Data

name		trackVaryData
input parameter (s)		None
output parameter (s)		jsVaryData(i)
pre-condition	description	jsFixedData(i).jsJavaScriptEnabled is not false.
	specification	∀i : Z | i ≥1 • jsFixData(i).jsJavaScriptEnabled ≠ F
post-condition	description	Create a JsEvent object jsEvent(i); Create a jsVaryData object jsEvent(i).
	specification	∃i : Z | i ≥1 • jsEvent(i) ≠ N ⇔ jsEvent(i).srcOfEvent ≠ N ∧ jsEvent.typeOfEvent ≠ N ∃i : Z | i ≥1 • jsVaryData(i) ≠ N ⇔ (jsVaryData(i) .jsEvent = jsEvent(i) ∧ (jsVaryData(i).jsForm ≠ N ∨ jsVaryData(i).jsForm == N))

Table 4-22: Track Vary Data

name		trackJsDataEntry
input parameter (s)		jsFixedData(i) and jsVaryData(i)
output parameter (s)		jsDataEntry(i)
pre-condition	description	If the fixed data does not exist, nothing will be tracked.
	specification	$\exists$ i : Z | i $\geq$1 • jsFixedData(i) $\neq$ N
post-condition	description	Two situations exist: vary data exist or not.
	specification	$\exists$ i : Z | i $\geq$1 • (jsVaryData(i) $\neq$ N $\Rightarrow$ jsDataEntry(i). jsFixedData = jsFixedData(i) $\wedge$ jsDataEntry(i).jsVaryData = jsVaryData(i)) $\vee$ (jsVaryData(i) == N $\Rightarrow$ jsDataEntry(i).jsFixedData = jsFixedData(i) $\wedge$ jsDataEntry(i).jsVaryData = N) $\wedge$ jsDataEntry(i).jsServerTime $\neq$ N $\wedge$ jsDataEntry(i).jsIpAddr $\neq$ N$\wedge$ jsDataEntry(i).jsSession.jsSessionId $\neq$ N

Table 4-23: Track JS DataEntry

Name		createCookie
input parameter (s)		jsFixedData(i).jsCookieEnabled jsFixedData(i).jsJavaScriptEnabled jsVaryData(i).jsCookies
output parameter (s)		jsDataEntry(i)
pre-condition	description	The browser is cookie supported, and a cookie created by JavaScript code is not yet available.
	specification	$\forall$ i : Z | i $\geq$1 • jsFixedData(i).jsCookieEnabled == T $\wedge$ jsFixedData(i).jsJavaScriptEnabled == T $\wedge$ jsVaryData(i).jsCookies.jsCookie == N
post-condition	description	A cookie is created.
	specification	$\exists$ i : Z | i $\geq$1 • jsDataEntry(i).jsVaryData.jsCookies.jsCookie $\neq$ N

Table 4-24: Create a Cookie

Name		checkJsSession
input parameter (s)		None
output parameter (s)		jsSession
pre-condition	description	None
	specification	None
post-condition	description	jsSession(i).sessionId $\neq$ N.
	specification	$\exists$ i : Z | i $\geq$1 $\bullet$ jsDataEntry(i).jsSession.sessionId $\neq$ N

Table 4-25: Check the JsSession

Name		jsAddEntry
input parameter (s)		jsDataEntry(i), JS_TABLE
output parameter (s)		JS_TABLE
pre-condition	description	JS_TABLE has no such entry; jsDataEntry(i) is not null.
	specification	$\exists$ i, j : Z | 1 $\leq$ i, 1 $\leq$ j $\leq$ sizeof(JS_TABLE) $\bullet$ jsDataEntry(i) $\neq$ N $\wedge$ $\neg$(JS_TABLE(j) == jsDataEntry(i))
post-condition	description	Add one entry into JS_TABLE.
	specification	sizeof(JS_TABLE′) = sizeof(JS_TABLE) + 1 $\wedge$ ($\exists$ i, j : Z | 1 $\leq$ i, 1 $\leq$ j $\leq$ sizeof (JS_TABLE′) $\bullet$ JS_TABLE′(j) == jsDataEntry(i))

Table 4-26: Add the Entry to JS_TABLE

4.6.1.1.2 Single-pixel Image Tracking

The single-pixel image is a trigger to the server-side tracking program. Since the tracking program runs on the server side, major data can be tracked using CGI variables. Besides, session data can be tracked separately.

.

- **User Defined Types**

An ICookie object is made up of two parts, shown in Table 4-27. Another data type, ImgCgiData, shown in Table 4-28, includes most CGI variables.

```
ICookie = structure of
        imgCookie : Cookie
        imgOtherCookie : S
    end
```

Table 4-27: User Defined Data Type: Icookie

```
ImgCgiData = structure of
        imgAuth_type : S
        imgContent_length : S
        imgContent_type : S
        imgDocument_root : S
        imgPath_info : S
        imgPath_translated : S
        imgQuery_string : S
        imgRemote_addr : S
        imgRemote_host : S
        imgRemote_user : S
        imgRequest_method : S
        imgReferrerURL: S
        imgScript_name : S
        imgServer_name : S
        imgServer_port : S
        imgServer_protocol : S
        imgServer_software : S
    end
```

Table 4-28: User Defined Data Type: ImgCgiData

ImgDataEntry, shown in Figure 4-29, is defined to represent an entry of items tracked by single-pixel image. In addition to CGI variables, this data type also includes cookies, session data, and server-side time when the single-pixel image triggers the tracking program.

```
imgDataEntry = structure of
        imgSession : Session
        imgCookies : ICookie
        imgRequest_time : S
        imgCgiData : ImgCgiData
    end
```

Table 4-29: User Defined Data Type: ImgDataEntry

- **Requirements Specification**

Single-pixel image tracking process involves session control and tracking and saving of tracked data.

name		trackImgDataEntry
input parameter (s)		None
output parameter (s)		imgDataEntry(i)
pre-condition	description	None
	specification	None
post-condition	description	Ensure an ImgDataEntry object exists.
	specification	$\forall$ i : Z | i $\geq$1 $\bullet$ imgCgiData(i).imgAuth_type $\neq$ N $\wedge$ imgCgiData(i).imgContent_length $\neq$ N $\wedge$ imgCgiData(i).imgContent_type $\neq$ N $\wedge$ imgCgiData(i).imgDocument_root $\neq$ N $\wedge$ imgCgiData(i).imgPath_info $\neq$ N $\wedge$ imgCgiData(i).imgPath_translated $\neq$N $\wedge$ imgCgiData(i).imgQuery_string $\neq$ N $\wedge$ imgCgiData(i).imgRemote_addr $\neq$ N $\wedge$ imgCgiData(i).imgRemote_host $\neq$ N $\wedge$ imgCgiData(i).imgRemote_user $\neq$ N $\wedge$ imgCgiData(i).imgRequest_method $\neq$N $\wedge$ imgCgiData(i).imgReferrerURL$\neq$N $\wedge$ imgCgiData(i).imgScript_name $\neq$ N $\wedge$ imgCgiData(i).imgServer_name $\neq$ N $\wedge$ imgCgiData(i).imgServer_port $\neq$ N $\wedge$ imgCgiData(i).imgServer_protocol $\neq$ N $\wedge$ imgCgiData(i).imgServer_software $\neq$ N $\forall$ i : Z | i $\geq$1 $\bullet$ imgDataEntry(i).imgSession $\neq$ N $\wedge$ imgDataEntry(i).imgCookie $\neq$ N $\wedge$ imgDataEntry(i).imgRequest_time $\neq$ N $\wedge$ imgDataEntry(i).imgCgiData = imgCgiData (i)

Table 4-30: Tracking Data Using Image

name		checkImgSession
input parameter (s)		None
output parameter (s)		imgSession
pre-condition	description	None
	specification	None
post-condition	description	imgDataEntry(i).imgSession.sessionId $\neq$ N
	specification	$\exists$ i : Z | i $\geq$1 • imgDataEntry(i).imgSession.sessionId $\neq$ N

Table 4-31: Check the ImgSession

Name		imgAddEntry
input parameter (s)		imgDataEntry(i), IMG_TABLE
output parameter (s)		IMG_TABLE′
pre-condition	description	IMG_TABLE does not have such an entry; and imgDataEntry(i) is not null.
	specification	$\exists$ i, j : Z | 1 $\leq$ i, 1 $\leq$ j $\leq$ sizeof(IMG_TABLE) • imgDataEntry(i) $\neq$ N $\wedge$ $\neg$(IMG_TABLE(j) == imgDataEntry(i))
post-condition	description	Add one entry into IMG_TABLE.
	specification	sizeof(IMG_TABLE′) = sizeof(IMG_TABLE) + 1 $\wedge$ ($\exists$ i, j : Z | 1 $\leq$ i, 1 $\leq$ j $\leq$ sizeof(IMG_TABLE′) • IMG_TABLE′(j) == imgDataEntry(i))

Table 4-32: Add the Entry to Database IMG_TABLE

4.6.1.1.3 HTTP Proxy Server Tracking

HTTP proxy server tracks requests and response between the user and the Web server.

- **User Defined Types**

To represent the data tracked using this approach, the ProxyDataEntry data type is defined. Table 4-33 shows the structure of ProxyDataEntry record.

ProxyDataEntry = structure of proxyRequstMethod : S proxyProtocol : S proxyAcceptFiles_type : S proxyAcceptLanguages : S proxyAcceptEncoding : S proxyUserAgent : S proxyHost : S proxyResponseStatus : S proxyURL : S proxyReferer : S proxyTime : S proxyCookies : Cookie proxyServerType : S proxySession : Session end

Table 4-33: User Defined Data Type: ProxyDataEntry

- **Requirements Specification**

HTTP proxy server tracking process mainly involves requests from the client to the Web server. Also, session data are tracked by the proxy server.

Name		checkProxySession
input parameter (s)		None
output parameter (s)		proxySession
pre-condition	description	None
	specification	None
post-condition	description	proxySession
	specification	$\exists$ i : Z | i $\geq$1 • proxyDataEntry(i).proxySession.sessionId $\neq$ N

Table 4-34: Check the ProxySession

Name		trackProxyEntry
Input parameter (s)		None
output parameter (s)		proxyDataEntry(i)
pre-condition	description	None
	specification	None
post-condition	description	Ensure a proxyDataEntry object exists.
	specification	$\forall$ i : Z | i $\geq$1 $\bullet$ proxyDataEntry(i).proxyRequestMethod $\neq$ N $\wedge$ proxyDataEntry(i).proxyProtocol $\neq$ N $\wedge$ proxyDataEntry(i).proxyAcceptFiles $\neq$ N $\wedge$ proxyDataEntry(i).proxyAcceptEncoding $\neq$ N $\wedge$ proxyDataEntry(i).proxyUserAgent $\neq$ N $\wedge$ proxyDataEntry(i).proxyHost $\neq$ N $\wedge$ proxyDataEntry(i).proxyResponseStatus $\neq$ N $\wedge$ proxyDataEntry(i).proxyLocation $\neq$ N $\wedge$ (proxyDataEntry(i).proxyReferer == N $\vee$ proxyDataEntry(i).proxyReferer $\neq$ N) $\wedge$ proxyDataEntry(i).proxyTime $\wedge$ (proxyDataEntry(i). proxyCookies = N $\vee$ proxyDataEntry(i). proxyCookies $\neq$ N) $\wedge$ proxyDataEntry(i). proxyServerType $\neq$ N $\wedge$ proxyDataEntry(i).proxySession.sessionId $\neq$ N

Table 4-35: Tracking Data Using HTTP Proxy Server

Name		proxyAddEntry
Input parameter (s)		proxyDataEntry(i), PROXY_TABLE
output parameter (s)		PROXY_TABLE′
pre-condition	description	No such an entry in PROXY_TABLE; proxyDataEntry(i) exists.
	specification	$\exists$ i, j : Z | 1 $\leq$ i, 1 $\leq$ j $\leq$ sizeof(PROXY_TABLE) $\bullet$ proxyDataEntry(i) $\neq$ N $\wedge$ $\neg$(PROXY_TABLE(j) == proxyDataEntry(i))
post-condition	description	Add one entry into PROIXY_TABLE.
	specification	sizeof(PROXY_TABLE′) = sizeof(PROXY_TABLE) + 1 $\wedge$ ($\exists$ i, j : Z | 1 $\leq$ i, 1 $\leq$ J $\leq$ sizeof(PROXY_TABLE′) $\bullet$ PROXY_TABLE′(j) == proxyDataEntry(i))

Table 4-36: Add the Entry to Database PROXY_TABLE

4.6.1.2 Specification of Traffic Analysis

Traffic analyses in this thesis mainly include user identification, session identification, path analysis, and clustering analysis. All analyses are based on the tracked data saved in the database.

- **User Defined Types**

A user can be identified by a user ID, which is string, as defined in Table 4-37. An entry for a user consists of the following items, shown in Table 4-38.

```
User: S
```

Table 4-37: User Defined Data Types: User

```
UserEntry = structure of
        user : User
        userSessionId : S
    end
```

Table 4-38: User Defined Data Types: UserEntry

The analysis results of unique users are saved in the table USER_TABLE in the database, as presented in Table 4-12.

```
Page = structure of
        url : S
        pageId : PageId
    end
```

Table 4-39: User Defined Data Types: Page

In the path analysis, we need to map each URL to the page ID. To achieve this, the data type Page is defined. The structure of Page is shown in Table 4-39.

Another data type, SessionPage, which consists of a group of ordered pages, is defined as shown in Table 4-40. The session analysis result can be described as a data type, PathEntry, which consists of session data and all the pages for that session, i.e., a SessionPage object. Path analysis results are saved in the table PATH_TABLE, a sequence of PathEntry objects. Table 4-41 shows the structure of PathEntry object.

```
SessionPage : seq PageId
```

Table 4-40: User Defined Data Types: SessionPage

```
PathEntry = structure of
        session : Session
        sessionPage : SessionPage
    end
```

Table 4-41: User Defined Data Types: PathEntry

Clustering is a process to group similar users. A cluster ID is used to differentiate a group from others. In this data type, uniquePage is a special kind of SessionPage, since all pages in uniquePage are unique. Table 4-42 shows the structure of a ClusterEntry object.

```
ClusterEntry = structure of
        user : User
        clusterId : Z
        uniquePage : SessionPage
    end
```

Table 4-42: User Defined Data Types: ClusterEntry

Clustering analysis results are saved in the table CLUSTER_TABLE, which consists of a sequence of ClusterEntry objects.

- **Requirement Specification**

User identification is based on session data when available. Otherwise, more data entries should be combined. All data about the user are saved in a newly created table USER_TABLE. Table 4-43 specifies how to identify a user by cookies while Table 4-44 specifies user identification by username.

name		identifyUserByCookie
input parameter (s)		IMG_TABLE, USER_TABLE
output parameter (s)		USER_TABLE' or none
pre-condition	description	A same cookie Id exists for two records.
	specification	None
post-condition	description	One same username for those two records.
	specification	$(\forall i, j : Z \mid 1 \le i, j \le \text{sizeof(IMG\_TABLE)} \bullet$ $\text{IMG\_TABLE}(i).\text{imgCookie} \ne N \wedge$ $\text{IMG\_TABLE}(j).\text{imgCookie} \ne N \wedge$ $\text{IMG\_TABLE}(i).\text{imgCookie} ==$ $\text{IMG\_TABLE}(j).\text{imgCookie}) \wedge$ $(\neg(\exists k, g : Z \mid 1 \le k, g \le \text{sizeof(USER\_TABLE)} \bullet$ $\text{USER\_TABLE}(k).\text{user.username} ==$ $\text{USER\_TABLE}(g).\text{user.username}) \Rightarrow$ $(\text{sizeof(USER\_TABL}'\text{E)} = \text{sizeof(USER\_TABLE)} + 1 \wedge$ $(\exists r, h : Z \mid 1 \le r, h \le \text{sizeof(MAX\_USER\_TABLE)} \bullet$ $\text{USER\_TABLE}'(r).\text{user.username} ==$ $\text{USER\_TABLE}'(h).\text{user.username}) \wedge$ $\text{USER\_TABLE}'(r).\text{sessionId} ==$ $\text{IMG\_TABLE}(i).\text{imgSession.sessionId})$

Table 4-43: Identify User by Cookies

name		identifyUserByUsername
input parameter (s)		JS_TABLE
output parameter (s)		USER_TABLE′ or none.
pre-condition	description	If a user name has been tracked by JS code, and the table USER_TABLE does not have such a user.
	specification	$\forall i : Z \mid 1 \le i \le \text{sizeof(JS\_TABLE)} \bullet$ JS_TABLE(i). jsFormData.username $\neq N \wedge$ $\neg(\exists\, i : Z \mid 1 \le i \le \text{sizeof(USER\_TABLE)} \bullet$ USER_TABLE(i).user.username == JS_TABLE(i).jsFormData.username)
post-condition	description	Create an entry in USER_TABLE.
	specification	sizeof(USER_TABLE′) = sizeof(USER_TABLE) + 1 $\wedge$ $(\exists\, i : Z \mid 1 \le i \le \text{sizeof(MAX\_USER\_TABLE)} \bullet$ USER_TABLE′(i).user.username == JS_TABLE(i).jsFormData.username)

Table 4-44: Identify User by Username

- **Unique Session Identification**

Session identification uses the data from the three data source tables. A user may have more than one session. Since we track data by session, a unique session ID identifies a unique session. Table 4-45 specifies session identification.

Data for a user saved in separate forms are related by session ID. Table 4-46 specifies the association of the data in table JS_TABLE with the other two tables.

- **Path Analysis**

Path analysis is based on session data in the table IMG_TABLE, since single-pixel image tracks users based on page views. All pages viewed are ordered by time in a session and constitute an entry in table PATH_TABLE. Each entry in PATH_TABLE is a node in our path tree (described in Chapter 3). Table 4-47 specifies the creation of a path node.

name		identifySession
input parameter (s)		JS_TABLE, IMG_TABLE, PROXY_TABLE
output parameter (s)		None
pre-condition	description	None
	specification	None
post-condition	description	Two same session IDs mean a same session
	specification	$\forall i, j : Z \mid 1 \leq i, j \leq$ sizeof(JS_TABLE, IMG_TABLE, PROXY_TABLE) $\bullet$ (JS_TABLE(i).jSession.sessionId == JS_TABLE(j).jSession.sessionId $\Rightarrow$ JS_TABLE(i).jSession == JS_TABLE(j).jSession) $\wedge$ (IMG_TABLE(i).imgSession.sessionId == IMG_TABLE(j).imgSession.sessionId $\Rightarrow$ IMG_TABLE(i).imgSession == IMG_TABLE(j).imgSession) $\wedge$ (PROXY_TABLE(i).proxySession.sessionId == PROXY_TABLE(j).proxySession.sessionId $\Rightarrow$ PROXY_TABLE(i).proxySession == PROXY_TABLE(j).proxySession)

Table 4-45: Identify Session

name		associateData
input parameter (s)		Any entry in JS_TABLE, IMG_TABLE, PROXY_TABLE
output parameter (s)		Associate data in one table with other two tables
pre-condition	description	None
	specification	None
post-condition	description	For any entry in a table, there exists at least an entry in each of other two tables, they share the same session ID.
	specification	$\forall i : Z \mid 1 \leq i \leq$ sizeof(JS_TABLE) $\bullet$ (JS_TABLE(i).jSession.sessionId $\neq$ N $\Rightarrow$ ($\exists j : Z \mid 1 \leq j \leq$ sizeof(IMG_TABLE) $\bullet$ IMG_TABLE(j).imgSession.sessionId = JS_TABLE(i).jSession.sessionId) $\wedge$ ($\exists k$: Z $\mid 1 \leq k \leq$ sizeof(PROXY_TABLE) $\bullet$ PROXY_TABLE(k).proxySession.sessionId = JS_TABLE(i).jSession.sessionId))

Table 4-46: Associate Data in Different Tables

name		createPathNode
input parameter (s)		IMG_TABLE, PAGE_TABLE
output parameter (s)		PATH_TABLE’
pre-condition	description	If no such an entry exists in PATH_TABLE.
	specification	$\forall i : Z \mid 1 \le i \le \text{sizeof(IMG\_TABLE)} \bullet$ $\neg(\exists j : Z \mid 1 \le j \le \text{sizeof(PATH\_TABLE)} \bullet$ IMG_TABLE(i).imgSession.sessionId == PATH_TABLE(j).session.sessionId)
post-condition	description	For all entries in a session, all pages ordered by time are stored in a string.
	specification	$\forall i : Z \mid 1 \le i \le \text{sizeof(IMG\_TABLE)} \bullet$ $(\forall j : Z \mid 1 \le j \le \text{sizeof(IMG\_TABLE)} \bullet$ IMG_TABLE(i).imgSession.sessionId == IMG_TABLE(j).imgSession.sessionId $\Rightarrow$ $(\forall k, g : Z \mid 1 \le k, g \le \text{sizeof(PAGE\_TABLE)} \bullet$ (IMG_TABLE(i).imgReferrerURL == PAGE_TABLE(k).url $\wedge$ IMG_TABLE(j).imgReferrerURL ==PAGE_TABLE(g).url) $\Rightarrow$ (sizeof(PATH_TABLE′) = sizeof(PATH_TABLE) + 1 $\wedge$ PATH_TABLE′(i).sessionPage = PAGE_TABLE(k).pageId + PAGE_TABLE(g).pageId $\wedge$ PATH_TABLE′(i).session.sessionId = IMG_TABLE(i).imgSession.sessionId)))

Table 4-47: Create a Path Node

- **Cluster Analysis**

Cluster analysis is based on user data in the tables USER_TABLE and PATH_TABLE. In cluster analysis, we only care about which page has been visited in a certain session, not the browsing order of those pages in that session. Obviously, two users show similar interests if they visited the same or similar pages no matter the path they followed. In the specification shown in Table 4-48, the function “clustering” and a constant “THRESHHOLD” are used without providing their details, since as presented in Chapter 3, standard clustering algorithms are available in the literature, therefore, instead of redefining them, they are used here directly.

name		identifyUserCluster
input parameter (s)		USER_TABLE, PATH_TABLE
output parameter (s)		CLUSTER_TABLE′ or none
pre-condition	description	Assign a cluster ID to all users; any user has an initial unique cluster ID.
	specification	$\forall i : Z \mid 1 \le i \le \text{sizeof(USER\_TABLE)} \bullet$ $(\exists j : Z \mid 1 \le j \le \text{sizeof(CLUSTER\_TABLE)} \bullet$ $\text{USER\_TABLE}(i).\text{user} ==$ $\text{CLUSTER\_TABLE}(j).\text{user}) \wedge$ $\text{CLUSTER\_TABLE}(j).\text{clusterId} \neq N \wedge$ $\neg (\exists g: Z \mid 1 \le k \le \text{sizeof(CLUSTER\_TABLE)} \bullet$ $\text{CLUSTER\_TABLE}(j).\text{clusterId} ==$ $\text{CLUSTER\_TABLE}(k).\text{clusterId}))$
post-condition	description	Identify user(j) is in a given cluster or not.
	specification	$\forall i : Z \mid 1 \le i \le \text{sizeof(CLUSTER\_TABLE)} \bullet$ $(\forall j : Z \mid 1 \le j \le \text{sizeof(CLUSTER\_TABLE)} \bullet$ $\text{clustering(CLUSTER\_TABLE}(i).\text{uniquePage},$ $\text{CLUSTER\_TABLE}(j).\text{uniquePage}) \le$ $\text{THRESHHOLD} \Rightarrow$ $\text{CLUSTER\_TABLE}'(i).\text{clusterId} =$ $\text{CLUSTER\_TABLE}'(j).\text{clusterId}$

Table 4-48: Identify User Cluster

4.6.2 Verification

The following sections verify the requirements captured by above specification.

4.6.2.1 Verification of Traffic Tracking

Verification of traffic tracking includes verifying the requirements for each of the three tracking approaches.

- **JavaScript Tracking**

Data tracked using JavaScript code consist of two parts, JsFixedData and JsVaryData. JsFixedData are those data which do not change, in the same session. For any browser accessing a given site, all data items in JsFixedData are always tracked. To ensure each item in JsFixedData has a value, in "initializeFixedData", all variables are initialized. When a user comes to a page, new fixed data items will replace those initialized values. This is achieved in "trackFixedData". JsVaryData exists when the user has some activities on a page. As shown in the post-condition of "trackVaryData", each item in JsVaryData has a value. Besides JsFixedData and JsVaryData, we also track the time when the tracked data are sent to the server-side and the IP address of the user, since client-side JavaScript cannot get the IP address. All these can be found in "trackJsDataEntry". For any user, since "JsFixedData" is about the operation environment of the user, therefore, no matter whether the "JsVaryData" exists or not, "trackJsDataEntry" always runs if "JsFixedData" exists. That is why we have a pre-condition in "trackJsDataEntry" to make sure "JsFixedData" exists before executing "trackJsDataEntry". To identify a user, we create client-side cookie and server-side session for the user if the cookie and session are unavailable. Since we are using JavaScript code to create a cookie, we should ensure that the user's browser supports JavaScript and the cookie's function is turned on. Also if such a cookie already exists, we do not need to create a new one. These are captured in the pre-condition of "createCookie". For the session created by the server, the session ID is sent to the client as another cookie, we need to do the same check as above. This is shown in "checkJsSession". Finally, all tracked data are sent to the table JS_TABLE in "jsAddEntry". To ensure the same data are not sent more than once, we use a pre-condition to make sure only new data entry can be sent to the database table.

- **Single-pixel Image Tracking**

Single-pixel image tracks user by using server-side sessions. All server-side programs can share a session. It is possible before the tracking program is triggered, other tracking processes, such as HTTP proxy or JavaScript code, have occurred. However, whether such a session exists or not, we only need to ensure a session exists after "checkImgSession". All data tracked using this approach remain unchanged for the same page. Therefore, we do not need to initialize these data items in "trackImgDataEntry", and as the post-condition, each of these data items has a value. All tracked data are sent to the table IMG_TABLE in "imgAddEntry", no duplicate data are sent.

- **HTTP Proxy Server Tracking**

Every time a user makes a request to a server and the server sends a response to the user, the data about the request and the response are tracked by the HTTP proxy server. First, session in this approach is checked as “checkProxySession”, whose post-condition ensures that a session exists for a user whether the session ID is newly created or it is an old one. In “trackProxyEntry”, the user’s data about requests, responses, and session are tracked through the post-condition which ensures that each of these data items has a value. Finally, all tracked data are sent to the table PROXY_TABLE in “proxyAddEntry”. A pre-condition prevents sending the data in duplicates.

4.6.2.2 Verification of Traffic Analysis

Verification of traffic analysis consists of verifying the requirements for user and session identification, path and cluster analyses.

- **Unique User Identification**

Unique user identification is done based on the two assumptions: a username identifies a user; two users with the same cookie are same user. In “identifyUserByUsername”, we check whether or not a username has been tracked by JavaScript code. If a username exists, that user is identified. Only a new user’s data can be sent to the table USER_TABLE. Therefore, before sending data, we check whether this user exists in that table or not as shown in pre-condition of “identifyUserByUsername”. In “identifyUserByCookie”, we check any two entries in the IMG_TABLE. If these two entries have the same cookie, that means these two entries are from the same user. The data of the user are added to the table USER_TABLE after checking to ensure no such a user exists in that table.

- **Unique Session Identification**

Session analysis is done on the assumption: only data in the same session are associated with the same session ID. In “identifySession”, for any two available entries in each of the three source tables: JS_TABLE, IMG_TABLE, and PROXY_TABLE, if they share the same session ID, they come from the same session. Also, since we track session ID during each tracking, therefore, session ID can be used to associate data from those three tables, as presented in "associateData".

- **Path Analysis**

Path analysis is based on the pages, which are ordered by time and browsed by a user in a session. A path is defined as a group of pages ordered by time in this specification. Since single-pixel image tracks a user by page views, therefore, path analysis uses data from the table IMG_TABLE. In the post-condition of "createPathNode", we specify that for any entry in the table IMG_TABLE, if another entry has the same session ID as the first one, then we add the second one's page string to the first one. In this way, all pages with the same session ID are added one by one to form a path node.

- **Cluster Analysis**

Cluster analysis assigns a unique cluster ID to each user in the table USER_TABLE. Since a user may have more than one session, we base our analysis on the assumption that if a user is interested in some pages, the user would visit the pages several times. Based on this, we only consider the session in which there are the most unique pages. As a result, each user has only one session to be considered. This is specified in "identifyUserCluster". In the pre-condition, we ensure that each user in USER_TABLE has a unique ID. In the post-condition, consider any two users, A and B, if the distance of clustering operation on the pages visited by A and B satisfies a pre-set threshold, we consider A and B belong to the same cluster and as a result, replace user B's cluster's ID with user A's cluster's ID.

4.7 Run-time Behavior

Traffic tracking and analysis in this thesis includes two parts: tracking and analysis. The main focus is on tracking, although some analyses are done as well.

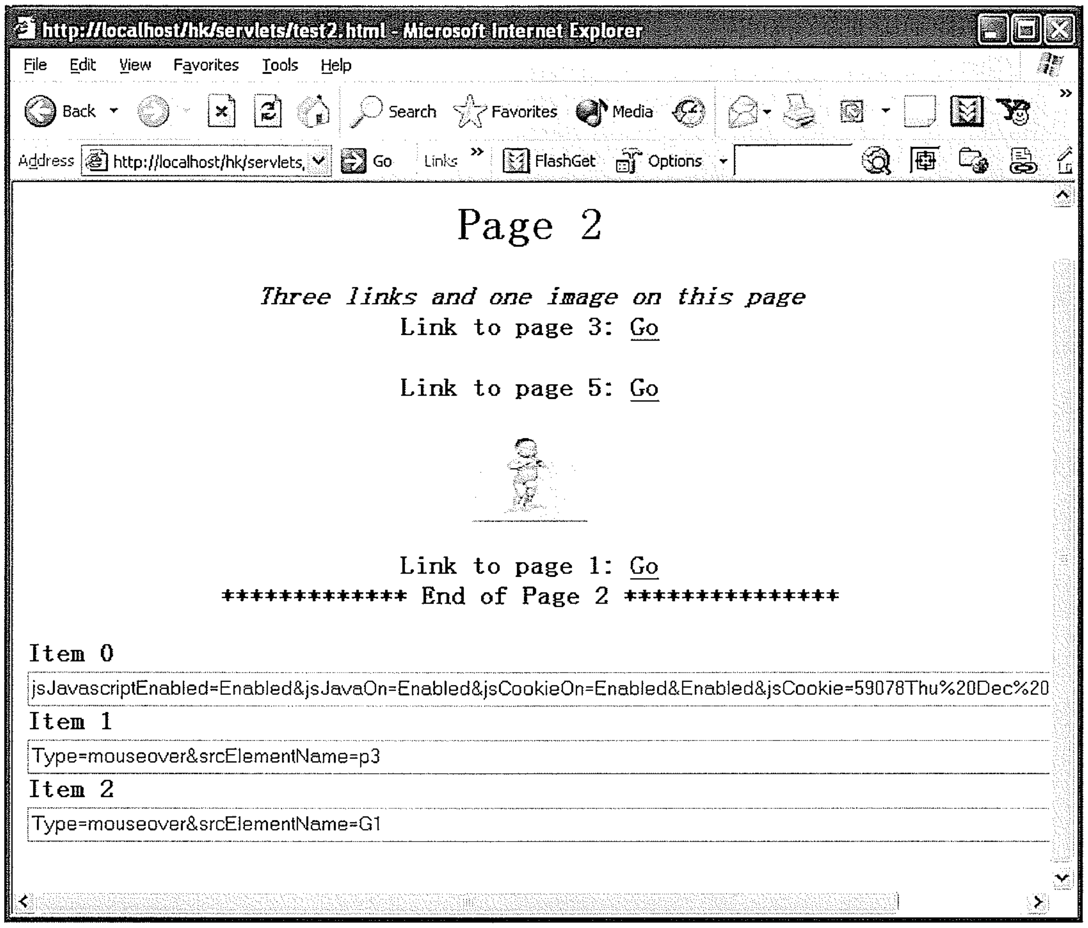
4.7.1 Traffic Tracking

The run-time behavior of traffic tracking module includes all behaviors from each of the three tracking approaches: JavaScript tracking, single-pixel image, and HTTP proxy server.

- **JavaScript Tracking**

JavaScript tracking can capture and send form data to the server program, which parses the required data items and send them to a database. The tracking mechanism is based on tracking user events. The tool can be configured to select what kinds of events should be tracked. In this implementation, we track mouse-over and mouse-click events for all links and images on any page; for form input, we register mouse-out, field-change, and key-pressed events; and mouse-out and field-changed for other types of form input such as text-area and text-field.

Figure 4.3 shows what happens when a user using MSIE 6.0 visits page 2, which has JavaScript tracking code. On the page, the form at the bottom is, in fact, hidden from the user. The JavaScript tracking code automatically creates this form every time a user with a JavaScript enabled browser arrives at this page. Item 0 of the form includes all fixed data and cookies information if the user turns on cookie function. The other items record user event types and source of the events. For example, in this form (the hidden part of Figure 4.3) the user moved the mouse over the link to page 3, and then put the mouse over the image.


http://localhost/hk/servlets/test2.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address http://localhost/hk/servlets,
Go
Links
FlashGet
Options

Page 2

Three links and one image on this page
Link to page 3: Go

Link to page 5: Go

Link to page 1: Go
************* End of Page 2 ***************

Item 0

jsJavascriptEnabled=Enabled&jsJavaOn=Enabled&jsCookieOn=Enabled&Enabled&jsCookie=59078Thu%20Dec%20

Item 1

Type=mouseover&srcElementName=p3

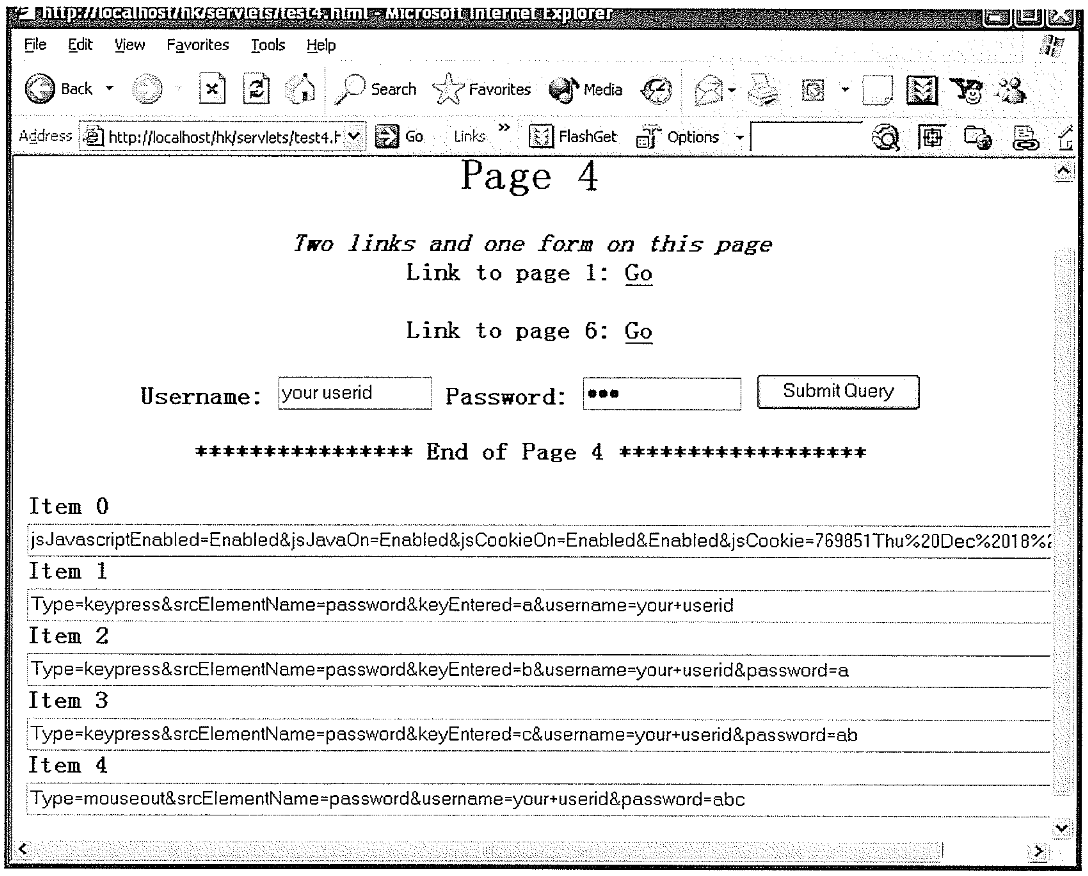
Item 2

Type=mouseover&srcElementName=G1

Figure 4.3: JS Tracking for MSIE Browsers: Link and Image Tracking

Although we can capture all the user's events on the page and send the records to the database, the user's last event before leaving this page is not shown on this form and other later run-time snapshots because we can not capture the screen (i.e., print out screen to get the snapshot) while leaving a page (normally this is the last event).

To imitate online shopping process on e-commerce sites, which usually ask for some user inputs, Figure 4.4 shows an example on how JavaScript tracking code tracks the user form data.


http://localhost/hk/servlets/test4.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/hk/servlets/test4.h Go Links FlashGet Options

Page 4

Two links and one form on this page
Link to page 1: Go

Link to page 6: Go

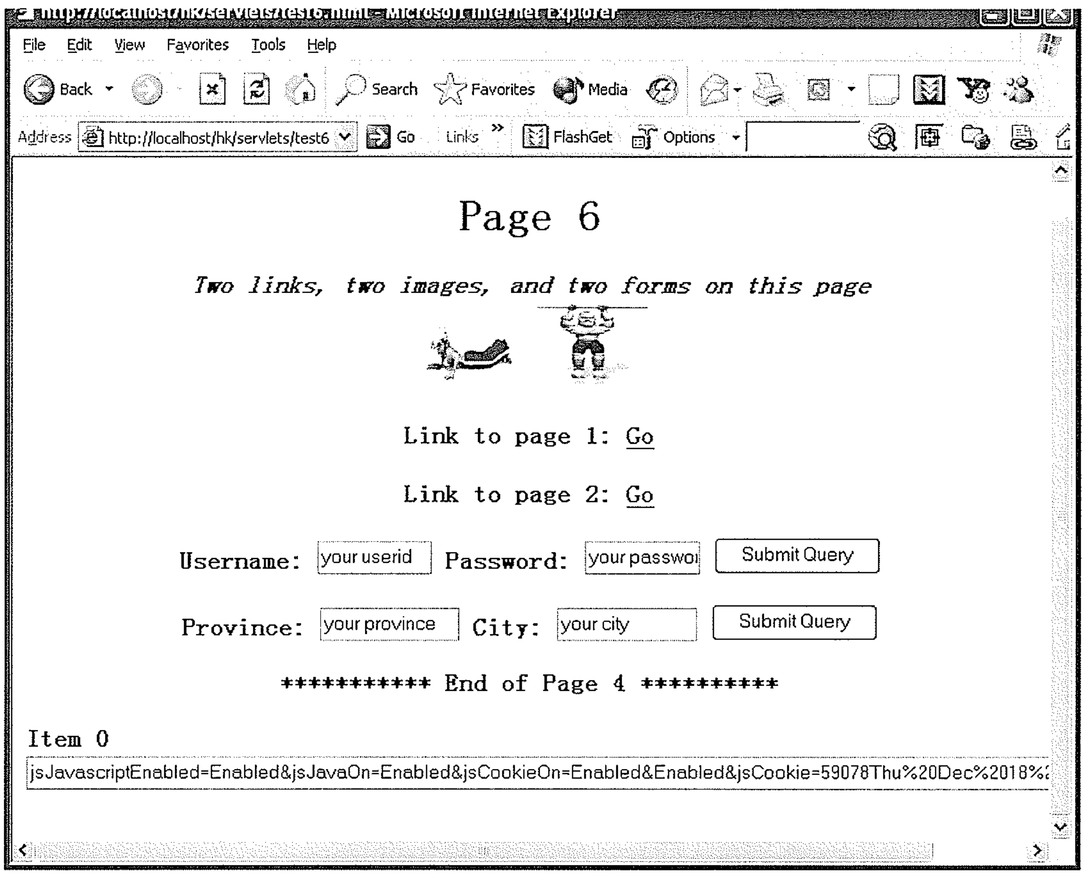
Username: your userid Password: ••• Submit Query

```
**************** End of Page 4 ******************

Item 0
jsJavascriptEnabled=Enabled&jsJavaOn=Enabled&jsCookieOn=Enabled&Enabled&jsCookie=769851Thu%20Dec%2018%
Item 1
Type=keypress&srcElementName=password&keyEntered=a&username=your+userid
Item 2
Type=keypress&srcElementName=password&keyEntered=b&username=your+userid&password=a
Item 3
Type=keypress&srcElementName=password&keyEntered=c&username=your+userid&password=ab
Item 4
Type=mouseout&srcElementName=password&username=your+userid&password=abc
```

Figure 4.4: JS Tracking for MSIE Browsers: Tracking User Input

On arriving at this page, the hidden form creates first line for fixed data. The pre-set user name is "your userid" and password is " your password". The second line of the tracking form shows that the user is changing the password and pressed the key " a" and input "a" in the password box. Then again, input "b" and "c" which are recorded in the next two lines as Item 2 and Item 3, respectively. The last item shows that the user moved out the mouse from the password box and now this item shows the whole value of the password is "abc", which is what the user has input. When the user leaves this page, these items are sent to the server-side program.

Usually, some pages might have more than one form. Figure 4.5 and Figure 4.6 show how the JavaScript code tracks input from multiple forms on a single page.


http://localhost/hk/servlets/test6.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/hk/servlets/test6 Go Links FlashGet Options

Page 6

Two links, two images, and two forms on this page

Link to page 1: Go

Link to page 2: Go

Username: your userid Password: your passwo Submit Query

Province: your province City: your city Submit Query

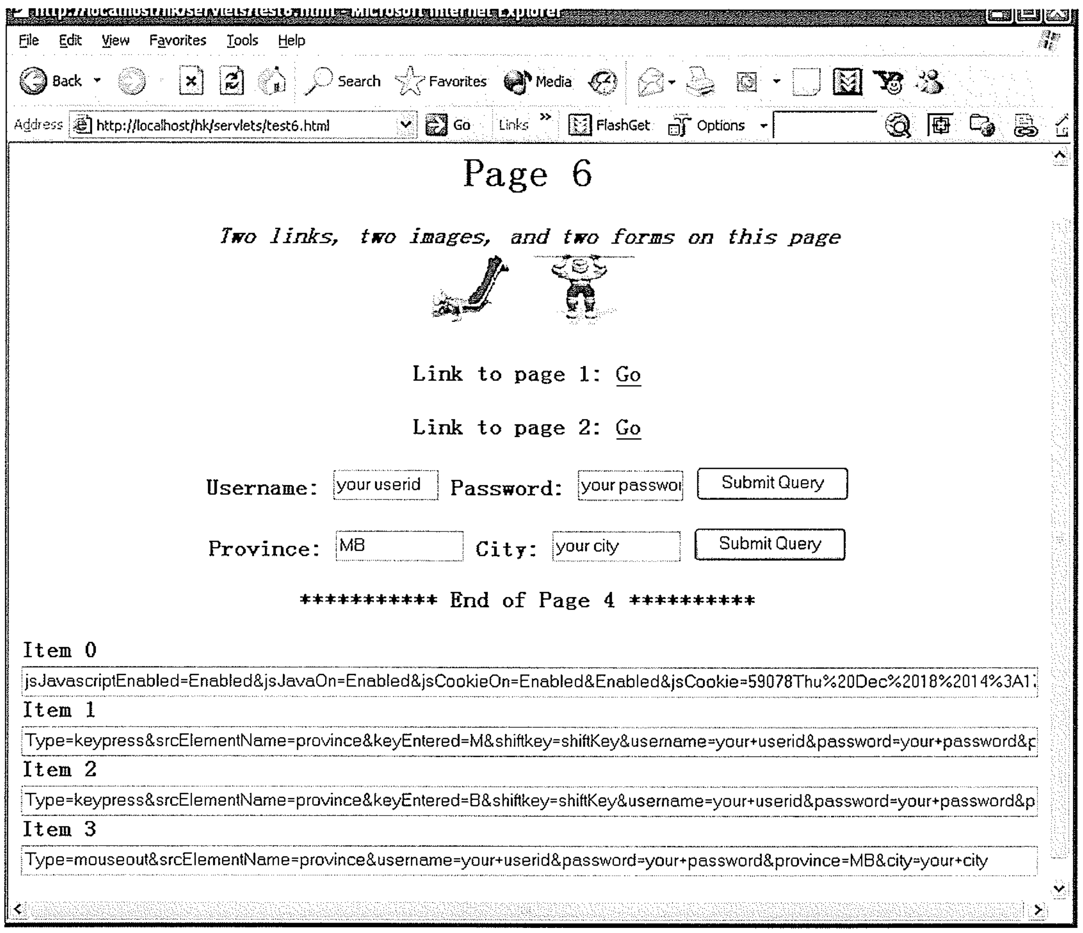
*********** End of Page 4 **********

Item 0

jsJavascriptEnabled=Enabled&jsJavaOn=Enabled&jsCookieOn=Enabled&Enabled&jsCookie=59078Thu%20Dec%2018%

Figure 4.5: JS Tracking for MSIE Browsers: Tracking Multiple Forms (1)

Figure 4.5 shows a situation when a user just arrives on a page. Since the user has not made any event yet, the hidden form has only one line for fixed data. Also, we can find that each of the input boxes has an initial value set by the site for the purpose of explanation. The two forms on this page are separate. In fact, the form can be any kind of forms, such as text-area, text-field, choice box, etc.


File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address http://localhost/hk/servlets/test6.html
Go
Links
FlashGet
Options

Page 6

Two links, two images, and two forms on this page

Link to page 1: Go

Link to page 2: Go

Username: your userid Password: your passwoi Submit Query

Province: MB City: your city Submit Query

*********** End of Page 4 **********

Item 0

```
jsJavascriptEnabled=Enabled&jsJavaOn=Enabled&jsCookieOn=Enabled&Enabled&jsCookie=59078Thu%20Dec%2018%2014%3A1
```

Item 1

```
Type=keypress&srcElementName=province&keyEntered=M&shiftkey=shiftKey&username=your+userid&password=your+password&p
```

Item 2

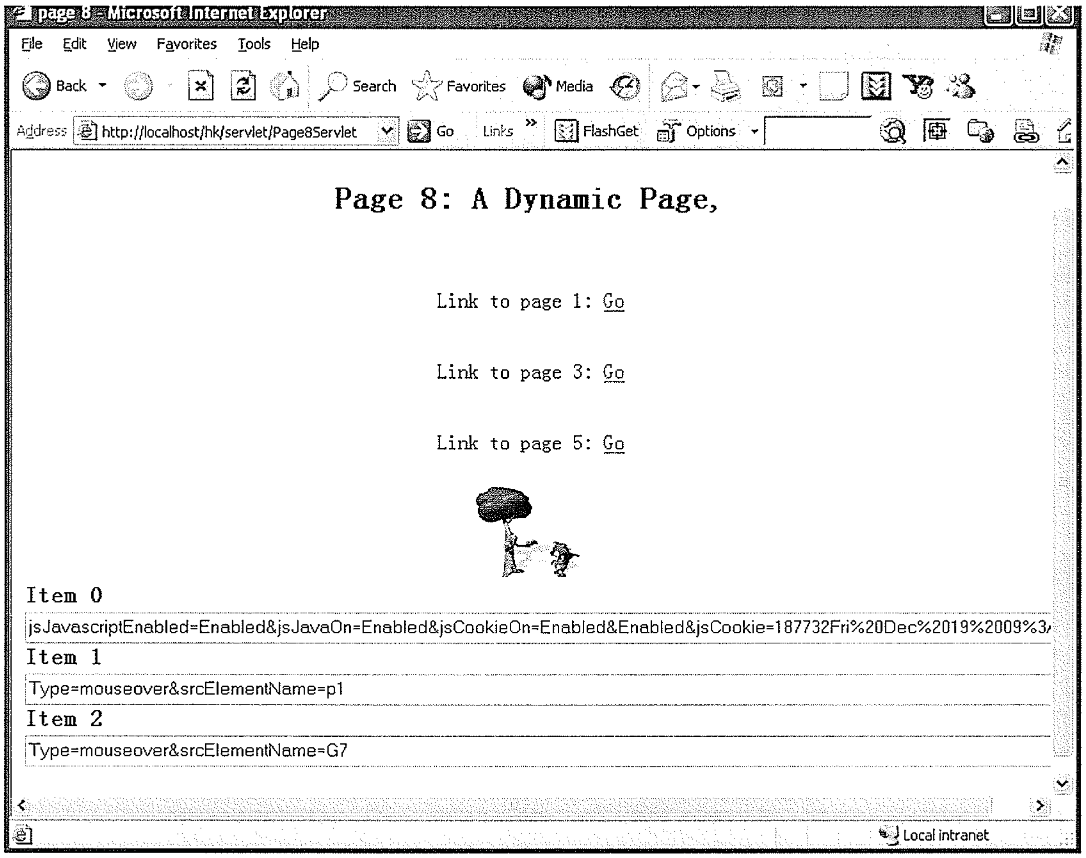
```
Type=keypress&srcElementName=province&keyEntered=B&shiftkey=shiftKey&username=your+userid&password=your+password&p
```

Item 3

```
Type=mouseout&srcElementName=province&username=your+userid&password=your+password&province=MB&city=your+city
```

Figure 4.6: JS Tracking for MSIE Browsers: Tracking Multiple Forms (2)

Figure 4.6 shows the user changed the value in the second form, which triggered the JavaScript code to capture data from all form inputs. Item 1 and Item 2 show that the user deleted the default value of the province input box and typed "M" and "B" successively. Item 3 shows the user moved the mouse out. As a matter of fact, even if the user does not change anything in any input form at all, any registered event can trigger the JavaScript code to capture data from all current fields in all the forms on the page.


page 8 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/hk/servlet/Page8Servlet Go Links FlashGet Options
Local intranet

Page 8: A Dynamic Page,

Link to page 1: Go

Link to page 3: Go

Link to page 5: Go

Item 0

jsJavascriptEnabled=Enabled&jsJavaOn=Enabled&jsCookieOn=Enabled&Enabled&jsCookie=187732Fri%20Dec%2019%2009%3

Item 1

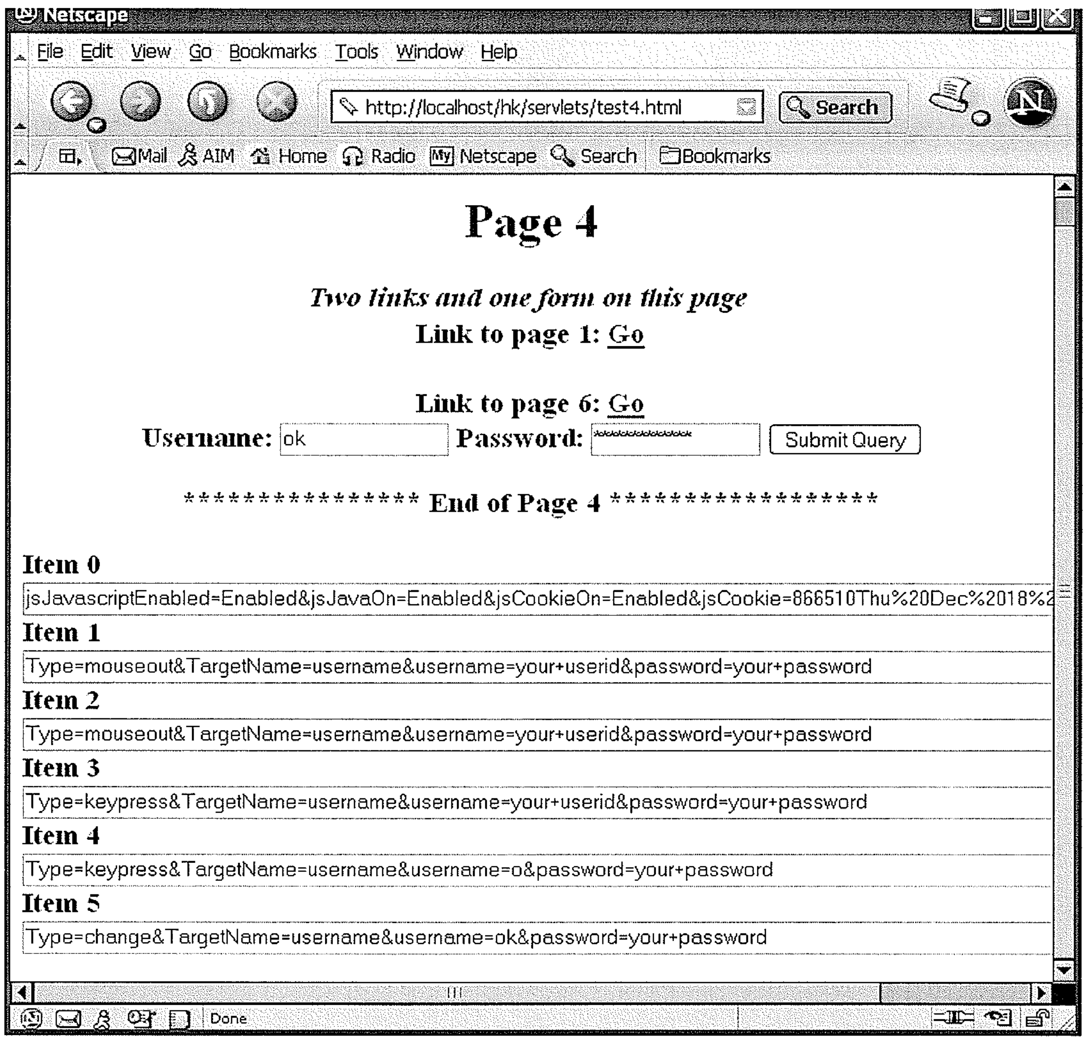
Type=mouseover&srcElementName=p1

Item 2

Type=mouseover&srcElementName=G7

Figure 4.7: JS Tracking for MSIE Browsers: Tracking on A Dynamic Page

Dynamic pages are widely used not just for e-commerce sites alone. Figure 4.7 shows that JavaScript tracking also works well for dynamic pages. The displayed page, page 8, is a dynamically created by a servlet based on the user's request. From Figure 4.7, we can see similar behavior as in Figure 4.3.


Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/hk/servlets/test4.html
Search
Mail AIM Home Radio Netscape Search Bookmarks
Done

Page 4

Two links and one form on this page

Link to page 1: Go

Link to page 6: Go

Username: ok **Password:** ************** Submit Query

**************** **End of Page 4** ******************

Item 0

```
jsJavascriptEnabled=Enabled&jsJavaOn=Enabled&jsCookieOn=Enabled&jsCookie=86651 0Thu%20Dec%2018%2
```

Item 1

```
Type=mouseout&TargetName=username&username=your+userid&password=your+password
```

Item 2

```
Type=mouseout&TargetName=username&username=your+userid&password=your+password
```

Item 3

```
Type=keypress&TargetName=username&username=your+userid&password=your+password
```

Item 4

```
Type=keypress&TargetName=username&username=o&password=your+password
```

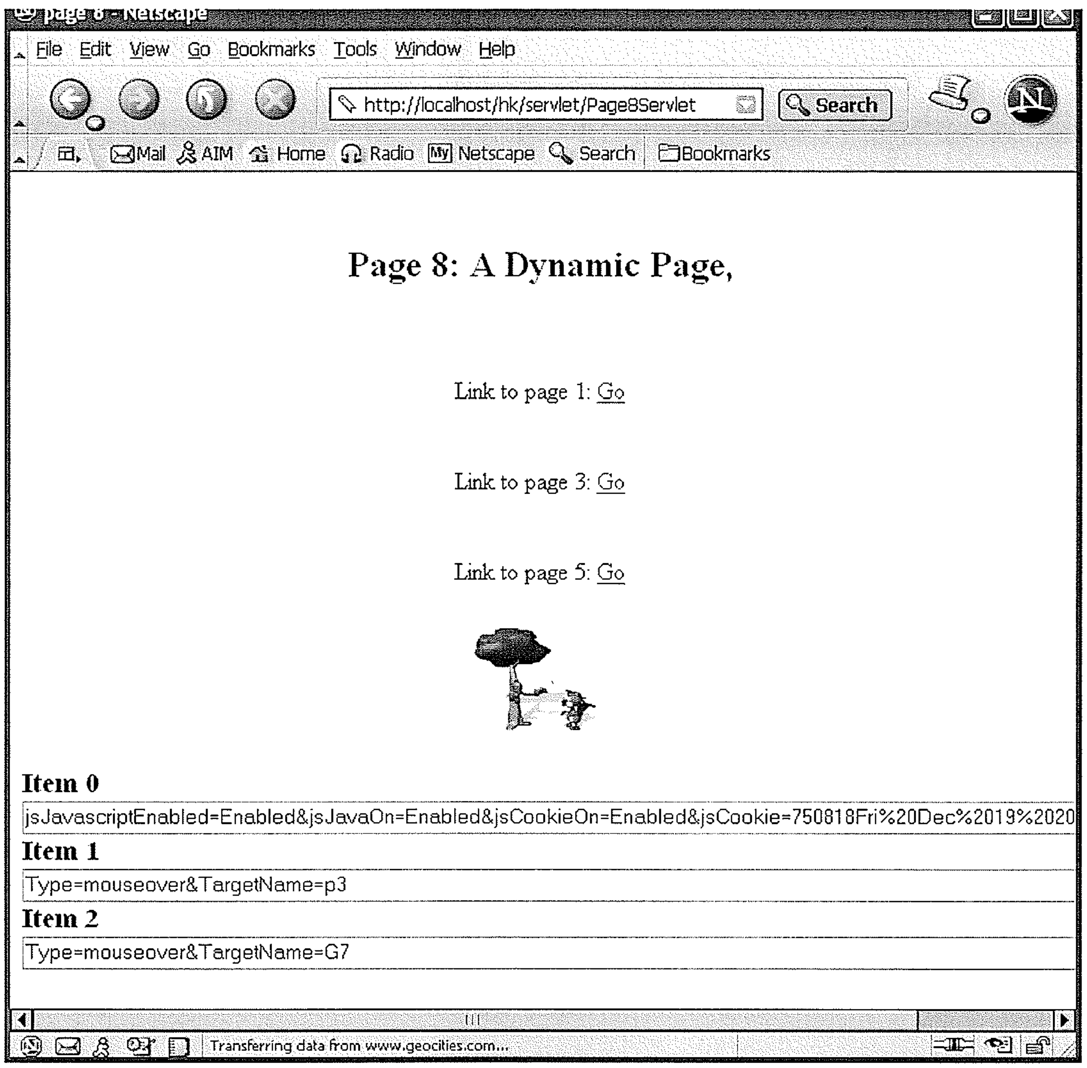
Item 5

```
Type=change&TargetName=username&username=ok&password=your+password
```

Figure 4.8: JS Tracking for NS Browsers: Tracking Form Data

Although only about 4% of Internet users use Netscape Navigator, a very small size compared with 94% MSIE users, to show that the JavaScript tracking can track the NS user events on any web page, Figure 4.8 shows an example which captures data on both static page and form data.

Figure 4.9 shows that for dynamic pages, the NS browser displays the same behavior as the MSIE browser.


page 8 - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/hk/servlet/Page8Servlet
Search
Mail AIM Home Radio Netscape Search Bookmarks

Page 8: A Dynamic Page,

Link to page 1: Go

Link to page 3: Go

Link to page 5: Go

Item 0

jsJavascriptEnabled=Enabled&jsJavaOn=Enabled&jsCookieOn=Enabled&jsCookie=750818Fri%20Dec%2019%2020

Item 1

Type=mouseover&TargetName=p3

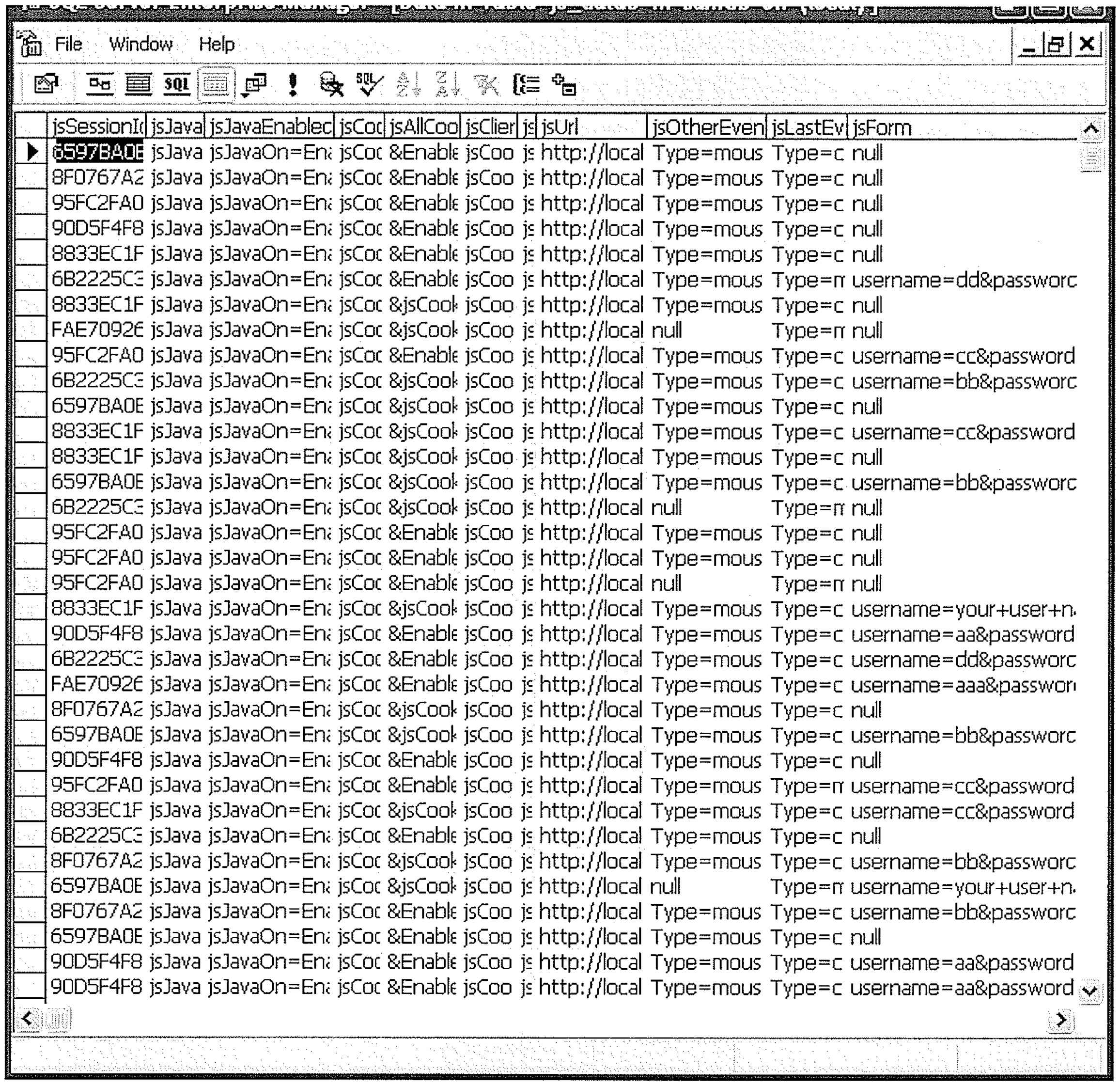
Item 2

Type=mouseover&TargetName=G7

Transferring data from www.geocities.com...

Figure 4.9: JS Tracking for NS Browsers: Tracking Data on A Dynamic Page

All the data tracked by JavaScript are sent to a database. Figure 4.10 shows these data in the database table.



jsSessionI	jsJava	jsJavaEnablec	jsCoc	jsAllCoo	jsClier	js	jsUrl	jsOtherEven	jsLastEv	jsForm
6597BA0E	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	null
8F0767A2	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	null
95FC2FA0	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	null
90D5F4F8	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	null
8833EC1F	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	null
6B2225C3	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=m	username=dd&passworc
8833EC1F	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	null
FAE7092E	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	null	Type=m	null
95FC2FA0	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	username=cc&password
6B2225C3	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	username=bb&passworc
6597BA0E	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	null
8833EC1F	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	username=cc&password
8833EC1F	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	null
6597BA0E	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	username=bb&passworc
6B2225C3	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	null	Type=m	null
95FC2FA0	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	null
95FC2FA0	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	null
95FC2FA0	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	null	Type=m	null
8833EC1F	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	username=your+user+n
90D5F4F8	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	username=aa&password
6B2225C3	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	username=dd&passworc
FAE7092E	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	username=aaa&passwor
8F0767A2	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	null
6597BA0E	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	username=bb&passworc
90D5F4F8	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	null
95FC2FA0	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=m	username=cc&password
8833EC1F	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	username=cc&password
6B2225C3	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	null
8F0767A2	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	Type=mous	Type=c	username=bb&passworc
6597BA0E	jsJava	jsJavaOn=En	jsCoc	&jsCook	jsCoo	js	http://local	null	Type=m	username=your+user+n
8F0767A2	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	username=bb&passworc
6597BA0E	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	null
90D5F4F8	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	username=aa&password
90D5F4F8	jsJava	jsJavaOn=En	jsCoc	&Enable	jsCoo	js	http://local	Type=mous	Type=c	username=aa&password

Figure 4.10: All Data Tracked by JavaScript Code

Figure 4.10 shows the data specified by our requirement specification. The most important data items captured by JavaScript tracking are the user events and form data. These two items cannot be obtained from other tracking approaches.

One disadvantage of JavaScript tracking comes from the browser's caching. Some pages may not be tracked. Although we can control our JavaScript code, we cannot control users' browsers. Another problem is with the NS browser, which supports much less browser events than MSIE. This fact leads to the problem of how to send data to the server-side. Although using other client-side technology such as applet can resolve this, this would raise performance and security issues, which come from applet technology. Since NS is not our target browser in this thesis, we defer issues relating to NS together with use of applets to future work.

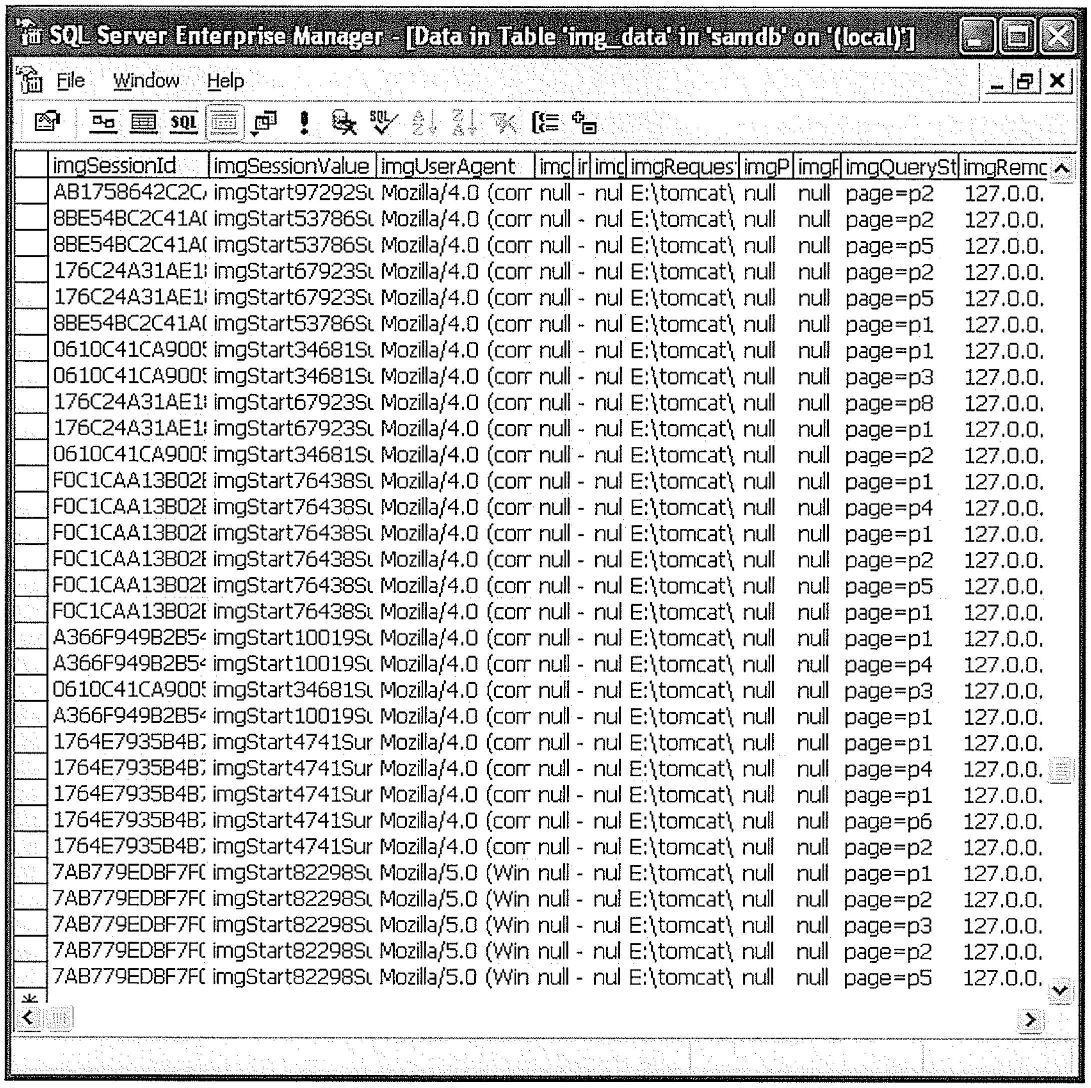
- **Single-pixel Image**

Single-pixel image is the main approach used in tracking users in this thesis. It works very well for both MSIE and NS. Since the client-side involves only downloading an image, the tracking code is on server-side, which is controllable.

There is a problem with tracking referrer page using this approach. Since the tracking image works at the background, it is impossible to get the "real" referrer page on any page. The referrer page we tracked is the current page on which the image sits. Consider the following example: when a user on page 1 moves to page 2, the tracking program on page 2 tracks the referrer page as page 2, and not page 1, since page 2 is the real referrer page for our tracking program. However, we are more interested in the referrer page of page 2.

Since the server-side tracking program is configurable, we set the program to none-cached. Thus, we can track each page in a session. In any session, we sort the pages by the request time. In this way, the last neighbor page is the referrer page of the current page.

Figure 4.11 shows the data in a user' s session from the database.


SQL Server Enterprise Manager - [Data in Table 'img_data' in 'samdb' on '(local)']
File
Window
Help

imgSessionId	imgSessionValue	imgUserAgent	imc	ir	imc	imgReques	imgP	imgF	imgQuerySt	imgRemc
AB1758642C2C,	imgStart97292St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p2	127.0.0.
8BE54BC2C41A(	imgStart53786St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p2	127.0.0.
8BE54BC2C41A(	imgStart53786St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p5	127.0.0.
176C24A31AE1!	imgStart67923St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p2	127.0.0.
176C24A31AE1!	imgStart67923St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p5	127.0.0.
8BE54BC2C41A(	imgStart53786St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
0610C41CA900!	imgStart34681St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
0610C41CA900!	imgStart34681St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p3	127.0.0.
176C24A31AE1!	imgStart67923St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p8	127.0.0.
176C24A31AE1!	imgStart67923St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
0610C41CA900!	imgStart34681St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p2	127.0.0.
F0C1CAA13B02E	imgStart76438St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
F0C1CAA13B02E	imgStart76438St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p4	127.0.0.
F0C1CAA13B02E	imgStart76438St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
F0C1CAA13B02E	imgStart76438St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p2	127.0.0.
F0C1CAA13B02E	imgStart76438St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p5	127.0.0.
F0C1CAA13B02E	imgStart76438St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
A366F949B2B5‹	imgStart10019St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
A366F949B2B5‹	imgStart10019St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p4	127.0.0.
0610C41CA900!	imgStart34681St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p3	127.0.0.
A366F949B2B5‹	imgStart10019St	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
1764E7935B4B;	imgStart4741Sur	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
1764E7935B4B;	imgStart4741Sur	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p4	127.0.0.
1764E7935B4B;	imgStart4741Sur	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
1764E7935B4B;	imgStart4741Sur	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p6	127.0.0.
1764E7935B4B;	imgStart4741Sur	Mozilla/4.0 (com	null	-	nul	E:\tomcat\	null	null	page=p2	127.0.0.
7AB779EDBF7F(	imgStart82298St	Mozilla/5.0 (Win	null	-	nul	E:\tomcat\	null	null	page=p1	127.0.0.
7AB779EDBF7F(	imgStart82298St	Mozilla/5.0 (Win	null	-	nul	E:\tomcat\	null	null	page=p2	127.0.0.
7AB779EDBF7F(	imgStart82298St	Mozilla/5.0 (Win	null	-	nul	E:\tomcat\	null	null	page=p3	127.0.0.
7AB779EDBF7F(	imgStart82298St	Mozilla/5.0 (Win	null	-	nul	E:\tomcat\	null	null	page=p2	127.0.0.
7AB779EDBF7F(	imgStart82298St	Mozilla/5.0 (Win	null	-	nul	E:\tomcat\	null	null	page=p5	127.0.0.

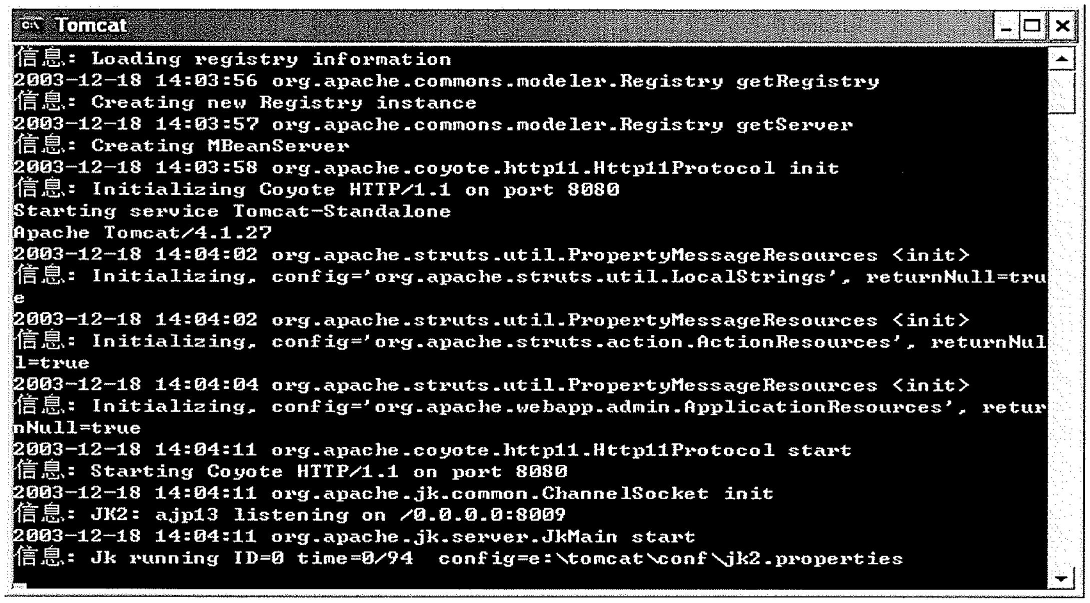
Figure 4.11: All Data Tracked by Single-Pixel Image

As stated previously, single-pixel image tracks user on a page view, so the column "imgCurPage" in Figure 4.11 records the page where the user was tracked. Besides working in an environment when the user turns off JavaScript, we can also control not to allow the user to cache the pages. These features allow single-pixel to be most efficient in tracking. Tracking users by page views makes later analysis much easier.

- **HTTP proxy server**

This thesis uses the local host as Web client, Web server, and application server where all our tracking programs reside. To start the proxy server, the Web server should be started first.

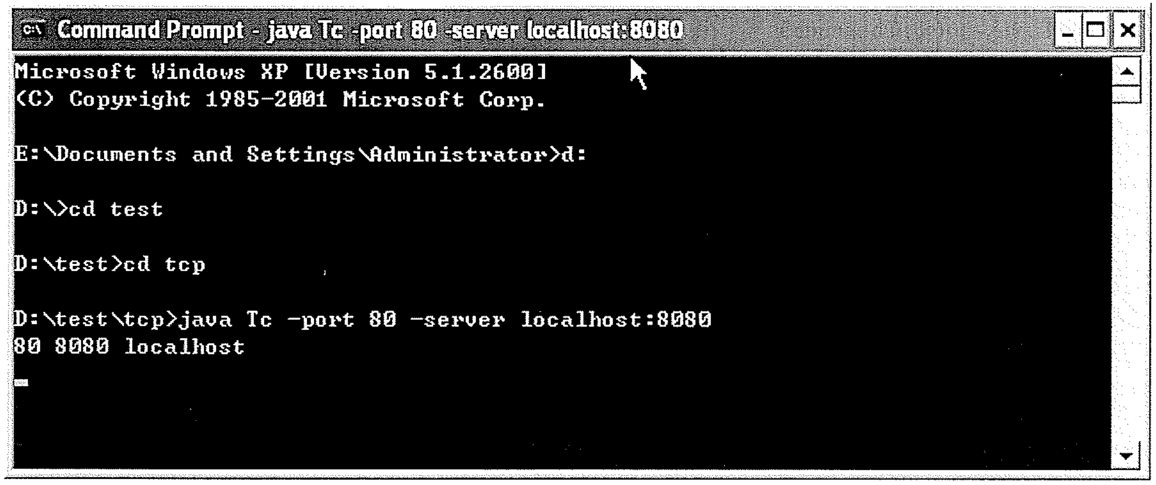
Figure 4.12 shows the start of the Web Server.


Tomcat

```
信息: Loading registry information
2003-12-18 14:03:56 org.apache.commons.modeler.Registry getRegistry
信息: Creating new Registry instance
2003-12-18 14:03:57 org.apache.commons.modeler.Registry getServer
信息: Creating MBeanServer
2003-12-18 14:03:58 org.apache.coyote.http11.Http11Protocol init
信息: Initializing Coyote HTTP/1.1 on port 8080
Starting service Tomcat-Standalone
Apache Tomcat/4.1.27
2003-12-18 14:04:02 org.apache.struts.util.PropertyMessageResources <init>
信息: Initializing, config='org.apache.struts.util.LocalStrings', returnNull=tru
e
2003-12-18 14:04:02 org.apache.struts.util.PropertyMessageResources <init>
信息: Initializing, config='org.apache.struts.action.ActionResources', returnNul
l=true
2003-12-18 14:04:04 org.apache.struts.util.PropertyMessageResources <init>
信息: Initializing, config='org.apache.webapp.admin.ApplicationResources', retur
nNull=true
2003-12-18 14:04:11 org.apache.coyote.http11.Http11Protocol start
信息: Starting Coyote HTTP/1.1 on port 8080
2003-12-18 14:04:11 org.apache.jk.common.ChannelSocket init
信息: JK2: ajp13 listening on /0.0.0.0:8009
2003-12-18 14:04:11 org.apache.jk.server.JkMain start
信息: Jk running ID=0 time=0/94  config=e:\tomcat\conf\jk2.properties
```

Figure 4.12: Web Server Starts on Localhost

The Web Server starts on the localhost's port 8080, which means that all requests to the Web server should go to http://localhost:8080. However, we used a proxy server to hide this real Web server. To achieve this, the proxy server starts on the standard port for http requests: 80. Figure 4.13 shows this process. After running the java program, the http proxy server starts on port 80 to receive responses and send requests to and from the real port (8080) of the Web server.


Command Prompt - java Tc -port 80 -server localhost:8080

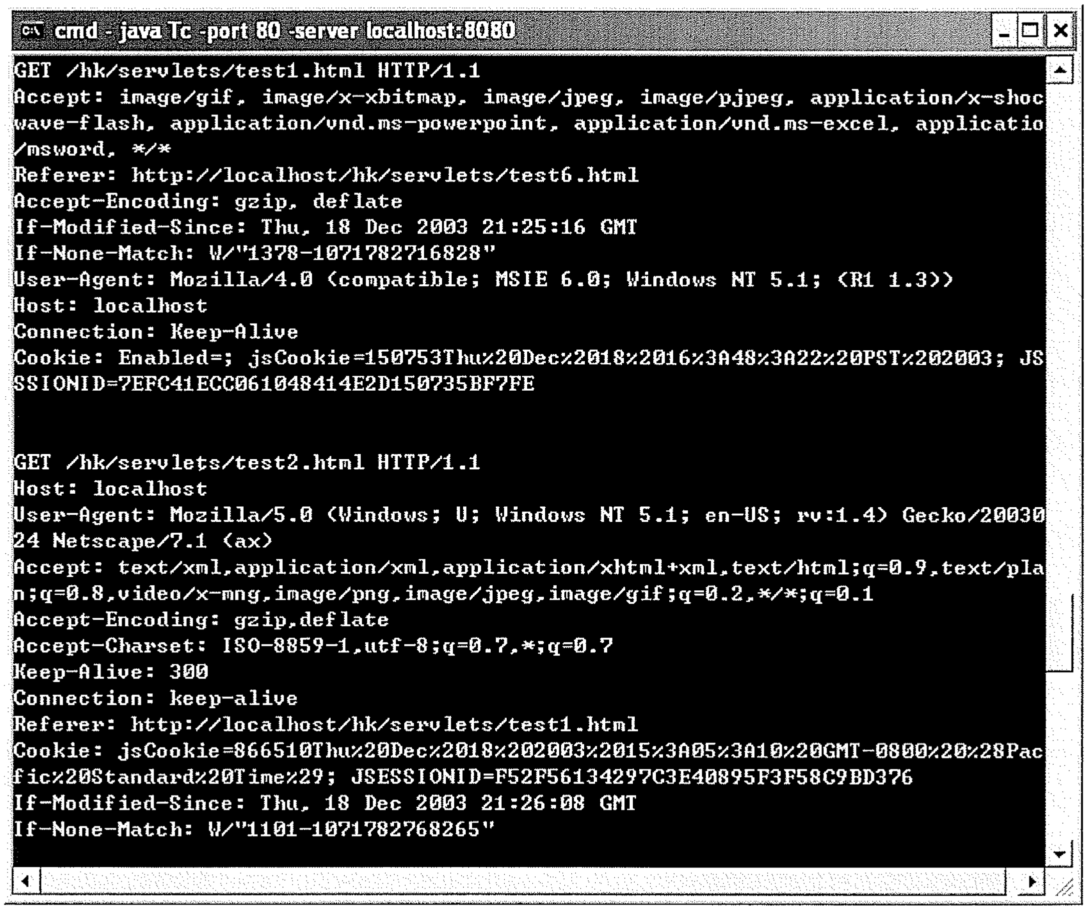
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

E:\Documents and Settings\Administrator>d:

D:\>cd test

D:\test>cd tcp

D:\test\tcp>java Tc -port 80 -server localhost:8080
80 8080 localhost
```

Figure 4.13: HTTP Proxy Server Starts

Figure 4.14 shows how the proxy tracks users and two requests were tracked from two different users and with two different browsers. One user used MSIE 6.0 and the other NS 7.0.

From both Figure 4.13 and Figure 4.14, we can find the address of the proxy server because they both show that the running port is 80, while the real port of the server is at 8080.

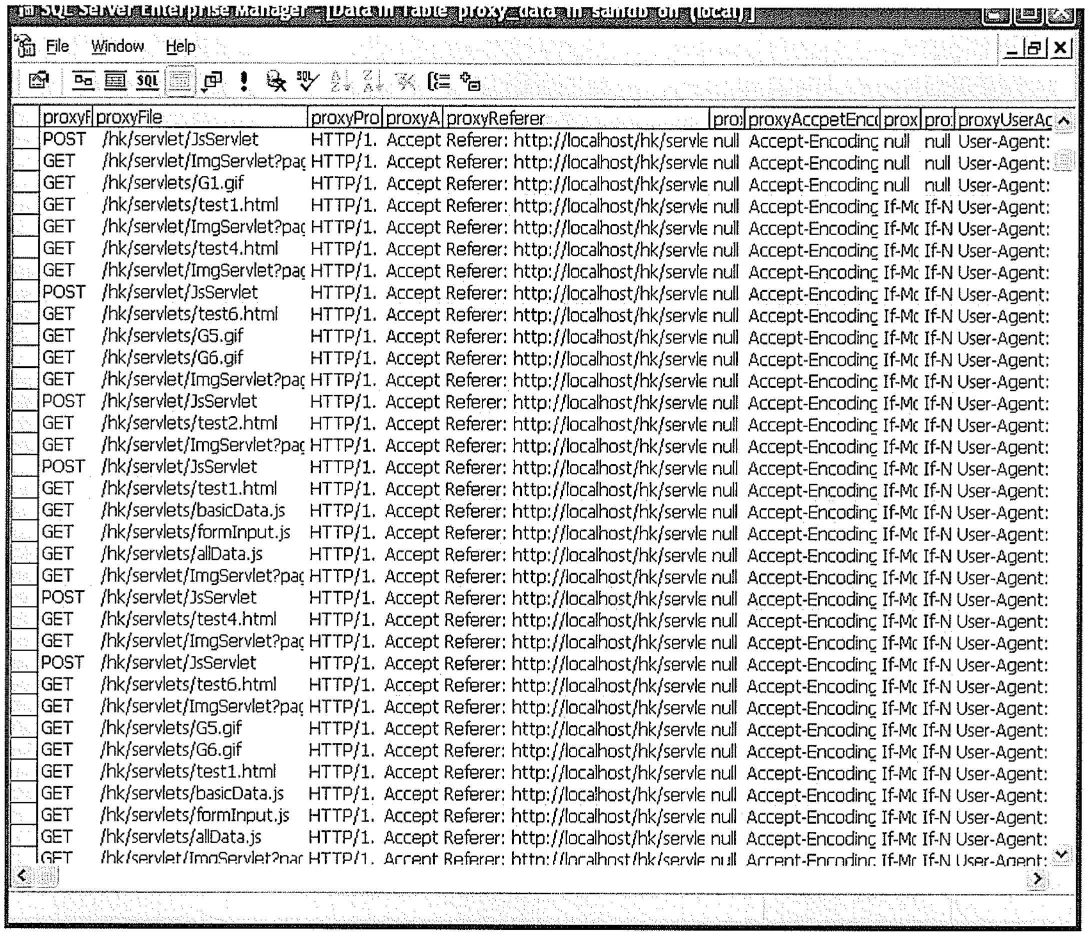
The proxy approach provides the only natural way to track any referrer page if available, since each request and response to and from the Web server must pass through this proxy server. A drawback of this approach is that it depends on the cookie values to integrate the data tracked by this approach into the other two sources. Although a user can disable the cookie function of the browser, Internet users must enable their cookie function while browsing as required by most Web sites to function correctly.



```
GET /hk/servlets/test1.html HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shoc
wave-flash, application/vnd.ms-powerpoint, application/vnd.ms-excel, applicatio
/msword, */*
Referer: http://localhost/hk/servlets/test6.html
Accept-Encoding: gzip, deflate
If-Modified-Since: Thu, 18 Dec 2003 21:25:16 GMT
If-None-Match: W/"1378-1071782716828"
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; (R1 1.3))
Host: localhost
Connection: Keep-Alive
Cookie: Enabled=; jsCookie=150753Thu%20Dec%2018%2016%3A48%3A22%20PST%202003; JS
SSIONID=7EFC41ECC06104841 4E2D150735BF7FE


GET /hk/servlets/test2.html HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.4) Gecko/20030
24 Netscape/7.1 (ax)
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/pla
n;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://localhost/hk/servlets/test1.html
Cookie: jsCookie=866510Thu%20Dec%2018%202003%2015%3A05%3A10%20GMT-0800%20%28Pac
fic%20Standard%20Time%29; JSESSIONID=F52F56134297C3E40895F3F58C9BD376
If-Modified-Since: Thu, 18 Dec 2003 21:26:08 GMT
If-None-Match: W/"1101-1071782768265"
```

Figure 4.14: HTTP Proxy Server Tracking

All the data tracked by the HTTP proxy server are also sent into the database. Figure 4.15 shows the data tracked by the proxy server approach. Most data found in Figure 4.15 are similar to those in a typical Web log file.


SQL Server Enterprise Manager - [Data in Table 'proxy_data' in 'samdb' on '(local)']
File Window Help

proxyF	proxyFile	proxyPro	proxyA	proxyReferer	pro	proxyAccpetEnc	prox	pro	proxyUserAc
POST	/hk/servlet/JsServlet	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	null	null	User-Agent:
GET	/hk/servlet/ImgServlet?pac	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	null	null	User-Agent:
GET	/hk/servlets/G1.gif	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	null	null	User-Agent:
GET	/hk/servlets/test1.html	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlet/ImgServlet?pac	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/test4.html	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlet/ImgServlet?pac	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
POST	/hk/servlet/JsServlet	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/test6.html	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/G5.gif	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/G6.gif	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlet/ImgServlet?pac	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
POST	/hk/servlet/JsServlet	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/test2.html	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlet/ImgServlet?pac	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
POST	/hk/servlet/JsServlet	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/test1.html	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/basicData.js	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/formInput.js	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/allData.js	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlet/ImgServlet?pac	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
POST	/hk/servlet/JsServlet	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/test4.html	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlet/ImgServlet?pac	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
POST	/hk/servlet/JsServlet	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/test6.html	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlet/ImgServlet?pac	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/G5.gif	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/G6.gif	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/test1.html	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/basicData.js	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/formInput.js	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlets/allData.js	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:
GET	/hk/servlet/ImgServlet?pac	HTTP/1.	Accept	Referer: http://localhost/hk/servle	null	Accept-Encodinc	If-Mc	If-N	User-Agent:

Figure 4.15: HTTP Proxy Server Tracking Data

4.7.2 Traffic Analysis

Traffic analysis in this thesis mainly focuses on answering those questions in Chapter 1. Those questions include the basic, advanced, and more advanced.

4.7.2.1 Basic Analysis

- **Which browsers and operation systems are the most common?**

We tracked user-agent data in all the three data sources, therefore, there is enough data to analyze a user's operation environment, such as operation and browsers. Since the single-pixel image approach tracks user by page view, our analysis is mainly based on this data source.

The analysis shows that in 15 tracked users, 14 users used MSIE 6.0 on Windows NT 5.1 operating system, while 1 user used NS on the same operating system. Figure 4.16 shows this analysis result.

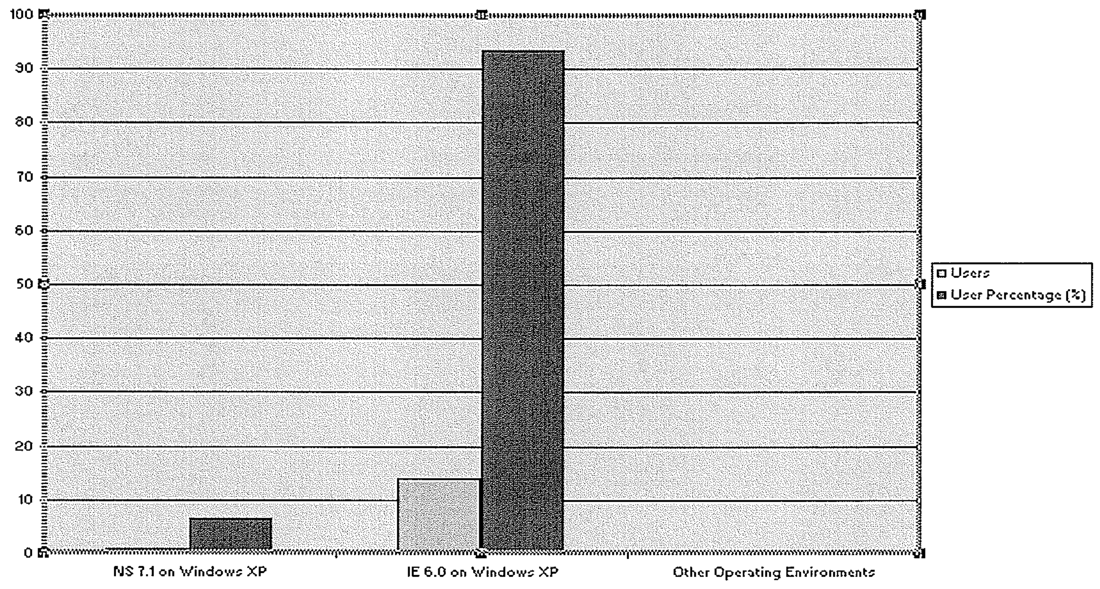


Figure 4.16: User Operation Environment

- **How many visitors visited this Web site?**

This question can be answered by analyzing the data from table js_data which stores usernames, and table img_data, which stores session and page views. Figure 4.17 shows that 16 visitors came to this site, 4 of them have registered while the other 12 are anonymous users.

Visitors	Registered	Anonymous
	aa	user 1
	bb	user 2
	aaa	user 3
	cc	user 4
		user 5
		user 6
		user 7
		user 8
		user 9
		user 10
		user 11
		user 12
Total	4	12
Percentage (%)	25	75

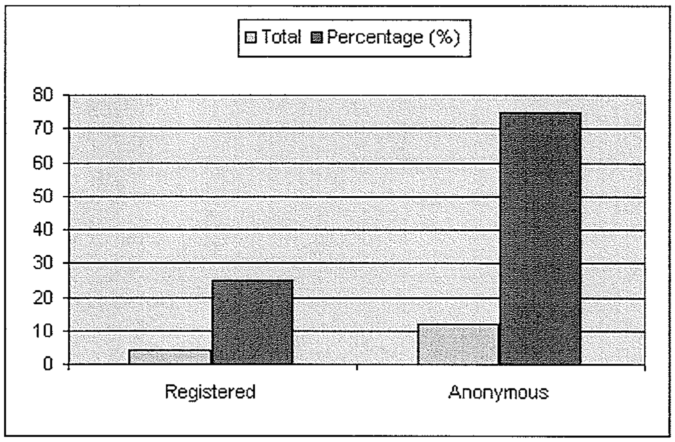


Figure 4.17: Visitors to This Site

- **Which page is most frequently requested?**

Based on the data tracked using single-pixel image approach, the request times of each page is shown in Figure 4.18.

Page	Times	Percentages (%)
page 1	44	35.77
page 2	22	17.89
page 3	13	10.57
page 4	17	13.82
page 5	8	6.50
page 6	18	14.63
page 7	0	0.00
page 8	1	0.81
page 9	0	0.00
Total	123	100.00

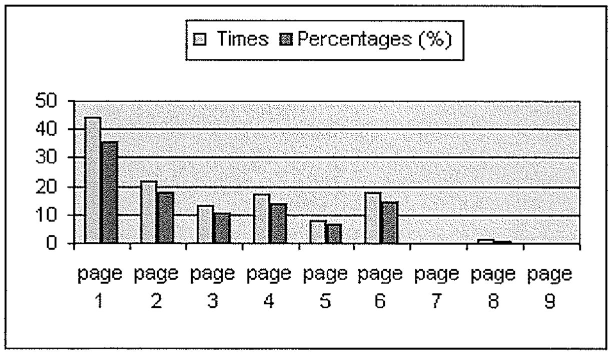


Figure 4.18: The Request Times of Each Page

From Figure 4.18, page 1 is the most requested page on this site.

- **From where are the visitors coming?**

This question involves finding the referrer pages. Figure 4.19 shows the statistical results for all referrer pages.

Page	Times	Percentages (%)
page 1	22	48.35
page 2	17	18.68
page 3	11	12.09
page 4	17	18.68
page 5	7	7.69
page 6	16	17.58
page 7	0	0.00
page 8	1	1.10
page 9	0	0.00
Total	91	100.00

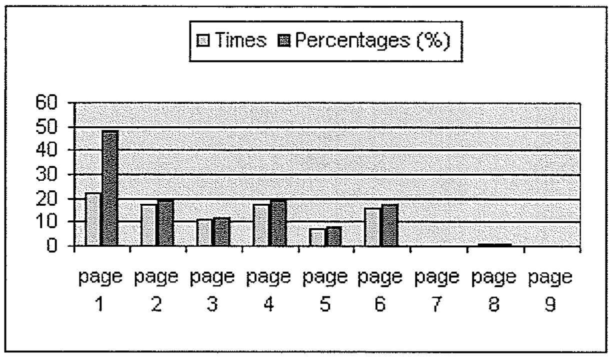


Figure 4.19: Referrer Pages

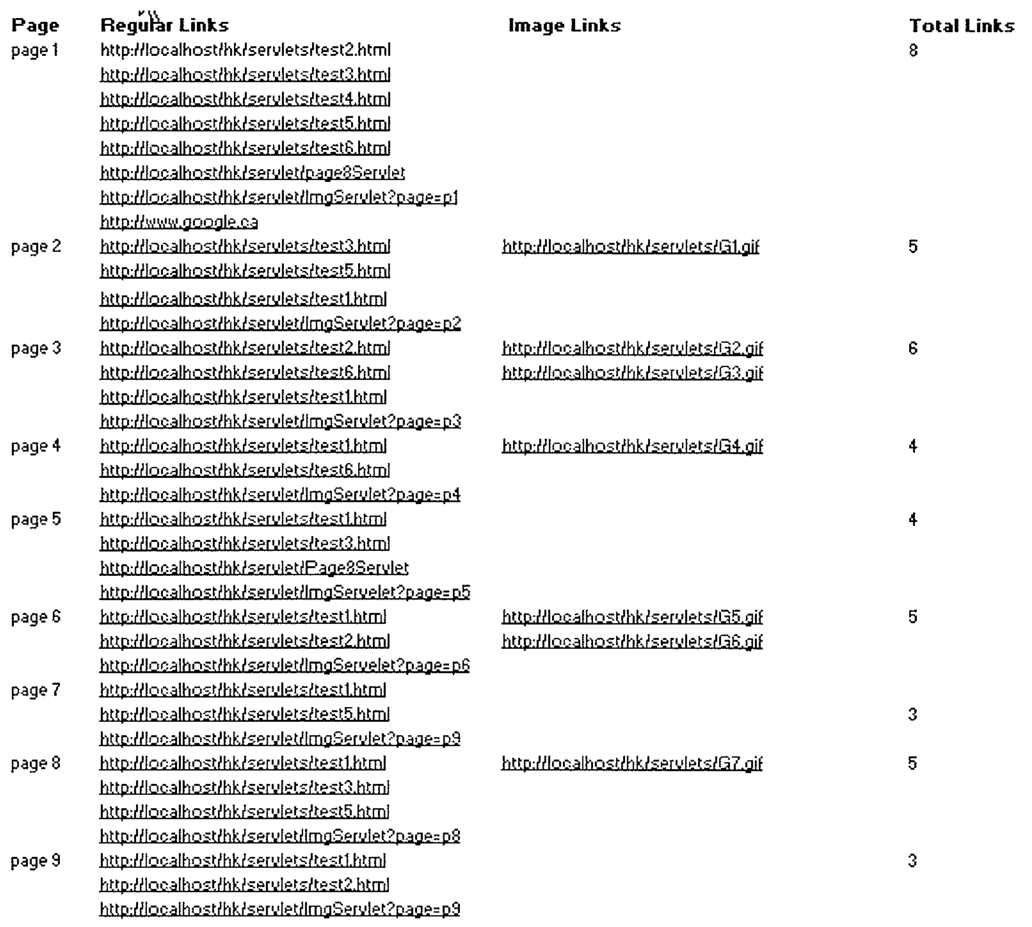
4.7.2.2 Advanced Analysis

Advanced analysis includes page content analysis, in which we parse out major content components of a Web page; path analysis, which allows us to know how people navigate through the site; and user clustering analysis, which serves marketing purposes based on the assumption: same browsing behavior comes from same interest.

- **Page Content Analysis**

HTML Web pages contains rich information in their texts, links and images, etc. Mining text information is another hot research area in data mining. We defer mining text information as future work, however, we show parsing link and image information as in Figure 4.20.

Page content can be used to help e-commerce sites owners in find a user's interest, and develop market strategies. For example, if a user stays on a page for a very long time, this may suggest some contents on this page are very attractive to this user. If this is the case for most users, that may suggest the business should put more relevant information on this page, since this page is attractive to most users.

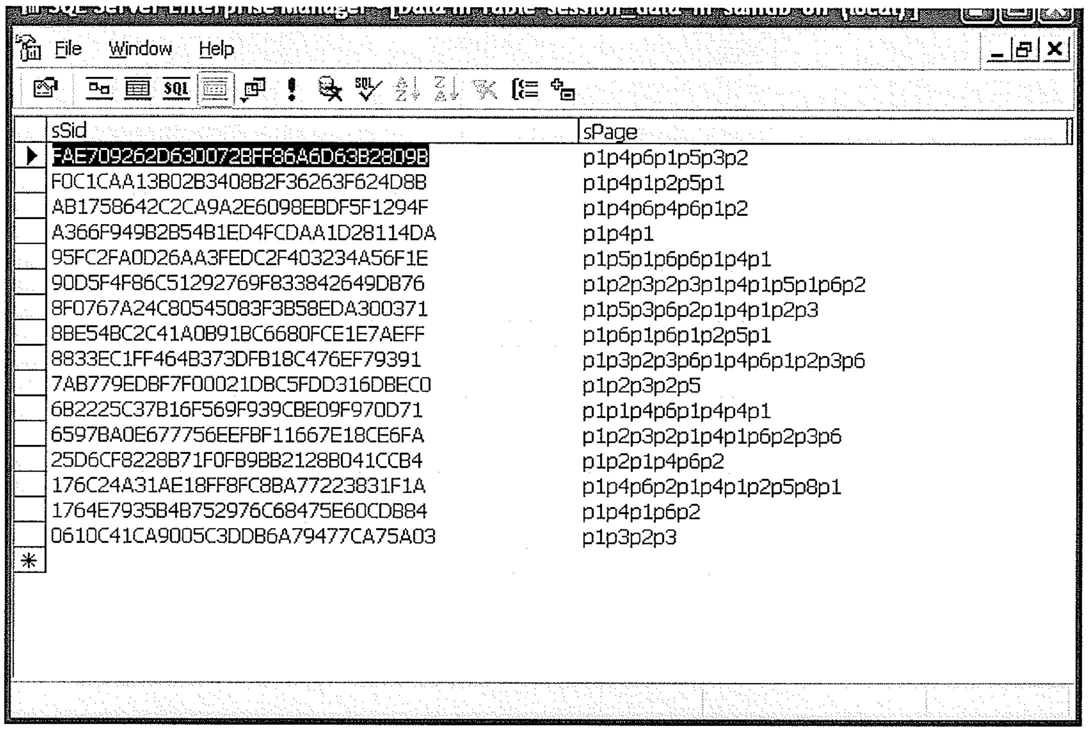


Page	Regular Links	Image Links	Total Links
page 1	http://localhost/hk/servlets/test2.html		8
	http://localhost/hk/servlets/test3.html		
	http://localhost/hk/servlets/test4.html		
	http://localhost/hk/servlets/test5.html		
	http://localhost/hk/servlets/test6.html		
	http://localhost/hk/servlet/page8Servlet		
	http://localhost/hk/servlet/ImgServlet?page=p1		
	http://www.google.ca		
page 2	http://localhost/hk/servlets/test3.html	http://localhost/hk/servlets/G1.gif	5
	http://localhost/hk/servlets/test5.html		
	http://localhost/hk/servlets/test1.html		
	http://localhost/hk/servlet/ImgServlet?page=p2		
page 3	http://localhost/hk/servlets/test2.html	http://localhost/hk/servlets/G2.gif	6
	http://localhost/hk/servlets/test6.html	http://localhost/hk/servlets/G3.gif	
	http://localhost/hk/servlets/test1.html		
	http://localhost/hk/servlet/ImgServlet?page=p3		
page 4	http://localhost/hk/servlets/test1.html	http://localhost/hk/servlets/G4.gif	4
	http://localhost/hk/servlets/test6.html		
	http://localhost/hk/servlet/ImgServlet?page=p4		
page 5	http://localhost/hk/servlets/test1.html		4
	http://localhost/hk/servlets/test3.html		
	http://localhost/hk/servlet/Page8Servlet		
	http://localhost/hk/servlet/ImgServelet?page=p5		
page 6	http://localhost/hk/servlets/test1.html	http://localhost/hk/servlets/G5.gif	5
	http://localhost/hk/servlets/test2.html	http://localhost/hk/servlets/G6.gif	
	http://localhost/hk/servlet/ImgServelet?page=p6		
page 7	http://localhost/hk/servlets/test1.html		
	http://localhost/hk/servlets/test5.html		3
	http://localhost/hk/servlet/ImgServlet?page=p9		
page 8	http://localhost/hk/servlets/test1.html	http://localhost/hk/servlets/G7.gif	5
	http://localhost/hk/servlets/test3.html		
	http://localhost/hk/servlets/test5.html		
	http://localhost/hk/servlet/ImgServlet?page=p8		
page 9	http://localhost/hk/servlets/test1.html		3
	http://localhost/hk/servlets/test2.html		
	http://localhost/hk/servlet/ImgServlet?page=p9		

Figure 4.20: Page Content Analysis

- **Path Analysis**

As presented in Chapter 3, this thesis uses a distinctive way to show the path in any session. Path analysis is mainly based on the data tracked by single-pixel image approach. On one hand, single-pixel image approach tracks users by page views, which simplify data process; on the other hand, single-pixel image approach can control no-cache on the client-side, which offers complete records of the pages browsed by a user than the other two approaches. Figure 4.21 shows path strings of each user's session.

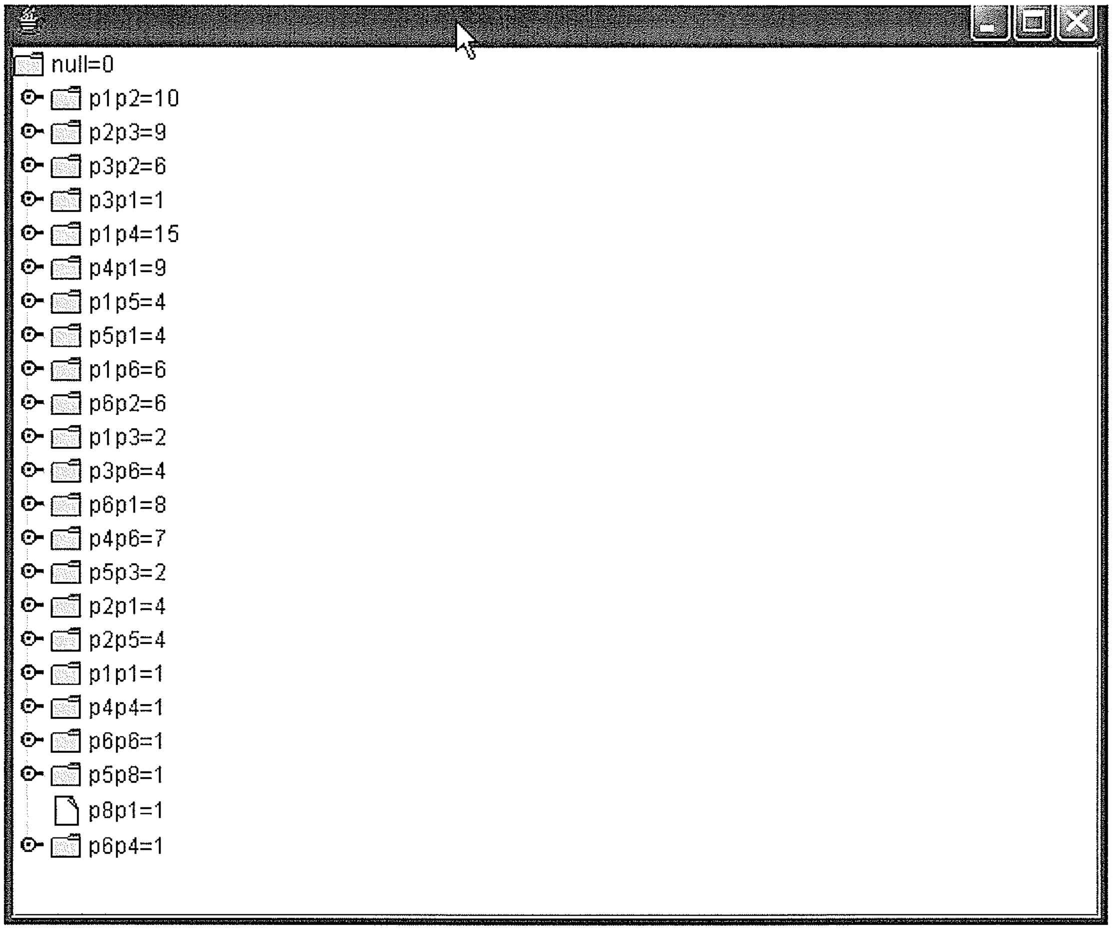


sSid	sPage
FAE709262D630072BFF86A6D63B2809B	p1p4p6p1p5p3p2
F0C1CAA13B02B3408B2F36263F624D8B	p1p4p1p2p5p1
AB1758642C2CA9A2E6098EBDF5F1294F	p1p4p6p4p6p1p2
A366F949B2B54B1ED4FCDAA1D28114DA	p1p4p1
95FC2FA0D26AA3FEDC2F403234A56F1E	p1p5p1p6p6p1p4p1
90D5F4F86C51292769F833842649DB76	p1p2p3p2p3p1p4p1p5p1p6p2
8F0767A24C80545083F3B58EDA300371	p1p5p3p6p2p1p4p1p2p3
8BE54BC2C41A0B91BC6680FCE1E7AEFF	p1p6p1p6p1p2p5p1
8833EC1FF464B373DFB18C476EF79391	p1p3p2p3p6p1p4p6p1p2p3p6
7AB779EDBF7F00021DBC5FDD316DBEC0	p1p2p3p2p5
6B2225C37B16F569F939CBE09F970D71	p1p1p4p6p1p4p4p1
6597BA0E677756EEFBF11667E18CE6FA	p1p2p3p2p1p4p1p6p2p3p6
25D6CF8228B71F0FB9BB2128B041CCB4	p1p2p1p4p6p2
176C24A31AE18FF8FC8BA77223831F1A	p1p4p6p2p1p4p1p2p5p8p1
1764E7935B4B752976C68475E60CDB84	p1p4p1p6p2
0610C41CA9005C3DDB6A79477CA75A03	p1p3p2p3

Figure 4.21: Path Analysis Result

Path analysis is an effective way to know users' browsing behaviors. A path string reveals two thing for each session data: how many pages in this site the user has browsed and what path the user browsed in this session. For example, in the first session, i.e., the first line in Figure 4.21, the user browsed 6 pages. The user started at p1 and followed the order (p1 $\rightarrow$ p4 $\rightarrow$ p6 $\rightarrow$ p1$\rightarrow$ p5 $\rightarrow$ p3 $\rightarrow$ p2) to browse, and then exit at page 2. When considering all data from all tracked sessions, we can also find how many pages have been requested in a certain time span, and which paths are usually followed by most users. From Figure 4.21, each user starts at page 1, which, in fact, is the homepage of this site. Also, notice that 6 out of 16 sessions starting path is p1 $\rightarrow$ p4.

Each of the above paths is a node of a path tree. Figure 4.22 shows the path tree built on the paths from Figure 4.21.

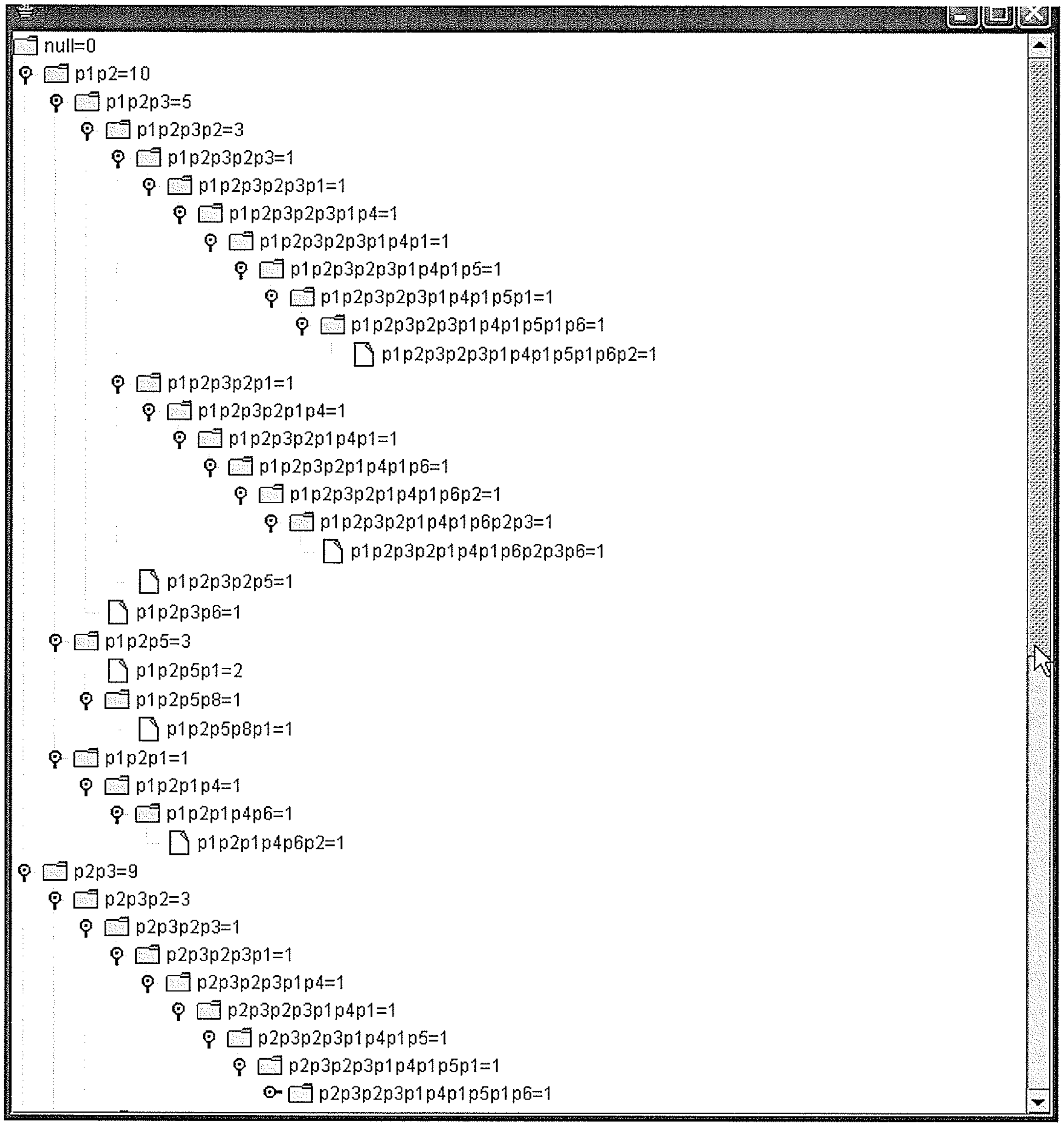


```
null=0
  p1p2=10
  p2p3=9
  p3p2=6
  p3p1=1
  p1p4=15
  p4p1=9
  p1p5=4
  p5p1=4
  p1p6=6
  p6p2=6
  p1p3=2
  p3p6=4
  p6p1=8
  p4p6=7
  p5p3=2
  p2p1=4
  p2p5=4
  p1p1=1
  p4p4=1
  p6p6=1
  p5p8=1
  p8p1=1
  p6p4=1
```

Figure 4:22: The Path Tree with Collapsed Branch Nodes

The top of the tree is the root node with the values (null, 0). The rest are sub-nodes or leaves, which follow the format (path string = occurrence times) as described previously in Chapter 3. To see more details of this tree, we expand all branch nodes, which are partially (too big to display here) shown in Figure 4.23.

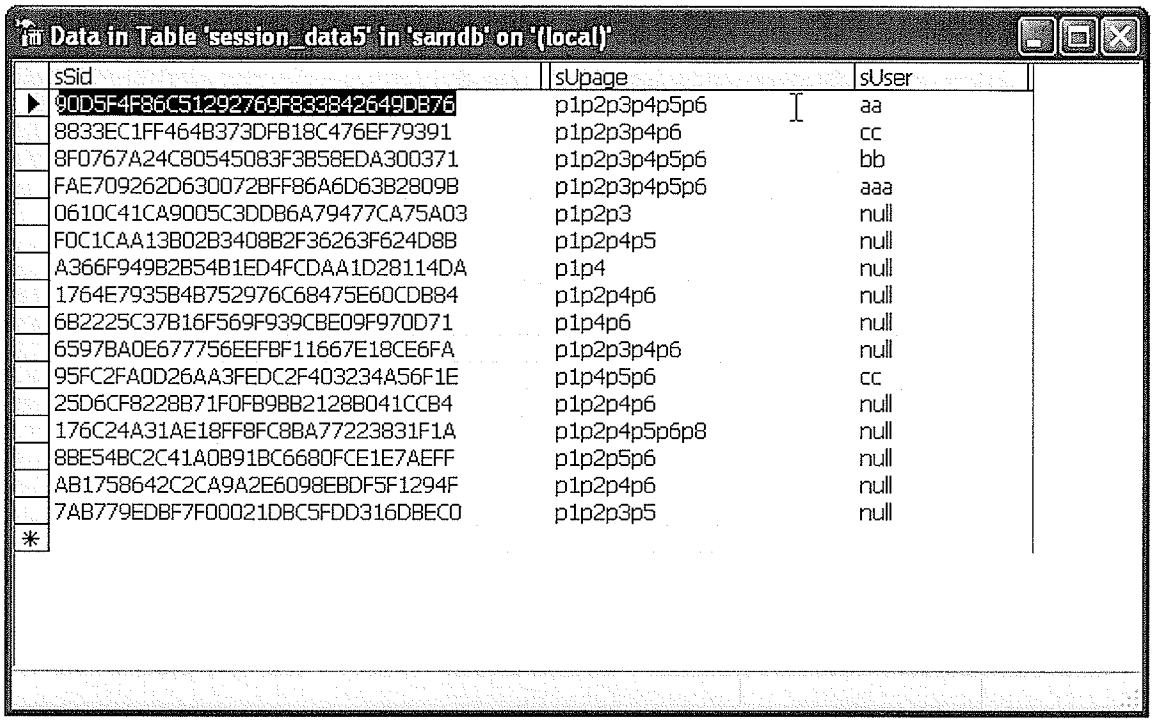
From the path tree, we can easily know which path is most used by the users to navigate the site. For example, in this case, the path "p1p4" occurred 15 times. This information is important for the site owner or the business to manage the site and develop marketing strategies. One example is to put the product advertisements on those pages, such as page 4, where users usually go.



```
null=0
  p1p2=10
    p1p2p3=5
      p1p2p3p2=3
        p1p2p3p2p3=1
          p1p2p3p2p3p1=1
            p1p2p3p2p3p1p4=1
              p1p2p3p2p3p1p4p1=1
                p1p2p3p2p3p1p4p1p5=1
                  p1p2p3p2p3p1p4p1p5p1=1
                    p1p2p3p2p3p1p4p1p5p1p6=1
                      p1p2p3p2p3p1p4p1p5p1p6p2=1
        p1p2p3p2p1=1
          p1p2p3p2p1p4=1
            p1p2p3p2p1p4p1=1
              p1p2p3p2p1p4p1p6=1
                p1p2p3p2p1p4p1p6p2=1
                  p1p2p3p2p1p4p1p6p2p3=1
                    p1p2p3p2p1p4p1p6p2p3p6=1
        p1p2p3p2p5=1
      p1p2p3p6=1
    p1p2p5=3
      p1p2p5p1=2
      p1p2p5p8=1
        p1p2p5p8p1=1
    p1p2p1=1
      p1p2p1p4=1
        p1p2p1p4p6=1
          p1p2p1p4p6p2=1
  p2p3=9
    p2p3p2=3
      p2p3p2p3=1
        p2p3p2p3p1=1
          p2p3p2p3p1p4=1
            p2p3p2p3p1p4p1=1
              p2p3p2p3p1p4p1p5=1
                p2p3p2p3p1p4p1p5p1=1
                  p2p3p2p3p1p4p1p5p1p6=1
```

Figure 4.23: The Path Tree with Expanded Branch Nodes

Business activities target one or more user groups. Clustering is one way to identify user groups. To identify whether a user belongs to a user group or not, we need know whether the user has the same interest as those in that group. To find a user's interest, we need only care about whether or not a user came to a Web page, not how the user came. Based on this view, instead of using path string, page string, which includes all unique pages in a session, is used in the clustering analysis. Figure 4.24 shows the page string for clustering analysis.


Data in Table 'session_data5' in 'samdb' on '(local)'

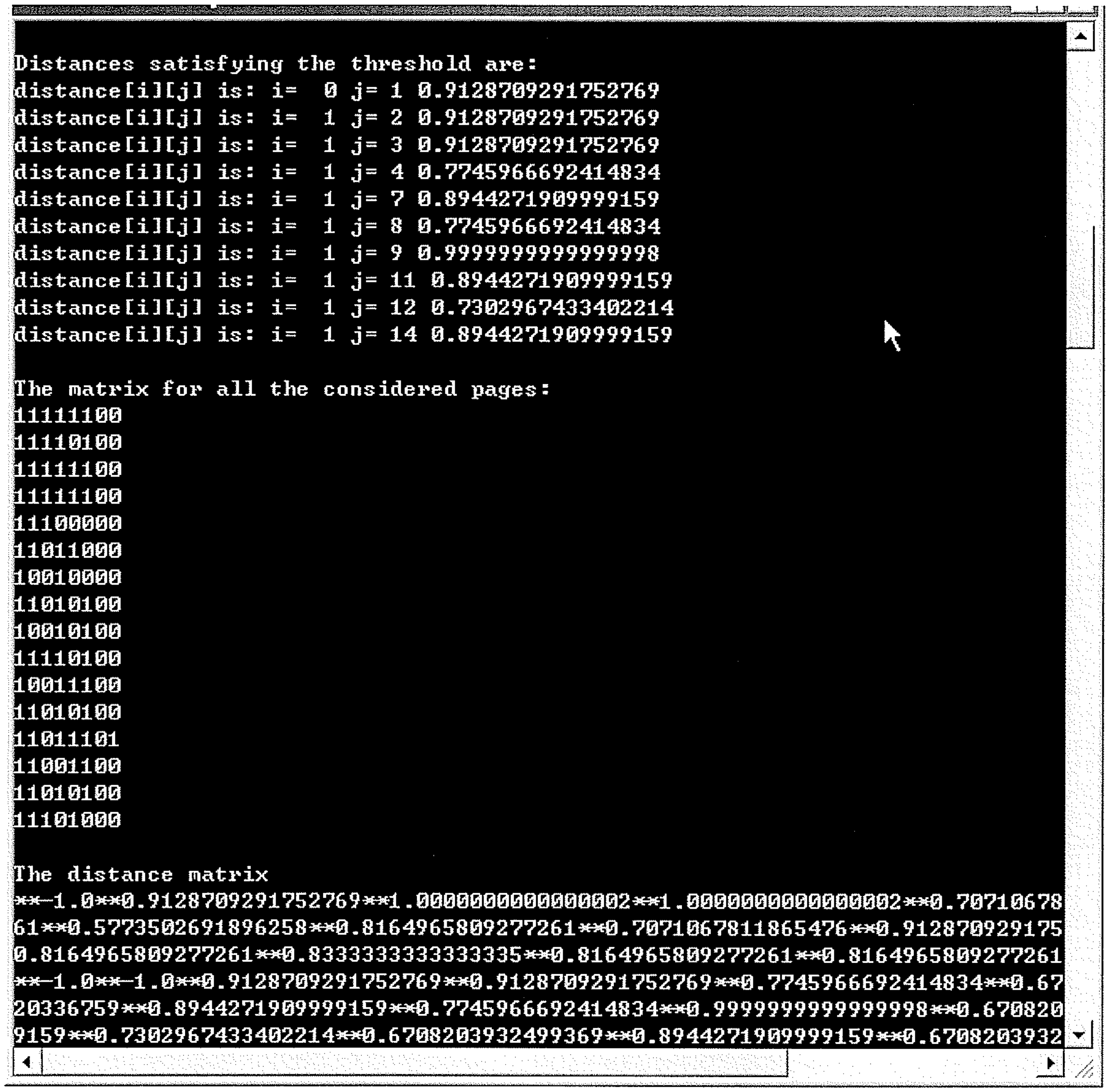
sSid	sUpage	sUser
90D5F4F86C51292769F833842649DB76	p1p2p3p4p5p6	aa
8833EC1FF464B373DFB18C476EF79391	p1p2p3p4p6	cc
8F0767A24C80545083F3B58EDA300371	p1p2p3p4p5p6	bb
FAE709262D630072BFF86A6D63B2809B	p1p2p3p4p5p6	aaa
0610C41CA9005C3DDB6A79477CA75A03	p1p2p3	null
F0C1CAA13B02B3408B2F36263F624D8B	p1p2p4p5	null
A366F949B2B54B1ED4FCDAA1D28114DA	p1p4	null
1764E7935B4B752976C68475E60CDB84	p1p2p4p6	null
6B2225C37B16F569F939CBE09F970D71	p1p4p6	null
6597BA0E677756EEFBF11667E18CE6FA	p1p2p3p4p6	null
95FC2FA0D26AA3FEDC2F403234A56F1E	p1p4p5p6	cc
25D6CF8228B71F0FB9BB2128B041CCB4	p1p2p4p6	null
176C24A31AE18FF8FC8BA77223831F1A	p1p2p4p5p6p8	null
8BE54BC2C41A0B91BC6680FCE1E7AEFF	p1p2p5p6	null
AB1758642C2CA9A2E6098EBDF5F1294F	p1p2p4p6	null
7AB779EDBF7F00021DBC5FDD316DBEC0	p1p2p3p5	null

Figure 4.24: Unique Pages in Each Session

- **Clustering Users**

Similar browsing behavior comes from similar interests. To group users would make the business to recommend some products to user A if user B has interest in that product and user A and user B are in the same group. To group users, clustering is used. Clustering operation is based on session data. Obviously, unique pages browsed by a user in each session provide more information of the user' interest than path taken by the user.

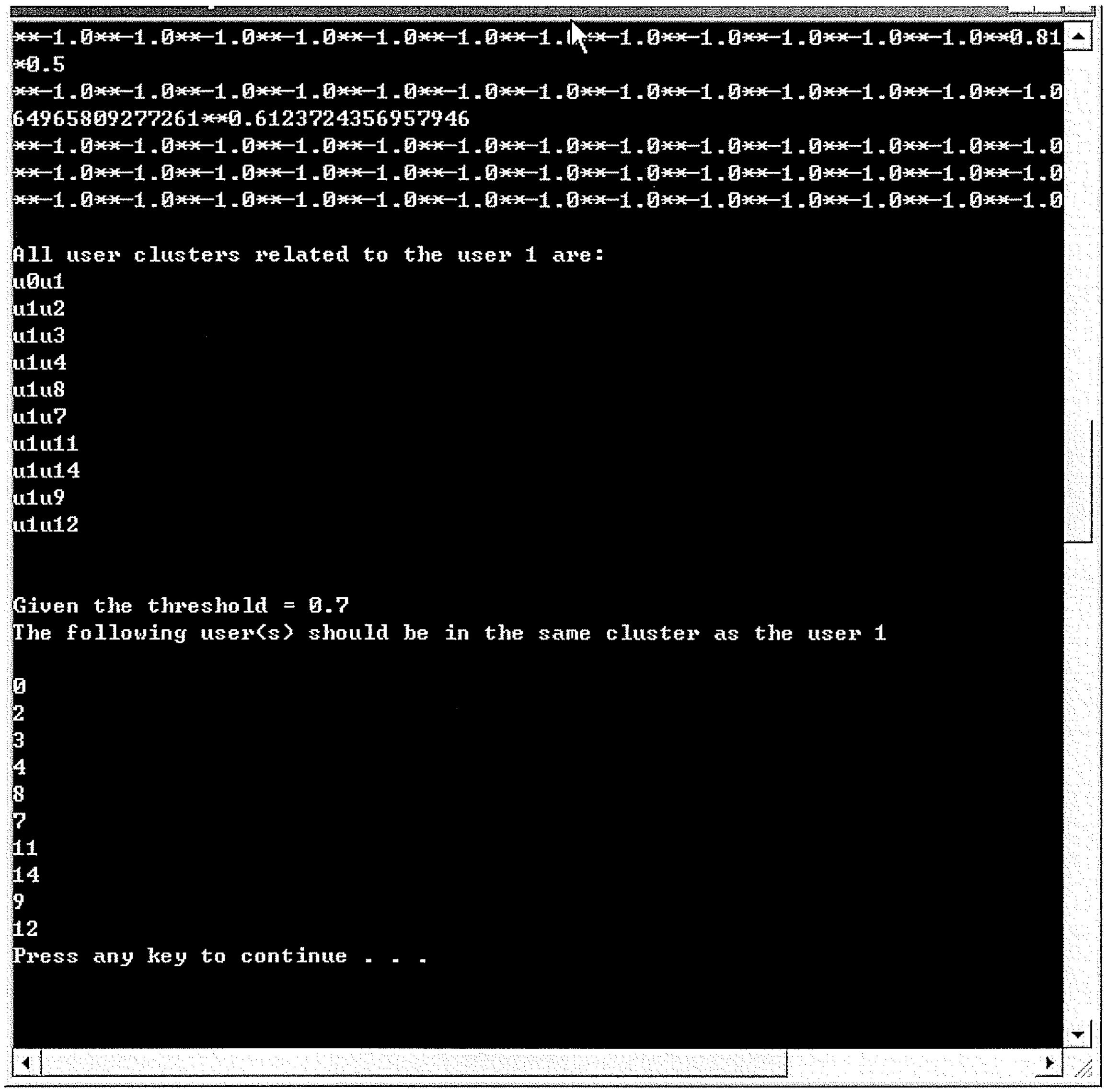
User clustering process involves finding distances between user transactions, formatting user transactions, and setting clustering threshold. Figure 4.25 shows the process to find all other users who share the similar interest with user cc (see Figure 4.24).



```
Distances satisfying the threshold are:
distance[i][j] is: i=  0 j= 1 0.9128709291752769
distance[i][j] is: i=  1 j= 2 0.9128709291752769
distance[i][j] is: i=  1 j= 3 0.9128709291752769
distance[i][j] is: i=  1 j= 4 0.7745966692414834
distance[i][j] is: i=  1 j= 7 0.8944271909999159
distance[i][j] is: i=  1 j= 8 0.7745966692414834
distance[i][j] is: i=  1 j= 9 0.9999999999999998
distance[i][j] is: i=  1 j= 11 0.8944271909999159
distance[i][j] is: i=  1 j= 12 0.7302967433402214
distance[i][j] is: i=  1 j= 14 0.8944271909999159

The matrix for all the considered pages:
11111100
11110100
11111100
11111100
11100000
11011000
10010000
11010100
10010100
11110100
10011100
11010100
11011101
11001100
11010100
11101000

The distance matrix
**-1.0**0.9128709291752769**1.0000000000000002**1.0000000000000002**0.70710678
61**0.5773502691896258**0.8164965809277261**0.7071067811865476**0.912870929175
0.8164965809277261**0.8333333333333335**0.8164965809277261**0.8164965809277261
**-1.0**-1.0**0.9128709291752769**0.9128709291752769**0.7745966692414834**0.67
20336759**0.8944271909999159**0.7745966692414834**0.9999999999999998**0.670820
9159**0.7302967433402214**0.6708203932499369**0.8944271909999159**0.6708203932
```

Figure 4.25: Clustering Users (1)

After clustering, a user's can be identified with a group based on the set threshold. Figure 4.26 shows the result of clustering operation to identify the group for user cc.



```
**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**0.81
*0.5
**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0
64965809277261**0.6123724356957946
**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0
**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0
**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0**-1.0

All user clusters related to the user 1 are:
u0u1
u1u2
u1u3
u1u4
u1u8
u1u7
u1u11
u1u14
u1u9
u1u12


Given the threshold = 0.7
The following user(s) should be in the same cluster as the user 1

0
2
3
4
8
7
11
14
9
12
Press any key to continue . . .
```

Figure 4.26: Clustering Users (2)

From Figure 4.26, when we set cluster threshold to 0.7, we find that user 1 should be in the user group with following users [0, 2, 3, 4, 7,8, 9, 11, 12,14] (see Figure 4.24), where the users are indexed with the rows, which start with 0.

4.7.2.3 More Advanced Analysis

- **How many people made purchases among the visitors?**

This question can be answered by checking the data tracked by single-pixel image method. Suppose page 8 is the confirmation page for purchase of this site, we know that only one user in one session arrived at that page, since page 8 occurred once in all sessions shown in Figure 4.21. Therefore, we can conclude that only one user made purchase among those 16 visitors.

- **What is user A's interest?**

Users' browsing behaviors reflect interests. User A's interest can be found by other members in the same cluster from the cluster analysis, which puts users with similar interest together. For example, if we know that another user B is interested in the information on page 5, we may infer that user A is interested in that information too if user A and user B are in the same user cluster. Another way to find user A's interest is to analyze the case individually. For example, suppose user A is the user who created the fifth record in Figure 4.21, we can reason that user A must have been interested in something on page 2, since the user visited that page three times during that session.

- **Which referrer results in the most purchases, not just hits?**

To answer this question, we must know whether a user have made a purchase or not. If yes, the referrer page of the purchase is the page we need to find. From the answer to the question, "How many people made purchases among the visitors?" we know only one user made a purchase. Then, we check the path tree and find that "p6" (i.e., page 6) is the referrer page for page 8 in that session. We can conclude that the page 6 resulted in that one purchase. In the case of several purchases, a group of referrer pages exist. Each referrer page in this group resulted in a purchase. The referrer page, which has the most occur times in the group results in the most purchases.

Chapter 5

Conclusion and Future Work

This work introduces the development and design of a tracking and analysis tool for e-commerce sites. This tool is targeted towards helping e-commerce organizations know more about their users in order to develop efficient marketing strategies. The main focus of the study is on the theory and approaches to track online user and analyze the tracked data. In brief, the study consists of five aspects: (1) a review of the present traffic tracking and analysis methods, and the description of online activities tracking and analysis techniques necessary for an e-commerce; (2) major challenges in developing such tool; (3) theory and methods for design of this tool; (4) evaluation of the design using formal specification; and (5) implementation of this tool.

The following sections present the contributions of this thesis and suggestions for future work.

5.1 Contributions

The design of the tracking and analysis tool in this thesis contributes to knowledge in both the tracking and analysis parts as follows:

- **User Activity Tracking:**

1. The combination of three tracking approaches (i.e., improved single-pixel approach, JavaScript approach, and HTTP proxy server) allows a user to be always tracked by at least one approach.

2. The improved single-pixel approach has more distinctive advantages over the general single-pixel technique, which requires JavaScript support in the user's browser. The improved single-pixel approach does not require JavaScript support to function.

3. Two main features make the JavaScript approach more effective and different. One is that it can track most keyboard and mouse events of online users. The user of this tool can configure what kind of events about a user should be tracked. The other feature is that it is capable of tracking any form input on the page.

4. The improved single-pixel and JavaScript approaches are designed to work on both static and dynamic Web pages.

5. In both the improved single-pixel approach and JavaScript approach, using Java servlet technology allows these approaches to be more compatible with many platforms than CGI programs which most tools use; and more Java servlet session control enables an easy control of users' sessions. Using Java servlet session technology results in a comparatively simple process of analyzing the tracked data later.

- **The Analysis of Tracked Data:**

1. A straightforward way to represent a path and an algorithm to build a path tree for all users' sessions of a site have been developed and the algorithm implemented. Path analysis is one of the most important approaches for Web site management and marketing in e-commerce. The path analysis support / provides statistical analyses or forecasts of the usage of the Web site.

2. Page contents (i.e., links and images) and clustering analysis are implemented. These analyses, together with path analysis, form the basis of finding users' interests, and developing personalization and recommendation for users.

3. In addition to the answers to the questions in basic analysis for traffic analysis, most of the advanced and more advanced questions are answered in the analysis.

5.2 Future Work

Potential future work of this thesis include:

1. Use a realistic e-commerce site to test this tool. Although implementation of this tool displays what we expected, we realize a software can behave very differently from testing results when it is used in a real environment.

2. Develop recommendation and personalization based on path analysis, page content analysis, and cluster analysis. Suppose we find from the path analysis that the page 3 (i.e., p3) usually follows a path p1p2. We can recommend the page 3 to a user when the user has finished browsing p1p2. Another way to do personalization is to combine path analysis with user clusters. We can recommend user B's path to user A if A and B are in the same group.

3. Do more analyses on the tracked data. One potential analysis is clustering analysis on path. Instead of creating user clusters, we create path clusters. If such path clusters can be found, we can develop Web recommendation in a different way than user clusters. For example, suppose {p1p2p3, p2p3p4} is such a cluster. If a user has finished page p1, p2, and p3 in that order (i.e., the finished path is p1p2p3), we can recommend page 4 to that user. In this case, the resulting path is p1p2p3p4, which can be thought of as p1→ p2p3p4, while p2p3p4 is in the same cluster as the finished path p1p2p3. Besides path analyses, more analyses on the data about users' events can be done, since we have tracked all registered user events.

4. Address privacy issues in a user friendly way. Web traffic analysis tools can enhance a Web site's efficiency through the analysis of the users' data, although more and more people are concerned about privacy issues today. Users tracking can benefit both the Web users and the business (the owner of the Web site). The business can develop and sell more products while the users can have easy navigation through the site and purchase customized products. The identity of individual users is not important to other analyses except for the purpose of

profiling the users in order to offer customized product or service offerings. To address privacy concerns in a user friendly way, the users should know what is being tracked about them. Also, the tracking process should not interrupt the users' normal navigation so the tracking is done in the background. To keep the tracking open to the users, P3P should be deployed on the Web server and configured to communicate with the uses' browsers. A user's browser can be configured to request a Web site's privacy policy, store the user's privacy preferences and check these preferences against the site policy before any processing. If the Web site's privacy policy does not match those preferred by the user, the user will be warned [86]. In this way, the user has the choice whether he/she can be tracked, what data can be tracked and how the tracked data can be kept and used.

5. Provide end-user friendly marketing support tools. The Web tracking and analysis tool developed in this thesis is mainly for the purpose of developing marketing strategies in e-commerce. The main focus of this thesis is on the research and application of developing such tool; end-user interface is ignored during the design and implementation. However, as an application software, an end-user friendly interface will be attractive and indispensable to the tool users. Since more than 90% Internet users currently use Windows operating system, a user interface compatible with Windows operating systems should be provided. Although Microsoft provides a lot of tools for interface design, considering the user interface will communicate with those server-side analysis programs, which use Java technology, Java Swing technology can be used in the future's interface design for this tool. Also, the "look and feel" feature makes Java Swing technology a powerful tool for user interface design.

6. Extend page content mining, such as mining text data from Web pages.

7. Extend the proxy server to handle other protocols, such as https.

References:

[1] *Building Confidence*, United Nations Conference on Trade and Development (UNCTAD), February 2000, pp.18.

[2] Malacinski A., Dominick S., and Hartrick T., "Measuring Web Traffic, Part 1 and Part 2", *DeveloperWorks*, IBM Corporation, March 2001.
(See http://www-106.ibm.com/developerworks/Web/library/wa-mwt1 and http://www-106.ibm.com/developerworks/Web/library/wa-mwt2.)

[3] "Driving Business Decisions in Web Time", *White Paper*, Accrue Software, Inc., March 2000.

[4] "Analyzing Web Site Traffic", *White Paper*, Sane Solutions, LLC., 2002.

[5] McClure M., "Web Traffic Analysis Software", *White Paper*, Accrue Software, Inc., 1999.

[6] Cunningham M., *Smart Things to Know about E-commerce*, Capstone, 2000.

[7] Berst J., *The Magnet Effect*, McGraw-Hill, 2000.

[8] Siebel T. M. and House P., *Cyber Rules: Strategies for Excelling at E-Business*, Doubleday, 1999.

[9] Korper S. and Ellis J., *The E-commerce Book: Building the E-Empire*, Academic Press, 1999.

[10] Deise M. V., Nowikow C., King P., and Wright A., *Executive's Guide to E-Business*, John Wiley & Sons, 1999.

[11] Alagar V. S. and Periyasamy K., *Specification of Software System*, Spring-Verlag, 1998.

[12] Ehikioya S. A., *Specification of Transaction Systems Protocol*, Ph.D. Thesis, Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, September 1997.

[13] Ehikioya S. A. and Barker K. E., "Towards a Formal Specification Methodology for Transaction Systems Protocol", *3rd Annual IASTED International Conference on Software Engineering and Applications (SEA'99)*, Scottsdale, Arizona, USA, October 6-8, 1999.

[14] Gamse K., "Mining for Gold in Your Web Traffic Logs", A Web Article on Internet Business. (Available at http://www.cyberspeaker.com/mininglogs.html.)

[15] Krishnamurthy B. and Rexford J., "Software Issues in Characterizing Web Server Logs", *W3C Consortium Workshop on Web Characterization*, Cambridge, MA, November 1998.

[16] Davison B. D., "Web Traffic Logs: An Imperfect Resource for Evaluation", *Proceedings of 9th Annual Conf. of the Internet Society*, June 1999.

[17] Morris R. and Lin D., "Variance of Aggregated Web Traffic", *Proceedings of IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000.

[18] Mah B., "An Empirical Model of HTTP Network Traffic", *INFOCOM 97*, Kobe, Japan, April 1997.

[19] Feldmann A., Greenberg A., Lund C., Reingold N., Rexford J., and True F., "Continuous Online Extraction of HTTP Traces from Packet Traces", *W3C Workshop on Web Characterization*, November 1998.

[20] Lamm S. E. and Reed D. A., "Real-Time Geographic Visualization of World Wide Web Traffic", *Electronic Proceedings of the Fifth International World Wide Web Conference*, Paris, France, May 6-10, 1996.

[21] Crovella M. E. and Bestavros A., "Explaining World Wide Web Traffic Self-Similarity", *Technical Report TR-95-015*, Computer Science Dept., Boston University, October 1995.

[22] Schechter S., Krishnan M., and Smith M. D., "Using Path Profiles to Predict HTTP Requests", *Proceedings of 7th International World Wide Web Conference*, Brisbane, Australia, 1998.

[23] Cunha C., Bestavros A., and Crovella M. E., "Characteristics of WWW Client-based Traces", *Technical Report TR-95-010*, Computer Science Dept., Boston University, Boston, MA, 1995.

[24] Masseglia F., Poncelet P., and Teisseire M., "Using Data Mining Techniques on Web Access Logs to Dynamically Improve Hypertext Structure", *ACM SigWeb Letters*, 8(3):13-19, October 1999.

[25] Zaiane O. R., Xin M., and Han J., "Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs", *Advances in Digital Libraries*, Santa Barbara,1998, pp. 19-29.

[26] Theusinger C. and Huber K., "Analyzing the Footsteps of Your Customers", *WEB Knowledge Discovery in Databases (WEBKDD)*, Boston, MA, USA, August 2000.

[27] Eick S. G., "eBinessness Performance Analysis", *Superiore G. Reiss Romoli (SSGRR)*, L'Aquila, Itality, Jul 31 - Aug 06, 2000..

[28] Shen L., Cheng L., Ford J., Makedon F., Megalooikonomou V. and Steinberg T., "Mining the Most Interesting Web Access Associations", *Proceedings of the World Conference on the WWW and Internet (WebNet)*, San Antonio, Texas, November 2000, pp. 489-494.

[29] Berendt B., "Web Usage Mining, Site Semantics, and the Support of Navigation", *WEB Knowledge Discovery in Databases (WEBKDD)*, Boston, MA, USA, August 2000.

[30] Schafer J. B., Konstan J. A., and Riedl J., "E-commerce Recommendation Applications", *Data Mining and Knowledge Discovery*, 2001.

[31] Sarwar B., Karypis G., Konstan J., and Reidl J., "Analysis of Recommendation Algorithms for E-commerce", *Proceedings of the ACM Conference on E-commerce (EC00)*, Minneapolis, October 2000.

[32] Glover E. J, Lawrence S., Gordon M. D., Birmingham W. P., and Giles C. L., "Recommending Web Documents Based on User Preferences", *ACM SIGIR '99 Workshop on Recommender Systems*, University of California, Berkeley, August,1999.

[33] *Unified Modeling Language (UML) Verson 1.3a*, Object Management Group, 1999. (See http://www.rational.com/uml/index.jsp.)

[34] Smith G., *The Object-Z Specification Language*, Kluwer Academic Publishers, 2000.

[35] Myllymaki J. and Jackson J., "Web-based Data Mining, Automatically Extract Information with HTML, XML, and Java", *DeveloperWorks*, IBM Corporation, June 2001.

[36] Hammer J., Garcia-Molina H., Cho J., Crespo A., and Aranha R., "Extracting Semistructured Information from the Web*", Proceedings of the Workshop on Management of Semistructured Data*, Tucson, Arizona, May 1997.

[37] Bergholz A. and Freytag J. C., "A Three-layer Approach to Semistructured Data", *Proceedings of the International Workshop on Theory and Application of Graph Transformations (TAGT)*, Paderborn, Germany, November 1998.

[38] Soderland S., "Learning to Extract Text-based Information from the World Wide Web", *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 1997.

[39] Hall M., *Core Servlets and JavaServer Pages*, Sun Microsystems Press, August 2000, pp 181.

[40] "Assessing Web Site Usability from Server Log Files", *White Paper,* Tec-Ed, Inc., December 1999.

[41] "Online Visitor Traffic Analysis", *White Paper,* Audientia, 2000.

[42] Wilson T., "Web Traffic Analysis Turns Management Data to Business Data", *TechWeb Network*, April 2, 1999.

[43] Drott M.C., "Using Web Server Logs to Improve Site Design", *Proceeding on the Sixteenth Annual International Conference on Computer Documentation*, Association for Computing Machinery (ACM), September 23-26, 1998, Quebec City, Canada, Page 43-50.

[44] Pei, J., Han, J., Mortazavi-asl, B., and Zhu, H., "Mining Access Patterns Efficiently from Web logs", *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2000).*

[45] *Illustrated Handbook for Web Management*, The Office of e-Envoy of UK Government, May 2002.
(Available at http://www.e-envoy.gov.uk/assetRoot/04/00/06/57/04000657.pdf.)

[46] Ackerman S. M., Cranor L. F. and Reagle J., "Beyond Concern: Understanding Net Users' Attitudes About Online Privacy", *AT&T Labs-Research Technical Report TR 99.4.3,* AT&T Labs-Research, April 14,1999.
(Available at http://www.research.att.com/projects/privacystudy.)

[47] "Privacy Notice Research, Final Results", Harris Interactive, Inc 2001.
(Available at http://www.ftc.gov/bcp/workshops/glb/supporting/harris%20results.pdf.)

[48] Catledge L. and Pitkow J., "Characterizing Browsing Behaviors on the World Wide Web", *Computer Networks and ISDN Systems*, 27(6), 1995.

[49] Cooley R., Srivastava J., and Mobasher, B. "Data Preparation for Mining World Wide Web Browsing Patterns", *Journal of Knowledge and Information Systems*, vol. 1, no. 1, pp. 5 -32,1999.

[50] Eick S. G., "eBusiness Performance Analysis", *Superiore G. Reiss Romoli (SSGRR2000),* L'Aquila, Itality, Jul 31 - Aug 06, 2000.

[51] Schechter S., Krishnan M., and Smith M. D., "Using Path Profiles to Predict HTTP Requests", *Proceedings of the Seventh International World Wide Web Conference,* Brisbane, Australia, April 1998.

[52] Mannila H., Toivonen H., and Verkamo A. I., "Discovering Frequent Episodes in Sequences", *Proceedings of the 1st Int'l Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210-215, Montreal, Quebec, 1995.

[53] Srikant R. and Agrawal R., “Mining Sequential Patterns: Generalizations and Performance Improvements”, *Proceedings of the 5th International Conference on Extending Database Technology (EDBT’96)*, pages 317, Avignon, France, March 1996.

[54] Theusinger C. and Huber K., “Analyzing the Footsteps of Your Customers”, *The 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mniing,* Boston, MA, USA, August 20, 2000.

[55] Mobasher B, Dai H, Luo T, Sun Y, and Zhu J. , "Integrating Web Usage and Content Mining for More Effective Personalization". *Proceedings of the International Conference on E-commerce and Web Technologies (ECWeb2000)*, Greenwich, UK, 2000.

[56] Mobasher B., Cooley R., and Srivastava J., “Automatic Personalization Based on Web Usage Mining”, *Communications of the ACM*, 43(8):142--151, 2000.

[57] “Web Site Personalization”, *IBM High-Volume Web Site Team*, January 2000.

[58] Sarwar B., Karypis. G., Konstan J., and Riedl J., “Analysis of Recommendation Algorithms for E-Commerce”, *Proceedings of the ACM Conference on E-Commerce (EC00)*, Minneapolis, October 2000.

[59] Han E.H., Karypis G., Kumar V., and Mobasher B., “Clustering Based On Association Rule Hypergraphs”, *Proceedings of SIGMOD'97 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'97)*, May 1997.

[60] Shen L., Cheng L., Ford J., Makedon F., Megalooikonomou V., and Steinberg T., “Mining the Most Interesting Web Access Associations*”, World Conference on the WWW and Internet (WEBNETC)*, 2000.

[61] Hill W., Stead L., Rosenstein M., and Furnas G., “Recommending and Evaluating Choices in a Virtual Community of Use”, *Proceedings of the Conference on Human Factors in Computing Systems (CHI95)*, pp 1942, 1995.

[62] Konstan J., Miller B., Maltz D., Herlocker J., Gordon L., and Ridl J., “GroupLens: Applying Collaborative Filtering to Usenet News”, *Communications of the ACM*, 40(3), pp.77-87, 1997.

[63] Resnick P., Iacovou N., Suchak M., Bergstrom P., and Riedl J., “GroupLens: An Open Architecture for Collaborative Filtering of Net News”, *Proceedings, of CSCW’94*, Chapel Hill, NC, 1994.

[64] Herlocker J., Konstan J., Borchers A., and Riedl, J., “An Algorithmic Framework Performing Collaborative Filtering”, *Proceeding of the 1999 Conference on Research and Development in Information Retrieval*, August, 1999.

[65] Spiliopoulou M. and Faulstich L.C., "WUM: A Web Utilization Miner", *Proceeding of International Conference on Extending Database Technology (EDBT) Workshop WebDB98*, Valencia, Spain, LNCS 1590, Springer Verlag, 1999.

[66] Buchner A.G.. and Mulvenna M.D., "Discovering Internet Marketing Intelligence Through Online Analytical Web Usage Mining", *SIGMOD Record*, (4) 27, 1999.

[67] Pei J., Han J., Mortazavi-asl B., and Zhu H., "Mining Access Patterns Efficiently from Web", *Proceedings Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00)*, pp. 396-407, Kyoto, Japan, April 2000.

[68] Shahabi C., Zarkesh A.M., Adibi J., and Shah V., "Knowledge Discovery from Users Webpage Navigation", *Proceedings of Seventh International Workshop on Research Issues in Data Engineering*, pp. 20-29, Birmingham, England, 1997.

[69] Joshi A. and Krishnapuram R., "Robust Fuzzy Clustering Methods to Support Web Mining", *Proceedings of Workshop in Data Mining and knowledge Discovery, SIGMOD*, pp. 15-1 - 15-8, 1998.

[70] Nasraoui O., Frigui H., Joshi, A., and Krishnapuram R., "Mining Web Access Logs using Relational Competitive Fuzzy Clustering", *Proceedings of 8th International Fuzzy Systems Association World Congress (IFSA 99)*, Taipei, August 1999.

[71] Yan T.W., Jacobsen M., Garcia-Molina H., and Dayal U., "From User Access Patterns to Dynamic Hypertext Linking", *Proceedings of the 5th International World Wide Web Conference*, Paris, France, May 1996.

[72] Hartiga J.A., *Clustering Algorithms*, John Wiley & Sons, Inc., 1975.

[73] "IP Address Frequently Asked Questions", *BCR Online Reference Database*, Sept. 2003. (Available at http://www.bcr.org/reference/IP/ipaddFAQ.html.)

[74] http://www.thecounter.com/stats.

[75] http://www.google.com/press/zeitgeist_nov03.html.

[76] Lubling O. and Malave L., "Developing Scalable, Reliable, Business Applications with Servlets", *technical article,* Sun Microsystems, April 1998.

[77] Ramirez A. O., "Three-Tier Architecture", *Linux Journal*, July 2000. (Available at http://www.linuxjournal.com/article.php?sid=3508.)

[78] Kinshuk and Yang A., "Web-based Asynchronous Synchronous Environment for Online Learning", *United States Distance Education Association Journal*, 17 (2), 5-17, 2003.

[79] Unhelkar B., *Process Quality Assurance for UML-Based Projects*, Addison-Wesley, 2003, p27.

[80] Sommerville I., *Software Engineering (4th Edition)*, Addison-Wesley 1992, p590.

[81] Epp S., *Discrete Mathematics with Applications*, PWS Publishing Company, 1996.

[82] Clinton W.J. and Gore A. Jr., "A Framework for Global Electronic Commerce", *White paper*, the United States government, July 1, 1997.

[83] Electronic Privacy Information Center's (EPIC) site about online guide to privacy resources. (See http://www.epic.org/privacy/privacy_resources_faq.html.)

[84] World Wide Web Consortium's (W3C) site about Platform for Privacy Preferences (P3P) Project. (See http://www.w3.org/P3P.)

[85] Canada government's site about privacy legislation. (See http://www.privcom.gc.ca/legislation/index_e.asp.)

[86] Robinson S., "Consider Privacy Issues in Tracking Web Server Activity", *World Wide Web (W3C) in the Press in 2003*, December 2003. (See http://www.w3.org/Press/Articles-2003.html.)

[87] "The Freedom of Information and Protection of Privacy Act (FIPPA)", June 28, 1997. (See http://www.gov.mb.ca/chc/fippa/index.html.)

[88] "Privacy Act", 1980-81-82-83, c.111, Sch. II "1", *Office of the Privacy Commissioner of Canada*, 1983. (See http://www.privcom.gc.ca/legislation/02_07_01_01_e.asp#001.)

[89] "The Personal Information Protection and Electronic Document Act (PIPED Act)", *Office of the Privacy Commissioner of Canada, 2000.* (See http://www.privcom.gc.ca/legislation/02_06_01_e.asp.)