Proposed Protocol to Solve Discovering Hidden Web Hosts Problem

Mohamed A. Khattab, Yasser Fouad, and Osama Abu Rawash

Department of Mathematics (Computer Science), Faculty of Science, Alexandria University, Egypt

Summary
Web search engines use web crawlers that follow hyperlinks. This technique is ideal for discovering resources on the surface web but is often ineffective at finding deep web resources. The surface web is the portion of the World Wide Web that is indexed by conventional search engines whereas the deep web refers to web content that is not part of the surface web. The deep web represents a major gap in the coverage of web search engines as believed to be of a very high quality and is estimated to be several orders of magnitude larger than the surface Web. Understanding the nature of the deep web resources as being massively increased give us a conclusion that to be efficiently explored need an approach based on two main concepts, the first concept is to solve the problem from the side of web servers and the second concept is to automate the discovery process. In this paper we developed and implemented the Host List Protocol model that is depending on such approach to discover hidden web hosts and provide a way to be indexed through web search engines.

Key words:
Deep web, hidden web, invisible web, search engines, web crawlers, host list protocol, deep web resources, discover web resources, discover hidden web

1. Introduction

The World Wide Web has been considered as the largest digital library in recent years. People all over the world use the web to find all sorts of information. The web is inherently distributed and the data on the web is stored in numerous sources. Often, these sources have central points that provide search capabilities which are web search engines [1]. Search engines are building their databases through obtaining the entries from allowing authors of web sites or webmasters to submit their own web URLs or through using web crawlers. Crawlers are programs that automatically traverse the World Wide Web, retrieving pages and building a local repository of the portion of the Web that they visit. Depending on the application at hand, the pages in the repository are either used to build search indexes, or are subjected to various forms of analysis (e.g., text mining). Traditionally, crawlers have only targeted a portion of the Web called the publicly indexable Web (PIW) [2]. This refers to the set of pages reachable purely by following hypertext links, ignoring search forms, pages that require authorization or prior registration and unknown domains and hosts which is called “invisible Web”. Jill Ellsworth [3] used the term “invisible web” in 1994 to refer to websites that are not registered with any search engine. Frank Garcia [4] used the term "invisible web" in 1996 to refer to websites that are possibly reasonably designed, but the designers of these websites did not bother to register them with any of the search engines. So, no one can find them. Another early use of the term “invisible web” was in a December 1996 press release from Personal Library Software Inc. (PLS) [5], the leading supplier of search and retrieval software to the online publishing industry, describing the AT1 invisible web search tool which was considered the first tool of the second generation of web search engines. The AT1 invisible web search tool combines the best search agent of PLS and database extraction technology to offer publishers and users something they have never had before which is the ability to search for content residing in hidden databases, those large collections of documents managed by publishers not viewable by web crawlers. AT1 also allows users to create intelligent agents to search newsgroups and websites with e-mail notification of results. The first use of the specific term "deep web" occurred 2001 in a study by Michael Bergman [6] in which Bergman avoided the term "invisible web". The objectives of this study were to quantify the size and importance of the deep web, characterize the content, quality, and relevance to information seekers of deep web, discover automated means for identify deep web search sites, direct queries to them and begin the process of educating the internet searching public about this heretofore hidden and valuable information storehouse. S. Raghavan, H. Garcia-Molina [7] proposed an application specific approach to hidden web crawling which was a simple operational model of a hidden web crawler that succinctly describes the steps that a crawler must take to crawl deep web and found, it is estimated that the deep web is several orders of magnitude larger than the surface web which places the size of the hidden web (in terms of generated HTML pages) at around 500 times the size of the surface web or publicly indexable web. The deep web resources may be classified into one or more of the following categories

Manuscript received August 5, 2009
Manuscript revised August 20, 2009

Hidden web hosts – web sites that had been designed and published on the internet but not registered with any of the search engines.
Dynamic content - dynamic pages which are returned in response to a submitted query or accessed only through a form.
Unlinked content - pages which are not linked to by other pages.
Private web - sites that require registration and login.
Contextual web - pages with content varying for different access contexts (e.g. ranges of client IP addresses or previous navigation sequence).
Limited access content - sites that limit access to their pages in a technical way (e.g., using the Robots Exclusion Standard).
Scripted content - pages that are only accessible through links produced by JavaScript as well as content dynamically downloaded from web servers via Flash or AJAX solutions.
Non-HTML/text content - textual content encoded in multimedia (image or video) files or specific file formats not handled by search engines. Our interest in this paper is to minimize the burden on web search engines as being the mostly used point of view in solving the problem of deep web resources and trying to solve the problem from the side of web servers in an automated manner.

2. Related Work

One way to access the deep Web is via federated search based search engines. Search tools such as "worldwidescience.org" [8] and "science.gov" [9] are global science gateways connecting users to national and international scientific databases and portals and accelerates scientific discovery and progress by providing one stop searching of global science sources through a web based query interface. Their crawlers identify and interact with searchable databases, aiming to provide access to deep Web content. Web harvesting is another way to explore the deep web in which human crawlers are used instead of algorithmic crawlers. In the paradigm of web harvesting, humans find interesting links of the deep web that algorithmic crawlers can not find. This human-based computation technique to discover the deep web has been used by the "StumbleUpon.com" [10] service since February 2002. StumbleUpon is a web discovery service that allows its users to discover and rate web pages, photos, and videos through a web based interface. In 2005, Yahoo! made a small part of the deep web searchable by releasing Yahoo! Subscriptions [11]. Yahoo search subscriptions enable users to search access-restricted content such as news and reference sites that are normally not accessible to search engines. Some subscription websites display their full content to search engine robots so they will show up in user searches, but when they click a link from the search engine results page a login or subscription page will be shown.

Researchers have been exploring how the deep web can be crawled in an automatic fashion. Raghavan and Garcia-Molina [7] presented an architectural model for a hidden web crawler that used key terms provided by users or collected from the query interfaces to query a web form and crawls the deep web resources. Ntoulas et al [12] created a hidden web crawler that automatically generated meaningful queries to issue against search forms. Their crawler generated promising results, but the problem is far from being solved. Since a large amount of useful data and information resides in the deep web. Web search engines have begun exploring alternative methods to crawl the deep web. Google's Sitemaps protocol [13] is an easy way for webmasters to inform search engines about pages on their web sites that are available for crawling. The simplest form of the Sitemaps protocol is an XML or a TEXT file that lists URLs for a site along with additional metadata about each URL (when it was last updated, how often it usually changes, and how important it is, relative to other URLs in the site) so that search engines can more intelligently crawl the site. Web crawlers usually discover pages from links within the site and from other sites. Sitemaps supplement this data to allow crawlers that support Sitemaps to pick up all URLs in the Sitemap and learn about those URLs using the associated metadata. Using the Sitemap protocol does not guarantee that web pages are included in search engines, but provides hints for web crawlers to do a better job during crawling web site. The Apache module mod-oai [14, 15] that allows web crawlers to efficiently discover new, modified, and deleted web resources from a web server by using OAI-PMH, a protocol which is widely used in the digital libraries community. Also, the mod-oai module allows harvesters to obtain archive-ready resources from a web server. The Sitemaps protocol and mod-oai module are mechanisms that allow search engines and other interested parties to discover deep web resources on particular web servers. Both mechanisms allow web servers to advertise the URLs that are accessible on them, thereby allowing discovery of resources that are not directly linked to the surface web.

Reviewing the proposed methods will give us a conclusion that all solutions depending on collecting data from users through web based interface, behavior of users and ratings, specialized harvesting technique like mod-oai or depending on the webmaster of the site like Sitemap Protocol which still did not provide a way to discover hidden web hosts. Most of the provided solutions to discover deep web resources focus on the web search engine point of view and try to solve the problem from the side of web search engine.

3. Specification and Design of the Host List Protocol Model

Sitemaps Protocol logically considered being the second phase of our Host-List Protocol as we need first of all to have a frontier of unknown sites or hidden hosts then provide a way to help crawlers of the web search engines learning about the entire links of the site to be crawled more intelligently. The role of the model is informing web search engines about unknown sites or hidden hosts in an easy and automated manner by proposing a protocol called Host-List Protocol (HLP), a new developed protocol, to enhance discovering techniques of deep web resources. Our model is designed to be implemented as a periodical script providing a way to inform web search engines about hidden hosts or unknown hosts. The virtual hosting feature, applied in Apache web server [16], allows one Apache installation to serve many different actual websites. For example, one machine, with one Apache installation could simultaneously serve www.example.com, www.example2.com. This feature, actually the virtual hosts, will be our target during the implementation process of our model. The algorithm of the HLP model is extracting hidden hosts, in the form of virtual hosts, from Apache web server using one of the open source technologies [17] which is PHP scripting language [18] and through utilizing an open standard technology in the form of XML language [19], building a frontier of extracted hosts then sending such hosts frontier to the web search engines that support our protocol via HTTP request in an automatic fashion through a cron job [20]. Hosts frontier is an XML file that lists virtual hosts extracted from the configuration file of the Apache web server "httpd.conf" after checking its configuration to make a decision about from where to extract virtual hosts, from "httpd.conf". This combination of open-source software and Apache web server, the most widely-installed web server in the world as of October 2008 Apache served over 51% of all websites according to Netcraft [21] which conducts a monthly web server survey, makes our model a suitable model for providing Host-List functionality to a broad segment of the web market.

3.1 The Architecture of the Model

The architecture of the model composed of five components which are cron job, "httpd.conf" configuration file, multi-functional PHP script, Host-List XML file and a list of search engines that support HLP protocol as shown in figure 1.

Let's take a closer look to each component of the model as illustrated in figure 1:

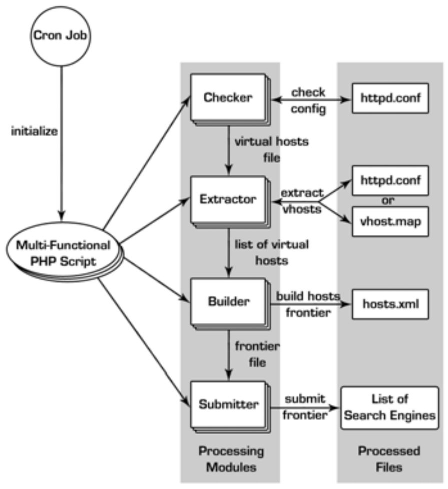

Figure 1 The architecture and main components of the HLP model

Cron job – is executed when the time/date specification fields of the crontab file, that controls the cron jobs, all match the current time and date. The role of our cron job is to initiate the multi functional php script and to repeat that automatically at a specified time/date.
"httpd.conf" –"httpd.conf" is the configuration file of the Apache web severs which contains the seeds of our model in the form of virtual hosts and has different configurations to represent these virtual hosts, using multiple virtual hosting systems on the same server (<VirtualHost> Directive) inside "httpd.conf" or using a separate virtual host configuration file "vhost.map".
Host-List XML file – "hosts.xml" consists of XML tags that represent the list of virtual hosts extracted from "httpd.conf" or from "vhost.map" file. The structure of the file is illustrated in figure 2.

```
 <?xml version="1.0" encoding="UTF-8"?>
 <hostsset>
    <host>www.site-1.com</host>
    <host>www.site-2.com</host>
    <host>www.site-n.com</host>
</hostsset >
```

Figure 2 Structure of the Host-List XML file

Multi-functional php script – is the brain of our model as being responsible for all tasks after has been initialized by the cron job which will have been run periodically on a given schedule. The PHP script is divided into four modules, the checker module, the extractor module, the builder module and the submitter module. The script moves from the initialize phase to the submit phase through a series of actions until the goal is achieved, sending the virtual hosts frontier to the search engines that support out HLP protocol. The main steps of our algorithm can be outlined as follows:

Step 1 – defines the Apache configuration file that will be used to check if the Apache web server is configured using multiple virtual hosting systems on the same server or using a separate virtual host configuration file.

Step 2 – defines the XML file that will contains the extracted virtual hosts and acts as the frontier file that will be send to the web search engines that support our HLP protocol.

Step 3 – defines the list of the web search engines that supports our HLP protocol and will receive the frontier file

Step 4 – initializes the virtual hosts file with an empty value and will be assigned a value later if the Apache web server is configured using a separate virtual host configuration file with such file.

Step 5 – declares the checker module that checks the Apache web server configuration, as mentioned above, and takes the configuration file defined above as an input, returns the value of the virtual hosts file depending on the result of checking.

Step 6 – declares the extractor module that extracts the virtual hosts from the virtual hosts configuration file or from the Apache configuration file according to the value returned from the checker module. The extractor module takes the configuration file and the virtual hosts file as inputs and returns the list of the extracted virtual hosts.

Step 7 – declares the builder module that builds the frontier XML file containing the extracted virtual hosts that returned from the extractor module. The builder module takes the list of the virtual hosts returned from the extractor module and the XML file declared above, in step 2, as inputs and returns XML frontier file formatted and structured to be sent to the list of web search engines.

Step 8 – declares the submitter module that submits the XML frontier file returned from the builder module to the list of web search engines. The submitter module takes the frontier XML file and the list of web search engines file as inputs, loops on the list of the web search engines sending a copy of the XML frontier file to each item of the list through sending an "HttpRequest" utilizing "client URL library (cURL)" [22].

Step 9 – is the main statement of our script that calls the above mentioned modules and executes each of them, passing the output of each module as an input to the other module.

List of supported search engines – is a file contains a list of web search engines that support our HLP protocol and will receive the frontier file of hidden hosts.

3.2 Modules of the Multi-Functional PHP Script

The four modules are run in a successive way during the execution of the multi-functional PHP script. The role and the architecture of each module as follows:

1. The checker module – The first module in action after the initialization process of the multi-functional PHP script by the cron job. The role of the checker module is considered to be as the role of the road sign in driving people to take a decision and be directed to a certain direction. The checker module is starting by checking the configuration file of the Apache web server, "httpd.conf", to find if the Apache is configured using multiple virtual hosting systems on the same server or using a separate virtual host configuration file and returns the value of the virtual hosts file. The checker module uses a list of C conditions C={C1, C2} in the checking process of the given F1 configuration file and make a decision D about from where to extract the list of H hosts H={H1, H2,H3, ….., Hn}. Depending on the value of D the extraction process will be done using a given file list F= {F1, F2}.

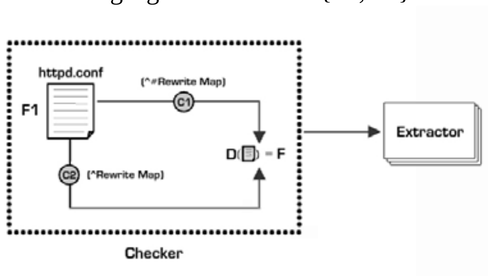

Figure 3 The process outline of the checker module to generate a decision D

A hypothetical scenario for the checker module F = {"httpd.conf", "vhosts.map"}, C= {"^Rewrite Map", "^#Rewrite Map"} is depicted in figure 3.

2. The extractor module – The second module in action after the execution of the checker module. The role of the extractor module is considered to be as the role of the explorer who searches for the treasures as being responsible for searching and extracting the virtual hosts that will be used to build the frontier file which will be sent to the web search engines that support our HLP protocol. The extractor module receives the decision D from the checker module and depending on the value of D the extractor module will execute a certain procedure. The value of the decision D will be assigned to one element of the set F which represents the file that contains the virtual hosts. The extraction process will vary in each procedure according to the value of F and uses a condition that belongs to the set C= {C3, C4} as follows:

[i] The set F of given files means that F1 is assigned to the first value of the set when the Apache is configured using multiple virtual hosting systems on the same server. The extraction criteria is depending on the F1 file format, filtering the contents of the file according to a condition C3 € C and extracting the values of the list H as H1, H2, H3, …, Hn. A hypothetical scenario for the extractor module with F1 = {"httpd.conf"} and C={"^servername "} is depicted in figure 4.

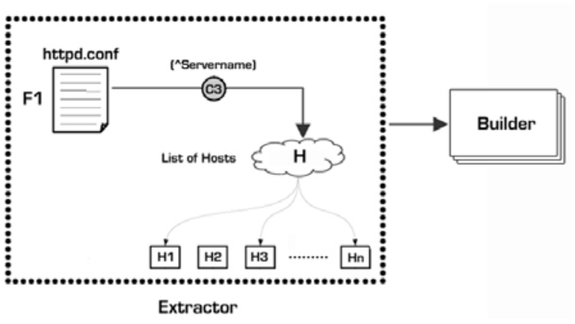

Figure 4 The process outline of the extractor module to extract the H list from the F1 file

[ii] The value of F is assigned to the second value of the set when the Apache is configured using a separate virtual host configuration file. The extraction criteria is depending on the F2 file format, filtering the contents of the file according to a condition C4 € C and extracting the values of the list H as H1, H2, H3,…, Hn. A hypothetical scenario for the extractor module with F2 = {"vhosts.map"} and C= {"^www "} is depicted in figure 5. The value of the set H= {H1, H2, H3,…, Hn} represents the output of the extractor module and the input of the successive module which is the builder module.

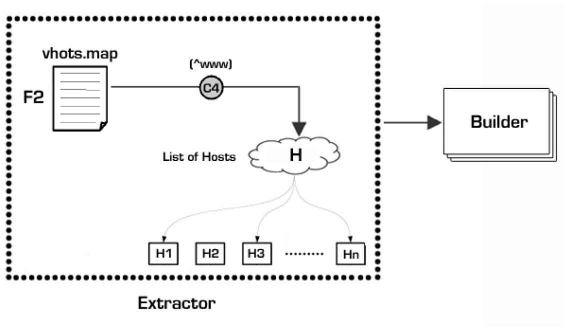

Figure 5 The process outline of the extractor module to extract the H list from the F2 file

3. The builder module – The third module in action after the execution of the checker and the extractor modules. The role of the builder module is considered to be as the role of the lumber in building the blocks of a building. The builder module is responsible for the creation process of the Host-List XML file that contains the extracted virtual hosts as adding each extracted host in the form of a block according to the XML file format illustrated in figure 2. The builder module receives the hosts set H from the extractor module and makes a mapping process in a one to one correspondence relationship to map each element that belongs to the set H into an element that belongs to the set B= {B1, B2, B3, …, Bn} where each element of the set B represents a building block of the XML frontier file.

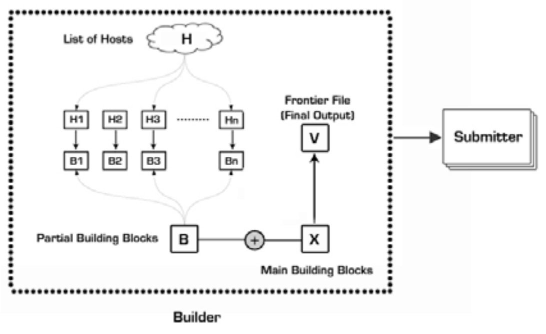

Figure 6 The process outline of the builder module builds the frontier B file

A hypothetical scenario for the builder module receives the set of virtual hosts H={"www.host1.com", "www.host2.com",…,"www.hostn.com"}and generates the set of XML blocks B = {"<host>www.host1.com</host>", "<host>www.host2.com</host>",…, "<host>www.host2.com </host>"} is depicted in figure 6. The value of the set B represents a partial output of the builder module which is the core of the frontier file contents. The final output V of the builder module, produced by adding the main building block X of the XML file to the set B, is the input of the successive module which is the submitter module.

4. The submitter module – The last module in action after the execution of the checker, the extractor and the builder modules. The role of the submitter module is considered to be as the role of the postman in delivering letters as being responsible for the posting process of the frontier file to the web search engines that support our HLP model.

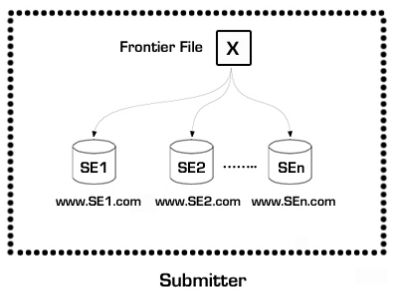

Figure 7 The process outline of the submitter module sends the frontier X file

The submitter module receives the file V from the builder module that contains the extracted virtual hosts and using a set of web search engines SE = {SE1, SE2, SE3, …, SEn} opens a session between the model and each web search engine and sending the X file through such session. A hypothetical scenario for the submitter module receives the file X and using the set of web search engines SE= {"http://www.SE1.com", "http://www.SE2.com",…, "http://www.SEn.com"} is depicted in figure 7.

4. Implementation and Final Results

As the objectives of our model are to solve the problem from the side of web servers and to automate the discovery process for the hidden resources then we need to proof that our model is capable of extracting such hidden hosts during posses the proposed solution in an automated manner. We here demonstrate the validity of our model through implementing the proposed solution during preserving the reliability of the system. So the reliability of the model according to the factor of execution time is another objective for our model also, to identify the suitable server category for the model from the analysis and evaluation of the obtained results.

The environment for the implementation of our scenario houses about fifty servers classified into five main categories according to their hardware configuration and all servers are running the same platform including the operating system, the underlying server daemons and the development environment. The implementation of our model was done using a virtual machine [23] simulator to simulate the behavior of the hosting web servers that will host our model and submit their hidden virtual hosts. The choice of the simulator for the implementation was "VMware Workstation 6.5.1" [24] was based on two reasons. The first reason because the VMWare provides an abstraction that is identical to the hardware underneath the host operating system as the VMWare executes mostly on the host hardware. The second reason because the VMWare is regarded widely as providing excellent performance [25]. All implementation scenarios performed on the same selected servers over VMWare according to their classification categories but with variant number of virtual hosts ranging from 103 to 593 hosts. The processors of all servers are the same in speed and cash but the hardware configuration differs in the number of running processors between single CPU, dual CPU and the used RAM space as ranging from 512MB to 3.264GB. The reason behind such categorization is to study and demonstrate the effect of the number of running processors and the used RAM space on the execution time of the extraction process and illustrated through analyzing the regression of the trendline of the data series obtained during our experiments. We made 5 different implementation scenarios according to the criteria mentioned above ranging from 103 tp 593 hosts and from 512MB to 3.264GB of RAM space and the distribution of the execution time relative to all server categories for all scenarios is illustrated in figure 8.

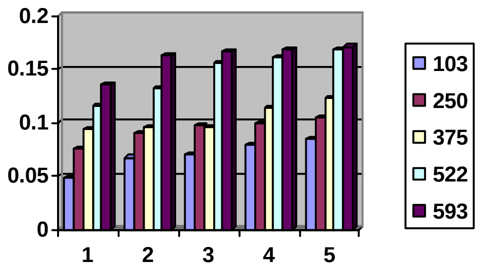

Figure 8 Distribution of the execution time relative to all server categories for all scenarios

From figure 8 we got that the best execution time was equal to 0.0476 seconds for a number of virtual hosts equal to 103 and was obtained over the first server's category. Also the figure illustrates that the best execution time for all other number of virtual hosts was obtained over the same server category. We made a regression analysis for the trendlines that represent the data series obtained through our implementation scenarios as trendlines are a graphic representation of trends in data series sloping upward to represent increased or decreased execution time over a change in server category. To determine the type of regression that represents the trend of data series for each implementation scenario we calculated the Coefficient of Determination (R2) value, R-Squad value, for all lines because it gives the proportion of the variance (fluctuation) of execution time over changing in server category. As the best R-Squad value for regression line that represents the trend of data series when the value is equal or near to 1 then we found that the logarithmic regression was the suitable regression type that good fit to the data series for scenarios 1, 2 and 4 which means that the rate of change in the execution time increases quickly as the hardware configuration is being less in the number of used processors or in the used RAM space. Whereas the polynomial regression with order 2 (one hill) was the suitable regression type that good fit to the data series for scenarios 3 and 5 which means that the execution time values fluctuates. By evaluating the obtained results for the type of regression line that good fit to the data series for each scenario we got that about 98% of the total variation in execution time can be explained by the variation in hardware configuration according to the server category through scenarios 1, 2 and 4. Whereas the total variation in execution time can be explained by the variation in hardware configuration according to the server category for scenario 3 is about 95%

and 91% for scenario 5. Hence from the evaluation of the obtained R-Squad values and the best execution time over all scenarios we got that the suitable server category for our model that preserve the system reliability is the first category.

5. Conclusion and Future Work

The deep web is massively increased in size and the gap in the coverage of web search engines for deep web resources is inherently increased hence, the deep web to be efficiently explored needs an efficient approach. The efficient approach that have the ability to explore the deep web and index its resources needs a smart characteristics and to be based on two concepts, the first concept is to solve the problem from the side of web servers not from the point of view of web search engines and the second concept is to automate the discovery process for such resources not to enforce webmasters of the web sites to register their sites in web search engines which means, automating discovery process is the applicable solution for such problem and all that concept needs is a smart way to do so. We enhanced the discovering techniques of deep web resources and explored an automatic fashion to bridge the gap in the coverage of web search engines for deep web resources. The enhancement was in understanding the nature of deep web resources and providing an approach based on the two main concepts mentioned above and done through the Host List Protocol model. As most research, this paper leaves open areas to be researched, recovered and enhancing the work provided through this paper. The avenues for future work are in many directions and we consider the following as the two main directions:

Integrated web search engine model Propose and implement the model of the web search engine to support our Host List Protocol model and identify how to handle the received virtual hosts' files. The handling process includes algorithms for filtering process of the new submitted hosts and the queuing process of such new hosts through the frontier of the web search engine preparing them to be crawled.

Protocol globalization As illustrated through our work the model was designed and implemented over The Apache web server. The proposed direction here is to propose and implement the Host List Protocol model on the available web server software other than the Apache web server.

References

[1] S. Lam, The Overview of Web Search Engines, University of Waterloo, Canada, February 2001.

[2] S. Lawrence and C. L. Giles. Searching the World Wide Web, Web Science magazine, 280(5360):98, 1998.

[3] Jill H. Ellsworth , Matthew V. Ellsworth, the internet business book, John Wiley & sons, Inc., New York, NY, 1994.

[4] The size of the World Wide Web. http://www.worldwidewebsize.com, retrieved on 2008.

[5] Garcia, Frank. Business and marketing on the Internet. Masthead 9 (1), 2001.

[6] Personal Library Software. PLS introduces AT1, the first 'second generation' internet search service, January 1996, http://web.archive.org/web/19971021232057/www.pls.com/news/pr961212_at1.html

[7] Kobayashi, M. and Takeda, K. Information retrieval on the web. ACM Computing Surveys (ACM Press), 32 (2): 144–173. doi:10.1145/358923.358934, 2000.

[8] Carlos Castillo, Effective web crawling, ACM SIGIR Forum, v.39 n.1, June 2005.

[9] Bergman, Michael K. The deep web: surfacing hidden value. The journal of electronic publishing, 7 (1), Aug 2001.

[10] Sriram Raghavan , Hector Garcia-Molina. Crawling the hidden web. In Proceedings of the 27th International conference on very large databases, pp.129-138, September 11-14, 2001.

[11] Eichmann, D. The RBSE spider: balancing effective search against web load. In Proceedings of the 1st World Wide Web Conference, Geneva, Switzerland, 1994.

[12] Pinkerton, B. Finding what people want: experiences with the web crawler. In Proceedings of the 1st World Wide Web Conference, Geneva, Switzerland, 1994.

[13] World wide science federated search engine. http://worldwidescience.org, retrieved on 2008.

[14] USA government science portal. http://www.science.gov, retrieved on 2008.

[15] Human based web discovery service. http://www.stumbleupon.com, retrieved on 2008.

[16] Yahoo subscriptions. http://search.yahoo.com/subscriptions.

[17] A. Ntoulas, P Zerfos, J Cho - proceedings of the 5th ACM/IEEE-CS joint conference on digital libraries, p.100-109, 2005.

[18] Sitemaps Protocol. http://www.sitemaps.org, retrieved on 2008.

[19] Michael L. Nelson, Joan A. Smith, Ignacio Garcia del Campo, Herbert Van de Sompel, Xiaoming Liu. Efficient automatic web resource harvesting. In Proceedings of the 8th ACM International Workshop on Web Information and Data Management (WIDM), 2006 .

[20] Mod Oai Project. http://www.modoai.org, retrieved on 2008.

[21] About the Apache HTTP server project. http://httpd.apache.org/ABOUT_APACHE.html, retrieved on 2008.

[22] Open Source Initiative (OSI). http://www.opensource.org, retrieved on 2008.

[23] "History of PHP and related projects", The PHP group. http://www.php.net/manual/en/history.php, retrieved on 2009.

[24] Bray, Tim; Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau. "Extensible Markup Language (XML) 1.0 (Fourth Edition)-Origin and Goals". World Wide Web Consortium, September 2006.

[25] GNU cron specification.

http://www.gnu.org/software/gcron/specification.html, retrieved on 2008.

[26] Web server survey archive. http://news.netcraft.com/archives/web_server_survey.html, retrieved on October 2008.

[27] PHP client URL library (cURL) manual. http://www.php.net/curl, retrieved on 2009.

[28] Jeremy Sugerman, Ganesh Venkitachalam, Beng-Hong Lim: Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor. In Proceedings of the 2001 USENIX Annual Technical Conference, p.1-14, Berkeley, CA, USA, June 2001.

[29] VMware Workstation. http://www.vmware.com/products/ws, retrieved on 2009.

[30] Samuel T. King, George W. Dunlap, Peter M. Chen: Operating system support for virtual machines. In Proceedings of the annual conference on USENIX Annual Technical Conference, p.6-6, San Antonio, Texas , June 2003.