Journal of Computer Science 1 (2): 283-289, 2005
ISSN 1549-3636
© Science Publications, 2005

Retrieving Information from the Invisible Web Using Mobile Agents

[1]Fabien-Kenzo Sato, [1]Samuel Pierre and [2]Roch H. Glitho
[1]Mobile Computing and Networking Research Laboratory
[2]Ericsson Research Canada

Abstract: This study proposes a model of information retrieval on the invisible Web by using the mobile agent paradigm. The developed architecture uses the power of a search engine to provide a list of sites of the invisible Web which are likely to be relevant and launches a dynamic search on these sites, thanks to mobile agents. To compare and experiment in real conditions, two versions were implemented: a version using the traditional client/server paradigm and a version using mobile agents. Client/server tests on actual Websites generated satisfactory qualitative results. A series of comparative experiments of the two versions implemented were carried out using a test site. Results show that the mobile agent version generates much less traffic and is thus faster than the client/server version, especially with low bandwidth. Moreover, as the mobile agents carry out calculations on the server rather than on the client's site, this approach relieves the resources of the client terminal. Thus, the mobile agent approach seems particularly advantageous in the case of weak resource terminals such as PDAs.

Key words: Information Retrieval, Mobile Agent, Search Engine, Invisible Web

INTRODUCTION

The Internet has become a major source of information for companies, organizations as well as for individuals. It has reached a tremendous size and it keeps growing exponentially. To find our way in this highly unstructured mass of information, tools such as search engines exist, but a significant proportion of the Web remains inaccessible to those who use traditional search tools. This study presents a new approach allowing access the invisible *Web* with a handy search tool.

The invisible *Web* represents all Web pages that remain unindexed by search engines. This includes many different kinds of pages. Some are isolated or too recent to be indexed. Some Webmasters deliberately hide their sites from search engines. Search engines ignore certain types of multimedia files. However, the most important issue is that it is very difficult for search engines to access dynamically generated Web pages.

This study will focus on this latter category. Dynamic sites represent the major main part of the information available on the Web. Most institutions, such as research centers or libraries, host their database online thanks to dynamic Websites. Data from this kind of Websites cannot be indexed by search engines although they represent the largest proportion of the invisible Web.

This study aims to present a new search engine functionality that would allow accessing dynamic sites. Two methods were developed: one using the traditional client/server approach and another, using mobiles agents. This allows to validate the use of mobile agents in the field of information retrieval.

BASIC CONCEPTS

There are currently many categories of search tools that function in different ways and produce various kinds of results. One must choose the most appropriate tools according to one's needs [1].

* Directories are built from the human selection and classification of Websites or pages according to their subject. They index only a description of the pages rather than their full text.
* Search Engines [2] (Fig. 1) are generated by computer robot programs called spiders or crawlers. They retrieve the full text information of the Web pages they visit. This data is stored in a huge index. When a user launches a query, the search engine looks in its index and ranks the results obtained according to its relevancy.
* Metasearch *Engines* make simultaneous queries to search engines and/or directories. They can be designed as software or as a Website. The major disadvantage of this type of search tool is that it is difficult to make precise simultaneous queries on different tools; generally a single search engine provides better results.

* There are many other search tools that are less well known [3]. Some of them are more specialized or innovative. Among those, spy agents, human search tools, peer-to-peer based tools and semantic based search engines like Exalead [4].

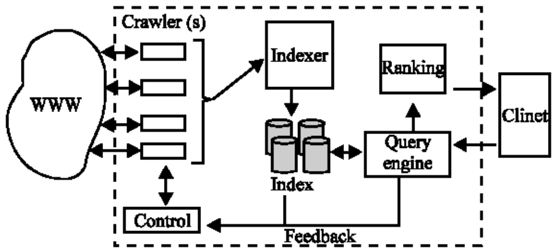


Fig. 1: Search Engine Architecture

Each search tool has its own characteristics and fits into a particular type of query. However, search engines remain the most powerful search tools. Present analysis will focus on this type of tools.
At the end of the year 2002, the search engine *Google* was indexing over 3 billion Web pages. Since the Web is so huge, it is very difficult to crawl all of its pages and a choice of priority must be established. Another crawling problem lies in its refreshment delays. The average delay for a search engine index remains a few weeks. However, some Web pages change many times a day, while others are modified every six months.
Indexed data storage can be challenging due to the very large amount of data. However, it can be well managed in distributed databases. The index must be accessed by the crawlers and by the query module.
Sherman and Price [5] classified the invisible Web into four categories:

* The Opaque Web represents the pages that could be technically indexed although they are not;
* The Private Web denotes the pages that are deliberately excluded from the search engines index by the Webmasters;
* The Proprietary Web designates Websites that require registration, whether it is free or not;
* The Truly Invisible Web depicts the part of the Web that cannot be indexed for purely technical reasons. The databases interfaced with a dynamic Website are not accessible to the search engines. This category constitutes the most interesting part of the invisible Web for research purposes in this field.

Although some metasearch engines and gateway pages claim to specialize in the invisible Web, they solely list a limited number of sites of the invisible Web. Those tools provide limited results as they are based on a human selection of sites.

An *agent* [6] is a software entity that acts for another (human or software) entity. It is autonomous, has its own goals and interacts within its own environment. The *mobility* of the mobile agent refers to its ability it to migrate from place to place and to carry on executions itself on those successive places.
Mobile agents potentially present many advantages. They reduce network traffic, they can be easily personalized, they allow clients to be disconnected during the execution on the server, the agent execution has the ability to survive a network node crash and they are portable to many different operating systems. The field of information retrieval seems to be one of the most promising for the use of this new technology [7].

MODEL AND IMPLEMENTATION

This study proposes an invisible Web search tool model that would have the power and simplicity of the search engines. Basically, as shown in Fig. 2, it was decided to add, to the traditional search engine architecture, a new module responsible to make dynamic searches in conjunction with the classical search engine.
The first important objective is to make the search engine index the hypertext links that points to dynamic sites. While crawling, the search engines cannot access the contents of dynamic sites as a query has yet to be launched. However, the homepage of sites that usually contain such a form can be identified. When a query is launched with the search engine, a list of links toward dynamic sites according to the keywords of the query can be provided. This constitutes the first component of our architecture.
The second component aims to exploit these links dynamically. The purpose aims to analyze the pages where they point. Then, a query will be generated according to the keywords and the homepage form. The page generated that way is analyzed in order to evaluate its relevance in respect to the user's query. Finally, all of the results are compiled onto a single Web page.
From this general architecture, different variations and scenarios are possible. The dynamic search module can be located with the search engine or on the user's terminal (as software). In this study, only the version with the module integrated into the search engine will be presented. Both scenarios can be implemented using a client/server or a mobile agent approach.
With the client/server approach, the dynamic search module communicates with the different Websites using the traditional client/server paradigm. First, the module downloads the homepage or access page of the dynamic site. Then, it must analyze the page and more specifically the forms included in it to generate the appropriate query. The effect of this query creates a new Web page on the server that will be downloaded by the module. This protocol is applied to all the Websites

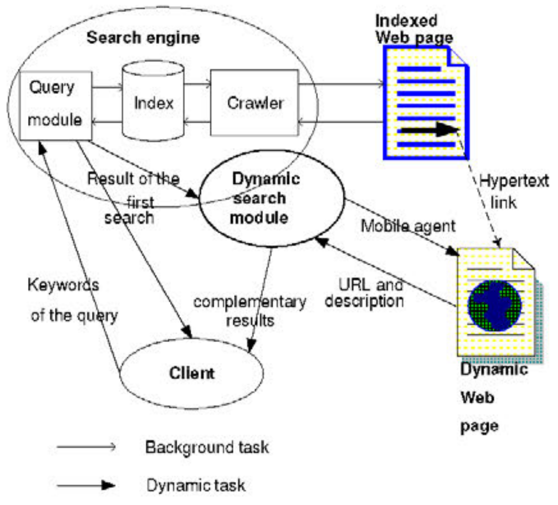
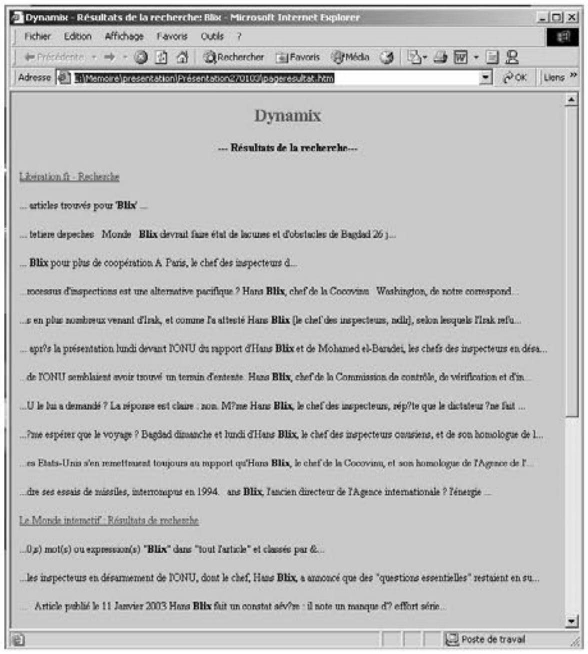


Fig. 2: Search Tool Architecture


Dynamix - Résultats de la recherche: Blix - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente Rechercher Favoris Média
Adresse E:\Memoire\presentation\Présentation270103\pageresultat.htm
OK Liens
Poste de travail

Dynamix

--- Résultats de la recherche---

Libération.fr - Recherche

... articles trouvés pour '**Blix**' ...

... tetiere depeches Monde **Blix** devrait faire état de lacunes et d'obstacles de Bagdad 26 j...

... **Blix** pour plus de coopération A Paris, le chef des inspecteurs d...

...rocessus d'inspections est une alternative pacifique ? Hans **Blix**, chef de la Cocovinu Washington, de notre correspond...

...s en plus nombreux venant d'Irak, et comme l'a attesté Hans **Blix** [le chef des inspecteurs, ndlr], selon lesquels l'Irak refu...

... apr?s la présentation lundi devant l'ONU du rapport d'Hans **Blix** et de Mohamed el-Baradei, les chefs des inspecteurs en désa...

...de l'ONU semblaient avoir trouvé un terrain d'entente. Hans **Blix**, chef de la Commission de contrôle, de vérification et d'in...

...U le lui a demandé ? La réponse est claire : non. M?me Hans **Blix**, le chef des inspecteurs, rép?te que le dictateur ?ne fait ...

...?me espérer que le voyage ? Bagdad dimanche et lundi d'Hans **Blix**, le chef des inspecteurs onusiens, et de son homologue de l...

...es Etats-Unis s'en remettraient toujours au rapport qu'Hans **Blix**, le chef de la Cocovinu, et son homologue de l'Agence de l'...

...dre ses essais de missiles, interrompus en 1994. ans **Blix**, l'ancien directeur de l'Agence internationale ? l'énergie ...

Le Monde interactif : Résultats de recherche

...0,s) mot(s) ou expression(s) "**Blix**" dans "tout l'article" et classés par &...

...les inspecteurs en désarmement de l'ONU, dont le chef, Hans **Blix**, a annoncé que des "questions essentielles" restaient en su...

... Article publié le 11 Janvier 2003 Hans **Blix** fait un constat sév?re : il note un manque d'? effort série...

Fig. 3: Results of the Search Compiled onto a Web Page

provided by the search engine and the results are compiled within a Web page.

With the mobile agent approach, the only difference is the means of communicating between the dynamic search module and the server. The module sends a mobile agent to each dynamic site it has to analyze. The agent generates and analyzes the page locally and returns only the URL and a short description of the page. As with the previous example, the results are compiled onto a Web page.

The main goal of this model aims to develop an efficient search tool for the invisible Web. It integrates itself in a preexisting and powerful tool. As the search is dynamically conducted, there is no need for data storage and the information remains up to date. Unlike some of the results produced by the search engines, this approach could not turn up dead links.

The model was partly implemented to experiment with it and to compare the client/server and mobile agent versions. Choices and simplifications had to be made regarding the conceptual model and implemented only the dynamic component of the model. That means actions were not performed at the level of the search engine. Thus, it was considered that this would provide a list of Websites to visit. To test result relevancy, we checked whether the queried keywords were present and indicated the page title, its hypertext link and a short excerpt around the keywords on a compiled results page (Fig. 3).

For the purpose of present experiment, the mobile agent platform *Grasshopper* was used. This platform respects both standards MASIF (Mobile Agent System Inter-operability Facility) and FIPA (Foundation of Intelligent Physical Agents). As Grasshopper is in Java, the program was written in this language.

We designed a package named Dynamix that uses the library HTMLParser. This library contains many tools to parse HTML code. The Dynamix package is composed of three classes. The first class analyzes the form contained on a Web page, the second class examines the page generated and the last class creates a Web page and compiles all results for the client/server version.

For the mobile agent version, we use the work done previously by installing the Dynamix package on the server side and by calling its methods within the mobile agent's code after the server migration. Finally, the agent returns on the user terminal and generate a Web page compiling the results.

EXPERIMENTS AND RESULTS

We wanted to design experiments that reflect the reality of the Internet as closely as possible. However, the main problem is that current Websites do not support mobile agents. The client/server model can be launched on any Website, although a mobile agent platform must be installed in order to use the mobile agent model. Hence, to avoid these problems, we created our own dynamic Website to compare the relative performance of both models. The sole way to conduct our comparative tests is to control the clients' sites.

Our experiments are twofold. First, the model will be tested in a concrete Internet environment thanks to the client/server paradigm and then, the relative performance of both versions, mobile agent and client/server, will be compared. Qualitative tests will be discussed. Then, the experimental environment used for the comparative tests will be described. Finally, the different tests conducted in order to assess the performance of our applications will be described.

Tests and Experiments: The model was tested by launching a simultaneous search on different Websites such as a daily newspapers sites (Fig. 3). Relevant results were obtained within a reasonable delay ranging from 5 to 20 sec. Given those conditions, the results generated by the application were deemed satisfactory.
In order to conduct comparative tests, a server that supported mobile agents was required; however, this is not currently available on the Internet. Hence, two computers were configured, one acted as the client and the other as the server. The server is equipped with superior hardware and a faster network connection. In addition, it was necessary to create and install a dynamic Website on the server.
The server AMD processor clocked at 1400 MHz and has 256 Mo of RAM. It has a 1 Mbps ADSL Internet connection and a Fast Ethernet 100 Mbps card. The Web server is composed of an Apache 2 server and a PHP3 server. In addition, the mobile agent platform Grasshopper 2 is installed with the Java packages HTMLParser and Dynamix.
The client is installed on a laptop with an 800 MHz Intel Pentium III processor with 128 Mo RAM. It can be connected to a network in two ways: with a 56 Kbps modem or a 100 Mbps Fast Ethernet card. Grasshopper 2 is installed with the Java packages HTMLParser and Dynamix.
The test Website is written in PHP3 and it allows us to generate pages of different sizes according to the keywords of the query. A very simple Website was created. It included a form on the homepage and five different pages that are generated upon request. The page lengths are 0.6, 4.9, 46, 111 and 666 Kbytes respectively, hence a wide range of lengths is investigated. A smaller page offers a search on a limited quantity of data, while a heavier page represents a search on a substantial branch of a site.

Response Time with Low Throughput: Both computers, the server with an ADSL connection and the client with a 56 Kbps modem are connected to the Internet. Thus, the throughput remains quite low. For both versions of our application, five series of queries were conducted, one for each page of our Website. The response time was measured by inserting a line that appraises the system time into the Java code of the application. Each query was repeated 40 times. The results are presented in Fig. 4 and 5.
In both cases, the standard deviation remains low. The execution time remains constant for the client/server version while it varies according to page length for the mobile agent version.
Figure 6 shows that the execution time of the client/server version is proportional to the page length.

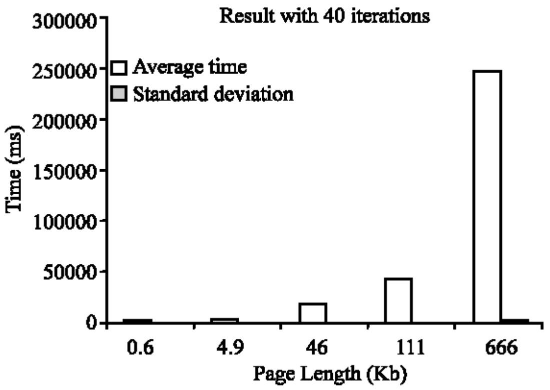


Fig. 4: Response Time for the Client/Server Version (56 Kbps)

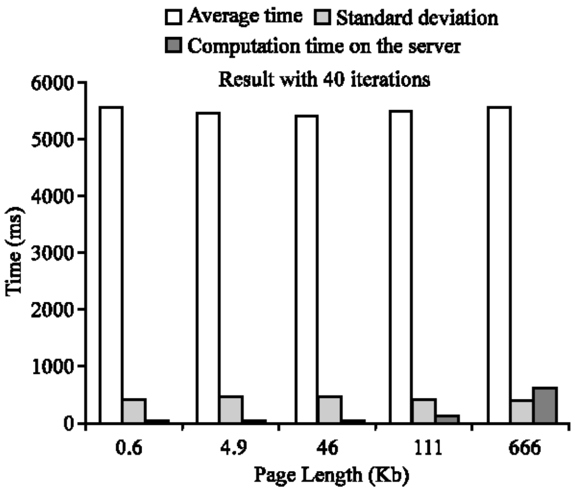


Fig. 5: Time Response for the Mobile Agent Version (56 Kbps)

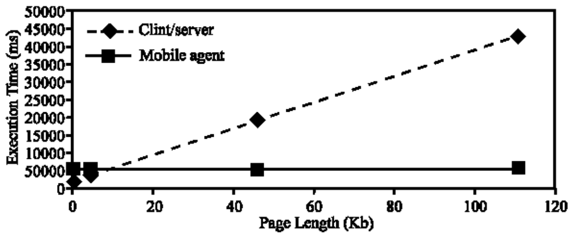


Fig. 6: Comparing Response Times (56 Kbps)

This is to be expected as the entire page has to be downloaded. The mobile agent, which is quite small, has a constant response time of about 5 sec. In this low throughput situation, the mobile agent version clearly appears to be the best solution.

Response Time with High Throughput: In the following series of experiments, the two computers are linked directly through their Fast Ethernet card; hence, they are both equipped with an 100 Mbps connection. We conducted the same tests as in the previous subsection. However, we performed 100 iterations

instead of 40 as the execution times are much shorter. Fig. 7 and 8 illustrate those results.
For the client/server version, the standard deviation is quite high compared to the average time when the page is relatively heavy. This is due to the insignificant transmission delay and the substantial fluctuations in the server execution time.
The execution time of the mobile agent is essentially caused by the creation delay. As shown in Fig. 9, this delay is higher than the response time of the client/server version even if the analyzed page is rather heavy. In this situation, the client/server version is advantageous although the other version has a very reasonable response time.

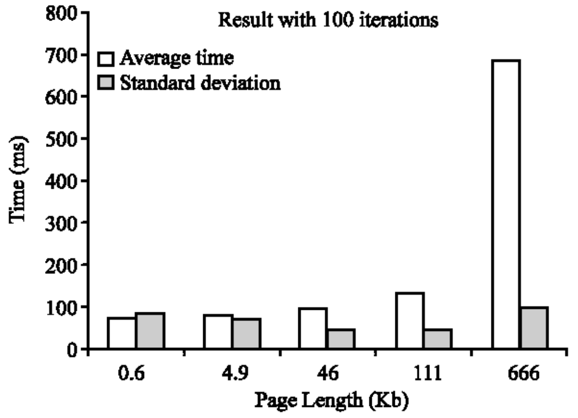


Fig. 7: Response Time of the Client/Server Version (100 Mbps)

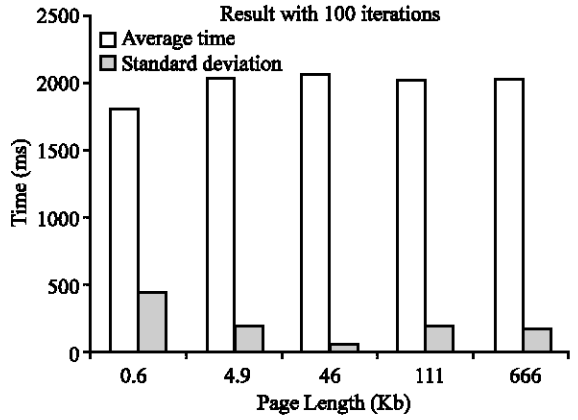


Fig. 8: Response Time of the Mobile Agent Version (100Mbps)

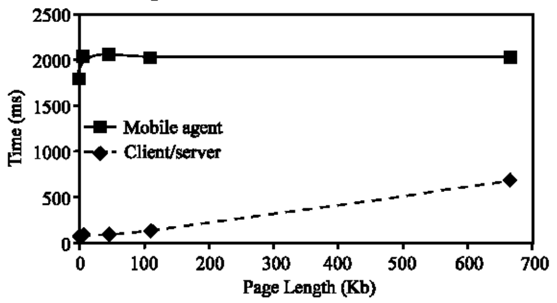


Fig. 9: Response Time Comparison (56 Kbps)

Traffic Evaluation: Response time is not the sole criterion for the comparative evaluation of these applications. Traffic is highly critical for applications used intensively (Google processed 130 million queries daily in 2002).
For this series of tests, connection speed is irrelevant. Thus, high throughput with Fast Ethernet cards was selected for its rapidity. The configuration remains is the same as described above. The network is isolated, hence there are no perturbations and reiterated measures are unnecessary. Traffic was measured with the EtherPeek software [http://www.wildpackets.com, March 2003].
For the client/server version (Fig. 10), traffic is logically proportional to the page length. There are only the TCP/IP control packets between the client and the server.
Traffic remains constant for the mobile agent version (Fig. 11). It is slightly more important from the client to the server as Grasshopper does not retransfer all of the mobile agent data when it returns home.
Notice that traffic comparisons of both versions (Fig. 12) are very similar to Fig. 6. That is because the response time mainly reflects the traffic generated in the case of a low throughput. The mobile agent version appears to be much more efficient from a traffic perspective. This result moderates the conclusions made in the second series of test. The client/server version was much faster although it generated much more traffic.

System Resource Consumption: The system resource consumption was the last criterion evaluated. An efficient application must not overload a server. The system configurations remain identical to those used in the experiments described previously. We measured the consumption of both processor and RAM thanks to Windows 2000 Task Manager that provides an historical record of the processor and RAM utilization rates. As the effects on the RAM were not significant, these measures are not presented. It is important to mention that the computers selected had a major effect upon the results.

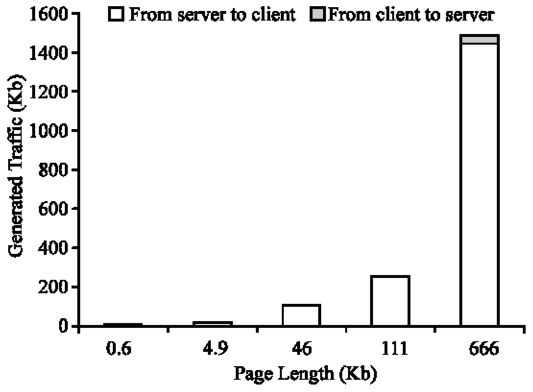


Fig. 10: Traffic Generated by the Client/Server Version

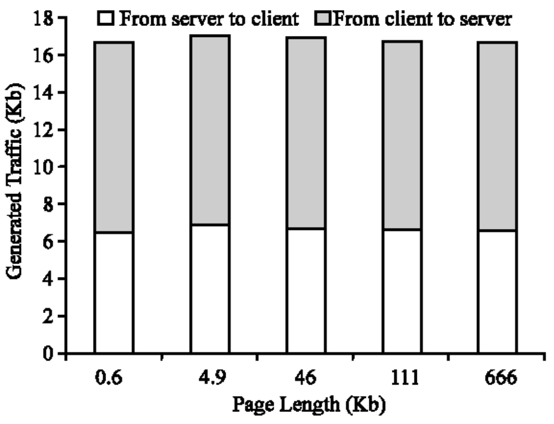


Fig. 11: Traffic Generated by the Mobile Agent Version

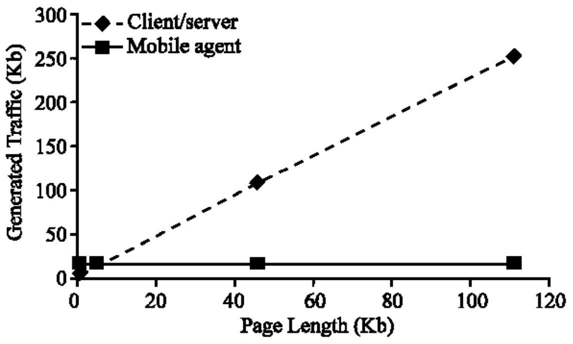


Fig. 12: Traffic Comparison

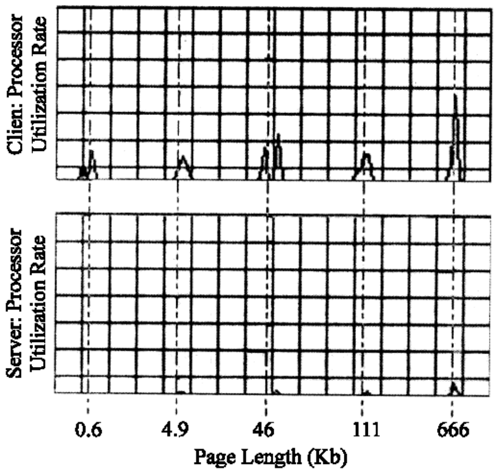


Fig. 13: Processor Utilization: Client/Server

As shown in Fig. 13, the processors utilization rates increase somewhat with the length of the analyzed pages in the client/server version. The server processor is not used very substantially. In the mobile agent version (Fig. 14), the client processor is used less and its utilization does not depend upon the page length. The server processor is used slightly more, especially when the analyzed page is heavier.

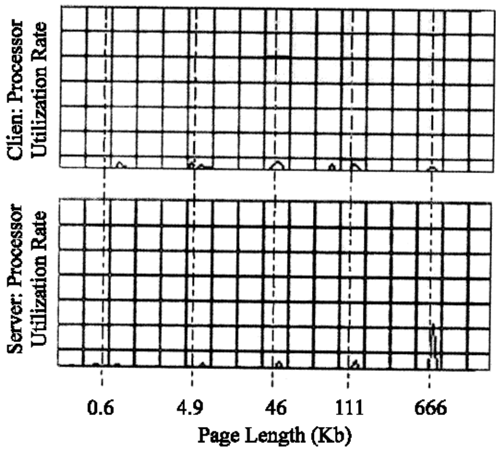


Fig. 14: Processor Utilization: Mobile Agent

The mobile agent approach seems to be advantageous when the client's computation capacity is low. To conclude on the server's utilization, further tests with simultaneous connections would be required.

CONCLUSION

This study addressed information retrieval on the invisible Web. We elaborated a model that supports dynamic search on the invisible Web and we implemented it in two distinct versions in order to evaluate the comparative performance of two approaches: client/server and mobile agent. However, we limited the implementation to a segment of the general architecture and we did not experiment with several clients simultaneously. This conclusion addresses the main contributions of this work and its limitations and it also offers a glimpse of future works.

An architecture was elaborated based on a search engine that can conduct search on dynamic Websites in real time. This model is built on two fundamental concepts. First, the search engine can index dynamic sites URLs and link them with keywords as it crawls the Web. Hence, the search engine can provide a list of dynamic Websites adapted to each query. Second, the Websites were analyzed after users conducted their queries, so the keywords of the queries were retrieved and the appropriate pages were generated on dynamic sites.

The flexible architecture does not require significant technical or material investments as it uses a preexistent search engine, nor does it require any data storage. We strived to exploit the fact that search engines are already user-friendly.

Another contribution of this work concerns the development of an application based on a mobile agent technology. It could be compared to a classical client/server application and the results are very encouraging.

The mobile agent version generates a creation delay and constant traffic. The client/server version needs to completely download all the data it has to process. In cases where client-server connections are low, it is advantageous to send a mobile agent. In cases where the connection is faster, the performance of the client-server version is better, although it creates much more traffic.

All of these results are consistent with the current theories and confirm the expected advantages of mobile agents in information retrieval. Local processing on the server relieves both the network and the client. The mobile agent approach would yield particularly interesting results in mobile computing with terminals equipped with a small computation capacity and low bandwidth.

Many challenges remain unsolved. These twin applications were only compared to a single search on one server. It would be interesting to conduct further searches on several sites. The advantages of more elaborate and more complex searches would certainly lean towards the mobile agent version due to better parallelization.

Personalizing mobile agents and rendering them evolutionary in order to adapt them to the different Websites could also improve this model. Moreover, a library of mobile agents that would perform enhanced searches on specific dynamic sites could be created.

REFERENCES

1. Bourdoncle, F., 1999. Panorama et perspectives des outils de recherche d'information textuelle sur Internet. In IDT/NET'99, Paris, pp: 8-10.
2. Arasu, A., J. Cho, H. Garcia-Molina, A. Paepcke and S. Raghavan, 2001. Searching the Web. ACM Transactions on Internet Technology, pp: 1-42.
3. Abondance, O.A., 2002. http://www.Abondance. com.
4. Exalead, B.F., 2002. http://www.exaleadcom.
5. Sherman, C. and G. Price, 2001, The invisible Web: Uncovering information sources search engines can't see, Information Today, Inc., Medford, USA.
6. Greenberg, M., J. Byington and D. Harper, 1998. Mobile agents and security, IEEE Communications Magazine, 36: 76-85.
7. Milojicic, D., 1999. Mobile agent applications. IEEE Concurrency, 6: 80-90.