World Applied Sciences Journal 19 (5): 666-673, 2012
ISSN 1818-4952;
© IDOSI Publications, 2012
DOI: 10.5829/idosi.wasj.2012.19.05.1083

Reducing Network Traffic and Managing Volatile Web Contents Using Migrating Crawlers with Table of Variable Information

[1,2]Niraj Singhal, [2]Ashutosh Dixit, [1]R.P. Agarwal and [2]A.K. Sharma

[1]Faculty of Electronics, Informatics and Computer Engineering, Shobhit University, India
[2]Deptartment of Computer Engineering, YMCA University of Science and Technology, India

Abstract: As the size of the web continues to grow, searching it for useful information has become increasingly difficult. Also study reports that sufficient of current internet traffic and bandwidth consumption are due to the web crawlers that retrieve pages for indexing by the different search engines. Moreover, due to the dynamic nature of the web, it becomes very difficult for a search engine to provide fresh information to the user. An incremental crawler downloads modified contents only from the web for a search engine, thereby helps reducing the network load. This network load further can be reduced by using migrants. The migrants migrate to the web server for downloading, filtering and compressing the documents before transferring them to the search engine side. In this paper a more network efficient approach for extracting the volatile information from the web server using migrants with the help of table of volatile information has been developed, which further helps in reducing the network load significantly.

Key words: Internet · Web · Search Engine · Information Retrieval · Volatile Information · Migrating · Crawler

INTRODUCTION

The study shows that today using Internet has become an integral part of human life and, out of the 6.93 billion population of the world 2.26 billion people (32.7%) use Internet [1. From.36 billion in 2000, the number of Internet users has increased to 2.26 billion in 2011 i.e. an increase of 528.1% from 2000 to 2011. It is not far away when one will start feeling that life is incomplete without Internet. Same growth rate is expected in future too and it is not far away when one will start feeling that life is incomplete without Internet.

Today the web has grown exponentially in size and as of today it contains billions of publicly indexable web documents [2 distributed all over the world on thousands of web servers. Large size and the absence of centralized control over its contents are two key factors for the success of the web. Unfortunately, same issues are responsible for problem in locating the desired information in time. In-fact quality is distributed very off-centered i.e. interesting pages are sparse in comparison with the rest of the contents. Hence, there is a need of a more effective way of retrieving information from the web. To cope up with this sheer volume of data available on the web, programs that run off of special websites called search engines; has been designed. Search engines help the user to find relevant pages from the web based on important words. A crawler downloads the web pages from www to be used by search engine later.

According to [3, the web is very dynamic and 52% of its contents change daily. Web crawlers recursively traverse the web and download web pages for search engines to create and maintain the web indices. According to [4, the need of maintaining the up-to-date pages in the collection causes a crawler to revisit the websites again and again. Due to this, the resources like CPU cycles, disk space and network bandwidth, etc. become overloaded and sometime a web site may crash due to such overloads on these resources. One study [5] reports that the current web crawlers have indexed billion of pages and about 40% of current internet traffic and bandwidth consumption is due to the web crawlers. According to [4], the maximum web coverage of any popular search engine is not more than 16% of the current web size.

Corresponding Author: Niraj Singhal, Faculty of Electronics, Informatics and Computer Engineering, Shobhit University India.

The centralized crawling techniques are unable to cope up with constantly growing web. Distributed crawling methods based on migrating (or mobile) agents, is an essential tool for allowing such access that minimizes network utilization and also keeps up with document changes. Using migrants (migrating crawlers), the process of selection and filtration of web documents can be done at web servers rather than search engine side which can reduce network load caused by the web crawlers [4,6]. Mouhammd Al-Kasassbeh [7] adopted the distributed model was to eliminate the scalability issues. Mobile agent combines with statistical methods based on the Wiener filter to collect and analyze the network data in order to detect anomalous behaviour in the network traffic.

In this paper we present functioning of a search engine and need of shifting from centralized crawling approach to distributed crawling using migrants. Use of migrants to extract volatile information from the web servers using table of volatile information has also been proposed. This enables web crawlers to select only modified information and bring it to the search engine side, leaving all other static information which already is in the search engine repository.

Related Work: A search engine [8] is a coordinated set of programs that is able to read every searchable page on the web, create an index of the information it finds, compare that information to a user's search request (i.e. query) and finally return the results back to the user. The invention of this technology has granted users quick and easy access to the knowledge they seek; by essentially categorizing web pages according to their relevancy in regards to a request, or query. A general web search engine has three parts a crawler, indexer and query engine.

Web search engines [3,4] employ crawlers to continuously collect web pages from the web. The downloaded pages are indexed and stored in a database. This continuous updation of database renders a search engine more reliable source of right and updated information. Web crawlers are small programs that peruse the web on the search engine's behalf and follow links to reach different pages. Starting with a set of seed URLs, crawlers extract URLs appearing in the retrieved pages and store pages in a repository database. The crawler has to deal with two main responsibilities i.e. downloading the new pages and keeping the previously downloaded pages fresh. However, good freshness can only be guaranteed by simply revisiting all the pages very frequently, but doing so is not possible as it puts unnecessary overload on the crawler. With the available bandwidth for conducting crawls which is neither infinite nor free, it is becoming essential to crawl the web in not only scalable but efficient way if some reasonable measure of quality or freshness is to be maintained. Amir *et al.* [9] proposed a new ranking framework named "NNRank" which uses the primitive features of web documents from the categories of content and context using an artificial neural network. The neural networks selected in their approach is a radial basis function or a principle component analysis neural network which due to their high convergence rate, have the capability to exhibit a high performance with a limited number of features.

Migrating Agents: In general, agent is an autonomous entity that acts on behalf of others in an autonomous fashion, performs its actions in some level of proactivity and reactivity and exhibits some levels of the key attributes of learning, co-operation and mobility [10]. Agents can be classified according to the actions they perform, their control architecture, the range and effectiveness of their actions, the range of sensitivity of their senses, or how much internal state they posses. Nwana [10] identifies seven type of agents i.e. collaborative agents, interface agents, migrating agents, information agents, reactive agents, hybrid agents and smart agents. Migrating agents (migrants) are computational software processes capable of roaming wide area networks such as WWW, interacting with foreign hosts, gathering information on behalf of its owner and coming back having performed the duties set by its user [10].

Mobility allows an agent to move, or hop, among agent platforms (as shown in Fig. 1). The agent platform provides the computational environment in which an agent operates. The platform from which an agent originates is referred to as the home platform and normally is the most trusted environment for an agent. One or more hosts may comprise an agent platform and an agent platform may support multiple computational environments, or meeting places, where agents can interact. They may cooperate or communicate with other agents making the location of some of its internal objects and methods known to other agents without necessarily giving all its information away.

The trends outlined in the previous section lead to the conclusion that mobile code and mobile agents, will be a critical near-term part of the Internet. Not because mobile code makes new applications possible, nor

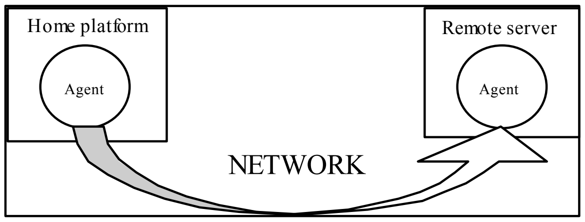


Fig. 1: Agent System Model

because it leads to dramatically better performance than (combinations of) traditional techniques, but rather because it provides a single, general framework in which distributed, information-oriented applications can be implemented efficiently and easily, with the programming burden spread evenly across information, middleware and client providers. Various trends leading to mobile agents [11,12] are information overload, diversified population, bandwidth gap, disconnected operation, customization (re-formatting, filtering, meta search etc) and avoid large transfers. In other words, mobile code gives providers the time and flexibility to provide their users with more useful applications, each with more useful features.

The key characteristics of mobile agents are as follows [10] migration, data acquisition, route determination and communication. Some of the advantages of mobile agents are [13,5] bandwidth, latency, asynchronous task execution, fault tolerance and peer-to-peer communication.

Distributed systems often rely on communication protocols that involve multiple interactions to accomplish a given task. This results a lot of network traffic. Mobile agents allow packaging a conversation and dispatching it to a destination host where the interactions can take place locally. Mobile agents are also useful when it comes to reducing the flow of raw data in the network. When very large volumes of data are stored at remote hosts, these data should be processed in the locality of the data rather that transferred over the network. The motto is, move the computations to the data rather than the data to the computations.

The agent approach use the bandwidth of the network to migrate an agent to a platform and allow it to continue to run after leaving a node, even if they lose connection with the node where they were created thereby provide the better utilisation on communication and allows parallel distributed applications. In simple words, an agent can move on to other machines when necessary and can delegate tasks to other mobile agents in order to achieve real parallel applications.

Managing Volatile Information: Cho and Garcia-Molina [14] devised the architecture for an incremental crawler and examined the use of an incremental versus a batch crawler under various conditions, particularly those where the entire web is not crawled. An incremental crawler refreshes existing pages and replaces less important existing pages with more important new pages. The design of an incremental crawler needs to address the two major issues i.e. how to keep the local collection fresh and, how to improve quality of the local collection.

Sharma *et al*. [15] has addressed the issue of managing the dynamic information more efficiently by introducing separate HTML tags for volatile information. These tags and information are stored in a separate file having same name with extension.TVI (Table of Variable Information). This TVI file is to be updated every time the changes are made to the hypertext document. The crawler needs to checks the contents of TVI file, only for any changes and accordingly updates its collection. The extraction of volatile information only at the document level would definitely reduce the network traffic due to its very less size as compared to the complete page. The explicit classification of the document into static part and volatile part would not only ensure greater stability but would also result in minimization of transmission errors in the static part of the document. Moreover, the effort required to update the changes in a document is also greatly reduced.

In the existing scheme of HTML document structure, the <SPAN> Tag is a container of any text element offering a generic mechanism for adding structure to documents. The <SPAN> Tag is primarily intended for specifying layout that can be used to specify volatile information with the help of class attributes: Vol1, Vol2 and Vol5 as shown in Fig. 2.

Similarly, the XML scheme of document structure offers more flexibility by allowing web page designers to use their own set of markup tags. These tags can be used to reflect the volatile information contained in a document as shown in Fig. 3.

It was suggested that the Tags and the information be stored separately in a file having same name but with different extension (say.TVI) as shown in Fig. 4.

This TVI (Table of variable information) file shall be updated every time the changes are made to the hypertext document. Every time when the latest version of the page is required all the Vol# Tags (volatile information tags) can be extracted out from the document along with their associated volatile information for updated information. This file containing the changed contents of a document

```
<BODY>
SHOBHIT UNIVERSITY, MEERUT
ADMISSION NOTIFICATION No. <SPAN
Class =Vol1 "> M-02/ADM/2012 -201 </SPAN>
Applications in the prescribed forms are invited
for the following courses offered at various
Faculties of the University for the academic year
<SPAN class = "Vol2 "> 2012 -2013 </SPAN>
.
Schedule for the Entrance Test:
Date: <SPAN class ="Vol3 "> 12.03.2012 and
Time : < SPAN class ="Vol4 "> 9 A.M. to 5 P.M.
</SPAN>,
Venue <SPAN class "Vol5"> Shobhit
University, Meerut </SPAN>
</SPAN>
.
</BODY>
```

Fig. 2: An HTML Document containing Volatile information

```
<BODY>
SHOBHIT UNIVERSITY, MEERUT
ADMISSION NOTIFICATION No.
<Vol1> M -02/ADM/2012 -2013
</Vol1>
Applications in the prescribed forms are
invited for the following courses offered
at various Faculties of the University for
the academic year < Vol2> 2012 -2013
</ Vol2>
.
Schedule for the Entrance Test:
Date: < Vol3> 12.03.2012 and Time : <
Vol4> 9 A.M. to 5 P.M. < / Vol4>,
Venue <Vol5> Shobhit University,
Meerut </ Vol5>
</ Vol3>
.
</BODY>
```

Fig. 3: An XML Document containing Volatile information

Vol	1	M-02/ADM/2012-2013
Vol	2	2012-2013
Vol	3	12.03.2012 and Time :
Vol	4	9 A.M. to 5P.M.
Vol	5	Shobhit University, Meerut

Fig. 4: TVI file containing only the Volatile information

would definitely be substantially smaller in size as compared to the whole document. For eg, the TVI file for the Fig. 2 or Fig. 3 will only contain the information shown in Fig. 4. It has been found that the size of a.TVI file is on an average 5% of the size of its corresponding main hypertext document.

Proposed Work: In traditional crawling the pages from all over the web are brought to the search engine site and then processed. When a page is brought to the search engine site and analysed, many a times it is found that it was not needed. In such cases the efforts made to send request to the web server and bringing the page to the search engine site seems to be useless.

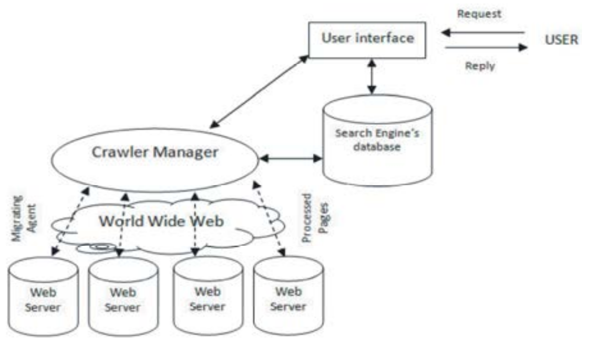


Fig. 5: Distributed crawling using migrating agents

```
While (true)
    While (CrUrls1 is not empty)
        url=SelectURLtoCrawl(CrUrls1)
        if website(url) allows mc
            newpage=page(url)

    newUrls=SelectEmbedUrls(newpage)
            newUrlsQ=newUrlsQ U newUrls
            process(newpage)
            collofpages=collofpages U newpage
            while newURLsQ is not empty

    url=SelectURLtoCrawl(newUrlsQ)
                newpage=page(url)

    newUrls=SelectEmbedUrls(newpage)
                newUrlsQ=newUrlsQ U
newUrls
                process(newpage)
                collofpages=collofpages
                U newpage
                compress(collofpages)
        AllUrls = AllUrls U newUrlsQ
        CrUrls1= (CrUrls1 – url ) U newUrlsQ
        CrUrls2= CrUrls2 U url U newUrlsQ
Where,
AllUrls=List of all Urls, CrUrls1=List of Urls to be
crawled and CrUrls2=List of Urls already crawled.
```

Fig. 6: Algorithm for Distributed crawling using migrating

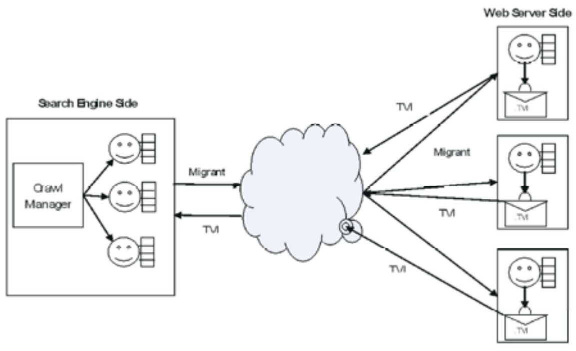


Fig. 7: Migration of Migrants to the Web servers

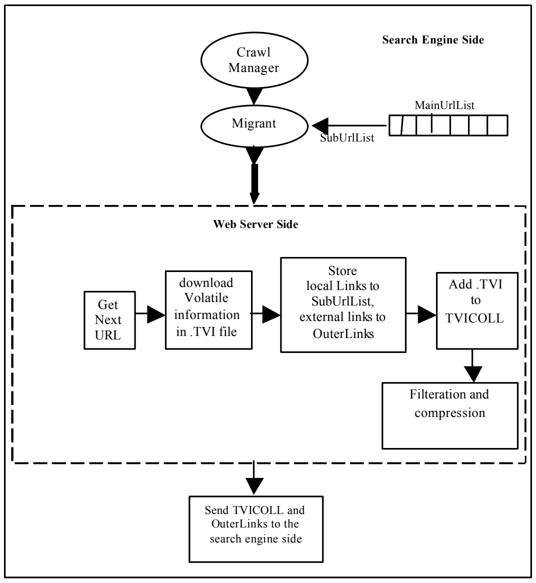

Crawl Manager
Search Engine Side
MainUrlList
Migrant
SubUrlList
Web Server Side
Get Next URL
download Volatile information in .TVI file
Store local Links to SubUrlList, external links to OuterLinks
Add .TVI to TVICOLL
Filteration and compression
Send TVICOLL and OuterLinks to the search engine side

Fig. 8: Transfer of Migrants and their functioning at the remote site

```
Crawl manager()
{
  //manages a list of URLs MainUrlList
  SubUrlList=Sublist of Urls selected from
  MainUrlList
  Creates and migrates a Migrant with
SubUrlList to
  Web server
  //...............
  // The Migrant goes to the web server
  // and extracts volatile information, local
links and
    outer links
  //............
  // On arrival back to the Crawl manager
    Changes are updated to the local
collection
  MainUrlList=MainUrlList U SubUrlList U
  OuterLinks
}
```

Fig. 9: Algorithm for Crawl Manager

```
Migrant( )
  Begin
      While (SubUrlList is not empty)
          Begin
              Pickup next Url;
              If robot.txt allows
              Begin
                  Open the HTML/ XML source file;
                  Download the .TVI file with the
                  same name as HTML/XML file;
              End
          Extract local links and add to SubUrlList
          Extract external links and add to OuterLinks
          End;
  Close the HTML/XML Source file;
  Close the .TVI file;
  Add .TVI file to TVICOLL
  End
```

Fig. 10: Algorithm for Migrants

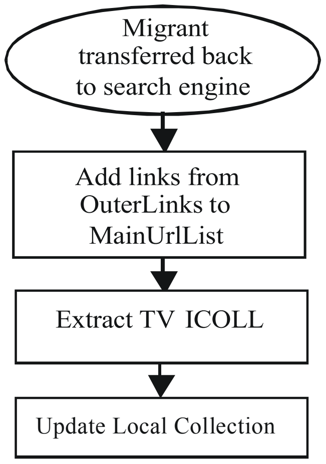
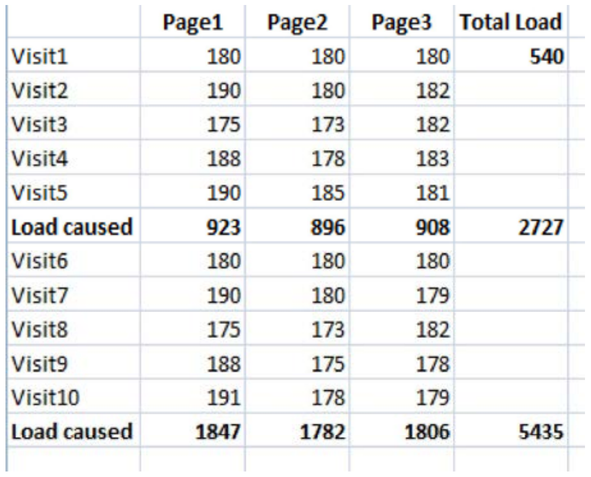
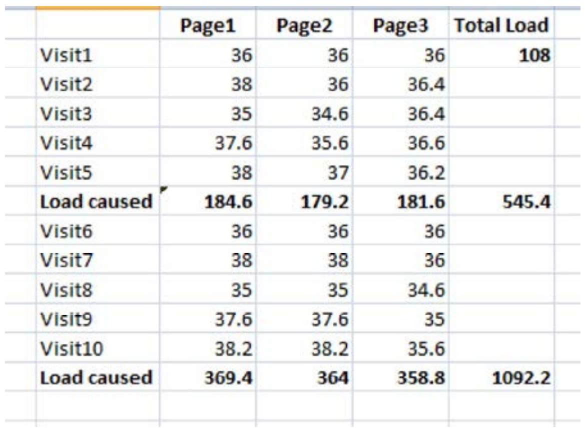

Migrant transferred back to search engine
Add links from OuterLinks to MainUrlList
Extract TV ICOLL
Update Local Collection

Fig. 11: Updation of Local Collection with Migrants



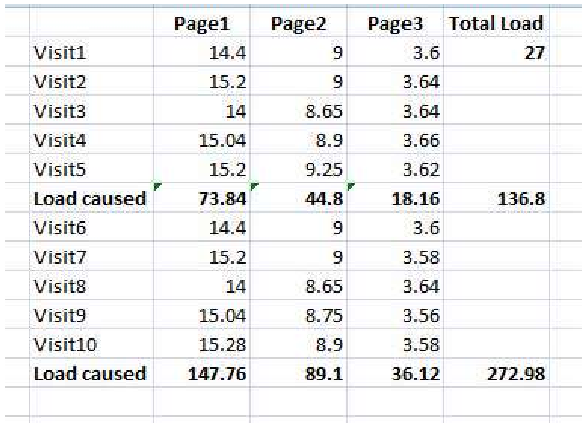
	Page1	Page2	Page3	Total Load
Visit1	180	180	180	**540**
Visit2	190	180	182	
Visit3	175	173	182	
Visit4	188	178	183	
Visit5	190	185	181	
Load caused	**923**	**896**	**908**	**2727**
Visit6	180	180	180	
Visit7	190	180	179	
Visit8	175	173	182	
Visit9	188	175	178	
Visit10	191	178	179	
Load caused	**1847**	**1782**	**1806**	**5435**

Fig. 12: Load caused using Centralized approach



	Page1	Page2	Page3	Total Load
Visit1	36	36	36	108
Visit2	38	36	36.4	
Visit3	35	34.6	36.4	
Visit4	37.6	35.6	36.6	
Visit5	38	37	36.2	
Load caused	184.6	179.2	181.6	545.4
Visit6	36	36	36	
Visit7	38	38	36	
Visit8	35	35	34.6	
Visit9	37.6	37.6	35	
Visit10	38.2	38.2	35.6	
Load caused	369.4	364	358.8	1092.2

Fig. 13: Load caused using migrating crawler



	Page1	Page2	Page3	Total Load
Visit1	14.4	9	3.6	27
Visit2	15.2	9	3.64	
Visit3	14	8.65	3.64	
Visit4	15.04	8.9	3.66	
Visit5	15.2	9.25	3.62	
Load caused	73.84	44.8	18.16	136.8
Visit6	14.4	9	3.6	
Visit7	15.2	9	3.58	
Visit8	14	8.65	3.64	
Visit9	15.04	8.75	3.56	
Visit10	15.28	8.9	3.58	
Load caused	147.76	89.1	36.12	272.98

Fig. 14: Load caused using decentralized approach with migrants and TVI

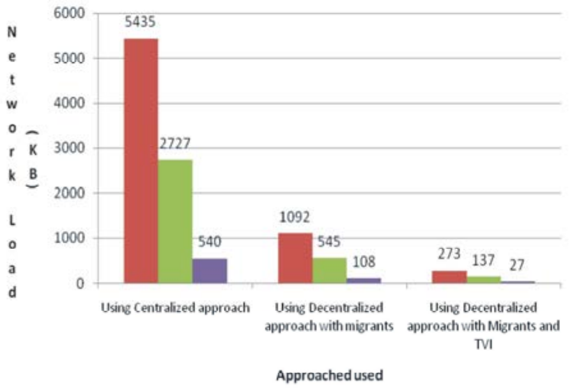


Fig. 15: Graph showing network load caused in various approaches

Moreover this results a lot of network traffic. In the distributed crawling with migrating agents approach, agents allow packaging a conversation and dispatching it to a destination host where the interactions can take place locally. By migrating to the location of the resource, a mobile agent can interact with the resource much faster than from across the network that reduces network traffic also.

The distributed crawling with migrating agents approach (Fig. 5) uses a crawler manager at the search engine site, that deputes migrating crawlers to the web servers with a list of URLs of respective web servers. The migrating crawler, on reaching a server crawls the pages, selects the best of the pages for its collection and comes back to the search engine with the collection. It reduces unnecessary overhead of bringing the unnecessary pages to the search engine site. The size of the collection can further be reduced by filtering the required specialized web pages and even compressing them.

This architecture is primarily designed to be used with IBM's aglets to build mobile agents. The aglets were not used for being not much compatible with Microsoft platforms. However for initial implementation remote procedure calls based on distributed client/server programs have been used to simulate the proposed architecture.

Algorithm for distributed crawling with migrating crawler (mc) approach is shown in Fig. 6.

Managing Volatile Web Contents Using Migrating Agents with TVI: In the migrating approach of crawling, migrants are allowed to migrate to a destination host where the interactions, downloading and processing of documents can take place locally on to the web server itself as shown above in Fig. 5. The main concern is to move the computations to the data rather than the data to the computations. By migrating to the location of the resource, a migrant can interact with the resource much faster than using HTTP across the network.

In the proposed system (Fig. 7) the major components are Crawl Manager and Migrants. The Crawl manager creates multiple worker threads named as Migrants and supplies to Migrant a set of URLs (SubUrlList). Where each URL in the SubUrlList provides the path to the document. The Migrant is a migrating crawling agent that along with SubUrlList goes to the remote site. It downloads.TVI file containing volatile information and stores all. TVI files to TVICOLL file. This process repeats until SubUrlList is exhausted. It also extracts the internal and external links and adds them to SubUrlList and OuterLinks file.

Unlike in traditional centralized approach in which all the job is to be done at the search engine side, this approach performs the job of selecting the volatile information at the place where information lies.

This process may be implemented in parallel and multiple migrants may be sent to many web servers to maintain the scalability. The flow diagram of the process is shown in Fig. 8. The algorithm of Crawl Manager is shown in Fig. 9. The working of a migrant is shown in Fig. 10.

After filtration and compression, TVICOLL and OuterLinks are sent back to the search engine side. Here the urls from the OuterLinks are added to the MainUrlList and local repository of the search engine is updated with the TVICOLL as shown in Figure 11.

Observations: To analyze and compare the approaches, a set of three websites are taken into consideration that contained information of volatile nature as well as static nature. First website contains approximately 8% of volatile contents, second containing 5% and third with least volatile data i.e. 2% only. The average size of a HTML page was 180KB so the network traffic caused using traditional centralized crawling approach was 540 KB. Whereas on using migrants, the pages were compressed at the server side and then the traffic load found was 108 KB. On using the proposed approach that collects only the volatile contents from the web pages it was found that the load came down to just 27 KB.

To measure the load in better manner the volatile contents of the pages were modified before every visit that gave slight variation in the size of pages and measurements of the load incurred dye to revisits of the crawler have been taken for each of the three approach i.e. centralized, migrating and using decentralized approach with migrants & TVI as shown in Fig. 12, Fig. 13 and Fig. 14 respectively.

It can be observed that, after five visits to the pages the load incurred has been found 2727 KB, 545 KB and 137 KB respectively and after ten visits the load was 5435 KB, 1092 KB and 273 KB respectively as shown in the Fig. 15.

CONCLUSION

Web is very dynamic in nature and it is the job of a search engine to provide fresh information to the user. Migrating crawlers are mobile computational software processes capable of roaming the world wide web, interacting with web servers that hosts web pages, gathering information on behalf of its owner and coming back having performed the duties set by its user. In this paper, a migrating crawling approach has been used wherein migrants after moving to the web servers downloads the.TVI (table of variable information) file only for maintaining the freshness of search engine repository. The preliminary experimental results show the reduction in network load by a fraction of 95% approximately in comparison to centralized approach and the network load reduced by a fraction of 75% approximately in comparison to migrating crawling approach. The freshness and quality of the collection also get improved significantly at search engine side.

REFERENCES

1. Internet World Stats, Worldwide internet users, available at http://www.internetworldstats.com (accessed on 24th March 2012).
2. Dixit, A., H. Kumar and A.K. Sharma, 2008. “Self Adjusting Refresh Time Based Architecture For Incremental Web Crawler”, International Journal of Computer Science and Network Security (IJCSNS), 8: 12.
3. Cho, J. and H.G. Molina, 2003. “Estimating Frequency of Change,” Computer Journal of ACM Transactions on Internet Technology, 3(3): 256-290.
4. Nath, R., S. Bal and M. Singh, 2007. “Load Reducing Techniques on the Websites and other Resources: A comparative Study and Future Research Directions,” Computer Journal of Advanced Research in Computer Engineering, 1(1): 39-49.
5. Yuan, X. and J. Harms, 2002. “An Efficient Scheme to Remove Crawler Traffic from the Internet,” in Proceedings of the 11th International Conferences on Computer Communications and Networks, pp: 90-95.
6. Singhal, N., R.P. Agarwal, A. Dixit and A.K. Sharma, 2011. “Information Retrieval from the Web and Application of Migrating Crawler”, 978-0-7695-4587-5/11 © 2011 IEEE, DOI 10.1109/CICN.2011. 99: 476-480, Print ISBN: 978-1-4577-2033-8.2011, Oct 2011.
7. Mouhammd Al-Kasassbeh, “Network Intrusion Detection with Wiener Filter-based Agent”, World Applied Sciences Journal 13 (11): 2372-2384.
8. Singhal, N., A. Dixit and A.K. Sharma, 2010. “Design of A Priority Based Frequency Regulated Incremental Crawler”, Published in proceedings of International Journal of Computer Applications (IJCA), Volume 1, No. 1, Article 8, pp 47-52, Harvard Press US 2010. ISSN: 0975–8887, 2010.
9. Amir Hosein Keyhanipour, Maryam Piroozmand and Kambiz Badie, “A Neural Framework for Web Ranking Using Combination of Content and Context Features”, World Applied Sciences Journal 6(1): 06-15.

10. Nwana, H.S., 1996. "Software Agents: An Overview," Knowledge Engineering Review, Cambridge University Press,
11. Fiedler, J. and J. Hammer, 1999. "Using the Web Efficiently: Mobile Crawling," in Proceeding of the 7th International Conference of the Association of Management (AoM/IAoM) on Computer Science, CA, pp: 324-329.
12. Lingnau, A., O. Drobnik and P. Domel, 1995. A HTTP-Based Infrastructure for Mobile Agents. In: Proceedings of the Fourth International World Wide Web Conference, Boston, USA.
13. Fiedler, J. and J. Hammer, 2000. "Using Mobile Crawlers to Search the Web Efficiently," International Journal of Computer and Information Science, 1(1): 36-58.
14. Cho, J. and H.G. Molina, 2000. "The evolution of the web and implications for an incremental crawler", In Proceedings of the 26th International Conference on Very Large Databases,
15. Sharma, A.K., J.P. Gupta and D.P. Agarwal, 2003. " A novel approach towards management of Volatile Information" Journal of CSI, 33(1): 18-27.