A Novel Parallel Domain Focused Crawler for Reduction in Load on the Network

Rajender Nath[1], Naresh Kumar[2]
[1]Professor, DCSA, Kurukshetra University, Kurukshetra, Haryana, India.
[2]Assistant Professor, MSIT, JanakPuri, New Delhi, India.

Abstract

World Wide Web is a collection of hyperlinked documents available in HTML format. Due to the growing and dynamic nature of the web, it has become a challenge to traverse all the URLs available in the web documents. The list of URL is very huge and so it is difficult to refresh it quickly as 40% of web pages change daily. Due to which more of the network resources specifically bandwidth are consumed by the web crawlers to keep the repository up to date. So, this paper proposes Parallel Domain Focused Crawler that searches and retrieves the Web pages from the Web, which are related to a specific domain only and skips irrelevant domains. It makes use of change in frequency of webpage change to fetch a web page from the web. It downloads only those pages that are changed and ignores those pages that are not modified since the last crawl. Experimentally, it has been found that the proposed crawler reduces the load on the network up to 40%, which is considerable.

Keywords: WWW, search engine, mobile crawler, network resources, Internet traffic, web change behavior.

1. Introduction

World Wide Web is a system of interlinked hypertext documents which is growing rapidly from few thousands of pages in 1993 to more than several billion of pages at present [1]. It is based on client server architecture that allows a user to search anything by providing the keywords to a search engine, which in turn collects and returns the required web pages from the Internet. Due to large number of pages present on the web, the search engine depends upon crawlers for collection of required pages. A crawler [15] (also called spider, walker, wanderer, worm or bot) uses the hyperlinks present in the document to download and store the web pages for the search engine. A search engine tries to cover the whole web and serve queries concerning to all possible topics. But according to [5], any search engine can cover only 16% of the entire web. And one study in [2], shows that the documents available on the web are very dynamic in nature and 40% of its contents changes daily. So, according to a study [3], to maintain the repository of search engine up to date crawler should revisit the web again and again. Due to which, the CPU utilisation, disk space, and network bandwidth etc. are consumed and overload the network. Currently web crawlers have indexed billions of web pages and 40% of Internet traffic and bandwidth utilization is due to web crawlers that download the web pages for different search engines [4].Many suggestions [6][7][8][9] were given to maintain the web repository by using mobile crawlers. In order to filter out unwanted data locally these mobile crawlers were sent to the remote site where the data actually resides. These crawlers can reduce the network load automatically by sending the filtered data only. This paper proposes an alternate approach by using the mobile crawlers and frequency of change. These crawlers go to the remote sites, remains there and skip those pages that are not modified since the last crawl and send only the compressed modified web pages to search engine for indexing.The rest of the paper is organized as follows: The related work is discussed in section 2. Section 3 describes the problems with current crawling techniques. Section 4 describes the proposed architecture of the mobile crawler system. Sections 5 describe the URL allocation approach used by the system and section 6 describe the working of the proposed system. Section 7 shows the experimental evaluation of proposed work. Section 8 concludes the paper.

2. Related Work

In [8], last modified date, number of keywords and number of URLs were used to check the change in the page. Their purpose was to determine page change behavior, load on the network and bandwidth preservation. It was found that 63% of the total pages changed that need to be downloading and indexing, thus reducing the 37% pages on the network. It was also shown that proposed scheme reduce the load on the network to one fourth through the use of compression and preserving bandwidth of network. A fast HTML web page change detection approach based on hashing and reducing the number of similarity computations was purposed by [10]. For change detection, the web page is first converted into the XML tree structure. Two trees were compared to calculate the similarity computation. The sub trees with highest similarity were considered to be more similar. Furthermore to speed up the algorithm Hashing was used in order to provide the direct access during comparison. The experimental results have shown that web page changes were detected in seconds for small

and medium sized web pages and few seconds for large web pages. In [11], the HTML structure (tags and attributes) were taken into consideration to improve the crawling performance. Based on this structure, a robot might use the text of certain HTML elements to prioritise the documents for downloading. They also calculated the relevancy of the text to a topic by considering all the words case insensitive. They performed his experiment on *satellite navigation systems* and showed that the speed of downloading was increased. Focused Structure Parallel Crawler proposed by [12], used click stream based Link Independent web page importance metric to minimize the communication overhead. They used parallel agent's collaboration approach without using any central coordinator, the concept of crawl history, duplicate detector; distiller and classifier etc. to prove the relevancy of document with query and achieve more precision in the performance of the crawler. A distributed and parallel crawling system was proposed by [1] that shows more coverage of the web and decrease the bandwidth consumption but such type of system just distribute and localized the load but did not do anything in reducing the load. A Novel Focused Crawling Approach proposed by [13] of crawl the pages related to a specific topic based on the terms of genre and content information. To retrieve the relevant pages they applied the querying policy on the syllabi of computer science and found that the proposed architecture returns 90% relevant results and good level of precision.

The focused crawlers reported by [5][6][9][14] have the following limitations:

[1]. The relevancy of any web page can be determined after downloading it on the search engine end only. So once a page is downloaded then what is the benefit of filtering it, because all the network resources are already used while downloading those pages.
[2]. The another limitation related to first one is the load on the network and bandwidth is considerably large due to the downloading.
[3]. One problem is related to the attemptation of the web space i.e. given a URL to fetch the web page require to determine quicklly to which domain the URL belongs to.

3. Problem Formulation

Many mobile crawling techniques [3][7][8][12] were introduced to overcome the limitations mentioned in previous section. These techniques describe that the mobile crawlers, crawl the web pages and stay there on the remote system to monitor any change that can take place in the pages allotted to them. But these techniques also have problems given below:

[1]. The problem of iterative computation for every web document affects the performance of the crawler.
[2]. If the mobile web crawler stays in the memory then it consumes significant segment of memory. So what happens when more mobile crawlers from different search engines stay there? This can produce scarcity of memory on remote system site because remote system have to perform there own task also.
[3]. Due to security reasons, remote site may not allow the mobile web crawler to stay there at the remote site.
[4]. If a web page changes rapidly and every time a mobile crawler catches these changes and send these changed page to search engine. Then it again consumes more resources of the network including the processing of the system. To overcome these limitations, this paper proposes architecture for mobile web crawlers that use the concept of different domains and frequency change estimator for efficient URL delegation.

4. Proposed Parallel Domain Focused Crawler (PDFC)

The mobile crawlers are developed as mobile agents and sent to the web servers where they download the web documents, process them and filter out non-modified web pages. Finally, the modified web pages after compression are transmitted back to the Search Engine. The architecture of the proposed PDFC is shown in Figure 1. The major components of the PDFC are Analyzer Module (AM), Old Database File (ODF), Link Extractor (LE), Domain Name Director (DND), Crawler Hand (CH), Crawler Manager (CM), Statistics Data Base (SD), Frequency change Estimator (FCE), Remote Site/Server (RS), Comparator (CoM). Each of these modules are discussed below:

4.1 Crawler Manager (CM)

The main tasks of CM are generation of mobile crawler, delegation of URL's to the mobile crawler for crawling the web pages based on the information taken from the FCE, construction / updation of the data in ODF, updation of SD, receivable of the crawled pages from the RS, decompression and indexing them. After extracting the URL from the downloaded web pages by LE, the task of URL delegation to the mobile crawlers are also performed by the CM with the help of CoM Module.

4.2 Domain Name Director (DND)

The job of DND is to select the specific domain for the URLs that are coming from the URL queue and forward it to the particular DNS queue (such as .org, .com, .edu etc.). Given a URL to the DNS queue, the number of pages to be downloaded and time depends upon the URL itself. The distribution of the load on the CH is likely to be different depending on the frequencyof demand of the domains.

4.3 Frequency Change Estimator (FCE)

This module identifies whether two versions of a web document are same or not and thus helps to decide whether the existing web page in the repository should be replaced with changed one or not. This module is the part of the client site and remains there. The FCE is used to calculate the probability of the page change means in how many days a page gets changed. This module maintains the page change frequency of every page at the SE and updates this information in SD every time a page is crawled. This information about the pages helps SE in deciding that which page is to be re-crawled and when. The CM uses this information for URL allocation to the mobile crawlers. Thus, the FCE filter out all those pages at the client site that has low probability of change and reduces the work load of the crawler on the network. The frequency change (F) is defined as follows:

F = -1 (for those URLs which are traversed first time).

F = 0 (for those pages which change at a rapid rate, so frequency parameter cannot be applied for such pages).

F = N (N>0; N= number of days since last change of page occurred).

4.4 Statistical Database (SD)

This module is maintained at the SE and contains information about all the pages crawled by the mobile crawlers. This module has the following fields:

a) *Name of URL:* It stores the name of the web page that is indexed by the SE in its database after downloading.

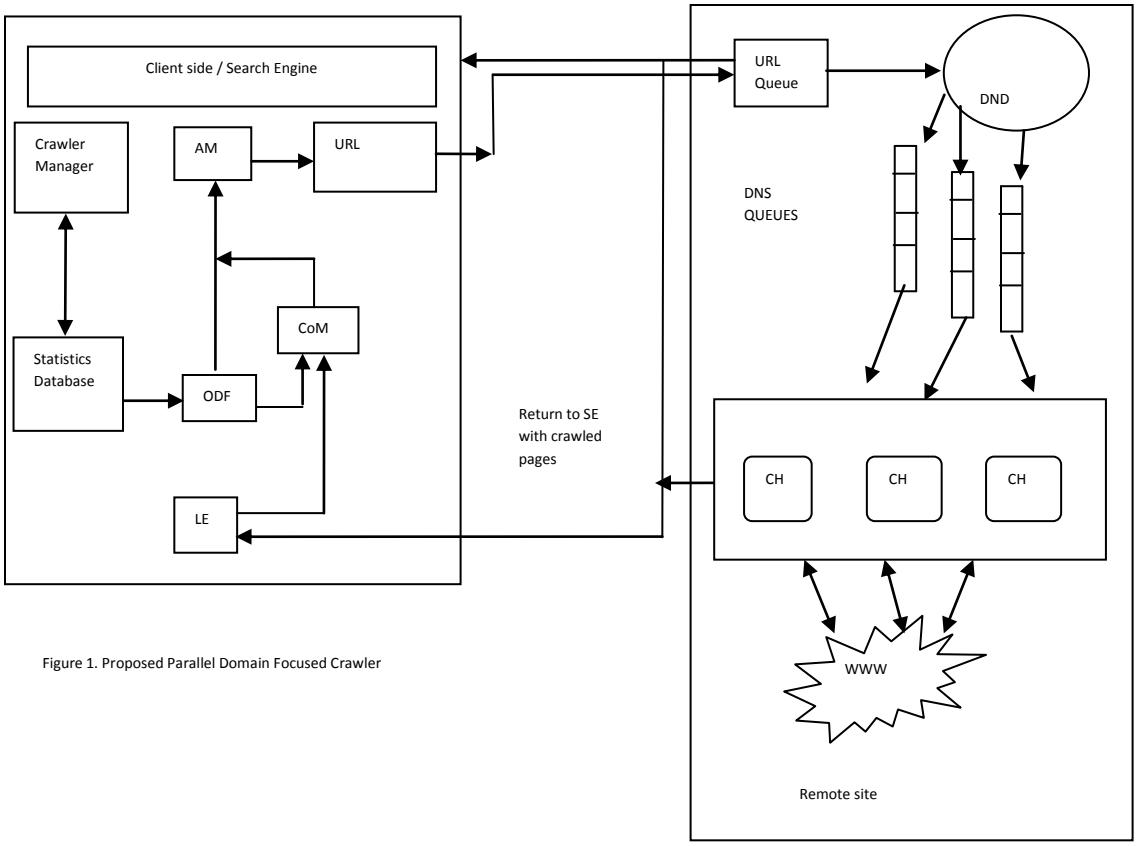

Figure 1. Proposed Parallel Domain Focused Crawler

a) *Parent URL:* It depicts the original URL entered by the user corresponding to which the URLs in the 'Name of URL' field were fetched. For example: say the URL entered by the user is http://xkcd.com/, Then this URL serves as the parent URL and http://dynamic.xkcd.com/random/comic serves as name of URL.
b) *ASCII count:* It shows the 'ASCII count' of a web page. It should be computed whenever the web page is downloaded.
c) *Last Modified Date:*It refers to the date when the particular URL had been crawled the last time.
d) *Frequency of change:*It refers to the number of days that it may take for a particular web page to change its content.
e) *File Path:* The location of the HTML file stored in the repository of computer (at SE end) is reflected in this field.

4.5 Old Database File (ODF)

The CM constructs one ODF for each mobile crawler and contains statistics about each HTML page to be crawled. This statistics is taken from the SD. This module is sent with the corresponding mobile crawler on to the RS with two things i.e. 'Name of URL' and ''ASCII count'' value of web page available in the repository.

4.6 Comparator Module (CoM)

The CoM checks to see if the downloading period of a web page is reached. If yes, then send this URL of Web page to AM for further processing otherwise reject this URL. The 'Last Modified Date' and 'Frequency of Change' is taken from the ODF that were computed when the URL was previously crawled. During this comparison if the difference between the current date and the last-modified date is less than or equal to the value of "frequency of change" field, then the new URL is not processed, rather, it is ignored. On the other hand, in case the difference between these two is greater than the "frequency of change" value, then web page is processed. This module also works on both sides i.e. SE and RS. At the RS it takes the current 'ASCII count' of crawled page and compares it with the old 'ASCII count' of the same page. If both are different then it means the web page is updated and sends to SE, otherwise reject the web page.

4.7 Analyzer Module (AM)

It works on the both sites (i.e. on the SE and RS). It scans and analyzes all the pages crawled by the mobile crawlers and extract 'ASCII count' from ODF for each web page. It also calculates the 'ASCII count' of the fetched page at the RS. It is sent at the RS with the mobile crawler and a copy of it is kept in the secondary memory of RS for future use by the mobile crawler.

4.8 Link Extractor (LE)

The main aim of LE is to take the downloaded web page coming from the CH to SE. This module extracts all the links from this web page and gives them to the CoM module for further processing.

4.9 Crawler Hand (CH)

CH as shown in figure 2 retrieves the URL from the DNS queue in FCFS fashion and sends request to web server. The Crawl Worker first searches the robot.txt file maintained by the web server to verify whether the URL, it is looking for to be download, is permitted for downloading or not. If it is among the list of restricted URLs then CH stops downloading process and discards the URL, otherwise the web document for corresponding URL is fetched and send to the Manager (as shown in Figure 2). The Manager calculates the corresponding 'ASCII count' of that web page with the help of ASCII calculator. CoM Compares the new and old 'ASCII count' of the corresponding page, if they are same then reject the web page otherwise send this page to SE after compression.

4.9.1 Crawling Process

Crawling process takes URL from the crawler worker and fetch the corresponding Web page (see Figure 2). These downloaded Web pages are sent to manager for further processing. After that the new URL is taken from the Crawler Worker and fetches the next Web page. The Web Crawler can run multiple Crawling Processes at a time, so they can share the load in between different processes. Each CH is independent of other means they have no communication with each other because the crawling hand gets the seeds URL from the independent and respective DNS queue only.

5. URL Allocation Approach

Once the URLs are fetched from URL queue then they are distributed using DND to the corresponding domain queues. The CH takes the URL's from the respective domain queue. It is assumed that the pages of one domain should remain on a single server. So, the CH accepts only that page that belongs to the same server domain only. Also ODF is sent with each CH that contains information about the pages to be crawled. The CM works as given below.

[1]. If the FCE has a value F = 0, then it is assumed that the information about the page change is not available because the web page is frequently changed and it is required to be processed the URL again in order to refresh the information available to SE. Furthermore, there is no need to send these pages to AM for scanning and analysis to collect the statistics about them in SD.

[2]. If the FCE has a value F = -1, it shows that the URL are not previously visited. Hence the URL should be crawled in order to get updated information on the SE end. Hence forward getting the web page at the SE end, the AM scans and analyzes the web page in order to get the statistics about the web page.

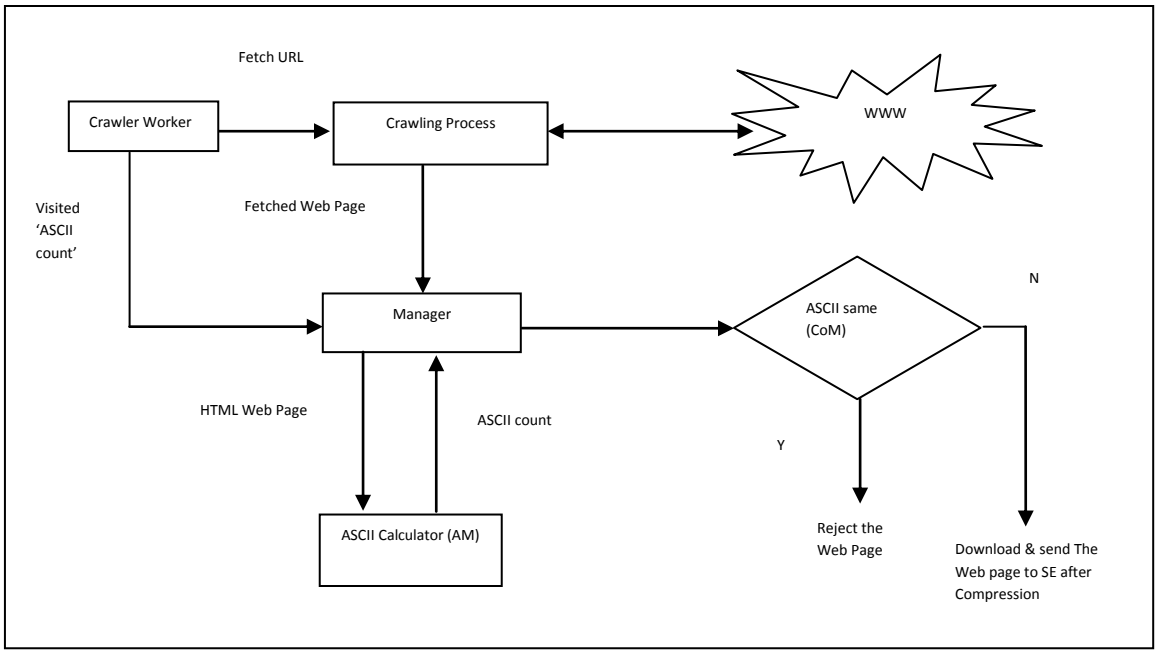

Figure 2. Crawler Hand and Web Page Change Detection

[3]. if the FCE has a value F =N, it shows that in how many days the page is to be crawled. This is analyzed based on the statistics available in SD. Whenever the exact time of frequency of change occurs then the corresponding URL is assigned to the crawler for downloading and statistics of web page is not required to be sent and the value of F in FCE is not updated correspondingly. Otherwise the web page is fetched and sent to the SE in order to update the information at the client side. Also, the web page is sent to the AM that will update the Statistics of the URL accordingly. In this way, the load on the network is reduced by retrieving and indexing the web pages directly whose F value in FCE is either 0 or -1. This reduction is achieved by not storing any information in ODF or not sending any information about the web page with the crawler on the RS. It also saves the CPU time on RS used in the analysis of these web pages.

6. Working of The PDFC

The working of the PDFC is given in flow chart as shown in Figure 3. The mobile crawler used in PDFC takes the ODF, the CoM and the AM with it on the RS from where the pages are to be crawled. The mobile crawler accesses those pages one by one whose URL's are given in the ODF, calculate the 'ASCII count' of the web page and compare this new ASCII count with the previous ASCII count of the same webpage. If both are equal then this webpage is rejected. Otherwise, the web page corresponding to the URL is sent to the SE for updation of information. This is not possible all the times that last modify date is available to SE for all the web page. So, all the web page whose last modify date is not available, is directly downloaded and sent to the SE for updation without any comparison of 'ASCII count' at the RS.

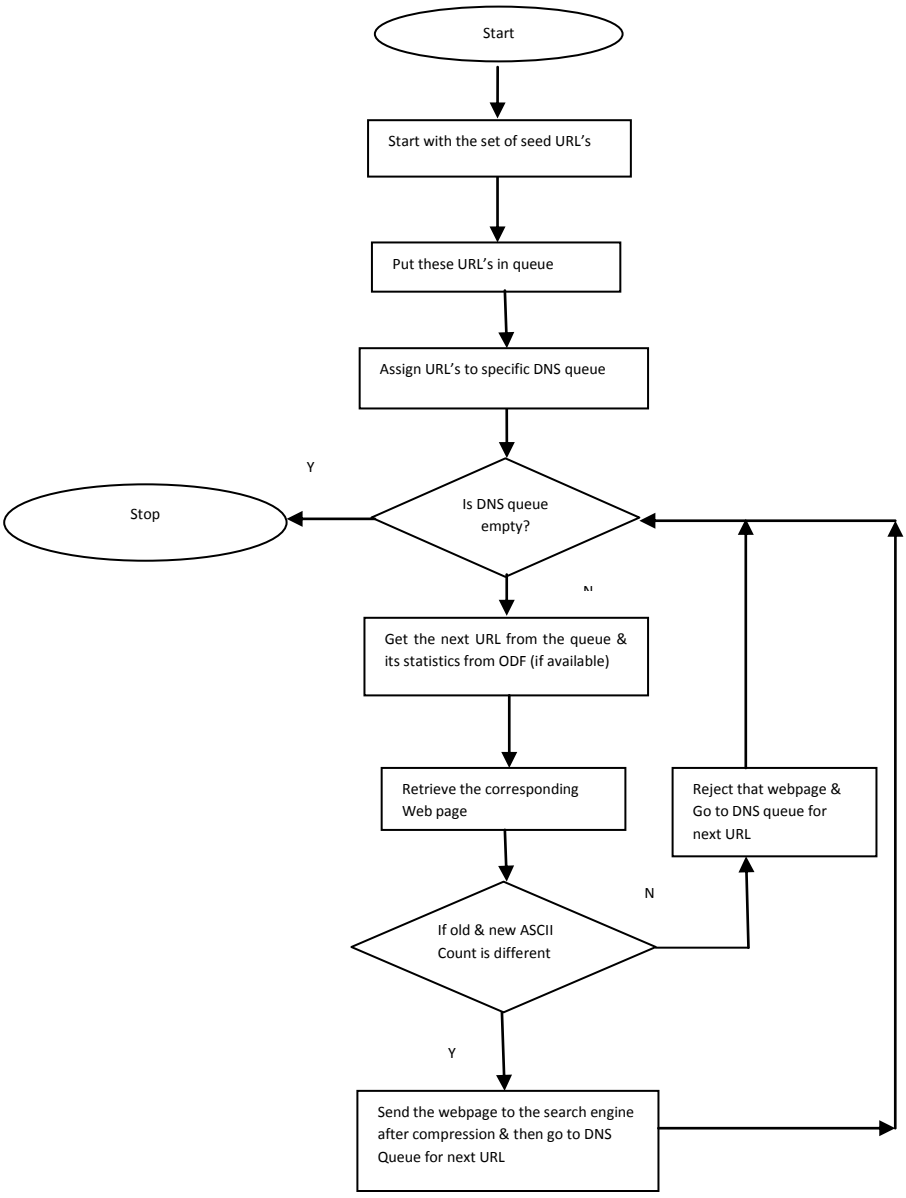

Figure 3. Flow Chart for Working of PDFC

7. Experimental Evaluation of the PDFC

PDFC is implemented in Java programming language using Netbeans IDE 7.0.1 (on Windows XP platform). The application required the JDK 1.6 version to function optimally. The database for storing information was created in MS Access. One hundred web sites were selected to perform the experiment and home pages of these web sites were downloaded. The experiment was carried over for thirty days and the data obtained from the experiment are available with the authors. On an average, each page was of 10 KB. The crawler of the PDFC visited the Remote Site (RS) and analyzed the pages for modification since the last crawl and returned only those pages that were actually modified. The crawler compares 'ASCII count' value available in ODF with 'ASCII count' value of the web page calculated at RS. The proposed PDFC is compared with the existing mobile crawlers on the following three parameters: Page change behavior, Load on the network and Bandwidth preserved are described with the help of bar charts.

[1]. **Page Change Behavior –** The page change behavior was computed based on the data collected during visiting 100 pages for 30 days. This Page change behavior was obtained by identifying the frequently changed pages and frequency of changed pages. The web pages that were found changed on every crawl are called frequently page change and the web pages that were found changed after an interval (not at a specific time) are called frequency of page change. This interval was measured in number of days. Frequency of page change is computed by summing the ASCII value of each character of a crawled HTML web page. Then compare this new computed ASCII count value with the old ASCII count value of the same webpage. If both of these values are found different then it means the newly crawled webpage is updated otherwise page is not updated. Based on above discussion and corresponding parameter representation explained in section 4.3,it was found that out of 100 web pages on an average 19 web pages were found changed due to frequency of page change and 41 pages were retrieved directly based on freshness of pages and with new URL. The bar chart given in Figure 4 shows the average number of pages retrieved due to different change criteria of PDFC. The bar chart in Figure 4 shows that out of 100 pages on the average only 60 (19+41) pages have been changed. Therefore the proposed PDFC sent 60 pages not 100 pages as in case of existing crawler. Thus saving the load of 40 pages on the network**.**

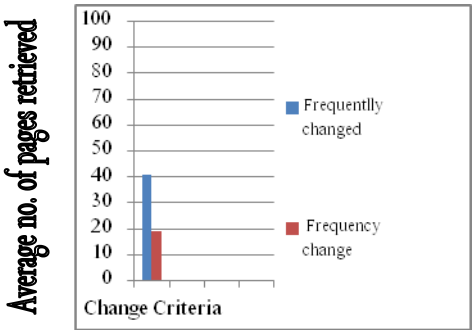

Figure 4. Average number of pages retrieved

[2]. **Load on the network**. The load on the network was measured by network graph manager and on an average it was 800Kbps when the existing mobile crawlers were used. Whereas on using PDFC without compression, the load is reduced to approximately half i.e. 480Kbps. As the PDFC retrieved on an average 60% page only, thus, the network load is approximately 60% when the crawler sends the pages to the SE without using any compression. This load is further reduced to approximately 170kbps by sing standard compressing tools (e.g. WINZIP), the pages can be compressed up to 30% of the original size. The bar chart in Figure 5 shows the comparison of load on the network (in Kbps) for different types of crawlers used. Below Figure 5 shows the comparison of three mobile crawlers – existing mobile crawlers (EMC), PDFC without compression (PDFCWC) and PDFC with compression (PDFCC). It is very much clear from the Figure 5 that PDFC reduce the load up to half when the compression is not used and reduced much more when compression is applied. Thus in both cases PDFC performs better than the EMC.

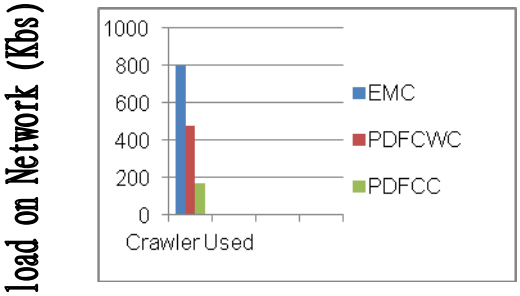

Figure 5. Load on the network.

[3]. **Bandwidth preserved –** According to Shannon- Hartley theorem, the data rate must be twice the bandwidth of the channel. So, if it is assumed that the channel is of 4 KHz without noise then the data rate of this channel would be 8 KBPS. Thus it takes 100 seconds to send the data using EMC. It takes approximately 60 seconds to send the filtered data (only those pages which are actually modified) with PDFC without compression and it takes approximately 21 seconds with PDFC with compression. The consumption of bandwidth used can be measured by software called

Bandwidth Meter Pro. The bar chart in Figure 6 had shown that EMC consumed bandwidth was 132 KHz where as in PDFC it was approximately 110 KHz. This combination of time and bandwidth has increased the data rate, which was saved by not crawling through unchanged websites. Hence, PDFC preserves bandwidth by reducing the traffic on the network by the way of filtering the unmodified pages at remote site.

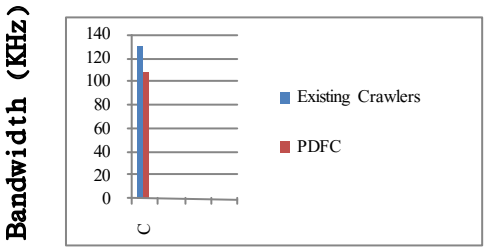

Figure 6. Comparison of Bandwidth

8. Conclusion

This paper has presented a novel Parallel Domain Focused Crawler for reduction in load on the network. The proposed crawler has made the use of the concept of ASCII count and shows experimental results in terms of reduction of load on the network, bandwidth preservation and saving memory. The proposed crawler has been intended by implementing it in Java programming language using Netbeans IDE 7.0.1. The experimental results have revealed that the proposed architecture had reduced the load on the network, has preserved bandwidth and has saved memory significantly.

References

[1] Shkapenyuk V. and Suel T., "Design and Implementation of A High Performance Distributed Web Crawler", in Proceedings of the 18th International Conference on Data Engineering, San Jose, California. IEEE CS Press, pp. 357-368, 2002.

[2] Cho J. and Garcia-Molina H., "Estimating Frequency of Change", in ACM Transactions on Internet Technology (TOIT), vol. 3, no. 3, pp. 256-290, 2003.

[3] Jan Fiedler, and Joachim Hammer, " Using the Web Efficiently: Mobile Crawlers", in Seventeenth Annual International Conference of the Association of Management (AoM/IAoM) on Computer Science, Maximilian Press Publishers, San Diego, CA, pages 324-329, August 1999.

[4] Yuan X. and Harms J., "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, 2002.

[5] Lawrence S. and Giles C., "Accessibility of Information on the Web", Nature, vol. 400, no. 6740, pp. 107-109, 1999.

[6] Nidhi Tyagi & Deepti Gupta," A Novel Architecture for Domain Specific Parallel Crawler ", in Nidhi Tyagi et. al. / Indian Journal of Computer Science and Engineering Vol 1 issue 1 ,pp 44-53, 2010.

[7] Odysseas Papapetrou and George Samaras," Minimizing the Network Distance in Distributed Web Crawling", R. Meersman, Z. Tari (Eds.): CoopIS/DOA/ODBASE 2004, LNCS 3290, pp. 581–596, 2004.

[8] Rajender Nath and Satinder Bal," A Novel Mobile Crawler System Based on Filtering off Non-Modified Pages for Reducing Load on the Network ", in The International Arab Journal of Information Technology, Vol. 8, No. 3, July 2011

[9] Rajender Nath and Naresh Kumar, " A Novel Architecture for Parallel Crawler based on Focused Crawling", in International Multiconference on Intelligent Systems, Sustainable, New and Renewable Energy Technology & Nanotechnology (IISN 2012), pp.36-39, March 2012.

[10] Hassan Artail, Kassem Fawaz," A fast HTML web page change detection approach based on hashing and reducing the number of similarity computations", in Data & Knowledge Engineering pp. 326–337, 2008.

[11] Ahmed Patel, Nikita Schmidt, "Application of structured document parsing to Focused web crawling", in Computer Standards & Interfaces pp. 325–331, 2011.

[12] F. Ahmadi - Abkenari, Ali Selamat, "A Click stream-based Focused Trend Parallel Web Crawler", in International Journal of Computer Applications (0975 – 8887), Volume 9– No.5, November 2010.

[13] Guilherme T. de Assis · Alberto H. F. Laender , Marcos André Gonçalves · Altigran S. da Silva, "A Genre-Aware Approach to Focused Crawling", in WorldWide Web (2009) 12:285–319, DOI 10.1007/s11280-009-0063-7.

[14] Joach im Hammer, Jan Fiedler, "Using Mobile Crawlers to Search the Web Efficiently", in International Journal of Computer and Information Science, 1:1, pages 36-58, 2000.

[15] Jungoo Cho, Hector Gracia-Molina, "Parallel Crawlers", in the proceedings of 11th international World Wide Web conference, Technical report, UCLA Computer science, 2002.