IPMicra: An IP-address based Location Aware Distributed Web Crawler

Odysseas Papapetrou and George Samaras
Department of Computer Science, University of Cyprus
75 Kallipoleos str., P.O. Box 20537, Nicosia, Cyprus
{cspapap,cssamara}@cs.ucy.ac.cy
Tel.: +35722892700
Fax: +35722892701

***Abstract*— Distributed crawling is able to overcome important limitations of the traditional single-sourced web crawling systems. However, the optimal benefit of distributed crawling is usually limited to the sites hosting the crawlers, the rest of the URLs are by large randomly distributed to the various crawlers. In this work, we propose a location-aware method, called IPMicra, that utilizes an IP address hierarchy, and allows crawling of links in a near optimal location aware manner. Our proposal outperforms earlier distributed crawling schemes by requiring one order of magnitude less time for crawling of the same set of sites.**

KEYWORDS: distributed crawling, web crawling, location aware crawling

I. INTRODUCTION

The challenging task of indexing the web (usually referred as web-crawling) has been addressed in many research publications. However, due to the current size, increasing rate, and high change frequency of the web, no web crawling schema is able to pace with the web. While current web crawlers managed to index more than 3 billion documents [2], it is estimated that the maximum web coverage of each search engine is around 16% of the estimated web size [3].

Distributed crawling [5], [6], [4], [1] was proposed to improve the situation. However, all the previous publications were not taking full advantage of the distributed nature of the application. While some of the previously suggested systems were fully distributed over the Internet (many different locations), each web document was not necessarily crawled from the most near crawler but from a randomly selected crawler. While the distribution of the crawling function was efficiently reducing the network bottleneck from the search engine's site and importantly improving the quality of the results, the previous proposals were not at all optimized.

In this work, we suggest IPMicra that facilitates crawling of each URL from the most near crawler (nearness in terms of network latency) without creating excessive load to the Internet infrastructure. We use data from the four Regional Internet Registries (RIRs) to build a hierarchical clustering of IP addresses, which assists us to perform an efficient URL delegation to the migrating crawlers. In addition to location aware crawling, IPMicra detects bottlenecks and provides load balancing in URL delegation and calibration of the existing delegations. The infrastructure runs with negligible network and processing overhead and has very promising results.

This short introduction is followed by a brief discussion on location aware web crawling. Then, we present and evaluate the IPMicra system and describe its advantages, compared to earlier work (centralized and distributed crawlers). Finally, section VI presents our conclusions.

II. LOCATION AWARE WEB CRAWLING

The high distribution of the migrating crawlers in several earlier distributed web crawlers favors a location aware delegation of the URLs to the crawlers. *Location aware web crawling* is distributed web crawling enhanced with a mechanism to facilitate the delegation of the web pages to the distributed crawlers so that each page is crawled from the nearest crawler (i.e. the crawler that would download the page the fastest). *Nearness* and *locality* are always in terms of network distance (latency) and not in terms of physical (geographical) distance. The distinction between the two types of distances (network and geographical distance) is important, since the two types of distances are not always analogous.

Location aware web crawling can make better use of network resources, the lack of them being the primary bottleneck in today's search engines. Modern search engines, instead of trying to reduce the network latency for each page, facilitate multiple crawling threads (issue many concurrent GET requests). *However,* this approach is not perfect, since it does not actually avoid the network bottleneck. Instead, location aware web crawling makes downloading of each web document faster (with multiple threads or not). The network resources in location aware crawling are being fully-facilitated.

In order to find the nearest crawler to a web server we use *probing*. Experiments showed that the traditional ICMP-ping tool, or the time that takes for a `HTTP/HEAD` request to be completed, are very suitable for probing. In the majority of our experiments (over 90%), the crawler with the smallest probing time was the one that could download the web page the fastest. Furthermore, in the rest of the cases, the sub-optimal delegation proposed from the probing function was not far away of the optimal one. Thus, the migrating crawler having the smallest probing result to a web server is possibly the crawler most near to that web server.

Evaluating location aware web crawling, and comparing it with distributed location unaware web crawling was actually simple. A distributed web crawling schema, i.e., UCYMicra [6], [5], was enhanced and, via probing, the URLs were optimally delegated to the available migrating crawlers. More specifically, *each* URL was probed from all the crawlers, and then delegated to the *nearest* one. Location aware web crawling outperformed its opponent (the original UCYMicra), which delegated the various URL randomly, by requiring *one order of magnitude less time*(1/10th) to download the same set of pages, with the same set of migrating crawlers and under approximately the same network load.

The data collected from this experiment also showed that there exist logical neighborhoods in the Internet, which will be most useful if we are able to discover and use for the delegation of the URLs to the crawler.

III. The IPMicra system

The initial extension for location aware web crawling had some impressive results. However, the extension could not easily scale, since it demanded extensive probing (each site should be probed from all the crawlers). Thus, we now propose IPMicra, a web crawling schema designed especially to minimize the need for probing and at the same time perform quality location-aware delegation of URLs to the available crawlers. IPMicra is powered from data collected from the Regional Internet Registries for performing the location aware delegation. The new system is also built over the same distributed web crawling infrastructure, UCYMicra [5], [6]. Thus, before describing the IPMicra approach in more detail, it is important to give a small introduction for UCYMicra and Regional Internet Registries (RIRs).

A. UCYMicra

UCYMicra [5], [6] was recently proposed as an alternative to distributed web crawling. The authors, realizing the limitations of the centralized web crawling systems and several other distributed crawling systems, designed and developed an efficient distributed web crawling infrastructure, powered from mobile agents. The web crawlers were constructed as mobile agents, and dispatched to collaborating organizations and web servers, where they performed *downloading of web documents, processing and extraction of keywords, and, finally, compression and transmission* back to the central search engine. Then, the so-called migrating crawlers remained in the remote systems and performed constant monitoring of all the web documents assigned to them for changes.

More specifically, the original UCYMicra consists of three subsystems, (a) the Coordinator subsystem, (b) the Mobile Agents subsystem, and (c) a public Search Engine that executes user queries on the database maintained by the Coordinator subsystem.

The Coordinator subsystem resides at the Search Engine site and is responsible for administering the Mobile Agents subsystem (create, monitor, kill a migrating crawler), which is responsible for the crawling task. Furthermore, the coordinator is responsible for maintaining the search database with the crawling results that it gets from the migrating crawlers.

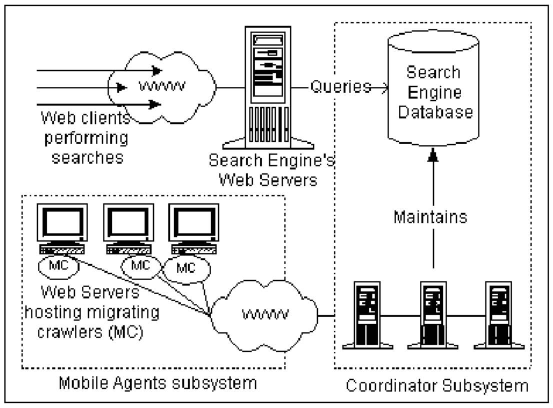
www
Web clients performing searches
Queries
Search Engine's Web Servers
Search Engine Database
Maintains
MC
MC
MC
Web Servers hosting migrating crawlers (MC)
www
Mobile Agents subsystem
Coordinator Subsystem

Fig. 1. UCYMicra basic components

The Mobile Agents subsystem is divided into two categories of mobile agents; the Migrating Crawlers (or Mobile Crawlers) and the Data Carries. The former are responsible for on-site crawling and monitoring of remote Web servers. Furthermore, they process the crawled pages, and send the results back to the coordinator subsystem for integration in the search engine's database. The latter are responsible for transferring the processed and compressed information from the Migrating Crawlers back to the Coordinator subsystem. Figure 1 illustrates the high-level architecture of UCYMicra.

The system involves the following tasks:

1) The user registers the machine that will host the migrating crawler using a web interface in the coordinator subsystem
2) A migrating crawler is then created and moved to the newly-registered system. Upon arrival, the crawler configures itself.
3) The crawler waits until its processing time arrives (the time span that the crawler is allowed to work). When the time comes, the crawler performs an initial crawling of the assigned web-pages.
4) The crawler processes the crawled web-pages and splits the processed results to packages. It compresses the packages and creates a data carrier for each package, to send it to the coordinator.
5) The data carrier transport the package back to the coordinator, where it is received from the coordinator, uncompressed, and finally integrated in the search engine's database
6) The crawler remains in the remote host and monitors the assigned web-pages/web-servers for changes. Upon detection of an update in the monitored web-pages, the crawler informs the coordinator for the update with the usage of a data carrier. Respecting its configuration, the crawler works only in the predefined time-spans.

The UCYMicra paradigm was easily acceptable from the users, and was appreciated and tempting to the web server administrators, since it could offer a quality-controlled crawling service without security risks (they could easily and efficiently set security and resource usage constraints). Actually, the use of UCYMicra was twofold. Powered from the portability of the mobile agents' code, the UCYMicra crawlers could easily be deployed and remotely administered in an arbitrary number of collaborating machines and perform distributed crawling in machines' *idle time* (similar to the seti@home approach [7]). Further on, the crawlers could be deployed in high-performance dedicated machines controlled from the search engine company, for performing efficient distributed crawling with very little communication overhead.

Due to its distribution, UCYMicra was able to outperform other centralized web crawling schemes, by requiring at least one order of magnitude less time for crawling the same set of web pages. The processing and compression of the documents to the remote sites was also important, since this reduced the data transmitted through Internet back to the search engine site, and also eliminated the processing and network bottlenecks. Furthermore, UCYMicra not only respected the collaborating hosts (by working only when the resources were unused) but also offered quality crawling - almost like *live update* - to the servers hosted in the collaborating companies (a service usually purchased from the search engines).

B. Regional Internet Registries

Regional Internet Registries (RIRs for short) are non-profit organizations that are delegated the task of handling IP addresses to the clients. Currently, there are four regional Internet Registries active in the world, APNIC, ARIN, LACNIC and RIPE.

Despite the anarchy currently experienced in the Internet, the four Regional Internet Registries manage to keep an updated database with IP addresses registered in their area. All the sub-networks (i.e. the companies' and the universities' sub-networks) are registered in their regional registries (through their Local Internet Registries) with their IP address ranges. Later on, if the subnet administrator wants to register another block of addresses, again the addresses are expected to be registered under the same organization name in the same RIR.

With the RIRs functionality, a hierarchy of IP ranges has been created in the Internet. We can consider the IP range hierarchy starting from the complete range of IP addresses (from 0.0.0.0 to 255.255.255.255). Then, the IP addresses are delegated to RIRs in large address blocks, and finally, they are sub-divided to LIRs (Local Internet Registries), where then are finally sub-divided to their customers-end users.

C. An outline of IPMicra

The basic idea behind IPMicra is to perform delegation of each URL to the nearest available migrating crawler by employing data collected from the Regional Internet Registries and a very limited number of probes. The IPMicra system is architecturally divided in the same three subsystems that were introduced in the original UCYMicra, and described in section (III-A): (a) the public search engine, (b) the coordinator subsystem, and (c) the mobile agents subsystem (consisted of the migrating crawlers and the data carriers). However, only the public search engine remains unchanged. The coordinator subsystem is enhanced with the functionality of building the IP hierarchy tree (which will be described in the following section) and coordinating the delegation of the subnets. The migrating crawlers are also enhanced with the functionality of probing the sites and reporting the results back to the coordinator.

IPMicra keeps all the functionalities of UCYMicra. More exactly, downloading, processing, and compression of data are all performed in a distributed manner, this way alleviating any processing bottleneck from the search engine. Only the changes, compressed and processed, are transmitted to the search engine's site, thus, alleviating the network bottleneck. Furthermore, as in UCYMicra, the system involves parking of the crawlers in the host systems, and distributed monitoring of the web-sites for detection of updates. In addition, IPMicra now employs an IP address hierarchy, in order to efficiently assign IP address ranges to the 'most near' available crawler, which will be able to crawl it with the least network latency.

Our evaluation (section IV) revealed that the IPMicra system not only outperforms traditional crawling, but, most importantly, significantly improves the performance of distributed crawling (namely UCYMicra), without adding any significant overhead. In fact, the added overhead due to the probing function is negligible.

D. The IP address hierarchy and crawlers placement

The basic idea is the organizing of the IP addresses, and subsequently the URLs, in a hierarchical fashion. However, since the introduction of classless IP addresses, we cannot make any assumptions about the *nearness* of two subsequent IP's (IP address 197.0.0.7 can reside in N.Y., and IP address 197.0.0.8 in Germany, with important network distance between). Thus, we use the `WHOIS` data collected from the RIRs to build and maintain a hierarchy with all the IP ranges (IP subnets) currently assigned to organizations (e.g., see figure 2). The data, apart from the IP subnets, contains the company that registers each subnet. Our experience shows that the expected maximum height of our hierarchy is 8. The required time for building the hierarchy is small, and the hierarchy can be easily loaded in main memory in any average system. While the IP addresses hierarchy does not remain constant over time, we found out that it is sufficient to rebuild it every three months, and easy populate it with the old hierarchy's data.

Once the IP hierarchy is built, the migrating crawlers are sent to affiliate organizations (in real life, these can be normal users or web-hosting organizations). Since the IP address of the machine that will host the crawler is known, we can immediately assign that subnet to the new crawler. In this way the various crawlers populate the hierarchy. The hierarchy can now be used to efficiently find the nearest crawler for every new URL, utilizing only a small number of probes. We stop probing as soon as we find a crawler that satisfies a threshold, called *probing threshold*. Probing threshold is the maximum acceptable probing time from a crawler to a page and it is set by the search engine's administrator depending on the required system accuracy. During our experiments (see section IV) we found a probing threshold set to 50msec to give a good ratio of quality for number of probes.

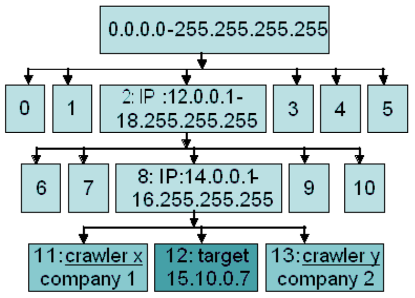

Fig. 2. A sample IP hierarchy. Subnet 11 and subnet 13 belong to company 1 and company 2 respectively. The mentioned hierarchy can be built from the bulk `WHOIS` information from the RIRs

E. The URL delegation procedure

The coordinator subsystem takes over the delegation of the URLs to the available crawlers. Based on the assumption that the sub-networks belonging to the same company or organization are logically (in terms of network distance) in the same area, we use the organization's name to delegate the different domains to the migrating crawlers. In fact, instead of delegating URLs to the distributed crawlers, we delegate subnets - IP ranges. We first find the *smallest* subnet from the IP hierarchy that includes the IP of the new URL, and check if that subnet is already delegated to a crawler. If so, the URL is delegated to this migrating crawler. If not, we check whether there is another subnet that belongs to the same company and is already delegated to a migrating crawler (or more). If such a subnet exists (and the probing result of the crawler having that subnet is smaller than the threshold), the new URL, and subsequently, the owning subnet, is delegated to this crawler. If there are subnets of the same company delegated to multiple crawlers then the new subnet is probed from these crawlers and delegated to the fastest. In fact, we stop as soon as we find a crawler that satisfies the probing threshold.

Only if this search is unsuccessful, we probe the subnet with the migrating crawlers, in order to find the best one to take it over. We navigate the IP-address hierarchy bottom up, each time trying to find the most suitable crawler to take the subnet. We first discover the parent subnet and find all the subnets included in the parent subnet. Then, for all the sibling subnets that are already delegated, the coordinator sequentially asks their migrating crawlers, and the migrating crawlers of their children subnets to probe the target subnet, and if any of them has probing time less than a specific threshold (probing threshold), the target subnet is delegated to that crawler. If no probing satisfies the threshold, the search continues to higher levels of the subnets tree. In the rare case that none of the crawlers satisfies the probing threshold, the subnet is delegated to the crawler with the lower probing result.

Probing of sites, as with any activity in the migrating crawlers, is always performed during the timespans that the migrating crawlers are allowed to work, when the host system and LAN are of low or no use (this is the only time of the day that the coordinator communicates with the migrating crawlers). This has the extra advantage that each site is probed from each crawler near the time that it would be actually crawled later if it was delegated in that crawler. This kind of probing takes the expected workload of the server to be crawled in the future crawling time under consideration, thus avoids overloading of the server in peak hours.

F. Dynamic calibration of URLs and load balancing

Internet is vastly changing, in structure and contents. Underlying connectivity between the servers, and routing tables of the intermediary routers change, invalidating past IPMicra delegations. Thus, the IPMicra algorithm must perform dynamic calibration of the delegations so that they remain optimal. Further on, the algorithm must be able to avoid bottleneck creation to crawlers that are in over-populated (from web documents) areas, and/or perform load balancing between the available crawlers.

Dynamic calibration: Our algorithm performs dynamic calibration of the URLs. During time, the Internet infrastructure changes, new nodes are added, routing tables in the intermediary routers change. For these reasons, we perform some additional counts during the crawling function, which efficiently detect changes that should cause re-delegation of some subnets. More specifically, each migrating crawler in the IPMicra extension counts the required time for every `HTTP/GET` request it issues for every web page. The new time is compared with the previous `HTTP/GET` times for the same web page, stored locally in the crawlers' memory. If the new time is sufficiently larger (a threshold defined from the search engine administrator) than the time demanded for the previous downloads of the same page, and if this repeats for more than one time continuously, then the subnet is re-delegated, so that a more suitable crawler is found. The same happens with every `HTTP/HEAD` command, which is often used from our crawlers in order to discover whether a particular page has changed.

Incorporating this extra task during web crawling, we manage to keep our delegations up to date and to efficiently detect any negative changes in the Internet infrastructure with respect to the existing delegations. Applying the dynamic calibration algorithm does not introduce any more load in the Internet infrastructure, since we only use information already available, produced from the standard commands issued for the web crawling task. Furthermore, the extra computational load added from the dynamic calibration algorithm is negligible.

Load balancing: Load balancing is also employed in the algorithm, in order to avoid overloading of certain web crawlers (in crowded areas), and at the same time perform near-optimal delegations. Each crawler has a maximum capacity, the size of the assigned web-pages that the crawler has to check each day. In the case where one crawler is found to be the best for one new subnet, but the inclusion of this subnet to the crawler's workload violates the crawler's capacity, then measures are taken toward one of the following: (a) either the overloaded crawler takes the new subnet and releases one of the old ones, which is then taken over from some other crawler, or (b) the new subnet is delegated to the second-best crawler.

We looked into several approaches to ensure that our suggestion is scalable. We were able to find many suitable approaches for this task. The problem could be translated and solved as a standard linear programming problem, or, we could follow a test-all-cases approach, testing all the possible combinations in order to find the best. Furthermore, we could use a simpler heuristic that finds the best URL to re-delegate by comparing the next-best time for all the URLs in the overloaded crawler. All the approaches target in minimizing the global cost, which is, in our case, defined as the sum of the time that takes each of the crawler to download the web pages assigned to it $\sum_{\forall i,j} P_{ij}$, where i=page, j=crawler, and

$$P_{ij} = \begin{cases} \text{time required to download i from j,} & \text{if i delegated to j} \\ 0, & \text{otherwise} \end{cases}$$

For the purpose of this work, we developed another heuristic based on the variance of the logical distance of each web-page (identified from the probing result) from all the migrating crawlers that probed the web page. Intuitively, small probing time variance implies that most of the probed crawlers have similar probing results. Thus, we expect to be easily able to find a near optimal crawler to take over a page. The execution of the load balancing algorithm is taking place in the coordinator. The coordinator uses the stored (in the database) probing results for all the subnets to select the subnet that can be removed from the overloaded crawler. This is the subnet with the smallest variance in its probing results. The crawler takes this subnet and checks if the next-best crawler for it (if any), according to the probing results, is available and can accept the subnet (i.e. will not get overloaded). If so, the subnet is re-delegated to this crawler. If not, we retry the approach with the next best subnet, until we find a subnet that can be efficiently delegated to a new crawler.

We found this heuristic to perform well. We selected this heuristic instead of a linear programming (which would be more effective) and the test-all-cases solution strictly because of the ease in implementation. Furthermore, the heuristic was selected over the heuristic based on next-best time, because it was expected to scale more gracefully, and function better over time. Our tests revealed that the heuristic was able to make the best load balancing decision in more than the 2/3 of the cases. Furthermore, in all the rest cases, the heuristic was able to find an acceptable solution. Unfortunately, due to space limitations we cannot present analytical results of our experiment here. While satisfied with this heuristic, part of our ongoing work is to apply and evaluate other load balancing algorithms.

IV. Performance and evaluation

Evaluating IPMicra, we should investigate four different aspects: (a) the overall time for performing the crawling function in IPMicra, compared with other distributed crawlers, (b) the percentage of the optimal delegations proposed from IPMicra (with and without load-balancing), (c) the time difference for crawling of the sub-optimal delegations, proposed from IPMicra, compared to the optimal ones, and, (d) the required probes for each delegation. We also experiment with different probing thresholds in order to find the one that would offer significant performance with low probing requirements.

We evaluated IPMicra with 2 different setups, with 4 and 12 migrating crawlers (the experiments were repeated multiple times for each setup) hosted from affiliated organizations world-wide, which were assigned a set of 1000 URLs to crawl. In each setup, the optimal delegation was also calculated (by facilitating a brute-force algorithm that is not described here). Crawling was performed with IPMicra and with location-unaware distributed web crawling (represented from a slightly modified - for matters of easier experimental setups - version of UCYMicra). In both the setups, IPMicra was able to perform a near-to-optimal location-aware assignment of URLs to the available crawlers (more than 75%), this way outperforming its opponent, UCYMicra, and requiring one order of magnitute less time *(near 1/10th of time)* to perform the crawling function.

We also experimented by varying the probing threshold (the minimum probing time for stopping the probing algorithm), and found out that a threshold set to 50msec would have the best quality:#probes ratio. Paying the cost of significantly less probes, when the probing threshold was set to 50msec, IPMicra was not able to perform the optimal location aware delegation, but the proposed delegation was very near to the optimal (more than 75% of the URLs were delegated to the nearest web crawler). Furthermore, all the sub-optimal delegations were practically very near to the optimal ones. The overhead from the IPMicra algorithm was less than 3 inexpensive probes per URL for our setup. Actually, as the IPMicra algorithm was getting calibrated with new data, we required less probes for every new URL. For example, while the average number of probes required for the delegation of all the URLs was 2.99 probes per URL, the average number of probes for delegation of the last 50 URLs (after the hierarchy was calibrated from the previous delegations) was 2.66 probes per URL.

V. Advantages of the IPMicra system

IPMicra has several advantages inherited from the mobile agents model, and its predecessor, UCYMicra. Furthermore, it handles load balancing and URL delegation better and in a more efficient way. More exactly, the new system has the following advantages:

1) Location aware crawling. It delegates the web sites to near migrating crawlers in order to take advantage of the lower network latency for faster crawling
2) IPMicra makes better use of the available bandwidth. While location unaware web crawlers (distributed or not) were trying to get over the network latency and increase the crawling rate by employing multiple crawling threads, the available bandwidth was still fully facilitated and was always a bottleneck. Location aware web crawling needs less time to download a web document and releases network resources faster. Just by re-arranging the delegation of the URLs to the available web crawlers, we can complete the crawling function more efficient. Therefore, we expect to avoid the network bottleneck during crawling.

3) Load balancing. It uses an efficient load balancing scheme to alleviate bottlenecks in the migrating crawlers.
4) IPMicra eliminates the need of the traditional centralized web-crawlers, since the new crawling paradigm can follow newly found links and performs efficient load balancing.
5) IPMicra eliminates the enormous processing bottleneck from the search engine's site, by delegating the processing task to the migrating crawlers.
6) IPMicra performs remote processing and compression (to the migrating crawlers) prior transmitting the results back to the search engine. The size of the transfered data back to the central point - the search engine - is reduced to an order of magnitude. More accurately, we transmit *less than 1/20th* of the size of the crawled data, without loosing any search-useful information.
7) Useless conditional GETs(`If-Modified-Since` headers) and `HEAD` requests do not any more facilitate network resources from the search engine's site, but are executed distributed. Thus, they do not contribute to the creation of a network or processing bottleneck in the search engine site.
8) Keeps up with document changes, since it can inexpensively monitor for changes.
9) Is upgradeable at real time, since the crawlers can perform update of their own code.
10) Is promising and acceptable from the users, due to the security constraints that can be set to the migrating crawlers, and at the same time, it can offer a fully configurable crawling service for the web server administrators (similar services are currently sold from commercial search engines).

VI. Conclusions

In this work, we proposed IPMicra, an extension of UCYMicra, that allows, based on the notion of 'nearness', crawling of links in a near optimal location aware manner. The motivating power behind IPMicra is an IP address hierarchy tree, which is build using information from the Regional Internet Registries. This hierarchy is used to delegate the web sites to near migrating crawlers in order to take advantage of the lower network latency for faster crawling. To our knowledge IPMicra is the first location aware distributed web crawler, and can offer an efficient and generic solution to todays web indexing problem. We view this work as an important step toward a truly distributed and scalable web crawler, that will be able to catch up to the expanding and rapidly changing web.

The location aware infrastructure developed in this work can be applied (as a framework) in any (fully or partially) distributed web crawler. The framework can even be applied in existing commercial approaches, like the Google Search Appliance or Grub. Furthermore, it can facilitate optimizations for distributed applications in the Internet in general. For example, this framework can efficiently enhance the load balancing schemes used from content delivery networks, such as Akamai.

References

[1] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. The Harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28(1–2):119–125, 1995.
[2] Google Inc. Google, September 2003. http://www.google.com/.
[3] S. Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400(6740):107–109, July 1999.
[4] LookSmart Ltd. Grub distributed internet crawler, 2003.
[5] Odysseas Papapetrou, Stavros Papastavrou, and George Samaras. Distributed indexing of the web using migrating crawlers. In *Proceedings of the Twelfth International World Wide Web Conference (WWW)*, 2003.
[6] Odysseas Papapetrou, Stavros Papastavrou, and George Samaras. Ucymicra: Distributed indexing of the web using migrating crawlers. In *Proceedings of the 7th East-European Conference on Advanced Databases and Information Systems, Dresden, Germany*, 2003.
[7] SETI. Search for extra terrestrial intelligence, January 2004.