Minimizing the Network Distance in Distributed Web Crawling

Odysseas Papapetrou and George Samaras

University of Cyprus, Department of Computer Science,
75 Kallipoleos str., P.O. Box 20537, Nicosia, Cyprus
{cspapap, cssamara}@cs.ucy.ac.cy

Abstract. Distributed crawling has shown that it can overcome important limitations of the centralized crawling paradigm. However, the distributed nature of current distributed crawlers is currently not fully utilized. The optimal benefits of this approach are usually limited to the sites hosting the crawler. In this work we describe IPMicra, a distributed location aware web crawler that utilizes an IP address hierarchy and allows crawling of links in a near optimal location aware manner. The crawler outperforms earlier distributed crawling approaches without a significant overhead.

1 Introduction

The challenging task of indexing the web (usually referred as web-crawling) has been heavily addressed in research literature. 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 [6], it is estimated that the maximum web coverage of each search engine is around 16% of the estimated web size [8].

Distributed crawling [10, 11, 9, 1, 3, 4] was proposed to improve this situation. However, all the previous work was 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 significantly improving the quality of the results, the previous proposals were not at all optimized.

In this work, we describe a near-optimal, for the distributed crawlers, URL delegation methodology, so that each URL is crawled from the nearest crawler. The approach, called IPMicra, facilitates crawling of each URL from the nearest crawler (where nearness is defined in terms of network latency) without creating excessive load to the Internet infrastructure. Then, the crawled data is processed and compressed before sent to the centralized database, this way eliminating the network and processing bottleneck in the search engine's central database

site. 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, provides load balancing taking into consideration the crawler's capacity and configuration. Furthermore, it dynamically adjusts to the changing nature of the Internet infrastructure itself.

This short introduction is followed by a brief description on related work, giving emphasis to UCYMicra, a distributed crawling infrastructure which we extend to perform location aware web crawling. We then introduce and describe location aware web crawling. Section 4 describes and evaluates our approach toward location aware web crawling, called IPMicra. Section 5 summarizes the advantages of IPMicra. Conclusions and future work are presented in section 6.

2 Related work

While the hardware bottleneck is easily (but not cheaply) handled in the modern web crawling systems with parallelization, the network bottleneck is not so easily eliminated. In order to eliminate the delay caused by the network latency (occurred mainly due to the network distance between the crawler and the target URLs), the modern crawlers issue many concurrent `HTTP/GET` requests. While this speeds up crawling, it does not optimize the utilization of the still limited network resources, and the overhead in hardware and network for keeping many threads open is very high. The network resources are not released (in order to be reused) as fast as possible. Furthermore, in most of the cases, the data is transmitted uncompressed (since most of the web-servers have compression disabled), and unprocessed to the central sink (the search engine), thus, its size is not reduced. Finally, the whole crawling process generates a big workload for the whole Internet infrastructure, since the network packets have to go through many routers (due to the big network distance of the crawler and the servers).

There were several proposals trying to eliminate the bottlenecks occurred in centralized crawling, such as [2, 5]. However, in the authors' knowledge, none of them was able to solve the single-sink problem. All of the crawled data was transmitted to a single point, uncompressed, and unprocessed, thus, requiring great network bandwidth to perform the crawling function (the nature of centralized systems). Thus, realizing the limitations of centralized crawling, several distributed crawling approaches have been proposed [10, 11, 9, 1, 3, 4]. The new approaches are based in the concept of having many crawlers distributed in the web, using different network and hardware resources, coordinated from the search engine, sharing the crawling workload. The crawlers sometimes run in the search engine's machines [3, 4], sometimes in customers' machines [11, 10], and sometimes in third parties (normal Internet users) [9]. The innovation in these approaches is that they mostly eliminate the network bottleneck in the search engine's site, since they reduce the size of the data transmitted to it (due to data processing, compression, and filtering before transmission). More exactly, while distribution introduces one more step - the step of transmitting the data

from the distributed crawlers back to the central search engine database - distributed crawlers do eliminate the network and processing bottlenecks in the search engine's site, since they can significantly reduce the size of the data (due to filtering and compression), and prepare the data (using distributed resources) for integration in the database.

As in the centralized crawlers, distributed crawlers also issued many concurrent `HTTP/GET` requests to minimize the network latency. However, as in the centralized crawling case, this approach is not the optimal, neither for network utilization, nor for the Internet infrastructure. More specifically, the distributed crawlers are forced to open many concurrent threads in order to cover the network latency, thus, they require more hardware and network resources. Furthermore, the network resources cannot be reused as fast as possible, since they are not optimally released. Finally, increased load occurs in the Internet infrastructure since the `HTTP/GETs` and `HTTP/HEADs` results are transmitted from the web servers uncompressed, unprocessed, and unfiltered, over a long network distance, through many routers, until they arrive in the distributed crawling points, for filtering and compression. *To remedy all these*, we now propose a truly distributed location aware web crawling, which minimizes the network latency in distributed crawling (between the distributed crawlers and the web-pages), speeds up the web crawling process, and also enables efficient load balancing schemes.

2.1 The UCYMicra System

UCYMicra [10, 11] was recently proposed as an alternative to distributed web crawling. Realizing the limitations of the centralized web crawling systems and several other distributed crawling systems we 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.

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 co-

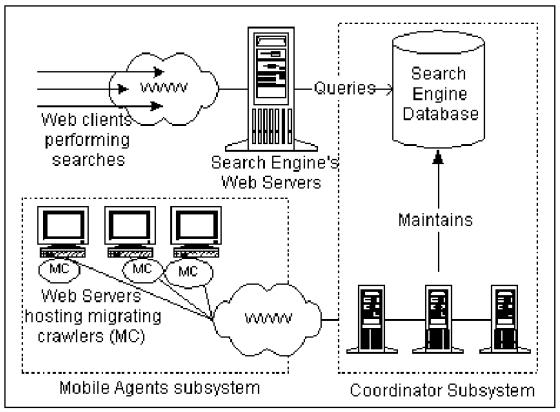

Fig. 1. UCYMicra basic components

ordinator 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 UCYMicra paradigm was easily received by 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 [12]. SETI users download and install a screensaver, which performs background processing while active, and sends the results back to the SETI team). 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 [11, 10]. 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).

3 Location aware web-crawling

The concept behind **location aware web crawling** is simple. Location aware web crawling is distributed web crawling that facilitates the delegation of the web pages to 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 purpose of finding the nearest crawler for each web-page is to minimize the time spent in crawling of the specific web-page, as well as the network and hardware resources required for the crawling function. This way, location aware web crawling can increase the performance of distributed web crawlers, promising a significant increase in web coverage.

Being distributed, the location aware web crawling approach introduces the load (small, compared to the gains of the approach) of transferring the filtered, compressed, and processed data from the distributed crawlers to the central database server. However, the search engine site's network is now released from the task of crawling the pages, which is now delegated in the distributed crawlers. This releases important network and hardware resources, significantly greater than the newly introduced load for transferring the data from the distributed crawlers back to the central search engine. Furthermore, optimization techniques, such as filtering, remote processing and compression, are enabled from the distributed crawlers and can be applied in the communication between the crawlers and the search engine, thus eliminating the network and processing bottlenecks in the search engine's site. In fact, distributed crawling, by combining filtering, processing, and finally compression, can reduce the size of the data transmitted to the search engine for integration in the database as much as one order of magnitude, without loosing any details useful for the search engine. Even further reduction in the size of data is available by adopting the distributed crawlers to the search engine's ranking algorithms.

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, the crawler with the smallest probing time was the one that could download the web page the fastest. 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 (e.g. UCYMicra) was actually simple. UCYMicra 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 "unaware" 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.

4 The IPMicra System

While location-aware web crawling significantly reduces the download time, building a location aware web crawler is not trivial. In fact, the straight-forward approach toward location aware web crawling requires each URL to be probed (i.e. `ping`)from all the crawlers, in order to find the most near web crawler to handle it. Thus, extensive probing is required, making the approach impractical. The purpose of IPMicra is to eliminate this impracticality. IPMicra specifically aims in reducing the required probes for delegating a URL to the nearest crawler. We designed and built an efficient self-maintaining algorithm for domain delegation (not just a URL) with minimal network overhead by utilizing information collected from the Regional Internet Registries (RIRs).

Regional Internet Registries are non-profit organizations that are delegated the task of handling IP addresses to the clients. Currently, there are four regional Internet Registries covering in the world: APNIC, ARIN, LACNIC, and RIPE NCC. 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. Via the RIRs a hierarchy of IP ranges can be created. Consider the IP range starting from the complete range of IP addresses (from 0.0.0.0 to 255.255.255.255). The IP addresses are delegated to RIRs in large address blocks, which are then sub-divided to the LIRs (Local Internet Registries); lastly they are sub-divided to organizations, as IP ranges, called subnets.

The IPMicra system is architecturally divided in the same three subsystems that were introduced in the original UCYMicra: (a) the public search engine, (b) the coordinator subsystem, and (c) the mobile agents subsystem. Only the public search engine remains unchanged. The coordinator subsystem is enhanced for building the IP hierarchy tree and coordinating the delegation of the subnets, and the migrating crawlers are enhanced for probing the sites and reporting the results back to the coordinator.

4.1 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. 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 it 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. Since the IP address of the machine that will host the crawler is known, we can immediately assign that subnet to the new crawler(e.g., crawler X

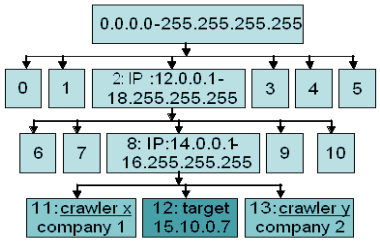

Fig. 2. A sample IP hierarchy. Subnets 11 and 13 belong to company 1 and company 2 respectively. Subnets 11 and 13 are assigned to crawlers X and Y respectively

is hosted by a machine belonging to subnet 11). 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. The populated hierarchy also enables calibrating and load-balancing algorithms (described later) to execute.

Updating the IP-address hierarchy is not difficult either. When we detect significant changes in the hierarchy data collected from the RIRs we rebuild the hierarchy from scratch (in our testing, rebuilding the hierarchy once a month was sufficient). Then, we pass the data from the old hierarchy to the updated one, in order to avoid re-delegations of already delegated URLs, and continue the algorithm execution normally. Any invalid re-delegations (i.e. important changes in the underlying connectivity of a web server or a web crawler), will be later detected, and the hierarchy will be calibrated (described later).

4.2 Probing

Since the introduction of classless IP addresses, the estimation of the network distance between two Internet peers, and subsequently, location aware web crawling, cannot be based in the IP addresses. For example, two subsequent IP addresses may reside in two distant parts of the planet, or, even worse, in the same part, but with very high network latency between. Therefore we needed an efficient function to estimate the network latency between the crawlers and the web-servers hosting the URLs.

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 (91% with ping and 92.5% when using HTTP/HEAD for probing), the crawler with the smallest probing time was the one that could download the web page the fastest. Thus, the migrating crawler having the smallest probing result to a web server is possibly the crawler most near to that web server.

Probing threshold: During the delegation procedure (described in detail in section 4.3) we consider a crawler to be suitable to get a URL if the probing result from that crawler to the URL is less than 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. In simple terms we can see the probing threshold as our tolerance on non-optimal delegation. During our experiments we found a probing threshold set to 50msec to give a good ratio of accuracy over required probes.

4.3 The URL Delegation Procedure

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. This is done in a *lazy evaluation* manner, that is, we try to delegate a subnet only after we find one URL that belongs to that subnet.

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 handled from that 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 exist, the new URL, and subsequently, the owning subnet, is delegated to this crawler. If there are more than one 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(section 4.2).

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, we sequentially ask 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), we delegate the target subnet to that crawler. If no probing satisfies the threshold, our 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.

The algorithm (see pseudo-code below) is executed in the coordinator subsystem.

```
for any newly discovered URL u {
  subnet s = smallestNonUnary(u);
  if (IsDelegated(s)){ // the subnet is delegated
    delegate u, s to the same migrating crawler;
    next u;
  }
  elseif (sameCompanySubnetDelegated(s.companyName)){
  // a subnet of the same company is delegated
```

```
    mc = migrating crawler that has the other subnet;
    mc.delegate(u, s) //the url and the subnet
  } else {
    while (s not delegated) {
      s = s.parent;
      if (IsDelegated(s)) { // check the parent
        mc = the migrating crawler that has subnet s;
        time = mc.probe(u);
        if (time<threshold)
          mc.delegate(u, s); //the url and the subnet
      }

      for every child of s until u is delegated {
        sch = s.child
        if (IsDelegated(sch)) { // check the child
          mc = migrating crawler that has subnet sch;
          time = mc.probe(u);
          if (time<threshold)
            mc.delegate(u, s)
        }
        if (allAvailableCrawlersProbed)
          delegate the subnet to the fastest crawler
      }
    }
  }
}
```

A URL delegation example: For clarity purposes an example is in order. The example references the IP address hierarchy presented in figure 2.

Subnet 2 in figure 2 has an IP range from 12.0.0.1 to 18.255.255.255. Subnet 8 is included in subnet 2 with an IP range from 14.0.0.1 to 16.255.255.255. Subnet 12 is a unary subnet for IP 15.10.0.7. The scenario includes probing for a URL that resides to IP 15.10.0.7. Querying the IP addresses hierarchy, we discover that the smallest subnet including the target IP is subnet 12, which however is unary. Thus, according to our algorithm, we ignore subnet 12, and use subnet 8 instead. Subnet 8 is not delegated in any crawler, so we check to see if any other subnet belonging to the same company is already delegated to any crawler. Assuming that no other subnet of the same company is delegated (organization name is stored in every node in the hierarchy), we continue by checking for neighbouring subnets that are delegated. Looking again in our hierarchy, we discover that while subnet 8 is not delegated to any crawler yet, subnets 11 and 13 (its children) are delegated to two different crawlers, x and y respectively. Therefore, we ask these two crawlers to probe the new subnet. If probing in either of the two crawlers' results in time less than the probing threshold, we delegate the new subnet to that crawler, or else we proceed to higher levels of hierarchy. However, since in this scenario, subnet 12 is a unary subnet, we delegate both subnets 8 and 12 to the faster crawler. Since the subnets 11 and 13 are already delegated and are lower in the hierarchy than subnet 8, this does not affect them (their delegation

supersedes the delegation of their father). Subnet 14, which is not yet delegated, stays un-delegated. If we need to delegate it in the future, we run the same algorithm until we find some crawler satisfying the probing threshold.

4.4 Load balancing and dynamic calibration

Our algorithm performs dynamic calibration of the URLs in order to follow the vastly changing Internet infrastructure. More specifically, the time required for each network action for each URL (i.e. `HTTP/GET`) is compared with the previous counts/statistics for the same URL. If the 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. In this way, with negligible processing, and no extra network overhead, the algorithm dynamically detects changes and calibrates the URL delegations.

IPMicra also performs efficient load-balancing. 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 a crawler gets overloaded, the coordinator removes the subnet(s) with the lower variance in their probing results (collected during their delegation, and stored in the coordinator), and delegates them to the next-best available crawler. 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. This heuristic performs well, and was preferred over other studied approaches (i.e. linear programming) due to the simplicity in implementation. Our tests showed that this heuristic was performing optimal decisions in more than 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.

4.5 Performance and Evaluation

The direct advantage and purpose of IPMicra is that it enables location-aware crawling in *distributed* crawling systems. As such, the evaluation of the new methodology must be focused in this exact point. In fact, what we need to compare is our distributed location-aware methodology with a representative of distributed crawling methodologies that does not account location during crawling. After all, distributed crawling *per-se* was already compared with centralized crawling [10, 11, 9, 1, 3, 4], and was found significantly better.

The case of various crawling optimizations that exist in other crawling systems (distributed or not) such as in-memory lexicon [2], DNS caching [5] and hardware acceleration [7] do not affect our approach, and do not need to be taken into account to our experiments. As such, we only need to examine the effects of the proposed location-awareness in distributed web crawling. Thus, we compare the IPMicra approach with a typical representative of distributed crawlers, i.e.

UCYMicra. We selected UCYMicra over other distributed crawlers for two reasons: (i) we had the UCYMicra crawlers already up and running, in a network of collaborated universities and organizations, and (ii) IPMicra was built over UCYMicra, so, it was using the same code to download and process pages, with the same optimization functions. Namely, the only practical difference between the two approaches was location awareness, thus, our measurements would be as objective as possible. That is, any differences in performance between the two crawlers, the location-aware Vs the location-unaware crawler, would be only due to the location awareness.

Before proceeding to describing our experiments and results, we have to stress once more that our selection to compare IPMicra with UCYMicra and not other approaches is because we now want to evaluate only the location-aware web crawling schema, and not several other optimization techniques existing in other proposals (either for distributed or for centralized crawling). In fact, most of these techniques can be applied in any distributed crawler, and in IPMicra. Thus, such techniques can combine with IPMicra and improve IPMicra's performance even more. IPMicra *per se* is also applicable in any other distributed crawler, in order to perform location aware web crawling.

We performed a two-phase evaluation and repeated each experiment several times to get statistical significance.

The first evaluation phase involved three experiments, with four coordinating crawlers, hosted from affiliated universities in four distinct geographical locations(USA, Greece, Cyprus, and London). The experiments included distributed crawling of 1000 distinct domain names, using three different variations: (a) Location unaware distributed crawling i.e. UCYMicra, (b) Optimal location aware distributed crawling, and, (c) IPMicra. Location unaware distributed crawling was performed with an enhanced version of UCYMicra, which was performing a random delegation of the URLs to the crawlers. The optimal location aware distributed web crawling was performed from another version of UCYMicra, which probed (with HTTP/HEAD) each URL from all the crawlers prior each delegation, and delegated each URL to the most near crawler (this was approaching the theoretically optimal location aware delegation). IPMicra was also executed in the same setup, as described before. However, since IPMicra's performance depends on the probing threshold, we experimented with many different thresholds (25msec to 125msec). We found a threshold set to 50msec with HTTP/HEAD as the probing function to give a good ratio of (accuracy:#required probes). Setting the threshold to a lower value i.e. 25msec was resulting to much higher accuracy (more than 90% optimal delegations) but required more probes for each delegation.

We found that location aware web crawling required **one order of magnitude less time (average 1/10th)** in the downloading process from the location-unaware version. The case was very similar with IPMicra, which also required one order of magnitude less time (with probing threshold set to 50msec and using HTTP/HEAD for probing function) compared to location unaware web crawling. The evaluation results are illustrated in figure 3 (the worst-case

scenario is the case where each URL is assigned to the farthest crawler). Note that even with only four crawlers the benefits are tremendous. In fact, as the number of crawlers increases the benefits increase as well. We expect the IP-address hierarchy to be instrumental in identifying the optimal number of crawlers for optimal location aware crawling.

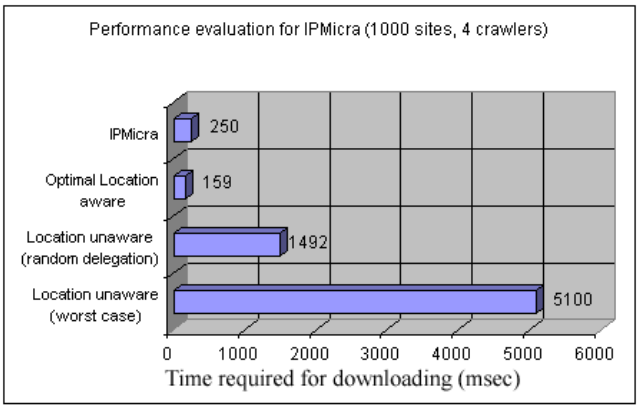

Fig. 3. IPMicra compared to the optimal location aware, the random, and the worst-case distributed crawling (1000 sites and 4 crawlers)

At the second evaluation phase, we included 12 crawlers (hosted in affiliated organizations and universities in USA, Europe and Australia) and 1000 randomly selected URLs - different than the previous. This experiment was to evaluate the accuracy of IPMicra in performing a location aware delegation, and the required probes for doing so. In this experiment, IPMicra was able to propose an optimal delegation in most of the URLs, by requiring very few probes. More specifically, with a probing threshold set to 50msec, IPMicra managed to perform the optimal delegation in 75% of the URLs, and required an average of only 3 probes per URL, compared to 12 needed for the brute-force approach presented in Section 3. With a probing threshold set to 25msec, IPMicra's accuracy was reaching to 90% accuracy (90% of the URLs were assigned in the nearest of the 12 crawlers), and required 6,5 probes for each URL. It is worth noting however that in all our experiments, the sub-optimal delegations were very near to the optimal ones, and always much better than a random delegation (from the delegation algorithm, one can realize that the maximum probing of any proposed non-optimal delegation was equal to the probing threshold, which however was significantly low in all cases). The effects of the probing threshold are illustrated in figure 4.

Due to practical difficulties (the difficulty of establishing controlled environment for our experiments in a number of distinct, world-distributed networks),

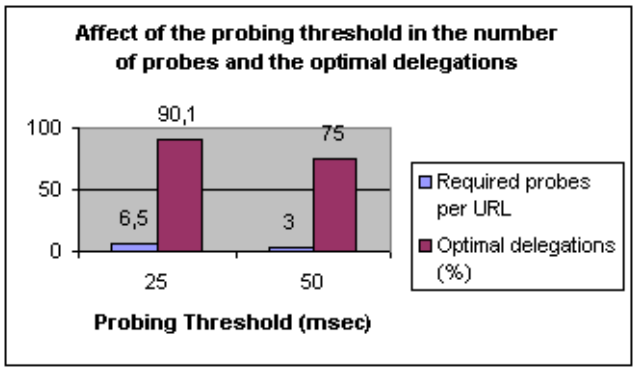

Fig. 4. Experimenting with probing threshold (25msec and 50msec), 12 crawlers and 1000 URLs

the two evaluations were made with a limited number of distributed crawlers and URLs. However, these crawlers were well distributed over the world (physically, and in network level), and they were significant for showing the advantages of the location-aware approach, and the effectiveness of IPMicra for performing location-oriented assignments of the IP subnets. Actually, we expect the approach to *react better with more collaborating crawlers*, since this will enable the algorithm to focus easier and faster to the most promising crawlers (without more probes). The crawlers populate in the hierarchy in a way that a number of IP subnets is automatically delegated (without probes) to them, and this knowledge is used for more effective future delegations. After all, the theoretical-ideal case of one IPMicra crawler in each subnet would result in 100% effectiveness of the approach - 100% optimal delegations, without any probing requirements (all the subnets would be optimally delegated to their own crawler). Furthermore, our experiments revealed an evolutionary nature of the approach (calibration in the course of time), promising more for the real-world deployment of the approach to hundreds of collaborated organizations, with the billions of URLs.

The adaptive/learning nature of IPMicra: In all our experiments, IPMicra was getting calibrated-optimized in the course of time, by facilitating *a priori* knowledge. For example, while the average number of probes for all the sites (phase 2 of the evaluation, with 12 crawlers) was 3 probes per URL, the average probing for the last 50 URLs was only 2.66 probes per URL. The fact that more delegations were performed in the IPMicra hierarchy - the hierarchy was getting *trained/calibrated* - was helping IPMicra to focus to the optimal crawler with less probes. The results of the previous experiment (with 6 and 12 crawlers) are also illustrated in graph 5. It is very important that the (linear) trendline in the graph is reducing, meaning that the required probes for each URL are also getting reduced in the course of time.

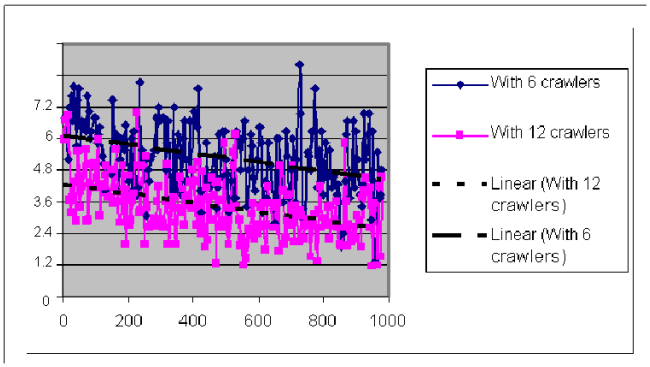

Fig. 5. The adaptive nature of IPMicra - Number of required probes per URL with probing threshold set to 50msec (for 1000 sites crawled from 6 and 12 crawlers)

5 Advantages of IPMicra

IPMicra has several advantages inherited from the mobile agents model, and its predecessor, UCYMicra. Furthermore, it supports load balancing and near optimal URL delegation. More specifically, IPMicra provides 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 not fully utilized 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 nearest 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 for URL delegation and re-delegation 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 introduces less overall load in the Internet infrastructure, since importantly less data is transmitted uncompressed over the Internet. The distance that the uncompressed data has to be transmitted (between the webservers and the distributed crawlers) is less or the two Internet points are connected with high bandwidth.
6. IPMicra has the important advantage of becoming dynamically calibrated in the course of time, for more focused (with less probes) searching for the

nearest crawler. Moreover, the system also detects important changes of the Internet's underlying network structure, and easily adjusts to them, to keep optimal delegations

Being distributed, IPMicra also inherits the advantages of distributed crawling. More specifically, not only it eliminates the enormous processing bottleneck from the search engine's site, by delegating the processing task to the migrating crawlers, but also it performs remote processing and compression (to the migrating crawlers) prior transmitting the results back to the search engine. The latter results to a significant reduction of the data transmitted back to the search engine's site (as in UCYMicra, we transmit **less than 1/20th** of the changed crawled data [10, 11]), without loosing any search-useful information. Also, useless conditional GETs(`If-Modified-Since` headers) and `HEAD` requests do not any more occupy network resources from the search engine's site, but are executed distributed. Moreover, due to the flexibility of the mobile agents paradigm, the whole system is upgradeable at real time (the migrating crawlers' code can be upgraded *live*), and uses negligible network resources for coordination. At the end, it is very promising and easily acceptable from the users, due to the security constraints that can be set to the migrating crawlers, and since it can offer a fully configurable crawling service for the web server administrators(similar services are currently sold from commercial search engines).

6 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 four 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.

IPMicra significantly improves the performance of distributed crawling by requiring one order of magnitude less time from a location unaware distributed crawler to crawl the same set of web pages. The performance is achieved just by re-arranging the URL delegations to the nearest crawlers. IPMicra also enables efficient load-balancing with negligible overhead.

This work 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 infrastructures 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. Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
3. Jan Fiedler and Joachim Hammer. Using the web efficiently: Mobile crawlers. In *Proceedings of the Seventeenth AoM/IAoM International Conference on Computer Science*, pages 324–329, San Diego CA, 1999. Maximilian Press Publishers.
4. Joachim Hammer and Jan Fiedler. Using mobile crawlers to search the web efficiently. *International Journal of Computer and Information Science*, 1(1):36–58, 2000.
5. Allan Heydon and Marc Najork. Mercator: A scalable, extensible web crawler. *World Wide Web*, 2(4):219–229, 1978.
6. Google Inc. Google, September 2003. http://www.google.com/.
7. Google Inc. Google search appliance, February 2004. http://www.google.com/appliance.
8. S. Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400(6740):107–109, July 1999.
9. LookSmart Ltd. Grub distributed internet crawler, 2003. http://www.grub.org.
10. 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.
11. 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.
12. SETI. Search for extra terrestrial intelligence, January 2004. http://setiathome.ssl.berkeley.edu/.