

# On Location Aware Internet

Stelios Erotokritou<sup>1</sup>, Odysseas Papapetrou<sup>2</sup>, George Samaras<sup>1</sup>, and Wolfgang Nejdl<sup>2</sup>

<sup>1</sup> University of Cyprus, Cyprus {cs02es1,cssamara}@cs.ucy.ac.cy

<sup>2</sup> L3S/University of Hannover, Germany {papapetrou,nejdl}@l3s.de

**Abstract.** An important aspect of performance for Internet-based applications is network delay (measured in terms of bandwidth and latency). The contribution and the basic motivation of Location Aware Internet is to enable clients to easily find the nearest (in terms of latency) out of a number of servers that can service a specific request. Location Aware Internet can significantly improve the performance of Internet applications such as WWW, web crawling and FTP. In this work we present our current findings, including two scalable approaches for implementing location awareness in distributed applications. Our approaches are compared to existing state-of-the-art approaches in a large-scale setup over PlanetLab. The work is motivated from and tested in a distributed web crawling application.

## 1 Introduction

Distributed Internet applications, the most famous being the web, are heavily depended on bandwidth and latency of their Internet connectivity. Low bandwidth or high latency can significantly decrease the performance of an application, irritate the user and increase failure rates of the application amongst others. The problem deteriorates in applications such as P2P and other highly changing-dynamic distributed database systems. In response, the network community is constantly working on improving Internet backbone *network level characteristics* such as routing algorithms. The distributed systems and Internet communities follow an orthogonal approach, suggesting a wiser facilitation of the duplication and mirroring that already exists in the Internet. Namely, they are suggesting selecting the nearest (in terms of network distance - which can be measured by latency amongst a few metrics) service provider replica, i.e. selecting the provider that will be able to provide answers the quickest, as far as network distance is concerned.

This work contributes to the second group of approaches, working in the application level. We try to provide efficient and scalable approaches for a node in a highly dynamic distributed system to select the nearest service provider over several possibilities. The two approaches suggested here, IPMicra and HiMicra, are feasible for large-scale Internet-based applications, and have moderate hardware and network requirements. Both approaches improve over time, optimizing their hierarchy-based structure for faster and more accurate detections of the

nearest replicas. *It should however be noted* that in this work we do not focus on finding the available replicas, but on finding the nearest replica from a *given* set.

Both our approaches, and the earlier state-of-the-art approaches are implemented and experimentally compared. We deploy a large scale experimental setup on PlanetLab, including more than 300 servers and 7000 clients. The experiments show that both our approaches are very promising for real-world usage and in most cases outperform the earlier approaches.

This short introduction is followed by a short description concerning Location Aware Internet requirements. Section 3 presents previous work on location aware Internet, and Sections 4 and 5 present our approaches, IPMicra and HiMicra. Section 6 documents our large-scale evaluation scenario and the comparison results. We finish with the conclusions and plans for future work.

## 2 Location Aware Internet

Location Aware Internet involves the ability of selecting the nearest server from a given group of servers able to offer a particular service. It should be stressed here that *nearness* refers to network distance and not to geographical physical proximity. A short physical distance between two Internet nodes, e.g. nodes residing in the same country, does not imply a short network distance and a small latency. Instead, expensive agreements between network operators are responsible for building the routing tables in the Internet infrastructure thus determining Internet latencies. It is often the case where a client faces a significant latency reaching a server which is physically in the same town, a few blocks away, compared to reaching a server in another part of the world.

Apart from physical distance, the current classless IPv4 addressing system does not enable any network distance estimation based solely on the IP addresses. Consecutive (in the same class C) IP addresses can belong to different companies, in completely remote locations, and with very different routing policies<sup>3</sup>.

Our experiments show the need for an easily deployed and sufficiently generic middleware supporting Location Aware Internet. In fact, applying location awareness in a distributed crawling setup we were able to reduce the time spent in network functions by one order of magnitude, compared to the location unaware counterpart. Even more impressive, the reduction compared to the worst-case of the location-unaware counterpart was approximately two orders of magnitude.

Optimally, Location Aware Internet services should be supported in the layer of a distributed middleware. This could be a pre-routing step in the middleware layer before the request gets translated to a network request, or a service directly invoked from the client. In a more direct and stand-alone solution, this functionality could be plugged in to existing load balancing functionalities such as DNS-based load balancing, returning the nearest available server with the lowest network distance with respect to the clients network location.

---

<sup>3</sup> The currently emerging IPv6 protocol *may* be able to support some kind of distance estimation based solely on IP addresses, yet this still remains unclear.

### 3 Related work

In this section we present two current state-of-the-art approaches for location awareness for the Internet. The first approach, Global Network Positioning, is based on providing global  $N$ -dimensional coordinates to the nodes, which minimize the error based on real, measured distances between them. IDMaps on the other hand is based in an extensive use of probers, called tracers, which are distributed over the Internet. Both these approaches were implemented and compared with IPMicra and HiMicra during our evaluation.

#### 3.1 GNP

The Global Network Positioning (GNP) framework was suggested in [6] as a new approach to predict Internet network distances. The approach models the Internet as a geometric space upon which a distance function could be applied. The geometric coordinates for each of the Internet hosts are calculated in a distributed manner.

The GNP framework used the concept of Landmarks which the authors defined as a set of cooperating hosts which would serve as a frame of reference for ordinary host coordinate deduction. The GNP framework consisted of a two part architecture.

In the first part of the GNP architecture, a small distributed set of hosts known as Landmarks were used to provide a set of reference coordinates, necessary to orient other hosts in the chosen geometric space  $S$ . The  $N$  number of Landmarks would simply measure the inter-Landmark round-trip times using ICMP ping messages and take the minimum of several measurements as their distance for each path. Using these distances, the computation of the  $k$ -dimensional coordinates where  $k < N$  could be cast as a generic multi-dimensional global minimization problem solved by the Downhill Simplex Method [5] which would derive the coordinates of the Landmarks.

In the second part, ordinary hosts could actively participate by deriving their own coordinates using the coordinates of the Landmarks in the geometric space  $S$ , and the minimum of several measurements for the distance between themselves and each of the Landmarks. Like deriving the Landmarks' coordinates, the computation of the host coordinates could also be cast as a generic multi-dimensional global minimization problem solved using the Downhill Simplex method.

Under the GNP framework, distances between hosts can be found using the distance function defined in the geometric space upon the computed coordinates of the hosts (i.e. euclidean distance). For finding the nearest service provider using the GNP framework, the users would place themselves in the framework and derive coordinates of their network position. They would then use the framework to retrieve the coordinates of all the service providers. The distance between the client and each of the service providers could then be calculated using the distance function, so that the one with the smallest distance is detected.

### 3.2 IDMaps

IDMaps was suggested in [2,3]. In its general structure, it was proposed as an Internet-wide architecture, which would measure and disseminate distance information on the global Internet. The primary motivation of IDMaps was to provide a service that would be able to give an estimate of the distance (in latency) between any two valid IP addresses on the Internet.

The IDMaps Architecture has a three-layer structure consisting of Tracers, IDMaps Servers and the End User applications.

The Tracers layer is the layer responsible for collecting the distances via probing. It includes three types of objects:(a) the Address Prefixes (APs), (b) the Tracers, and finally, (c) the raw distances - further differentiated into two types, those between Tracers (Tracer-Tracer VLs) and those between Tracers and APs (Tracer-AP VLs).

Address Prefixes were defined by the authors as a consecutive IP address range within which all hosts with assigned IP addresses are equidistant (with some tolerance) to the rest of the Internet. Tracers were defined as systems or light-weight measurement processes that would be widely deployed on the Internet over time. These tracers would optimally be distributed around the Internet so that every AP would be relatively close to one or more Tracers.

The functionality of the Tracers would be to measure the distances between themselves and all other Tracers (Tracer-Tracer VLs) and the distances between themselves and the APs (Tracer-AP VLs). In turn they would report their results to the IDMaps servers. The communication between the servers and the tracers follows the Distance Information Protocol (DIP) [4]. DIP is used for reporting measurement results from Tracers to Servers and sending feedback messages from Servers to Tracers.

The IDMaps Servers combine the measurement results from all the Tracers and use triangulation to build a distance map of the virtual network topology (in the APs granularity). They are thus able to provide a query/reply service to end-user applications by estimating network distances. When a client application (e.g., web browser) queries for the distance between two hosts, the IDMaps Server identifies the corresponding APs of the two hosts and estimates the distance between these two APs by calculating the length of the shortest path in the distance map, using triangulation.

## 4 IPMicra

IPMicra was proposed in [8] and was one of the first approaches to use detailed analysis of the allocated IP address space for the implementation of Location Awareness on the Internet. IPMicra can inexpensively and effectively find the closest (to the requesting client) server - out of a set of available servers, to provide a particular service.

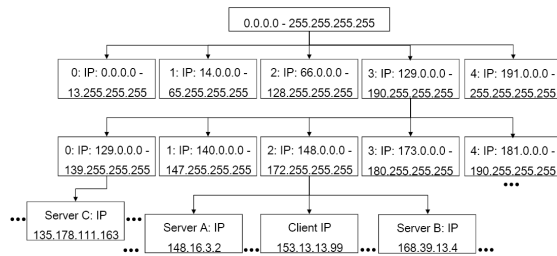
IPMicra is different from GNP and IDMaps in the sense that it does not use coordinates for the nodes or build a virtual network topology. All the service

providers (not necessarily of the same service) are placed in a hierarchy, which is then used for finding the nearest service provider for each service with respect to the requesting client.

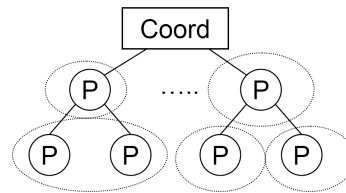
IPMicra was originally proposed in the context of a location-aware distributed web crawling application (thus the acronym is built from IP-addresses MIgrating CRAwlers). In its original setup, a number of available distributed crawlers provided the service of web crawling to the web. IPMicra’s purpose was to guide the delegation procedure of each web site to its nearest available web crawler, the one that would require the least amount of time to download it.

However, the location-aware IPMicra service is not bound to distributed web crawling. The experimental setup using the web crawling model was only an application presenting the capability of IPMicra as an effective tool for Location Awareness and Location Aware Internet. This generic concept could be used in many different applications available on the Internet and the WWW, for instance for mirror selection, Peer to Peer, and FTP.

IPMicra is based on information collected from the Regional Internet Registries to build an IP address hierarchy (Fig. 1), which is then used for providing a location aware service.



**Fig. 1.** IPMicra: Probers and sites populate the IP-Micra hierarchy



**Fig. 2.** HiMicra: The probers (P) are building the HiMicra hierarchy. Each prober is assigned multiple IP addresses ranges

#### 4.1 Regional Internet Registries

Regional Internet Registries (RIRs) are nonprofit corporations established for the purpose of administration and registration of Internet Protocol (IP) address space (both IPv4 and IPv6) and Autonomous System (AS) numbers. There are currently 5 RIRs in operation, (a) American Registry for Internet Numbers (ARIN) for North America, (b) RIPE Network Coordination Centre (RIPE NCC) for Europe, the (c) Asia-Pacific Network Information Centre (APNIC) for Asia and the Pacific, (d) Latin American and Caribbean Internet Address Registry (LACNIC), and (e) African Network Information Centre (AfriNIC).

Each of the RIRs hold information for the IP address space allocation that they are responsible for in their respective RIR database. For each address space

allocation they include details such as the address space allocated and details of the persons or organizations to which the address space is allocated to, which includes address and country of organization, name of assigned administrator to the address space, organization name, and other details. This data is often referred to as WHOIS data.

These RIR databases can be made available after request from the RIR organizations. For the IPMicra Algorithm it was necessary to obtain the RIR data. Specifically, the required data were the following: For each subnet we required: (a) the subnet id, (b) the subnet name, (c) the administrator name and / or (where applicable) the company name, (d) the subnet range (the first and last IP in the subnet address space).

## 4.2 The IPMicra algorithm

The basic idea behind IPMicra is to employ data collected from the Regional Internet Registries for performing delegation of subnets to the nearest from a set of available servers which can provide a particular service. The delegation process of IPMicra requires the construction of the IP hierarchy tree.

Using the WHOIS data collected from the RIRs, IPMicra builds and maintains a hierarchy with all the IP ranges (IP subnets) which have been allocated to organizations. Apart from the data for the IP subnets, IPMicra also maintains information concerning the company to which the subnet belongs. With the use of this IP address hierarchy, IP address ranges can be assigned to the “nearest” available request server, which will be able to serve a request with the least network latency.

The initial step of the IPMicra algorithm is to populate the hierarchy with the available servers offering a service. Knowing the IP address of one available server offering a service, IPMicra immediately assigns the subnet to which that particular IP address belongs, to that server (for this particular service). In this way the various servers populate the hierarchy.

An optional step then is to train the hierarchy. The IP address hierarchy is “*trained*” using a number of target IP addresses. For each training target IP address all the available servers measure their distance to the respective IP address. The training target IP address is delegated to the closest server to the site in terms of the used probing measurement. Training of the IP address hierarchy is carried out in order to increase the accuracy of the results and effectiveness of IPMicra. While training helps, our experiments show that there is no requirement for excessive training; In our evaluation scenario involving more than 300 servers, 500 URLs were sufficient for training purposes.

The hierarchy can then enable a location aware service, as explained in section 4.3

*Updating the IP address hierarchy:* The IP address hierarchy is based on the RIR data. Updating the hierarchy with the daily or monthly updates of this data is not difficult [8]. However, in our case, we avoided performing these updates, keeping the hierarchy constant. In our experiments, the performance of IPMicra

did not decrease, even after two months of keeping a constant hierarchy. This is mainly because IPMicra detects and corrects out-dated/expired delegations.

### 4.3 The IPMicra delegation process

Based on the assumption<sup>4</sup> that the sub-networks belonging to the same company or organization have fast interconnection between them (even if they reside far away from each other), IPMicra uses the organization’s or administrator’s name to delegate the different domains to the available servers. In fact, instead of finding the nearest server for each IP address, the nearest server for a whole IP range (a subnet in the RIR data) is found. We thus refer to this procedure as delegating IP subnets to the available servers.

The delegation is based on probing (i.e. PING), for distance measuring between the nodes. Thus the delegation process’s accuracy is parameterized by the **probing threshold**. The probing threshold defines the maximum acceptable probing time from a server to a client-target IP, for concluding that the server is near enough for the given client.

IPMicra first finds the smallest subnet from the IP hierarchy that includes the target(client’s) IP and checks to see if that subnet has already been delegated to an available server. If so, the target IP is delegated to that server. If not, IPMicra checks the hierarchy to find whether another subnet that belongs to the same company of the target IP has already been delegated to one or more servers. The subnet is assigned to the available server with the lowest probe, less than or equal to the acceptable probing threshold.

If the above search is unsuccessful, IPMicra navigates the IP-address hierarchy *bottom-up*, each time trying to find the most suitable server to which the subnet should be delegated. The parent subnet is searched and all the already-delegated subnets included in the parent subnet’s children are detected. IPMicra sequentially probes the target IP from the servers delegated to those subnets until an acceptable probing is found. If no probing satisfies the probing threshold, the search continues to higher levels of the IP hierarchy subnet tree.

In the rare case that none of the available servers satisfies the probing threshold, the subnet (and consequently the target IP) is delegated to the server with the lowest probing result.

The reader is reminded that IPMicra can offer location awareness for more than one type of services. The IPMicra hierarchy can be enriched with all kinds of services and nodes, and the matching of services to clients can be performed based on criteria (i.e. request for the nearest server offering **Service X**).

Our large-scale evaluation (Section 6) shows that IPMicra performs sufficiently accurate delegations, requiring only limited probing. It also scales well in Internet-size applications, due to the IP-based hierarchy which significantly reduces the probing requirements. More importantly, it copes dynamically with routing changes; in fact, previous evaluation [8] showed how it improves during

---

<sup>4</sup> This assumption is not restrictive. If it is not true, IPMicra algorithm will detect it and not use it

time, since the hierarchy gets more trained and keeps being updated with the infrastructure changes.

## 5 HiMicra

Similarly to IPMicra, HiMicra was also initially proposed using the distributed web crawling model [7]; in fact its early evaluation was through a similar simulation procedure as that of IPMicra, as was described in the previous section.

The initial evaluation showed that HiMicra is very promising for Internet-scale applications. However, just as IPMicra, the capabilities of HiMicra do not lie solely in the web crawling model. In fact, HiMicra is a generic and complete location-aware Internet middleware, which can be used in any distributed Internet-scale application.

HiMicra, unlike GNP and IDMaps, does not use coordinates or virtual network topology maps or the IP address hierarchy. Instead, HiMicra is based on a hierarchy created from the available servers which are able to provide a service for a specific application. This enables the client to inexpensively find the nearest server offering a specific service.

The hierarchy in HiMicra is built in a network-oriented approach, however, unlike IPMicra, it is not build from the RIR data. The available servers are organized in branches - in network neighborhoods, so that each group of siblings is expected to have similar network latencies to the rest of the Internet.

### 5.1 Constructing the HiMicra Hierarchy

The hierarchy in HiMicra(Fig. 2) is built in a network-oriented approach. To construct the hierarchy, a set of training URLs (also called training sites, having the same functionality as landmarks in GNP) needs to be selected. A moderate number of training URLs is sufficient for a good hierarchy to be built (i.e. 500 URLs). Our results have also shown that choosing the URLs to encompass a distributed data set (i.e. each site from a different network) can significantly increase the performance of the results. In our experiments, a random selection of URLs from a moderate-size URLs database built in a *breadth-first* crawling approach, proved to uniformly select sites from around the world.

All the training URLs are probed (i.e. with PING) from all the available servers. The probing results are required for the HiMicra hierarchy to be constructed. Note that this *URL-based* training makes no assumption for the location-aware application. The location-aware application can be anything, having no relation with URLs, i.e. a P2P system. The reason we select URLs for probing is because the web servers *almost always* respond to PING messages.

Imagining an  $N$ -dimensional Euclidian geometric space  $S$  where  $N$  is the size of the training set, we can place each of the servers in  $S$ , giving each server *coordinates* based upon the collected measurement distances. That is, the coordinates will be  $N$ -dimensional, and they will be derived using the distances between each of the servers and each of the training URLs.

To start building the hierarchy we create an artificial node which will act as the root of the hierarchy and represents the coordinator of the location-aware service. The artificial node is given the coordinates of the origin of the imaginary geometric space  $S$ . Since this node acts as the root of the hierarchy, all servers that we have at our disposal are descendants of this node in the hierarchy.

The HiMicra hierarchy can be built with any number of branching factors (denoted with  $BF$  for short) meaning maximum number of children that a parent node in the HiMicra hierarchy can have (a typical number for  $BF$  was 2 or 3 in our experiments). To proceed with building the hierarchy after constructing the artificial root node (level 0), we select the  $BF$  number of nodes, which, along with the root node, create the *simplex* in our space  $S$ , whose total surface area of polygonal faces is greatest out of all possible combinations.

After selecting the  $BF$  nodes that will act as the children of the artificial root node (level 1), the HiMicra algorithm separates the remaining descendants of the root as descendants of each of the  $BF$  children that have already been selected (the reader is alerted for the difference between descendants and children). For each node of the remaining descendants of the root, we find the Euclidian distance between the node and each of the selected children of the root in our geometrical space  $S$ . The node is assigned as a descendant to the selected child node with the smallest Euclidian distance. After the separation, each of the root descendants now also becomes a descendant of one of the children.

Once the process of assigning the descendants has been completed, the procedure which was described for the artificial root node is recursively repeated for its children and the set of descendants each of the children now has. The children in turn will have their own children with their own descendants and they too will recursively execute the same procedure. This procedure is carried out until the node executing it has no descendants, or when it has a number of descendants lower than that of  $BF$ . In the second case, there is no need to find the combination of the greatest simplex as described earlier as it is not possible to define the simplex. Instead, the descendants it has are termed as its children (they become leaves in the hierarchy) and the process ends. When all the nodes have carried out this procedure, the HiMicra hierarchy is built and ready to provide a location-aware service.

Due to the construction mechanism of the HiMicra hierarchy, the similarity between siblings in the hierarchy is expected to grow as we move down to the hierarchy (toward the leaves). In fact, the algorithm proposes as siblings (in the leaves level) mostly nodes residing in the same network, since they are forming the smallest simplex. On the other hand, very distant nodes (in network terms) can be proposed as siblings in internal, high levels of the hierarchy i.e. near the root.

*Updating the HiMicra hierarchy:* It is possible for the HiMicra hierarchy to get out-dated. This can result to false proposals from HiMicra for the nearest service provider. These changes can easily be detected, using a time history of the time required to service a client request. With this method, we can easily become aware of any significant deviations from expected time values. A relative

frequency of these occurrences will signal the probability of errors in the hierarchy, or that the hierarchy no longer conforms with the network structure. In this case, the HiMicra hierarchy has to be rebuilt from the beginning so that it conforms to the new network infrastructure and conditions. However, since changes which could occur in the network structure are rare, the number of times the hierarchy will have to be rebuilt will be very low (in our experiments, we did not observe significant decrease in the HiMicra proposals during a two-months timespan with the same hierarchy); therefore the cost of re-building the hierarchy is of no relevant concern.

## 5.2 The HiMicra delegation Process

HiMicra uses the generated hierarchy to provide a location-aware service to a requesting client. More precisely, it performs a *best-first* search in the hierarchy, in order to inexpensively locate a suitable server to handle a specific client request. A suitable server for a client would be a server that satisfies an acceptable **probing threshold** in respect to the client. The value of the acceptable probing threshold accordingly affects the effectiveness of HiMicra. Our experiments have shown that as the acceptable threshold value increases, the results of HiMicra deteriorates in accuracy, yet, the required probings are significantly reduced.

The HiMicra location aware service can operate in one of two ways: (a) in IP address granularity, or (b) in subnet granularity. They are both very similar, but the latter decrease the number of probing measurements required without harming the accuracy.

*IP address granularity:* The coordinator is responsible for coordinating the best-first search algorithm and requesting the probing of the requesting client site from the servers. Thus, when the coordinator is queried for a location aware service, it first asks from the top-level servers in the hierarchy to probe the domain of the requesting client site. The probing results with the identity of each server are then added in an ascendingly ordered list. The coordinator then removes the smallest-first probing result from the list. If the probing result is smaller than or equal to the acceptable probing threshold, then an acceptable server with respect to the client has been found. Thus the client's location is delegated to that server. Otherwise, the coordinator asks from the children of that server in the hierarchy to probe the domain of the client and add their probing results in the ordered list. The coordinator then continues in a similar manner, removing the first probing result from the ordered list, and asking the respective server to probe the IP address of the client, until a server with a probing result less than or equal to the acceptable threshold is found. The search also ends in the case where all the servers are probed, and none are found suitable for the client in terms of distance and acceptable threshold, in which case, the client's site is delegated to the best of the servers (according to the lowest of the probing results).

*Subnet granularity:* The second service approach for HiMicra also uses RIR data to detect subnets. Note however that not all the RIR data hierarchy is used (as in IPMicra). HiMicra uses only the IP ranges defining the leaves of the

RIR hierarchy, to detect the smallest subnets. It then uses this information to delegate subnets to servers, instead of delegating the single client IPs. It therefore attempts the procedure of the first service approach that was explained above only upon subnets that have yet to be met and delegated.

Upon receiving a request, the coordinator identifies the subnet of the client IP's and checks if it has already been delegated to a server. If so, it returns the already-identified server without any probing. Otherwise, the same steps as were described for the IP address granularity configuration are executed. Only this time not only the IP address is delegated to the server, but also the complete subnet is delegated to that server. The motivation behind the second approach is that anything of the same subnet is expected to be *network-close* to each other; thus requests originating from the same subnet are expected to be optimally served by the same server. This technique saves significant probings, what we defined as a cost to the implemented algorithms.

## 6 Evaluation

This work involved two evaluation requirements:(a) evaluating the significance of Location Aware Internet, and (b) comparing the state-of-the-art Location Aware Internet approaches on their accuracy and probing requirements. Both the evaluation experiments were performed over a large-scale setup, including around 500 service providers deployed over PlanetLab and 10000 clients. After removing suspicious results: (a) URLs with a permanent downloading failure from most the clients, (b) clients having very high download time to all URLs, or (c) clients with high download time variance for the same URL - each download time averaged over 4 downloading attempts, we were left with 316 service providers and 7086 clients<sup>5</sup>. From now on, we will be referring to this filtered data set of this experimental setup, involving only the filtered data. To the best of our knowledge, this is the most significant and largest experiment for location aware Internet involving real Internet measures, under real network load.

### 6.1 Evaluation setup

The trivial approach in evaluating location aware Internet would involve deploying an identical service on a part of PlanetLab nodes, and then using other randomly selected nodes to request the services. The time spent in network functions in location aware and the location unaware setup would then be compared, to see the significance of location aware selection. This however would limit our experiment to the *closed* network of *mostly academic* PlanetLab nodes, without interacting with real-life Internet services.

---

<sup>5</sup> For practical reasons, as we explain later, the functionality of the servers and the clients was reversed, yet without changing the results

*Reversing the experimental setup to get a bigger-scale experiment:* We thus followed a different approach based on building an equivalent setup by reversing clients and servers: *instead of trying to find the nearest of the available servers for each client, we tried to find the nearest client for each server*, so that each server is at the end serving exactly one client and the total network time is minimized. The advantage of this approach is that it enables us to use for service providers arbitrarily many publicly available web servers, residing outside of the PlanetLab network and under real network load scenarios. The clients for these servers are downloaders/crawlers running upon PlanetLab nodes, and our target is to download each server (one URL from each web server) from exactly one client. Note the similarity of the above scenario with a distributed web crawling setup, where optimally each server should be downloaded from its nearest client. In the web crawling scenario, the clients are actually the web servers, requesting a crawling service which can be offered from any of the available web crawlers.

The evaluation included downloading all web-servers (only one URL from each web-server) from all clients-downloaders. Each network interaction was timed, and averaged over 4 attempts. The multiple attempts for each download were also used for detecting malfunctioning nodes, i.e. servers or clients, which, for any reason (e.g. temporary failures), were having many failures. These clients or servers were filtered from the experimental results, for not risking integrity of the final evaluations. The filtering resulted to a total of 316 PlanetLab nodes and 7086 URLs.

The downloading times were used to build the following scenarios:

**Location Aware Internet scenario:** In this case, each URL was downloaded from the nearest client

**Location Unaware scenario:** In this case each URL was downloaded from a random client. This is the performance expected from a current Internet setup

**Worst case scenario:** In this case each URL was downloaded from the farthest client. This denotes an upper bound for any random setup

The same evaluation setup was used for evaluating our proposed algorithms for Location Aware Internet - which are proposed in this work - and comparing them with earlier state-of-the-art algorithms, namely those of GNP and IDMaps. For this part of our experiments we also included probing each server (using the PING tool). In all our measures, probing and the actual downloading, we specifically disabled the use of DNS, network or other caching applications, so that our measurements would be as objective as possible. GNP was parameterized in the number of landmarks and dimensions, and IDMaps in the number of traces. However, due to space limitations, not all the evaluation setups are presented here.

All the experiments were repeated for two cases: (a) for near-concurrent execution: all the nodes were almost concurrently downloading the web pages, and (b) for night execution: each of the nodes was executing the downloading during their local night times of 1am to 5am. This was done in order to reduce external inferences from local Internet users. Furthermore, two probing approaches were

facilitated in each case: (i) time required for completing an HTTP/HEAD request, and (ii) PING time. Both the probing approaches were already evaluated in previous work [8], and serve well as an inexpensive distance metric for Internet. Since the results in the above combinations were similar, we now present only the combination for near-concurrent execution, using the PING tool for probing (for the complete results the reader is cited to the project website [1]).

## 6.2 Evaluation results

*Evaluating Location Aware Internet:* Our evaluation, including 7086 URLs and 316 PlanetLab nodes showed an improvement over one order of magnitude in the time that was spent to network functions when Location Aware Internet was used. Table 1 summarizes the results over 1000 runs. The 'Location Aware' case is the case where each URL was delegated to its nearest downloader, the downloader that could download it the faster. The 'Location Unaware' case included a completely random assignment of URLs to downloaders, while the worst case included the assignment of each URL to its furthest downloader (the one that would require the most time).

| Configuration    | Average download time per URL | Complete download time |
|------------------|-------------------------------|------------------------|
| Location Aware   | 52 msec                       | 368 sec                |
| Location Unaware | 713 msec                      | 5058 sec               |
| Worst case       | 7123 msec                     | 50479 sec              |

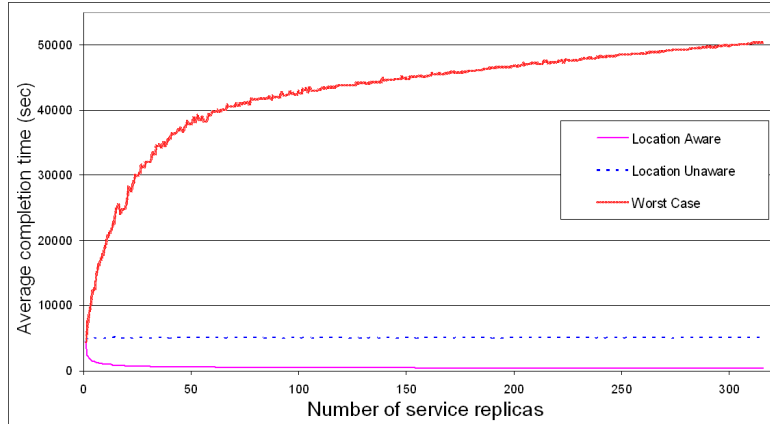
**Table 1.** Download times for the different cases

When each URL was downloaded from the nearest PlanetLab node, the average download time for each of the URLs was only 52 msec. The Location Unaware case required 713 msec, more than one order of magnitude worse than the optimal. Yet significantly worse was the performance of the worst-case scenario requiring an average of over 7 seconds per URL. The worst-case scenario serves as an upper bound for the random case (how bad can the performance get). The results are very similar to results reported in previous smaller-scale experiments again on real Internet measures [8].

While 7 seconds per URL, for the worst case, may sound too unrealistic based on the current user's experience, this is actually the case for specific pairings of URLs and clients, due to very bad routing strategies in the nodes between. The reader is reminded that our result processing even included a preliminary task of removing suspicious results, like too high downloading time for all URLs or frequent failures.

It was also expected that the benefit - performance increase - of Location Aware Internet would grow when more service replicas became available. For verifying this we re-ran the experiment for many scenarios, keeping a stable set of 7086 URLs, but each time increasing the number of PlanetLab nodes (downloaders) - from 1 to 316. The downloaders were each time randomly selected from all the available PlanetLab nodes. As expected, when many PlanetLab nodes were used, there was a higher probability that there was at least one of

them *near-enough* to each of the 7086 URLs, so that the performance did not suffer. The results were averaged over 250 executions for each number of downloaders. Figure 3 summarizes the results, comparing the Location Aware case (best case) with the worst and the Location Unaware (random) case.



**Fig. 3.** Benefit of Location Aware Internet grows as more service replicas are added

*Comparing the different Location Aware approaches:* We used the experimental setup described earlier to compare the accuracy of our suggested location aware algorithms with earlier state-of-the-art approaches. HiMicra on subnet granularity and IPMicra were compared with GNP and IDMaps. The evaluation task involved implementation of all four algorithms and their execution in the described setup for proposing the nearest of the PlanetLab nodes for each URL.

All the algorithms were evaluated on the same setup, involving 316 PlanetLab nodes and 7086 URLs. Initiating the algorithms involved a training procedure:

**IPMicra:** Several training scenarios were tested: 0, 100, 250, 500, 750, 1000 training URLs, leaving 7086, 6986, 6836, 6586, 6336, 6086 for testing URLs (the URLs involved in the training were not used in the testing set).

**HiMicra:** Several training scenarios were tested: 100, 250, 500, 750, 1000 training URLs, leaving 6986, 6836, 6586, 6336, 6086 for testing URLs (the URLs involved in the training were not used in the testing set).

**IDMaps:** Several scenarios of the IDMaps algorithm were run with number of Tracers ranging from 3 to 125 each one belonging in a different subnet.

**GNP:** Several scenarios of the GNP framework were run with number of Landmarks ranging from 3 to 25, each one belonging in a different subnet. Different number of dimensions ranging from 2 to 24 were also used.

IPMicra and HiMicra were parameterized by the probing threshold and the branching factor. IDMaps and GNP were parameterized by the number of Tracers, and number of Landmarks and dimensions used respectively. Each of the evaluation configurations was repeated 1000 times.

IPMicra’s and HiMicra’s results were averaged over the executions for each configuration. However, for GNP and IDMaps results, we selected the best results for each of the configurations. For instance, the maximum value for optimal and suboptimal delegations over the 1000 repetitions were selected for each configuration. This was because IDMaps and GNP propose that a wise selection of the tracers/landmarks *may* benefit the results. They propose some heuristics, which however are not always successful. We thus chose to use the best found results, so that the algorithms are not punished for these heuristics.

For each of the scenarios, the following were measured:

**Optimal proposals of nodes (percentage):** None of the algorithms was perfectly accurate in predicting the nearest of the available PlanetLab nodes for each URL. The correct-optimal proposals (where the proposed node was also the nearest node from all available for the URL) were measured, and the percentage of optimal proposal to all the proposals was calculated

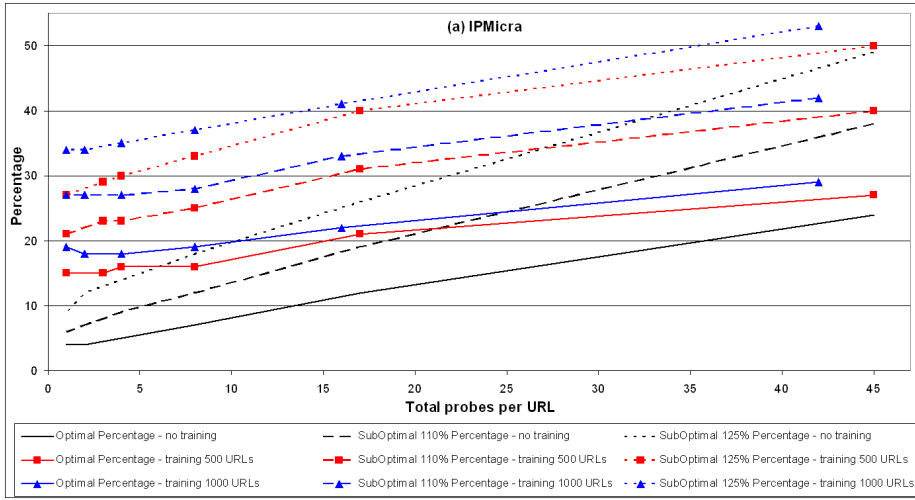
**Sub-optimal proposals of nodes (percentage):** The algorithms were often proposing sub-optimal proposals, which were very close to the optimal proposals and good enough for most web applications. We thus measured the percentage of the sub-optimal proposals, for which the download time was satisfactory: equal or less than (a) 110% and (b) 125% of the optimal download time - the time for the nearest node.

**Required number of Probes per URL:** All the algorithms were based on *probing* for constructing their respective structures of hierarchy/Internet map/Space and for placing the nodes upon them. These probes should be as few as possible, since they generate additional network load for the user and the Internet. We compare the required number of probes per URL since each of the evaluations involved a slightly different number of URLs. We also compare the probes required for the training of each approach.

**Average difference from optimal download time:** The ultimate target of Location Aware Internet is to reduce the total network time, so the complete time spent in downloading should be compared. However, since each of the scenarios involved a slightly different number of URLs, we compared the average of the difference between the optimal and the proposed download time for each URL instead of the total download time. This was found using the following equation:  

$$\sum_{\forall URL} \frac{optimalDownloadTime(u) - proposedDownloadTime(u)}{\#URLs}$$

Our evaluation results are summarized in Fig. 4 to 7. Figures 4 to 6 relate the number of average probes per URL in each approach with the accuracy of the algorithms, that is, the percentage of the optimal and sub-optimal delegations. Similarly, Fig. 7 relates the average difference from the optimal download time per URL (as defined earlier) with the number of the average probes per URL. The probing threshold in HiMicra and IPMicra are not mentioned directly in these graphs (they ranged from 10 to 100msec) since they are only for parameterizing the required quality of the algorithms, in expense of probes. Similarly, the tracers, landmarks and dimensions in the other two algorithms are expressed through the required probes. We follow this approach since we are not interested in any



**Fig. 4.** Performance of IPMicra compared to the required probes per URL

specific details of each of the algorithms, we are only interested in comparing their performance for specific probing requirements.

The IPMicra approach is the most accurate in detecting the nearest Planet-Lab node (optimal proposals). It is also the most successful in proposing near-optimal results (sub-optimal proposals) and keeps the average downloading difference from the optimal low, compared with the other approaches. The overall probing requirement for the IPMicra algorithm for this performance is also significantly lower than the others, denoting the scalability and suitability of IPMicra for a distributed environment. Our evaluation confirmed the suitability of IDMaps for large-scale scenarios, approaching the performance of IPMicra, yet with higher probing requirements. HiMicra also appears promising for exploitation in scenarios where the whole RIR hierarchy cannot be employed, while GNP appears not so suitable for such a large-scale distributed setup.

*Training requirements:* While IPMicra and HiMicra have higher probing requirements for initiating their hierarchy (Table 2) compared to GNP and IDMaps (Table 3), this is not a drawback of the algorithms since it is performed only once, when the hierarchies are initiated. The hierarchies of IPMicra and HiMicra are continuously maintained so there is no requirement for rebuilding them often. New nodes easily integrate in both the hierarchies, only by probing the set of training URLs (in the same way as GNP and IDMaps place new nodes).

| Training sites | IPMicra | HiMicra |
|----------------|---------|---------|
| 0              | 0       | N/A     |
| 500            | 158000  | 158000  |
| 1000           | 316000  | 316000  |

**Table 2.** Training requirements for IPMicra/HiMicra

| Tracers/Landmarks | 3 | 5  | 10 | 15  | 20  | 25  | 50   | 75   | 100  | 125  |
|-------------------|---|----|----|-----|-----|-----|------|------|------|------|
| IDMaps/GNP        | 3 | 10 | 45 | 105 | 190 | 300 | 1225 | 2775 | 4950 | 7750 |

**Table 3.** Training requirements for IDMaps/GNP in different configurations

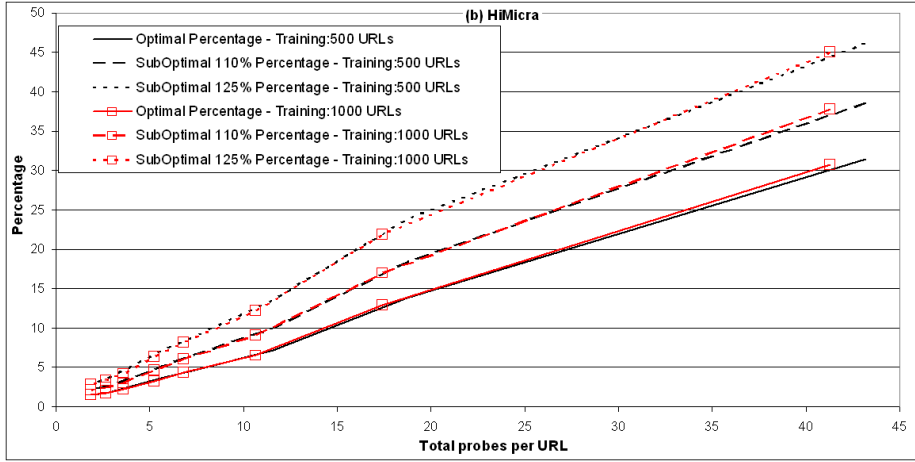


Fig. 5. Performance of HiMicra compared to the required probes per URL

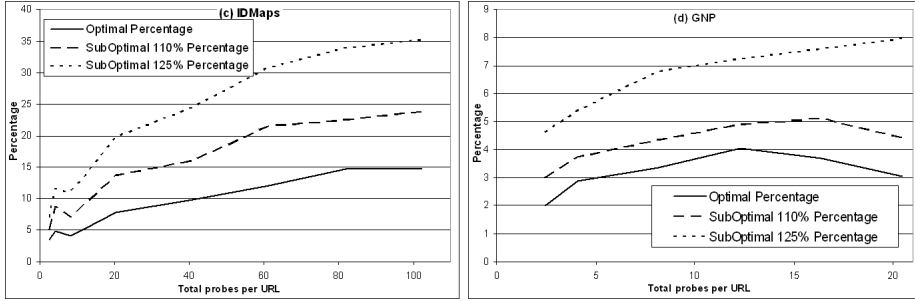


Fig. 6. Performance of IDMaps and GNP compared to the required probes per URL

## 7 Conclusions and Future Work

In this work we proposed two generic scalable Location Aware Internet algorithms. The proposed algorithms can easily reside in the middleware of an Internet-wide distributed system, be a stand-alone service, or even an inexpensive plug-in to a DNS load balancing system. Both the algorithms are based on a hierarchy; the first one that of IPMicra, enriched with data collected from the Regional Internet Registries, and the second populated from various Internet nodes, clients and servers. Both have the advantage of keeping constantly updated and optimized with any changes that may occur in the Internet layer. IPMicra outperforms the existing state-of-the-art approaches, and HiMicra performs in most scenarios also better than earlier works.

Continuing the work on Location Aware Internet, we plan to evaluate the algorithms at a larger *timescale* to see how optimized they can get during their lifecycle, and how they react on massive infrastructure changes i.e. a large backbone provider such as MCI changes its global routing policies. A fitting evaluation scenario for this is a Location-Aware selection for mirrors in large software packages, like SourceForge or other similar file mirroring systems.

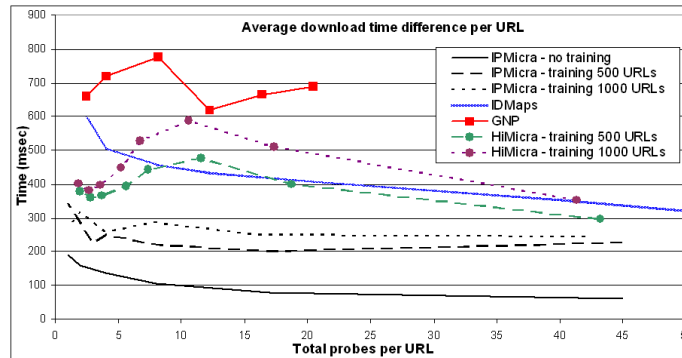


Fig. 7. Average difference from Optimal download time/required probes per URL

We are also examining other variations of the algorithms which will enable them to become fully distributed, and be solely deployed over weak and unstable collaborative systems, like P2P. Our current path to this is targeting a lower granularity hierarchy (compared to the hierarchies that are currently built from the two algorithms). Furthermore, specifically for HiMicra we are currently looking into ways of constructing a balanced hierarchy tree, as this will give a minimum hierarchy depth and upper bound the required probing to a theoretical maximum of  $O(\log_{\beta} n)$ , where  $\beta$  is the branching factor of the hierarchy and  $n$  is the number of nodes.

## References

1. Location aware internet project homepage. available at <http://www2.cs.ucy.ac.cy/locationawareinternet/>.
2. Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. IDMaps: a global internet host distance estimation service. *IEEE/ACM Transactions on Networking*, 9(5):525–540, 2001.
3. Paul Francis, Sugih Jamin, Vern Paxson, Lixia Zhang, Daniel F. Gryniewicz, and Yixin Jin. An architecture for a global internet host distance estimation service. In *IEEE INFOCOM 1999*, pages 210–217, New York, NY, March 1999. IEEE.
4. Yixin Jin, Beichuan Zhang, Vasileios Pappas, Lixia Zhang, and Suigh Jamin. DIP: Distance Information Protocol for IDMaps. In *IEEE Symposiums on Computers and Communications (ISCC)*, 2003.
5. J.A. Nedler and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.
6. T. S. Eugene Ng and Hui Zhang. Predicting internet network distance with coordinates-based approaches. In *INFOCOM*, 2002.
7. Odysseas Papapetrou. Location-aware web crawling with the use of migrating crawlers. Master’s thesis, Dept. of Computer Science, University of Cyprus, 2004.
8. Odysseas Papapetrou and George Samaras. Minimizing the network distance in distributed web crawling. In *CoopIS/DOA/ODBASE (1)*, pages 581–596, 2004.