Global Name Resolution Services for the Mobile Internet using a Distributed In-Network Hash Table (DIHT) *

Tam Vu, Akash Baid, Yanyong Zhang, Marco Gruteser, Dipankar Raychaudhuri

WINLAB, Rutgers University
{tamvu, baid, yyzhang, gruteser, ray}@winlab.rutgers.edu

ABSTRACT

This paper presents the design and evaluation of the distributed in-network hash table (DIHT) approach for realizing fast global name resolution services for the future mobile Internet. The proposed global name resolution service (GNRS) supports mobility at-scale in the future Internet in which a clean separation of the “name" of a network object from its “address" or point of attachment takes place. The GNRS is intended as a fast network-level service which can be queried by both end-points and in-network routers in order to obtain bindings between a GUID (globally unique identifier for the name) and the current routable network address (or addresses). Our approach distributes these name-to-address mappings amongst Internet ASs by directly hashing a name to produce an IP address announced by the AS which stores the mapping for that name. This in-network single-hop hashing technique leverages the IP reachability information which is readily available at the network layer and achieves low lookups latencies without any DHT maintenance overheads. The proposed DIHT technique is described in detail and specific technical problems such as address space fragmentation and reduction of latency through replication are addressed. Evaluation results are presented from a large-scale discrete event simulation of the Internet with ∼26,000 autonomous systems using real-world traffic traces from the DIMES repository. The results show that the proposed DIHT method evenly balances storage load and achieves lookup latencies with a mean value of ∼50 ms and 95th percentile value of ∼100 ms, considered adequate for the mobility scenarios under consideration.

1. INTRODUCTION

This paper introduces a distributed in-network hash table (DIHT) approach to realize a global scale name resolution service for mobility support in the future Internet. Large-scale in-network global name resolution services under consideration here are motivated by the dramatic growth of mobile devices and applications on the Internet. There are over 6 billion cellular devices in use today with a significant fraction of these providing data services in addition to voice and text. For example, a white paper from Cisco [5] predicts a cross-over of mobile Internet traffic volumes over that of fixed hosts by the year 2015. Current approaches for handling mobility involve a combination of cellular network protocols (e.g. 3GPP) [2] and mobile IP [11] but it is widely recognized that these solutions have serious limitations in terms of scale and service flexibility.

*Research supported by NSF Future Internet Architecture (FIA) grant CNS-1040735

This has motivated a number of "clean-slate" future Internet architecture projects aimed at investigating fundamentally new approaches to meeting anticipated needs such as large-scale mobility, security/privacy or content support [23].

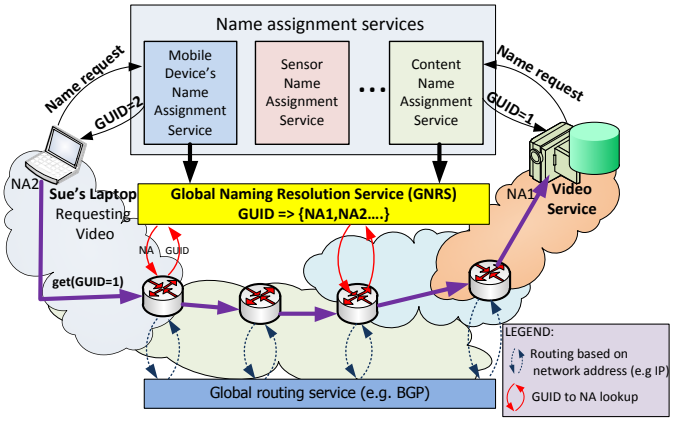

Figure 1: Global name resolution service concept supporting separation of network names and addresses

The MobilityFirst project [20] represents one of these architectural efforts with a particular focus on supporting large-scale, efficient and robust mobility services in the future Internet. The MobilityFirst architecture (which is still evolving) is based on a clean separation between the “names" of end-users or other network-connected objects, and their routable addresses or locators. Separation of names and addresses makes it possible for mobile devices to have a permanent, location independent name or globally unique identifier (GUID) which can then be mapped to a set of routable network addresses (NA) corresponding to the current point(s) of attachment. This concept of separating names from addresses has been proposed in earlier work, such as [8, 3] but is usually viewed as an overlay service above the network similar in spirit to DNS. The MobilityFirst architecture aims to integrate a global name resolution service (GNRS) as a basic network-layer service which can be efficiently accessed both by end-user devices and in-network routers, base stations and access points. This concept is illustrated in Figure 1 which shows the layering of functionality in the proposed MobilityFirst architecture. In this approach, a human-readable name such as ”Sue's laptop" is mapped to a GUID through one of many possible application level services deployed by the network provider or independent third-party providers. The GUID is then assigned to the mobile device (or other network-connected object) and entered into the network-level GNRS service shown in

the figure. The GNRS is a distributed network service which is responsible for maintaining the current bindings between the GUID and network address(es) (NA's). Mobile devices (or routers at their point of attachment) update the GNRS with current NA values resulting in a table entry such as <GUID: NA1, NA2, NA3, optional properties>. The technical problem addressed here is that of realizing a scalable GNRS service with ∼10 Billion GUID entries (i.e. network-attached objects) with lookup latencies fast enough to support anticipated mobility speeds and application usage patterns. We emphasize that since the GNRS can also be queried by the in-network routers for dynamic GUID:NA resolution for in-transit packets, low latency in the query response procedure is a critical requirement for the design.

Our approach addresses this problem through an in-network single-hop hashing technique that leverages the IP reachability information readily available at the network layer. It distributes the GUID to address mappings amongst Internet ASs. To look up a mapping, one can directly hash the GUID to produce an IP address of the AS that stores the mapping. Thus, this technique can achieve low lookup latency and minimum maintenance overhead without the need for more churn-tolerant DHTs. While the following sections describe this service in the context of IP, it is flexible enough so that it can also be used with other identifier schemes and addressing structures.

2. SINGLE HOP IN-NETWORK HASHING

The DIHT design is motivated by two key insights - (a) Centralized overlay resolution services similar to DNS that relies on extensive caching cannot deal with fast updates which is a basic requirement for the use of name/address separation. In addition, to store the mappings of billions of hosts and handle their updates/queries, much larger dedicated infrastructure support than the current DNS would be required; (b) Traditional DHT schemes such as Chord [26], CAN [24], etc. as well as their optimized variations such as those described in [21, 12], etc. aim to solve the problems with centralized architecture but invariably introduce a fundamental trade-off between service latency and table-size/maintenance overhead. We argue that minimization of both these parameters - latency and memory as well as network-traffic - is critically essential for name/address separation to yield any substantial benefit over existing schemes.

The way we meet these scalability requirements is by leveraging the globally available BGP reachability information to distribute the GUID:NA mappings among all the participating ASs in the Internet. With our scheme, every GUID is directly hashed to an existing IP address and its mapping is stored in the AS that announces that IP's reachability using BGP updates. This results in the resolution of all query/updates in exactly one overlap hop without requiring any table maintenance overheads over what is already available through BGP. To illustrate this technique, next we present the sequence of required operations assuming the use of the existing IP address space but the same technique can be realized to work with any future addressing scheme such as IPv6, AIP [3] or HIP [22].

In order to store the mapping of its GUID:NA, a host X forms a *GUID insert message* and sends it to the border gateway router of the AS that it belongs to. The border gateway applies a predefined consistent hash function on the GUID and maps it to a value IP_x in the IP space. Using the IP prefix announcements from its BGP table, the border gateway sends the mapping to be stored at that AS which announced the ownership for IP_x. A second host Y, in the same or a different AS who wishes to find the mapping entry for X, sends a *GUID query message* to its border gateway which then follows a similar hashing scheme to reach the AS maintaining the GUID:NA mapping for X. Here we assume that all ASs have the same view of the prefix announcements and delay the discussion on dealing with prefix aggregation, misconfiguration and route hijacking to Section 2.3. As an enhancement over this baseline scheme, the border gateway router of host X uses K different hash functions and stores the GUID:NA mapping for X at K different places. The benefits from this replication scheme are discussed in Section 2.2 while Figure 2 illustrates an example of the update and query process with $K = 3$.

We note that in contrast to traditional DHT schemes, this approach assumes the participation of network routers in the DIHT scheme in terms of responding to update and query messages and storing the relevant GUID:NA table entries. DIHT does not require any additional table maintenance overhead since we leverage the IP reachability information already made available by the BGP routing protocol. Since the hashing is done locally by the client's border gateway, inserts and queries are fast and do not consume any network bandwidth. In addition, it is noted that unlike many recent proposals [9, 15, 14, 19], we do not distribute the mapping table based on assumptions of aggregability of the GUID space. Thus, our scheme is suitable for flat address spaces, which is the proposed design in the MobilityFirst Architecture as well as other proposals such as AIP [3] and HIP [22]. In the next sections, we discuss the implementation challenges involved in our scheme and describe suitable modifications to the baseline technique to overcome them.

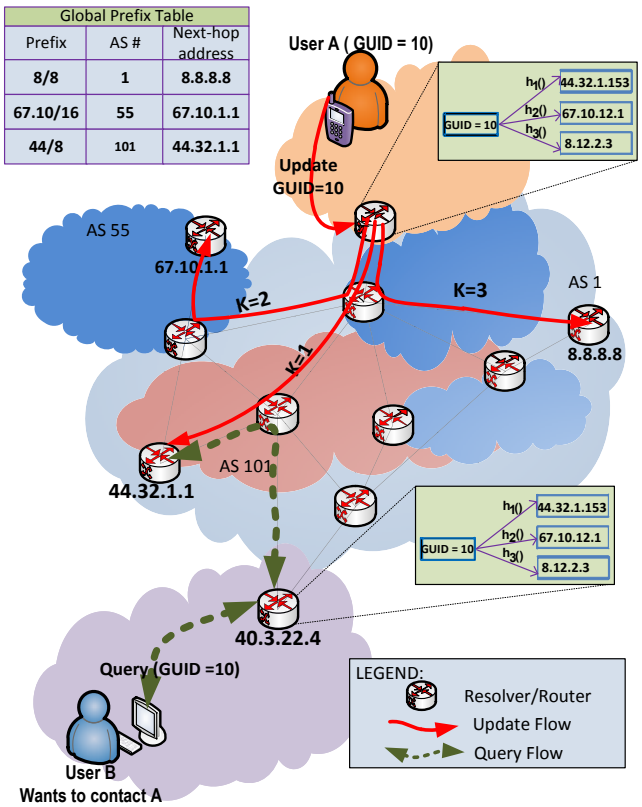

Figure 2: Proposed DIHT method with K=3 independent hash functions

2.1 IP Hole Effect

The key advantage of our approach compared to existing solutions is the use of direct hashing from name to the address of

the storage location leading to resolution in a single overlay-hop. While this results in low latency lookups and updates, the fragmented allocation of the address space presents a potential problem. Continuing with our insert-query example from Section 2, the hashed result IP_x could fall into an IP address space that no AS announces in the global default free zone. To illustrate the extent of this problem, we analyze the current IP assignment status. At present, while there are 2^{32} possible IP addresses for IPv4, only 86% of them are allocated to various entities [4], the rest being reserved for different purposes including multicast, limited multicast, loopback address and broadcast. Among those allocated address, only 63.7% of them are announced by the ASs, presumably because they do not require reachability to the rest of their IP space. This leads to an overall 55% announcement over the possible range of IPv4 addresses which results in a 45% chance that a randomly hashed IP_x will belong to the set of unannounced addresses. We address the IP hole problem by making the border gateway rehash the resulting IP_x values that fall into an IP hole. This process is repeated up to M times, following which if the hashed IP still falls in the unannounced block, we pick the announced IP address that has the minimum *IP distance* to the current hashed value. Given two k-bit addresses, A and B, their IP distance is defined as:

$$IP\_distance_{[A,B]} = \sum_{i=0}^{k-1} |A_i - B_i| * 2^i$$

We further define the IP distance between an address and an address block as the minimum IP distance between that address to all address in the block. We note that assigning mapping responsibility through IP distance introduces 'unfairness' since an AS that announces an IP which is adjacent to a large set of reserved addresses would have to store a large number of mappings. However, using $M = 10$ rehashes, reduces the probability of staying inside the IP hole to 0.009 (using the current IP assignment status), thus resulting in very few cases which require this indirect method of IP distance based insert/lookup. Algorithm 1, which summarizes the steps taken by the gateway to deal with the IP hole problem guarantees that a valid hashed result is always found. Since hashing, rehashing and prefix matching processes are done locally by the border gateway, these operations introduce very little delay and do not add any traffic overhead to the network. Also, we note that the solution described above is specific for the current IP address space but a similar scheme can be used for any other future addressing scheme.

Algorithm 1: Hashing GUID to address space

input : GUID - the $GUID$ to be hashed
M - maximum number of rehashing
output: An address guaranteed to be found in prefix table

```
number_of_tries ← 0;
result ← hash(GUID);
while (number_of_tries < M) do
    if Longest_Prefix_Matching(result) > 0 then
        return result; //ended here if found
    // no prefix was found
    result ← hash(result );
    number_of_tries ← number_of_tries + 1;
//No match found after M hashes
min_distance ← 2^32;
foreach prefix i in the Prefix Table do
if IP_distance(result,prefix i) < min_distance then
    min_distance ← IP_distance(result to the prefix);
return An address in the prefix that has min_distance
```

2.2 Replication

To enhance DIHT's reliability and further reduce access latencies, we increase the number of resolvers responsible for each mapping entry by having the border gateways employ K parallel hash functions at the time of update and query. During insertion or update, the border gateway applies Algorithm 1 K times using independent hash functions on each iteration to obtain K valid addresses. The mapping is stored in all the K ASs responsible for the resulting addresses, thus creating storage replicas for each entry in the mapping table. To lookup a mapping entry, the gateway router selects the closest AS (in terms of path cost obtained from the BGP table) among the K ASs that maintain the required mapping. This replicating technique leads to two important benefits - reduced lookup latencies and increased resilience to random failures. Firstly, with randomized replication, a gateway is more likely to find a nearby AS which stores the required mapping. We show in Section 3 that as K increases from 1 to 5, the 95th percentile lookup latency reduces from 202ms to 91ms. Secondly, as the number of replicas increases, the resilience of the system is also greatly improved as the probability of two distant routers failing simultaneously is very low. Figure 2 illustrates an example of the update and query process with $K = 3$. Finally, we note that this method interleaves the mapping replicas between ASs in contrast to exact replication of all the mappings from one AS to another, thus adding resilience random failures.

The improvement in latency and reliability, however, comes with an increased storage requirement and enhanced complexity in the update process. A careful analysis based on the bounds on requirements and capabilities of the network is thus required to select an appropriate value for K.

2.3 Network Dynamism

BGP Churn: Changes in the prefixes announced occurs when an AS withdraws a previously announced prefix or announces a new prefix. Since a change in the prefix announcements directly influences our mapping lookup scheme, we analyze its potential effects. A long term study of the BGP churn evolution [7] shows that a major reason for churn in the BGP tables is router configuration mistakes or other anomalies. The actual rate of prefix announcement and withdrawal is fairly small with new prefix announcements dominating prefix withdrawals. When an AS withdraws a certain prefix, all its stored mappings that correspond to the withdrawn prefix will neither be queried nor updated since any new query/update processing will be based on the current announcement status. This results in what we call *orphan mappings*. If an AS continues to store such mappings its storage memory is wasted, thus we propose the following planned withdrawal protocol: An AS T_1, before withdrawing a prefix announcement, uses Algorithm 1 for each GUID that corresponds to the prefix being withdrawn and finds a new storage place T_2 for that GUID. While doing so, it excludes its own addresses and thus is guaranteed to find a new AS for each GUID. T_1 then sends *GUID insert messages* to the new ASs corresponding to each affected GUID and finally deletes its own copy of the mappings. Note that this process does not involve the original publisher of the GUID to be involved, thus not requiring it to follow any periodic consistency check protocol.

New prefix announcements can similarly result in orphan mappings since any prior hashes resulting in those IP addresses would follow the IP hole procedures and rehash to reach a different AS. In addition, this could result in an AS getting a query for a GUID

whose mapping is not stored in its storage table. To solve this problem, an AS can adhere to the following protocol to correct the mapping errors in the system: On receipt of a GUID query which cannot be found in its mapping table, an AS follows Algorithm 1 (again by excluding itself) and finds the AS that was storing the mapping in the absence of the new announced prefix. It then contacts that AS sending a special *GUID migration message* through which the corresponding mapping is transfered from the prior AS to its rightful AS. We note that subsequent queries/updates for this GUID follows the normal procedure without incurring this additional overhead. We also emphasize that using these simple table rectification protocols guarantee that orphan mappings are promptly removed from the system and table sizes don't inflate over time.

Router Failure: An AS can lose part or whole of its mapping storage due to router failure or intentional maintenance. Dealing with these anomalies is straightforward by leveraging the replication technique explained in Section 2.2. Every mapping query by a host or router is accompanied by a timeout procedure in case the chosen AS does not respond within a stipulated time. Following a timeout, the host or router can select the next closest AS that stores the mapping required. To further improve resilience and reliability of the mapping, a query can be sent in parallel to P different ASs, with $P \leq K$.

3. EVALUATION

In this section, we first present qualitative arguments to estimate storage requirements and traffic overhead, followed by results from a large-scale event-based simulation to study the query latency and load distribution in our scheme. We show that consistent hashing proportionally distributes the GUID:NA mapping such that the storage at any particular AS does not exceed feasible limits. We also find that the 95th percentile query latency is bounded below 100 ms using modest amount of replication, thus meeting the goal of fast resolution.

To analyze the storage requirements in absence of specifications about the GUID/NA lengths and related headers, we make the following assumptions. We assume flat GUIDs of length 160 bit, each associated with a maximum of 5 NAs (accounting for multi-homes devices) of length 32 bits each. 32 bits of additional overhead per mapping entry is assumed which could include type of service, priority and other side information. Each mapping entry thus has a size of 160 + 32x5 + 32 = 352 bits. We assume a total of 5 Billion GUIDs, roughly equal to the present number of mobile devices, and a replication factor of $K = 5$. Based on the average prefix announcement by individual ASs as determined from a current snapshot of the BGP table [4], the storage requirements per AS, assuming roughly proportional distribution, is only 173 Mbits.

The update traffic overhead is also a key parameter of interest in ensuring scalability. The DIHT technique reduces the traffic overhead in comparison to other mapping schemes by: (a) Ensuring single overlay-hop path to storage location, (b) Not adding any table maintenance traffic as required in DHT based schemes. Using a broad estimate of 50% of the 5 Billion GUIDs being those of mobile hosts which update their GUID:NA mapping at an average rate of 100 updates/day, the world-wide combined update traffic would be $\sim$5 Gb/s, a minute fraction of the overall Internet traffic of $\sim 50\text{x}10^6$ Gb/s as of 2010 [5].

Next we describe our event-based simulation setup and present results which characterize the critical parameters of query latency and load distribution for our scheme.

3.1 Simulation setup

3.1.1 Event-based Simulator Architecture

The discrete-event simulator consists of $\sim$26000 nodes, each of which emulates the relevant operations of an individual AS. The connectivity graph of the network, inter-AS and intra-AS connectivity latencies, and the announced IP prefix list are derived from measurement driven data as described in Section 3.1.2. Three types of event: GUID inserts, GUID updates and GUID lookups are pregenerated and organized into a global queue. The global queue guarantees that the order of events on each run are the same and will lead to the same outcome. The event controller processes events in the global queue in sequential order following the sequence of operations outlined in 2 to process each kind of event. For each query event, the total latency is computed and stored as a data point. The simulation concludes with data gathering from each AS in terms of number of GUIDs stored and average query/update rates. [1]

3.1.2 Data Sources

We use the AS-level topology of the current Internet as our network model by extracting the following real-measurement data from the DIMES database [25]: (i) Connectivity graph containing 26,424 ASs and 90,267 direct links between them, (ii) Average end-to-end latencies between each pair of AS and within each AS. The DIMES database provides end-to-end median latency for about 9 million pairs of hosts which are either within the same AS or in different ASs. From this dataset, we extract the average inter-AS and intra-AS latency since we only work with a AS-level network topology in our simulation. Due to the inherent incompleteness of real-trace data, intra-AS latency numbers are not available for about 6% of the ASs that are involved in the storage or transit of the mapping data. For these ASs, we use the median value (3.5 ms) of the set of available intra-AS latencies as a working solution. We note that there are other measurement driven sources for AS topology data, for example [18] and [6], but we found DIMES to have a more complete view of the AS-level graph compared to any other database. For route selection, we use minimum latency paths between each pair of source and destination AS and make a conscious decision of not employing one of many path inference schemes such as [10, 27] that aim at also incorporating estimated policies at each AS. The dynamic nature of AS relationships, multi-homed networks and hidden/misconfigured policies prevalent in the Internet limits the accuracy of such schemes, introducing an uncharacterizable source of error in the results. We chose to present our results with the caveat of AS policy ignorance instead of adding unknown inference errors.

Since our scheme allocates GUIDs to ASs according to the prefix announcements, we use a complete list of IP prefixes advertised in the Internet default free zone, as seen by APNIC's router at DIX-IE in Tokyo, Japan [4]. This dataset consists of roughly 330,000 prefixes spanning close to 52% of the 32 bit IP address space which is consistent with recent estimates [7] about the size of the prefix tables in DFZ routers. We confirm our results with two other prefix tables taken from BGP routers in the continental USA and Europe respectively and observe similar trends.

To discard any location bias and to incorporate the global scale of operation, we use another dataset from DIMES to characterize the distribution of the source of GUID insert and query. Each GUID in our simulation is originated from a randomly picked source AS, where the probability of choosing a certain AS is weighted in proportion to the number of end-nodes found in that AS as per the DIMES database. The end-result emulates a real deployment scenario in which more populated areas (characterized by high num-

[1]The source code for our simulation is made available at [1]

K	Round Trip Query Latency(ms)		
	Mean	Median	95th percentile
1	77.8	57.9	202.0
5	51.3	41.7	90.9

Table 1: Query Latency Statistics for K = 1, 5

ber of IP end-nodes) will generate more GUID queries compared to less populated areas. We note that the nature of trace collection employed by DIMES might introduce a bias due to non-uniformity of vantage point distribution (though shown to be small), however we do not take this into account in our simulation.

3.2 Results

We present two sets of results that characterize the query latency and the load distribution of our scheme respectively.

3.2.1 Query Latency

The most critical performance measure of a mapping scheme is the delay incurred between making a query and receiving a reply from the appropriate mapping source. Low query latency is a key requirement for efficient mobility support using the name and locator split mechanism. We characterize the query latency of our scheme by running a set of simulations as per Section 3.1 with 100K GUID inserts and queries and varying values for the replication factor K, from 1 to 5. By repeated trials, we verified that the latency averages converge with 100K data points and further increasing the number of queries does not provide any additional insights. Figure 3 plots the Cumulative Distribution Function (CDF) of the round trip query latency for different values of K. The effect of increasing K, as described in Section 2.2 can be clearly seen with the leftward shift of the CDF curve as we increase the value of K. In particular, the mean, median and 95^{th} percentile query latencies of $K = 1$ and $K = 5$ cases are tabulated in Table 1 which shows a marked decrease in the tail of the latency distribution. The low query latency values and a relatively thin tail distribution for $K = 5$ validates the effectiveness of our scheme and shows that it can be used even under very stringent latency requirements.

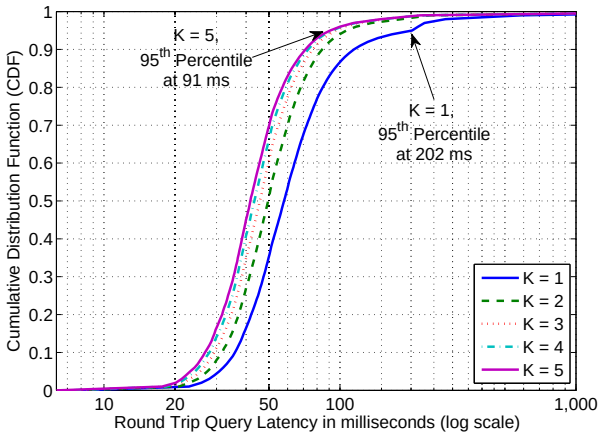

Figure 3: Empirical CDF of the round trip query latency

3.2.2 Load Distribution

To ensure that the scheme scales and no particular AS is assigned a disproportionally large number of GUIDs, fairness in load distribution is an important factor. Using the current IP addresses as an example of the NA space, we show that despite the heavily fragmented allocation of this space, our scheme exhibits fairness of GUID assignments. We measure fairness in terms of the Normalized Load Ratio (NLR) at each AS, which is defined as the ratio of GUID percentage assigned to that AS over the IP percentage advertised by that AS. For example if an AS announces a $/8$ prefix, corresponding to 0.39% of the 32 bit IP space and gets assigned 20,000 out of a total of 1 Million GUIDs i.e 2% of GUIDs, then its normalized load would be 2/0.39 $\simeq$5.

To evaluate the load distribution, we run a second experiment with 10 Million GUID inserts and a fixed value of $K = 1$. A higher number of inserts provide sufficient number of data points for averaging which is required to cover the wide variability in the number and size of IP prefix announcements. Simulations with higher values of K show similar trends and we only present the $K = 1$ case for clarity. Figure 4 shows the CDF of the normalized load over all the ASs in the network for this simulation. Close to 70% of the ASs lie in the flat vertical portion of the curve, indicating a fair assignment with narrow tails. The median NLR value of $\sim$2 is expected since in addition to its fair share of GUIDs, each AS is also allocated a portion of the GUIDs that fall in the IP holes as described in Section 2.1. We note that as we further increase the number of inserts, the tail distributions become narrower and the distribution of NLR values asymptotically converges to a fixed value.

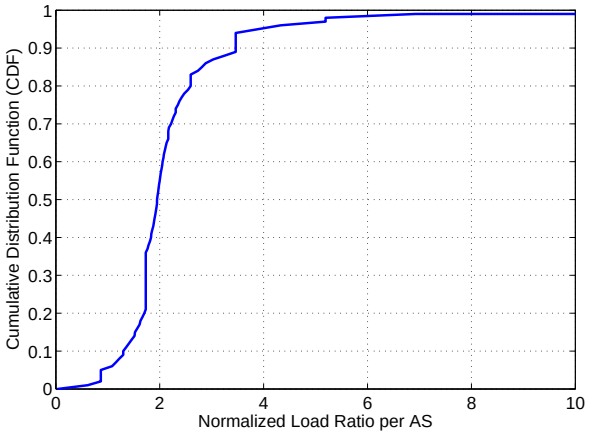

Figure 4: Empirical CDF of normalized Load per AS

4. RELATED WORK

Being a critical component of locator/identifier separation schemes, various architectures for mapping identifiers to locators have been proposed and studied. Most of the early mapping schemes [9, 15, 14, 19] assumed aggregatable identifier spaces and proposed ideas based on that vantage point. However, this assumption is too restrictive making such schemes not applicable to many recent mainstream proposals such as HIP [22], AIP [3] and MobilityFirst [20] which propose flat identifiers. Our approach, in contrast, targets a flexible resolution service by not making any assumptions about identifier hierarchy or locator structure.

There are some recent mapping architecture proposals that incorporate flat identifier space such as DHT-MAP [17], SLIMS [13]. However these approaches either incur high lookup latency, making it not applicable to highly mobile environment, or prohibitive management overhead which limits scalability. For example, the DHT based scheme in [17] can entail up to 8 logical hops introducing an average latency of about 900ms as per their assumptions.

In contrast, our scheme aims for much lower latencies by employing the one-hop hashing approach and ensures minimum man-

agement overhead for feasible deployment on a global scale. We argue that making use of network entities and the IP reachability information already available through the underlying routing infrastructure provides a practical and scalable approach to realize mapping resolvers. Reference [16] uses the similar in-network hashing scheme to target the different but related problem of name-based routing.

This work also focuses on a global-scale simulation to validate the design, which has been neglected in most of the prior works referenced above. Reference [14] is a recent exception which presents a trace based simulation using the iPlane dataset [18]. Our simulation approach is more realistic than that of [14] on two counts: (a) We use a larger dataset from DIMES [25] to extract AS level connectivity and latency information. The DIMES dataset is based on measurements from $\sim$1000 vantage points compared to $\sim$200 for iPlane, resulting in information for about twice the number of ASs as compared to iPlane; (b) To generate resolution lookup events, [14] uses DNS lookup traces from two particular source locations which introduces a significant locality bias in their results. In contrast, we globally distribute lookup source locations by weighting the chances of choosing a particular source location (source AS) in proportion to the available data on number of end nodes near that location. The basic intuition here is to mimic realistic deployment where more lookup requests will be generated from more densely populated areas.

5. CONCLUDING REMARKS

In this paper, we presented the motivations, design and evaluation of DIHT, a scheme for low latency, scalable name resolution service for future mobile Internet. DIHT distributes name to address mappings amongst Internet routers using an in-network single-hop hashing technique which derives the address of the storage router directly from the name. In contrast to other DHT-based techniques, DIHT does not require any table maintenance overhead since we use the IP reachability information already available through the inter-domain routing protocol. In addition, DIHT supports arbitrary name and address structures making it more widely applicable than most prior techniques. Through a large-scale discrete-event simulation, we show that the proposed DIHT method achieves low latencies with a mean value of $\sim$50 ms and 95th percentile value of $\sim$100 ms and good storage distribution among participating routers.

The main technique of leveraging the routing information for mapping distribution can be used to realize other variations of the distribution technique. For example, GUIDs can be directly hashed to AS numbers or allocation sizes could be varied based on parameters that reflect the economic incentives of the ASs. In the context of mobility support in the Internet, the benefits offered by caching-based schemes like the DNS come with a heavy cost on increased update complexity. We plan to extend the scope of this work by studying a feasible in-network caching scheme that builds on top of the DIHT scheme. Since our scheme interacts with the hosts, the inter-domain routing protocol and the Internet routers, security is a critical requirement at each level. The MobilityFirst project [20], takes a holistic approach towards self certification based security, which can tie well into the relevant aspects of our scheme. Our future work plan also includes incorporating the transient effects of BGP updates, misconfigurations and router failures in our simulation framework and studying the effects of various measurement biases on our results.

6. ACKNOWLEDGEMENTS

We would like to thank Jennifer Rexford for her insightful comments. We also thank Noa Zilberman and Udi Weinsberg from DIMES for help with the latency dataset. This work has been supported by NSF Future Internet Architecture (FIA) grant CNS-1040735

7. REFERENCES

[1] DIHT Simulation source code - MobilityFirst Project. http://www.winlab.rutgers.edu/ tamvu/NRS/sourceCode/, 2011.
[2] 3GPP: Mobile Broadband Standard. http://www.3gpp.org/.
[3] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable Internet Protocol (AIP). In *Proc. ACM SIGCOMM*, Aug. 2008.
[4] BGP Routing Table Analysis - DIX-IE Data. http://thyme.apnic.net/current/.
[5] Cisco: Global Mobile Data Traffic Forecast Update, 2009-2014,.
[6] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov. Internet mapping: From art to science. In *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*, pages 205–211, 2009.
[7] A. Elmokashfi, A. Kvalbein, and C. Dovrolis. BGP Churn Evolution: a Perspective from the Core. *2010 Proceedings IEEE INFOCOM*, pages 1–9, 2010.
[8] D. Farinacci, D. Fuller, D. Oran, and D. Meyer. *Locator/ID separation protocol (LISP)*. IETF Internet Draft, draft-farinacci-lisp-12.txt, Sep. 2009.
[9] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. *LISP Alternative Topology (LISP+ALT)*. IETF Internet Draft, draft-ietf-lisp-alt-06.txt, March 2011.
[10] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Trans. Netw.*, 9:733–745, December 2001.
[11] S. Gundavelli, K. Leung, V. Devarapalli, C. K., and P. B. *Proxy Mobile IPv6*. IETF Internet Standard, RFC 5213, Aug. 2008.
[12] A. Gupta, B. Liskov, and R. Rodrigues. One hop lookups for peer-to-peer overlays. In *Proceedings of the 9th conference on Hot Topics in Operating Systems - Volume 9*, pages 2–2, Berkeley, CA, USA, 2003. USENIX Association.
[13] J. Hou, Y. Liu, and Z. Gong. Silms: A scalable and secure identifier-to-locator mapping service system design for future internet. *Computer Science and Engineering, International Workshop on*, 2:54–58, 2009.
[14] L. Jakab, A. Cabellos-Aparicio, F. Coras, D. Saucez, and O. Bonaventure. LISP-TREE: a DNS hierarchy to support the lisp mapping system. *IEEE J.Sel. A. Commun.*, 28:1332–1343, October 2010.
[15] D. Jen, M. Meisel, D. Massey, L. Wang, Z. B., and Z. L. *APT: A Practical Transit Mapping Service*. IETF Internet Draft, draft-jen-apt-01.txt, May 2008.
[16] C. Kim, M. Caesar, and J. Rexford. Floodless in SEATTLE: a scalable ethernet architecture for large enterprises. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pages 3–14, New York, NY, USA, 2008. ACM.
[17] H. Luo, Y. Qin, and H. Zhang. A DHT-Based Identifier-to-Locator Mapping Approach for a Scalable Internet. *IEEE Trans. Parallel Distrib. Syst.*, 20:1790–1802, December 2009.
[18] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: an information plane for distributed services. In *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, pages 367–380, 2006.
[19] L. Mathy and L. Iannone. LISP-DHT: towards a DHT to map identifiers onto locators. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 61:1–61:6, 2008.
[20] MobilityFirst Future Internet Architecture Project. http://mobilityfirst.winlab.rutgers.edu/.
[21] L. Monnerat and C. Amorim. D1HT: a distributed one hop hash table. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 10 pp., 2006.
[22] R. Moskowitz and P. Nikander. *Host Identity Protocol (HIP) Architecture*. IETF Internet Standard, RFC 4423, May 2006.
[23] NSF Future Internet Architecture Project. http://www.nets-fia.net/.
[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, pages 161–172, 2001.
[25] Y. Shavitt and E. Shir. DIMES - Letting the Internet Measure Itself, http://www.netdimes.org/.
[26] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Trans. Netw.*, 11:17–32, February 2003.
[27] U. Weinsberg, Y. Shavitt, and E. Shir. Near-deterministic inference of AS relationships. In *Proceedings of the 28th IEEE international conference on Computer Communications Workshops*, INFOCOM'09, pages 377–378, 2009.