DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet[⋆]

Tam Vu[∗], Akash Baid[∗], Yanyong Zhang[∗], Thu D. Nguyen[†],
Junichiro Fukuyama[‡], Richard P. Martin[∗†], Dipankar Raychaudhuri[∗]

[∗]WINLAB, Rutgers University, {*tamvu, baid, yyzhang, rmartin, ray*}*@winlab.rutgers.edu*
[†]Department of Computer Science, Rutgers University, *tdnguyen@cs.rutgers.edu*
[‡]Toyota InfoTechnology Center USA, *jfukuyama@us.toyota-itc.com*

***Abstract*—This paper presents the design and evaluation of a novel distributed shared hosting approach, DMap, for managing dynamic identifier to locator mappings in the global Internet. DMap is the foundation for a fast global name resolution service necessary to enable emerging Internet services such as seamless mobility support, content delivery and cloud computing. Our approach distributes identifier to locator mappings among Autonomous Systems (ASs) by directly applying K>1 consistent hash functions on the identifier to produce network addresses of the AS gateway routers at which the mapping will be stored. This direct mapping technique leverages the reachability information of the underlying routing mechanism that is already available at the network layer, and achieves low lookup latencies through a single overlay hop without additional maintenance overheads. The proposed DMap technique is described in detail and specific design problems such as address space fragmentation, reducing latency through replication, taking advantage of spatial locality, as well as coping with inconsistent entries are addressed. Evaluation results are presented from a large-scale discrete event simulation of the Internet with ∼26,000 ASs using real-world traffic traces from the DIMES repository. The results show that the proposed method evenly balances storage load across the global network while achieving lookup latencies with a mean value of ∼50 ms and 95^{th} percentile value of ∼100 ms, considered adequate for support of dynamic mobility across the global Internet.**

I. Introduction

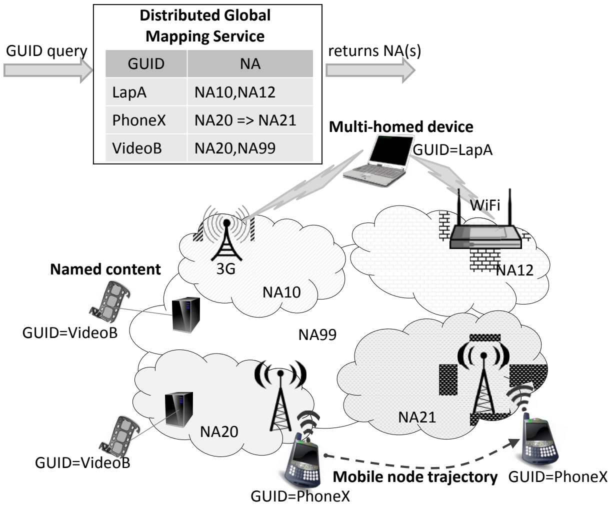
The concept of separating identifiers from routable addresses or locators has been advocated by a number of authors in the networking community [1], [2], [3], [4]. Separation of names from addresses makes it possible to avoid implicit or explicit binding of sources and destinations to the network's actual topology. Using existing terminology, the *identifier* names a communicating object, such as a particular mobile phone, while the *locator* identifies an address the network can use to route messages. For example, a phone connecting to different 3G, 4G, and WiFi networks would get a separate locator for each network. However, the identifier, which in this case could be the International Mobile Subscriber Identity (IMSI) number, would remain the same. The goal of this work is to explore the feasibility of identifier based communication under the assumption of large-scale dynamic mobility of named objects. In the above example, programmers should be able to send messages to a particular phone based on its IMSI number rather than to an IP address. We take the position that identifiers can also be used to name abstract entities and services; they need not to be tied to a particular device.

Identifier based communication has many advantages, including simplified implementation session management, multihoming, mobility, disconnection, authentication and security [1], [2], [3], [4]. When there is a high degree of dynamism between the communicating entities and the network (as in most mobile service, content retrieval and cloud computing scenarios), using identifiers to define network-attached objects is more appropriate than using locators. Intuitively, it is easier to work with networking primitives based on identifiers when the locator changes faster than the timescales of the communication session. For example, a voice call may last 30 minutes, but a mobile device in a vehicle may change its network attachment points many times during this period.

Realizing an identifier based protocol stack has several challenging aspects; the key design issue we address in this work is the dynamic binding of identifiers to locators. That is, when the user presents the networking stack with an identifier, the networking subsystem must quickly return a set of locators, or *network addresses* (NAs) back to the user. We address the challenge of providing a fast global name resolution service at Internet scale in this paper, and describe and evaluate a specific *Direct Mapping (**DMap**)* scheme for achieving a good balance between scalability, low update/query latency, consistency, availability and incremental deployment.

We take note of two trends in the Internet community that have significant bearing on the design of a global name resolution scheme. First, a flat identifier space is preferred to the hierarchical domain names currently used in the Internet. The use of flat, location independent identifiers is a central tenet of a number of clean slate proposals such as AIP [5], HIP [3], ROFL [6] and MobilityFirst [7]. The key advantage of flat labels lies in their use for direct verification of the

[⋆]This research was supported in part by NSF Future Internet Architecture (FIA) grant CNS-1040735.



GUID	NA
LapA	NA10,NA12
PhoneX	NA20 => NA21
VideoB	NA20,NA99

Fig. 1. Distributed global identifier to locator mapping service

binding between the name and an associated object (see [8] for a detailed discussion). As a result, name resolution schemes that rely on the hierarchical structure of the name such as the Domain Name System (DNS) or LISP-TREE [9] are not suitable for supporting such a flat identifier space.

The second trend is that due to its separation from the network attachment point, global names or identifiers will tend to belong to end-users or application providers rather than to the network, as is currently the case with IP. Hosts and other network-attached objects (content, computing services, etc.) are not owned by any Internet Service Provider (ISP), but they just happen to be connected to a particular Autonomous System (AS). Most name resolution schemes propose to store the mappings of the identifiers belonging to an AS inside that AS only [10]. The same rationale, however, does not work for a host-based identifier space because hosts (or content) do not belong to any particular AS, especially with the increasing number of mobile hosts which often have multiple simultaneous points of network attachment. Thus we challenge the assumed constraint of ownership based storage and design a scheme with network-wide sharing of the identifier-locator mappings independent of the AS boundaries.

Motivated by these trends, we propose a dynamic identifier to locator mapping management scheme called DMap which supports a flat space of a identifiers, referred to as *Globally Unique Identifiers*, or GUIDs. A GUID is a long bit sequence, such as a public key, that is globally unique and long enough that the chance of a collision is infinitesimally small. Each end host, such as laptops, mobile phones, servers and virtual machines can have a GUID. In addition, even abstract objects, such as a piece of content or a particular context, can have GUIDs. Each GUID is associated with one or more network addresses (NAs) that it attaches or belongs to. For example, the NAs of a multi-homed laptop in Figure 1 includes the NA of its 3G service provider and the NA of the network that its WiFi interface attaches to. We denote the identifier to locator mapping as the GUID→NA mapping.

To perform the mapping service for a given GUID, DMap applies $K (K > 1)$ hashing functions onto it to produce a list of K network addresses, which are IP addresses in today's Internet, and stores the GUID→NA mapping in the ASs that announce those network addresses. By doing so, DMap spreads the GUID→NA mappings amongst ASs, such that an AS will host mappings of other ASs, as well as have its mappings hosted by others. A key advantage of this *shared hosting* approach is that it allows the hosting ASs to be deterministically and locally derived from the identifier by any network entity. DMap is simple yet efficient. It leverages the routing infrastructure to reach the hosting AS in a single overlay hop; it does not require a home agent, unlike mobile IP and existing cellular networks. Further, the potential shortcoming of the direct mapping scheme, lack of locality, is addressed by having multiple copies of the mappings that are stored in multiple locations. We further improve the design by including a local copy of the mapping within the AS that the GUID is residing in (this AS may change as the host moves).

Through detailed simulation studies, we show DMap achieves a 95^{th} percentile round trip query response time of below 100ms, which is important to support the fast growing class of mobile devices connected to the Internet. Our results also show that DMap can proportionally distribute GUID→NA mappings among ASs, which is critical to scale our system to support billions of GUIDs and NAs associated with a global scale network.

The rest of paper is organized as follows. In Section II, we provide the background and motivation for DMap. The working of DMap and how DMap addresses several technical challenges are discussed in Section III. We present detailed simulation evaluation results in Section IV, and an analytical model in Section V. Finally, we have the related work in Section VI and the concluding remarks in Section VII.

II. Background and Motivation

A. Identifier and Locator Separation

While there is broad agreement on identifier locator separation [1], [11], implementation proposals vary widely along two main design dimensions: (i) what does an identifier correspond to? and (ii) how is it mapped to a locator? There are two main approaches. In the *router-based* proposals such as LISP [12], Six/One [13] and APT [14], the identifiers identify the network endpoints, and hosts can be reached by specifying the endpoint through which they are connected to the network. Thus a host has to acquire a different endpoint identifier every time it changes its network attachment point (though patches to work around this problem have been proposed [15]). In contrast, there is an alternative *host-based* approach in which identifiers are designated to end hosts, resulting in each host maintaining its identifier irrespective of changes in its point of attachment to the network. HIP [3], MILSA [4] and MobilityFirst [7] proposals have shown the distinct benefits of having host-based identifiers in terms of mobility support, multi-homing support and security, evidently at the cost of requiring changes in the host-side protocol stack.

B. Requirements of Host-Based Identifiers

We believe that a host-based mapping scheme must meet the following requirements:

- **Flat Identifiers:** The mapping architecture needs to support structure-less, flat identifiers.
- **Low Latency:** Since mobility is directly handled using dynamic identifier to locator mapping, latency requirements are much stricter in host-based schemes.
- **Low Staleness:** Fast mobility support also requires that the identifier-locator mappings be updated at a time-scale smaller than the inter-query time.
- **Storage Scalability:** Since flat identifiers would lead to substantially more number of identifier to locator entries, the mapping scheme needs to scale to the order of billions of entries instead of thousands [16].

The above requirements call for a fundamental shift from traditional mechanisms such as MobileIP, DNS and DHT. While it is applicable at small scale, the mapping scheme of MobileIP incurs high overhead since all mappings are resolved by the home agent regardless of its distance to correspondents. A home agent acting as a relaying node on the data plane in tunnelling mode makes MobileIP not scalable to global Internet scale. On the other hand, since it relies on extensive caching, DNS cannot deal with fast updates. In addition, to store the mappings of billions of hosts and handle their updates/queries, a much larger dedicated infrastructure than the current DNS would be required. Traditional Distributed Hash Table (DHT) schemes and their optimized variations, e.g., [17], [18], aim to solve the problems of centralized solutions but invariably introduce a fundamental tradeoff between service latency and table/maintenance overhead. A detailed discussion of other existing mechanisms along with their pros and cons are presented in Section VI.

C. Incentive for Shared Hosting

To address the problems above, in this work, we propose DMap which is built on the principle of shared hosting of the locator to identifier mappings among all the ASs in the network. A concern that may arise naturally with shared hosting is incentive: *why would Network Operator A store and manage Network Operator B's identifiers?* As we argued above, with host-based identifiers, the concept of site-dependent or provider-dependent identifiers are diluted specially in the case of mobile hosts. For fixed legacy hosts, we assert that just like peer-to-peer file sharing systems and TCP congestion control, cooperative schemes that result in a common good with a small individual cost have a natural incentive mechanism for deployment as long as the individual cost of participation is reasonable. In particular, the incentives for foul-play, i.e., not storing or answering mapping requests in this case, would depend on the benefit vs. possible penalty of non-compliance. Both technical solutions (such as reputation management in peer-to-peer systems) and non-technical policy bindings (analogous to Network Neutrality arguments) can be invoked to force/persuade ASs to fairly participate in the scheme. However, in this paper, we focus on the architectural and performance aspects of the scheme and leave the design of the incentive mechanisms open.

III. Direct Mapping (DMap)

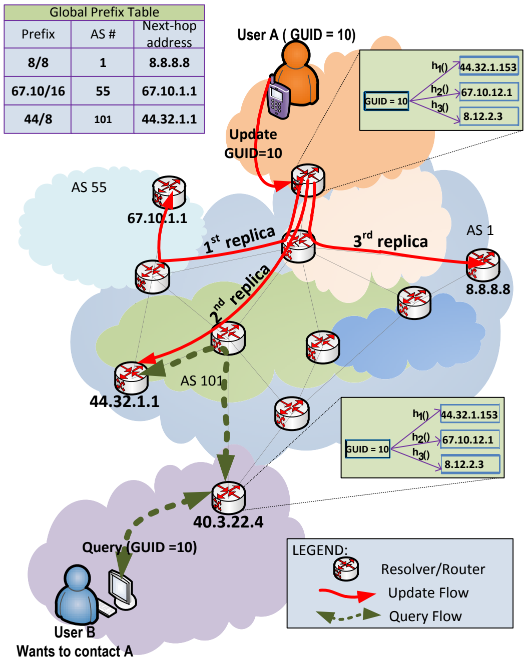
In DMap, each GUID→NA mapping is stored in a set of ASs. Each GUID is directly hashed to existing network addresses and its mapping is thus stored within the ASes corresponding to these network addresses.

A. Overview of DMap

In designing our mapping method, we strive to minimize update/lookup latencies as well as the amount of state information that needs to be maintained. We achieve these goals by leveraging the globally available BGP reachability information to distribute the GUID→NA mappings among all the participating ASs. In our scheme, DMap first hashes a GUID to an existing network address, and then stores its GUID→NA mapping within the AS that announces this network address. This results in exactly a single overlay hop for all the update/lookup requests without introducing any additional state information on each router. Next, we look at an example to illustrate this approach. In this example, we assume the usage of the existing IP address space, but we note that the same technique can be easily extended to any future addressing scheme such as IPv6, AIP [5] or HIP [3].

Let us suppose host X, with GUID G_x, is attached to NA N_x. X first sends out a *GUID Insert* request, which is captured by the border gateway router in its AS. The border gateway router then applies a predefined consistent hash function on G_x and maps it to a value IP_x in the IP space. Based upon the IP prefix announcements from its BGP table, the border gateway router finds out which AS owns IP_x and sends the $G_x \rightarrow N_x$ mapping to that AS. Later, suppose host Y wishes to look up the current locator for GUID G_x. Y sends out a *GUID Lookup* request. After the request reaches Y's border gateway router, the border gateway runs the same hash function to identify the AS that stores the mapping. Every time when X changes its association and connects to a different AS, it needs to update its mapping by sending out a *GUID Update* request. Update requests are processed similarly as insert and lookup requests.

Using the above approach, a GUID's mapping is hashed to a random AS, without considering the locality between the GUID and its lookup requests. This lack of locality may potentially lead to unnecessarily long lookup latencies. Thus, instead of storing a mapping at only one AS, we consider having K replicas of the same mapping stored at K random ASs. Having K replicas can significantly reduce the lookup latency as the requesting node can choose the closest replica (e.g., based upon the hop count between itself and the hosting ASs). Meanwhile, it will not have a big impact on the update latency as we can update the replicas in parallel. With K mapping replicas, the lookup latency becomes the shortest latency among the K ASs, while the update latency becomes the largest among the K ASs. Figure 2 illustrates an example update and lookup process with $K = 3$. Finally, we note that



Global Prefix Table		
Prefix	AS #	Next-hop address
8/8	1	8.8.8.8
67.10/16	55	67.10.1.1
44/8	101	44.32.1.1

Fig. 2. DMap with K=3 independent hash functions

important DMap parameters, such as which hash functions to use and the value of K, will be agreed and distributed before hand among the Internet routers.

Compared to other mapping schemes, one distinct feature of DMap is the direct participation of network routers in storing GUID→NA mappings and in responding to updates and lookups. DMap does not require any additional state information as the IP reachability information is already made available by the BGP routing protocol. In addition, we note that unlike many recent proposals [19], [20], [9], [10], DMap does not distribute GUID mappings based on the assumption of the aggregate-ability of the GUID space. Our scheme is suitable for flat address spaces, which has been pointed out as a desirable feature for the Future Internet [5], [3].

B. Handling Unallocated Network Addresses

DMap hashes a GUID to an IP address, and stores the GUID mapping in the AS that announces this IP address. Due to fragmentation in the IP address space, it is possible that the hashed IP address is not announced by any AS. This problem is referred to as the *IP hole problem*. To understand the extent of this problem, we take a close look at today's IP address space. At present, 86% of the 2^{32} IP addresses available in IPv4 are allocated to various entities [21]; the rest are reserved for other purposes including multicast, limited multicast, loopback address, broadcast, etc. Among the allocated addresses, 63.7% of them are announced by one of the ASs. This leads to an overall 55% announcement ratio over the entire IPv4 address space, 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 finding a deputy AS through rehashing if the IP address after the first hash falls into a hole. After $M-1$ rehashes, if the resulting address still falls into an IP hole, we pick the deputy AS as the one that announces the 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 addresses in the block. In this way, we can guarantee that a deputy AS can always be found.

There is a concern that the above method may introduce load imbalance among ASs: the AS that announces an IP address that is adjacent to a large set of reserved addresses (thus unannounced) may become a popular deputy AS and needs to store a large number of mappings. Fortunately, the probability of reaching an IP hole after M hashes decreases rapidly with increasing M. For instance, this probability is as low as 0.034% for $M = 10$. As a result, the chances that we need to resort to the ASs that announce the IP addresses with the minimum IP distances to the holes are very low.

Algorithm 1 summarizes the steps taken by the border gateway to deal with the IP hole problem. Since hashing, rehashing and prefix matching processes are done locally by the border gateway, these operations introduce very little delay to the network.

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
nearestPrefixID = findNearestPrefix(result);
return An address in nearestPrefixID;
```

When extending DMap to other network address schemes, such as IPv6, we need to rethink how we deal with the IP hole problem as these network address spaces may have substantially more holes than used address segments. To address such sparse address spaces, we propose to use a two-level indexing method to index each announced address segment: bucket ID and segment ID within that bucket. Suppose we have N buckets, each with a capacity of S segments. We make N large so that S can be kept small. Given a GUID, we run two hash functions, the first one mapping the GUID to bucket ID, and the other one mapping the GUID to the segment ID. Figure 3 illustrates the bucketing scheme.

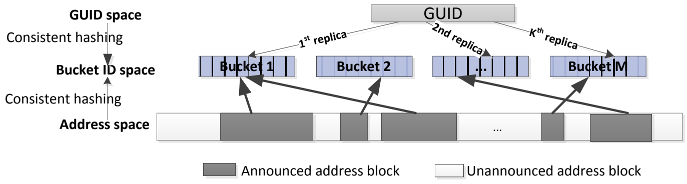


Fig. 3. Bucketing scheme handling non-contiguous address space issue

C. Spatial Locality and Local Replication

The main advantage of DMap lies in its simplicity: hashing a GUID to a random AS. However, this random placement ignores locality, and so may degrade performance. Having multiple replicas partially addresses this problem, but it still has the inherent problem of a direct hashing scheme: the GUID mappings are stored at faraway ASs when the host and requestor are close to each other. Thus, we enhance the baseline DMap for an expected common case of when a requesting node is attached to the same AS as the GUID that it is resolving. To leverage this *spatial locality*, DMAP stores an additional replica of a GUID mapping at its attached AS. When a host registers/updates its GUID, it creates/updates a local copy (at its attached AS) in addition to creating/updating the K "global" copies. When a node needs to lookup a GUID, it sends out a local and a global lookup simultaneously. When the host and the requester are from the same AS, the local request should lead to significantly reduced lookup latency.

D. Inconsistent GUID→NA Mappings

1) BGP Churn: Since a change in the prefix announcements directly influences DMap, we analyze the potential effects of BGP churn. A long term study of BGP churn evolution [22] shows that a major reason for churn in the BGP tables is router configuration mistakes or other anomalies. Changes in prefix announcements occur when an AS withdraws a previously announced prefix or announces a new prefix. The actual rate of new prefix announcement and prefix withdrawal is small, with the former dominating the latter.

When an AS withdraws a certain prefix, all the mappings previously hosted by the AS whose GUIDs are hashed to the withdrawn IP addresses will become unaccessible, resulting in what we call *orphan mappings*. To address this problem, we let the withdrawing AS run the IP hole protocol to find a deputy AS for these mappings before withdrawing. It sends a *GUID insert messages* to the deputy AS and deletes its own copy of the mapping. Subsequent queries will then hit an IP hole. Following the same IP hole protocol, they will reach the deputy AS and find the mapping.

Announcing new prefixes can also result in orphan mappings. The GUIDs that were originally hashed to these IP addresses had followed the IP hole procedure to a "deputy" AS, and announcing these addresses now can make the mappings on the deputy AS orphan mappings. As a result, queries that reach the announcing AS will not find the mappings, while the mappings on the deputy AS become inaccessible. To solve this problem, when the announcing AS receives a query and finds the mapping missing, it sends a *GUID migration message* to the deputy AS to relocate the mapping to itself. This operation could cause a negligible one-time overhead, which only occurs for the first query after the announcement.

2) Mobility: Mobility can also lead to inconsistencies in DMap. Suppose host X, with GUID G_x, is connected to AS A. As a result, DMap has the mapping $(G_x : A)$. Then suppose X moves to AS A' at time t_0, and its mapping will be updated to $(G_x : A')$ at time t_1. While we expect $t_1 - t_0$ to be small, it is possible for a querying node to get the old mapping right after X has moved. The querying node will then be unable to communicate with X. In this case, the querying node should mark the mapping as obsolete, and keep checking until it receives an updated one.

3) Router Failure: An AS can lose part or all of its mappings due to router failure. This is a rare event, but we need to address the resulting complication. If a lookup request reaches an AS, but cannot find the mapping due to this problem, the requestor will wait for a timeout. Following the timeout, the requestor will contact the next mapping replica (remembering we have K replicas in total). We note that the probability for K Internet routes to fail at the same time is extremely low, and thus our replication strategy also improves system resilience and reliability.

IV. Evaluation

In this section, we present the results from a detailed performance evaluation of the DMap scheme using a mix of qualitative reasoning and event-based simulation.

A. Storage and Traffic Overhead

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 bits, each associated with a maximum of 5 NAs (accounting for multi-homed 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 meta 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 [21], the storage requirements per AS, assuming proportional distribution, is only 173 Mbits. This storage requirement is quite modest, even if it is multiplied several times to include non-mobile devices as well as future growth.

The update traffic overhead is also a key parameter of interest in ensuring scalability. The DMap technique reduces the traffic overhead in comparison to other mapping schemes by: (a) Ensuring a single overlay-hop path to a storage location, (b) Not adding any table maintenance traffic as required in DHT schemes. Using a broad estimate 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 worldwide combined update traffic would be $\sim$10 Gb/s, a minute fraction of the overall Internet traffic of ~ 50x10^6 Gb/s as of 2010 [16].

B. Query Response Time and Load

The round trip response time of a query is composed of: (i) K longest prefix matchings at the local gateway router, (ii) network latency between the query source and the chosen destination AS, (iii) the queuing and processing delay of the mapping server at the destination AS and (iv) the return network latency between the destination AS and the query source. Since routers use fast longest prefix match algorithms, requiring on order of 100 instructions per lookup, i.e. ~30 nanoseconds on a 3 GHz processor [23], we ignore this component in our evaluation. Also sufficient resources are assumed at the mapping server to make the queueing and processing delay very small compared to the round trip latency. Note that this process does not add any delays to the normal data packets as the steps described above are only applied to GUID query packets and are assumed to be handled at a separate compute layer at the gateway router.

1) Simulation and Workloads: We develop a discrete-event simulator consisting of ~26000 nodes, each emulating an 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 below. We consider three types of events: GUID inserts, GUID updates and GUID lookups.[1]

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 an 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.

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 (DFZ), as seen by APNIC's router at DIX-IE in Tokyo, Japan [21]. 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 [22] 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 that contains the number of end-nodes connected to ASs to characterize the distribution of the source of GUID insert and query. Each GUID in our simulation originates 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.

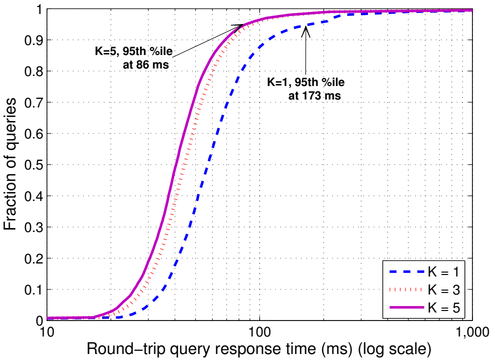


Fig. 4. Round trip query response times

The number of queries for any GUID depends on its popularity amongst Internet hosts. In order to capture the effects of the wide variations in host popularity, we use a Mandelbrot-Zipf distribution [26], [27] to model the varying host popularity. The Mandelbrot-Zipf distribution defines the probability of accessing an object at rank k out of N available objects as:

$$p(k) = \frac{H}{(k+q)^{\alpha}}, \tag{1}$$

where $H = 1/\sum_{k=1}^{N} 1/(k+q)^{\alpha}$, with α determining the skewness and q affecting the "flatness" of the peak. We use a value of $\alpha = 1.02, q = 100$ following the arguments in [27].

2) Results: We present two sets of results that characterize the query response time and the load distribution of our scheme respectively.

a) Query Response Time: We evaluate the query response time for DMap by inserting 10^5 GUIDs and generating 10^6 queries according to the popularity model. By repeated trials with increasing number of GUIDs/queries, we verified that the response times converged after reaching the above configuration and larger numbers are not necessary. When we store a mapping at multiple locations, i.e., $K > 1$, in the results below, we assume that the querying node has sufficient information to choose the location with the lowest response time. We note that in today's Internet, this information is only partially available, but at the least, each AS has hop count information for reaching all other ASs through the routing protocol. Using least hop count instead of lowest response time leads to similar results albeit with marginally increased latencies. We would also emphasize that many techniques are being proposed to better estimate the response times [28].

Figure 4 plots the cumulative distribution function (CDF) of the round trip query response times with varying K values. We make two observations. First, with $K = 5$, 95% of the queries complete within 86ms. This is well within the range

[1] The source code for our simulator is available at [24]

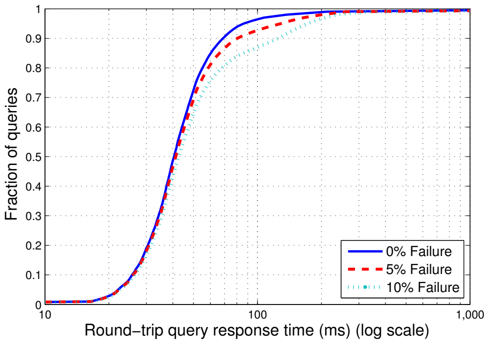


Fig. 5. Effect of BGP Churn on query response times.

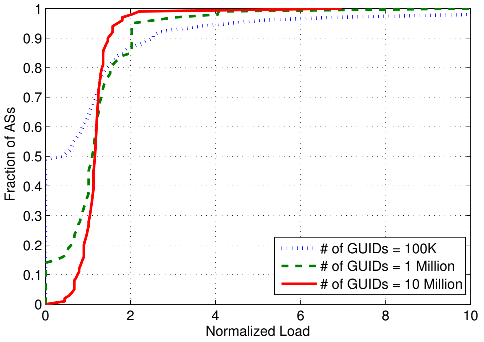


Fig. 6. Normalized Load Ratio per AS

needed for voice call handoffs [29]. Indeed, given that many WiFi and IP handoff protocols are often on the order of 0.5-1 second [30], [31], DMap updates would not introduce an undue additional burden. Second, storing each GUID mapping in multiple locations can significantly reduce query response times, as it allows a querying node to choose the replica that is "closest" to itself, thus addressing the locality of the requests. The effect of increasing K 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 I, which shows a marked decrease in the tail of the response time distribution.

However, even the curve for $K = 5$ has a relatively long tail. This long tail arises from a few queries originating from those ASs with unusually long intra-AS response times, according to the DIMES dataset. For example, the 18 queries with the longest response times all originated from AS 23951, a small AS registered in Indonesia with a one-way latency of more than 2.3 seconds on each of its outgoing links.

b) Impact of BGP Churn: The above study assumes that the BGP table at the query origin exactly reflects the current state of the Internet. However, BGP tables at different places in the Internet can be inconsistent because of new prefix announcements or prefix withdrawals. This inconsistency may have an adverse impact on the overall query response times as the query may reach an AS which does not host the requested mapping. In this situation, the AS will then reply with a "GUID missing" message, and the querying node will have to contact another replica. Thus, a query may require multiple round-trips to different ASs for each resolution. We note that the probability of two churns occurring at the same time for the same GUID is negligible. Here, we conduct a set of experiments to quantify the impact of this inconsistency caused by BGP churn. In the experiments, we vary the percentage of prefixes that are newly announced or withdrawn from 0 to 10%. Figure 5 plots the CDF of the query response times for $K = 5$ and 0% to 10% lookup failures. A 5% failure rate, which already seems pessimistic according to [32], [33], shifts the median and 95th percentile from 40.5ms and 86.1ms to 41.3ms and 129.1ms, respectively.

K	**Round Trip Query Response Time (ms)**		
	Mean	**Median**	**95th percentile**
1	74.5	57.1	172.8
5	49.1	40.5	86.1

TABLE I
QUERY RESPONSE TIME STATISTICS FOR K = 1, 5

c) Storage Distribution: We next study the distribution of GUID→NA mappings amongst ASs to evaluate DMap's ability to spread the storage load proportional to the size of the ASs. We measure storage load using the *Normalized Load Ratio* (NLR) at each AS, which is defined as the ratio of the percentage of GUIDs assigned to an AS divided by the percentage of IP addresses advertised by that AS. For example if an AS announces a $/8$ prefix, corresponding to 0.39% of the 32 bit IP space and is assigned 20,000 out of a total of 1 Million GUIDs, i.e., 2% of GUIDs, then its normalized load would be $2/0.39 \simeq 5$. Ideally, each AS's NLR would be 1.

Figure 6 plots the CDF of the NLR when we inserted from 10^5 to 10^7 GUIDs, with $K = 5$. We observe that for 10^7 GUIDs, 93% of the ASes had NLRs between 0.4 and 1.6. Further, we observe that as the number of GUIDs increases from 10^5 to 10^7, the CDF becomes much sharper around NLR equal to 1 (with a shorter tail). This suggests that DMap can distribute the storage load better when the system scales. These results show that DMap does a very good job of spreading out the storage load proportional to the percentage of the IP space that an AS claims.

Interestingly, the median NLR value is 1.16. The fact that the median NLR value is greater than 1 is expected since in addition to its fair share of GUIDs, many ASs are also allocated a portion of the GUIDs that has hashed values after M retries falling in the IP holes as described in Section III-B.

V. Analytical Model for Query Response Time

In this section we present an analytical model for an upper bound on the query response time and parametrically study its dependence on the Internet topology and replication factor K. While the simulation framework of Section IV shows the response time performance of DMap based on the current Internet topology, this analytical model allows us to estimate its performance based on a predicted future Internet model.

A. The Jellyfish Model

An accurate parametric model of the Internet topology is known to be a difficult endeavor due to the inherent complexities in routing policies, detour paths and limited visibility of the intra-AS structure [34]. The Jellyfish model, however, has been found to closely follow the evolution of the Internet topology at a relatively coarse scale [35]. In this paper, we build a Jellyfish model based on the PoP-level Internet topology. We first label the node with the highest degree as the root v_0 and the maximal clique[1] containing v_0 as the *core*, denoted as $Shell$-0. Let v be a non-root node and j a non-negative integer. The smallest path length[2] from v to a node in the core is its *distance to the core*. We use $Shell$-j to denote the set of nodes whose degree is more than 1 (intermediate nodes) and whose distance to the core is j. We use $Hang$-j to denote the set of nodes whose degree is 1 (leaf nodes) and whose distance to the core is $j+1$. These are standard notations in the Jellyfish model [34]. Then we have

$$Layer(j) = Shell\text{-}j \cup Hang\text{-}(j-1) \quad \text{for } j \geq 1,$$

and $Layer(0) = Shell$-0. We further denote the total number of layers in the Internet PoP topology as N and the percentage of nodes in layer j by r_j; if n is the total number of nodes in G, then $r_j = |Layer(j)|/n$. The separation of one degree nodes at each layer distinguishes between stub connections and transit connections which makes the model much closer to the Internet topology than a standard tree structure.

B. Upper Bound for Query Response Times

Following the above model, if we assume no peer links between the nodes inside each layer, then the distance between any two nodes s and t (in layers j_s and j_t respectively), $d(s,t)$, is at most $j_s + j_t + 1$. Note that since the core forms a completely connected graph, all hops in the core are of length one. To drive a simple parametric upper bound for the query response time, we assume that the network address space is uniformly distributed among the PoPs and all addresses within a PoP behave in an identical fashion. Following the algorithm described in Section III, let the source of a GUID query belong to PoP s and let $t_1, t_2, \ldots, t_K$ be the destination PoPs for the query determined by the K hash functions $h_1, h_2, \ldots, h_K$ applied on the GUID G. Assuming a linear relationship between PoP path length and response time, the query response time $\tau(s,G)$, is thus given by

$$\tau(s,G) = c_0 \cdot \min_{1\leq i\leq K} d(s,t_i) + c_1, \tag{2}$$

where c_0 and c_1 are constants. In order to average over all possible source and destination PoPs, we treat $d(s,t_i)$ as a random variable and find its probability distribution based on the r_j values defined in the previous subsection. In the analysis, we use the standard notations $Pr(\cdot)$ and $Pr(\cdot \mid \cdot)$ for the probability and conditional probability of argument events, and $\mathbb{E}(\cdot)$ for the expected value of a random variable.

Given a uniformly selected source PoP, we have $Pr(s \in Layer(j)) = r_j$. Note that since DMap actually chooses a network address uniformly over all possible addresses, the j^{th} layer could include a different number of addresses than the ratio r_j. Our analysis assumes the uniform distribution of addresses among PoPs but the model can be easily extended to non-uniform distributions by considering weights w_s proportional to the number of addresses in PoP s. Since accurate estimates of such a distribution is not directly available through any of the Internet measurement frameworks, we assume $w_s = 1$ for all s. Based on the same assumptions, we have $Pr(t_i \in Layer(j_i)) = r_{j_i}$ for each $i = 1, 2, \ldots, K$ and $j_i = 0, 1, 2, 3, 4, \ldots, N-1$. Thus the conditional probability of $d(s,t_i) \geq l+1$ given that $s \in Layer(j)$ is at most the percentage of nodes in $Layer(l-j) \cup Layer(l+1-j) \cdots \cup Layer(N-1)$. (Since s is in $Layer(j)$, $d(s,t_i) = l+1$ when t_i is in $Layer(l-j)$, $d(s,t_i) = l+2$ when t_i is in $Layer(l+1-j)$ and so on in the worst case). In other words,

$$Pr\left(d(s,t_i) > l \mid s \in Layer(j)\right) \leq p_{j,l},$$

where $p_{j,l} \overset{def}{=} r_{l-j} + r_{l+1-j} + r_{l+2-j} \cdots$

$$\Rightarrow Pr\left(\min_{1\leq i\leq K} d(s,t_i) > l \;\Big|\; s \in Layer(j)\right) \leq p_{j,l}^K,$$

$$\Rightarrow Pr\left(\min_{1\leq i\leq K} d(s,t_i) \leq l \;\Big|\; s \in Layer(j)\right) > 1 - p_{j,l}^K,$$

$$\Rightarrow Pr\left(\min_{1\leq i\leq K} d(s,t_i) \leq l\right) > \sum_{j=0}^{N-1} r_j (1 - p_{j,l}^K).$$

This provides an upper bound for the CDF of average distance from the source to the closest destination. Thus the probability that $\min_{1\leq i\leq K} d(s,t_i) > l$ is at most $1 - q_l$, where we define q_l as:

$$q_l \overset{def}{=} \sum_{j=0}^{N-1} r_j \left(1 - p_{j,l}^K\right),$$

Finally, noting that the diameter of our PoP graph is $(N-1) + (N-1) + 1 = 2N-1$ or less,

$$\begin{aligned} &\mathbb{E}\left(\min_{1\leq i\leq K} d(s,t_i)\right) < \sum_{l=1}^{2N-1} (1 - q_l) \\ &\Rightarrow \mathbb{E}\left(\tau(s,G)\right) < c_0 \left(\sum_{l=1}^{2N-1} (1 - q_l)\right) + c_1. \end{aligned} \tag{3}$$

C. Analytical Results

We use the formulation derived above to study the response time upper bound in three different scenarios with varying number of replicas K. The first scenario reflects the current Internet topology, for which we use measured data from the iPlane project [36] for setting the parameters $r_j, j = 1, 2, \ldots, N$. The data set shows a graph of 193,376 nodes within 8 layers and more than 60% of the nodes residing in layers 3 and 4. The next two scenarios model the medium-term

[1] clique: a completely connected subgraph of G.

[2] path length: the number of edges in the path, *i.e.*, that of the involved PoPs in the path minus 1.

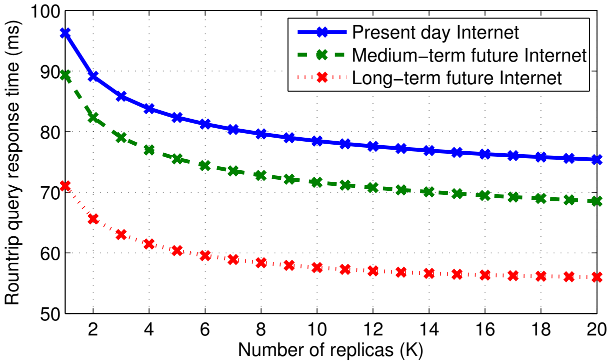


Fig. 7. Analytical upper bound of query response times with the Internet topology evolution.

and long-term future Internet topologies. In order to come up with models for the future Internet, we leverage the following two distinct trends observed from the widely regarded CAIDA measurement framework [37]: (i) The number of nodes are growing almost linearly with time, (ii) The topology graph is getting flatter with time, i.e. ASs are obtaining more direct paths to the core. Extrapolating these trends, we model the medium-term (5-10 years) future Internet as having 20% more nodes than present contained in 6 layers. Similarly, the long-term (25-30 years) future Internet model contains double the number of nodes contained in 4 layers. Figure 7 shows the analytical upper bound of average query response time for the three scenarios using the measured least squared error values for $c_0, c_1 = 10.6, 8.3$. The plot shows that based on the predicted future Internet topology, response time upper bounds for DMap queries become smaller with the evolution. Also, the analysis clearly indicates that increasing the replica number results in diminishing returns beyond a few replicas. We note that actual values for the query response upper bound will typically be smaller than the ones obtained in this analysis, since we did not consider the presence of peering links between nodes in the same layer.

VI. Related Work

Given the importance of locator/identifier separation schemes in both current and future networks, various architectures for mapping identifiers to locators have been proposed and studied. Most of the early mapping schemes [19], [20], [9], [10] 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 [3], AIP [5] and MobilityFirst [7] 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 [38], SLIMS [39]. However these approaches either incur high lookup latency, making it not applicable to highly mobile environment, or high management overhead which limits scalability. For example, the DHT based scheme in [38] 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 management 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 [40] uses a 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 [9] is a recent exception which presents a trace based simulation using the iPlane dataset [36]. Our simulation approach is more realistic than that of [9] 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 ∼1000 vantage points compared to ∼200 for iPlane, resulting in information for about twice the number of ASs as compared to iPlane; (b) To generate resolution lookup events, [9] 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.

VII. Concluding Remarks

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

In further work, we plan to consider other variations of the proposed DMap distribution scheme - for example GUIDs can be hashed directly to AS numbers or allocation sizes can be varied to reflect economic incentives at ASs. We also plan to extend the scope of this work by studying a feasible in-network caching methods that builds on top of the basic DMap 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 [7], takes a holistic approach towards self certification based security, which tie in 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.

On the validation and evaluation front, there is an ongoing effort to implement a proof-of-concept global scale DMap system using the GENI (global environment for network innovation) framework. A first DMap prototype was demonstrated at the GENI Engineering Conference-12 in Kansas City and efforts are currently under way to fully instrument the latency and overhead measurements necessary to evaluate scalability and performance. If the GENI experiments successfully confirm DMap performance, there are also further plans to use the proposed technique as part of a complete identifier-based protocol stack in the MobilityFirst future Internet architecture project.

Acknowledgements

We would like to thank Jennifer Rexford for her insightful comments. We also thank Noa Zilberman and Udi Weinsberg from DIMES for their help with the latency dataset.

References

[1] J. Saltzer, *On the Naming and Binding of Network Destinations*, IETF Internet Standard, RFC 1498, Aug. 1993.

[2] C. J. Bennett, S. W. Edge, and A. J. Hinchley, "Issues in the interconnection of datagram networks," Jul. 1977.

[3] R. Moskowitz and P. Nikander, *Host Identity Protocol (HIP) Architecture*, IETF Internet Standard, RFC 4423, May 2006.

[4] J. Pan, R. Jain, S. Paul, and C. So-in, "MILSA: A New Evolutionary Architecture for Scalability, Mobility, and Multihoming in the Future Internet," *Selected Areas in Communications, IEEE Journal on*, vol. 28, no. 8, pp. 1344–1362, Oct. 2010.

[5] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable Internet Protocol (AIP)," in *Proc. ACM SIGCOMM*, Aug. 2008.

[6] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, "ROFL: Routing on Flat Labels," in *In SIGCOMM*, 2006, pp. 363–374.

[7] MobilityFirst Future Internet Architecture Project, http://mobilityfirst.winlab.rutgers.edu/.

[8] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, "Naming in content-oriented architectures," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ser. ICN '11, 2011, pp. 1–6.

[9] 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.*, vol. 28, pp. 1332–1343, October 2010.

[10] L. Mathy and L. Iannone, "LISP-DHT: towards a DHT to map identifiers onto locators," in *Proceedings of ACM CoNEXT 2008*, pp. 61:1–61:6.

[11] E. T. Li, *Recommendation for a Routing Architecture*, IETF Internet Standard, RFC 6115, Feb. 2011.

[12] D. Farinacci, D. Fuller, D. Oran, and D. Meyer, *Locator/ID separation protocol (LISP)*, IETF Internet Draft, draft-farinacci-lisp-12.txt, Sep. 2009.

[13] C. Vogt, "Six/one router: a scalable and backwards compatible solution for provider-independent addressing," in *Proceedings of MobiArch '08*, 2008, pp. 13–18.

[14] D. Jen, M. Meisel, D. Massey, L. Wang, B. Zhang, and L. Zhang, "Apt: a practical tunneling architecture for routing scalability," *UCLA Computer*, pp. 1–9, 2008.

[15] D. Farinacci, V. Fuller, D. Lewis, and D. Meyer, *LISP Mobile Node*, IETF Internet Draft, draft-meyer-lisp-mn-05.txt, May 2011.

[16] "Cisco visual networking index: Global mobile data traffic forecast update, 2010-2015," Cisco White Paper, Feb. 2011.

[17] L. Monnerat and C. Amorim, "D1HT: a distributed one hop hash table," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, Apr. 2006, p. 10.

[18] A. Gupta, B. Liskov, and R. Rodrigues, "One hop lookups for peer-to-peer overlays," in *Proceedings of Network Systems Design and Implementation*. USENIX Association, 2004, p. 14.

[19] 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.

[20] 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.

[21] BGP Routing Table Analysis - DIX-IE Data, http://thyme.apnic.net/current/.

[22] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "BGP Churn Evolution: a Perspective from the Core," *2010 Proceedings IEEE INFOCOM*, pp. 1–9, 2010.

[23] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, "Small forwarding tables for fast routing lookups," *SIGCOMM Comput. Commun. Rev.*, vol. 27, pp. 3–14, October 1997.

[24] "DMap Simulation source code - MobilityFirst Project," http://www.winlab.rutgers.edu/ tamvu/NRS/sourceCode/, 2011.

[25] Y. Shavitt and E. Shir, DIMES - Letting the Internet Measure Itself, http://www.netdimes.org/.

[26] S. A. Krashakov, A. B. Teslyuk, and L. N. Shchur, "On the universality of rank distributions of website popularity," *Computer Networks*, vol. 50, no. 11, pp. 1769 – 1780, 2006.

[27] O. Saleh and M. Hefeeda, "Modeling and caching of peer-to-peer traffic," in *Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 249–258.

[28] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica, "Pathlet routing," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 111–122.

[29] 3GPP: Mobile Broadband Standard, http://www.3gpp.org/.

[30] V. Vassiliou and Z. Zinonos, "An analysis of the handover latency components in mobile ipv6," *Journal of Internet Engineering*, vol. 3, no. 1, pp. 230–240, Dec 2009.

[31] A. Mishra, M. Shin, and W. Arbaugh, "An empirical analysis of the IEEE 802.11 MAC layer handoff process," *SIGCOMM Comput. Commun. Rev.*, vol. 33, pp. 93–102, April 2003.

[32] S. Y. Qiu, P. D. McDaniel, F. Monrose, and A. D. Rubin, "Characterizing address use structure and stability of origin advertisement in inter-domain routing," in *Proceedings of the 11th IEEE Symposium on Computers and Communications*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 489–496.

[33] R. Mahajan, D. Wetherall, and T. Anderson, "A study of BGP origin as changes and partial connectivity," Oct. 2001. [Online]. Available: http://www.routeviews.org/papers/nanog_origin.pdf

[34] Y. He, G. Siganos, and M. Faloutsos, "Internet topology." in *Encyclopedia of Complexity and Systems Science*, 2009, pp. 4930–4947.

[35] S. L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos, "A simple conceptual model for the Internet topology," in *IEEE GLOBECOM '01*, vol. 3, 2001, pp. 1667–1671 vol.3.

[36] 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*, ser. OSDI '06, 2006, pp. 367–380.

[37] "CAIDA : The cooperative association for internet data analysis," 2011. [Online]. Available: http://www.caida.org/

[38] H. Luo, Y. Qin, and H. Zhang, "A DHT-Based Identifier-to-Locator Mapping Approach for a Scalable Internet," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, pp. 1790–1802, December 2009.

[39] 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*, vol. 2, pp. 54–58, 2009.

[40] C. Kim, M. Caesar, and J. Rexford, "Floodless in SEATTLE: a scalable ethernet architecture for large enterprises," in *Proceedings of ACM SIGCOMM 2008*. New York, NY, USA: ACM, pp. 3–14.