A Hierarchically Aggregated In-Network Global Name Resolution Service for the Mobile Internet

Yi Hu, Roy D. Yates, Dipankar Raychaudhuri
WINLAB, Rutgers University
{yihu, ryates, ray}@winlab.rutgers.edu

***Abstract*—The shift in Internet usage from wired to wireless devices motivates the design of name-based network architectures that support mobile wireless users. A key challenge is the design of a scalable global name resolution service (GNRS) that rapidly and reliably resolves name identities to network addresses under high mobility. In this work, we present GMap, an in-network GNRS that provides fast lookups through a hierarchical organization that exploits spatial locality queries while retaining the simplicity and scalability of a random replicated DHT approach. To provide a lightweight solution, GMap avoids per-entity state maintenance by deploying K replicas uniformly for each identifier-to-locator mapping. In the presence of skewed lookup popularity, server workloads are balanced by caches along the routing paths of lookup queries. This is in contrast to overlay name resolution services (such as the recently proposed scheme called "Auspice") in which placement needs to be optimized for each entry. Specific design issues are addressed for the proposed GMap service, including cache cost, server churn behaviors, and expected server capacity. Compared to DMap, a prior in-network GNRS, evaluation results show that GMap reduces the 95th percentile query latency from 100ms to around 20ms, with the maximum server workload deviation reduced more than fifty-fold.**

I. Introduction

"Mobility" has become the new norm for Internet usage as wireless/mobile devices have overtaken wired PCs as the primary end-user device [11]. However, the Internet does not support mobile access as the normal use case because it employs an IP address to identify a network entity as well as to locate it for network routing. This conflation of the identifier and the locator causes problems in supporting a network entity switching between multiple connection points (i.e., mobility) or simultaneously using multiple connection points (i.e., multi-homing). Five research teams under the NSF initiated Future Internet Architecture (FIA) projects [30] have been working on clean-slate redesign of the Internet to address the fundamental shift of the Internet usage. The proposed architectures include NDN, NEBUKA, XIA, MobilityFirst and ChoiceNet, all of which propose name-based designs in which a separation of name identifiers and routable addresses enables seamless mobility support. This separation removes limitations imposed by binding a network entity to its network connection point.

A. GNRS Background

To realize naming-address separation, services should be provided to resolve a network entity's human descriptive name (e.g., John's laptop model A) to the routing addresses (e.g., AS10-PoP1) that identify its points of network attachment. Given that routing addresses generally change much faster than human descriptive names, especially in high mobility settings, a complete resolution service is split into two parts: the Name Certification Service (NCS) for resolving a human descriptive name of a network entity to its global unique identifier (GUID), and Global Name Resolution Service (GNRS) for resolving a GUID to its network address (NA)[1]. A GUID is a public key based flat identifier, which is cryptographically verifiable and serves as an authoritative identifier for a network entity. A GUID may be a network device such as a tablet, mobile applet, server or sensor, or an abstract network entity such as networked content, virtual machines, virtual networks, or a set of devices specified by a network application context.

The NCS takes charge of certifying a GUID from its associated human language descriptions, including verification of ownership, adjudication of name space collisions, and provision of attribute registration for the GUID so that it can be searchable through the attributes. The basic GNRS function is to dynamically bind a GUID to an NA; that is, GNRS servers record a mapping of GUID to an NA for each NCS-certified GUID. In addition, a GUID record may contain optional fields such as access policies to the mapping, usage preferences on network addresses. If the GUID refers to a context-specified group of network entities, the address field lists the member GUIDs of the group. By recursively resolving the NA mappings of member GUIDs, the GNRS enables context-aware services such as multicast or anycast.

B. Architectural Positioning

Architectural positioning is a fundamental difference between GMap and other name resolution service schemes. GMap is among the few schemes (e.g.,DMap [41]) that are positioned at network layer, where name servers are co-hosted by the routers belonging to each autonomous system (AS). By contrast, other name resolution service schemes (e.g., Auspice [38], CoDoNS [35], [32]) and the legacy DNS rely on servers provided at the application layer. We argue that an in-network GNRS provides the foundation for advanced network architectures and services.

- Network reachability, which requires provision of both name resolution services and routing functions, should be provided at the network layer.
  As the Internet evolves from a packet network to service network, resolving names to addresses becomes a joint effort in routing for connecting network attached entities.

[1]The NA may refer to either a single point of attachment or, in the multihomed case, a collection of attachment points.

This is different from the existing Internet that is based on address abstraction where network layer with routing service is enough to provide reachability.

- In-network GNRS enables a self-sufficient network.
  A GUID service layer on top of the routing function in the network layer provides name-based service APIs for application users who only need to specify the name of service and the name of the involved network entities. As a result, advanced name-based services such as "get sensor data from island X" or "find a vacant taxi in Times Square" can be provided in the absence of any application services, which makes a self-sufficient advance network. Such self-sufficiency is desired both at network startup or in case of network disruption and disconnection since the named services can be provided as long as routers are functioning. For example, a virtual network (VN) is desired for critical service delivery after a disaster. With in-network GNRS support for VN membership management and application specific routing in VN, a QoS application can be delivered by the VN, whereas an application-layer GNRS will fail.
- Providing name resolution service at network layer is critical for organic network growth.
  For network operators, a new network design always starts from small scale and grows to large, which requires a lightweight and scalable GNRS scheme. Our in-network DHT based GMap provide an inexpensive solution matching the network growth given that the name servers are co-hosted by routers. While per-entity optimization based GNRS (e.g., Auspice [38], CoDoNS [35]) requires extra server resources to maintain the workload state of each GUID record, imposing significant overhead to support expected trillions of GUIDs.
- In-network name resolution avoids premature identity-to-locator resolution that can occur at the application layer. Since the NA of an entity is irrelevant to the basic semantics of applications, only the routing schemes need to know the network address of an entity. If the entity's network address changes during he application session, an in-network GNRS enabled router can perform late-binding to re-query the GNRS to obtain the updated NA without re-establishing the application connection. By contrast, an application layer GNRS will fail to provide such transparency.

C. Performance and Scalability

With advances in mobile technology and emerging mobile applications, the number of networked devices per person keeps increasing. The total number of networked devices is predicted to reach 50 billion by 2020 [12]. In addition, networked content is also fast growing. Thus, a key challenge in providing a high performance global name resolution service (GNRS) is that the scheme must scale to support from tens of billions up to trillions of GUIDs. Consequently, deployment at such scale of per-GUID optimization schemes such as Auspice [38] and CoDoNS [35] are likely to be costly.

GMap achieves fast lookup performance (comparable to Auspice as shown in Section III-B) through a locality-aware replica placement process that exploits the spatial locality patterns shared by networked objects. Locality measurement studies on search engine queries, tweets, phone call logs, and network content request [5], [6], [10], [23] have categorized network objects into three groups based on the spatial locality of their queries: citywide, countrywide and worldwide locality. In addition, mobility studies [20], [33] on network entities show common movement patterns featuring transitions between two to three major locations with movement areas that also exhibit a hierarchy of city, country, and global scale. Such common hierarchy is introduced for locality and mobility of sorts of network objects are driven by common factors such as culture background, language in use, time zone partition, and political policies. GMap harnesses these locality and mobility characteristics to provide a hierarchical geo-locality aware replica placement.

Specifically, in GMap, replica servers of each GUID are placed hierarchically in geographic locations by dividing K total replicas into K_1 global replicas, K_2 regional replicas and K_3 local replicas (K=K_1+K_2+K_3) based on the GUID's current location. Local replicas are placed in the same city of the GUID; regional replicas would be placed in the same country; and global replicas are placed without any geographic constraint for the purpose of reliability and load balancing.

While GMap's uniform geo-aware K replica placement is a low overhead solution for fast lookups, it may lead to server workload congestion in the presence of skewed GUID popularity. Through workload evaluation, we will see that a small percentage of servers (no more than 0.01% of total servers) have workloads more than 1000x of the mean server workload. Such imbalanced server workloads are caused by hotspot GUIDs that have orders of magnitude higher popularity. The query workload of a hotspot GUID would overwhelm its K servers. A proposed solution is to deploy a number of replica servers for each GUID in proportion to its query popularity [35]. However, this requires each server to maintain the query popularity state for each GUID and synchronize it over all servers in order to adjust the number of replicas.

GMap deploys small caches along the routing paths of lookup queries to distribute the workload of hotspot GUIDs over all servers along the path. A small cache size (e.g., 0.05% of total GUIDs) is used to alleviate server congestion at hotspot replica servers without resorting to per-GUID state maintenance. To mitigate obsolete caches due to GUID's mobility, we use a probabilistic go-through mechanism to proactively refresh the cache for hotspot GUIDs and use a time-to-live (TTL) timer to invalidate the expired mappings of infrequently visited cache entries. Our evaluation results will show that a cache size equal to 0.05% of total GUIDs is sufficient to reduce the ratio of the maximum server workload to the mean server workload from over 10^3 to under 20. As a GMap name server could consist of a cluster of routers located in a point-of-presence (PoP), it is practical to share the workload among routers in a PoP to accommodate peak workloads that are within 20 fold of the typical workload.

Our major contributions are summarized as follows:

- We design a lightweight in-network global name resolution scheme GMap providing fast lookup service through

geo-locality aware replica placement and server workload balance through a novel caching technique.
- We evaluate GMap performance through modeling network entities mobility and query locality in real world geographic and demographic Internet topologies. Comparisons with latest in-network and application layer GNRS schemes are conducted.
- We quantify the name server capacity requirement in GMap through benchmark testing and analysis.

The rest of the paper is organized as follows. Section II presents GMap design details. Section III evaluates the performance of GMap. In Section IV, we quantify the server capacity requirement in GMap. Section V reviews related work and we conclude the paper in Section VI.

II. GMap Design

In this section, we first highlight GMap's design requirements and corresponding mechanisms. We then present the replica placement scheme and the operation flows in GMap. Next, we describe the cache design for balancing name server workload. Lastly, we introduce the name server availability management scheme.

GMap is designed to provide robust and efficient GUID-to-NA resolution service. Such name resolution service provides a GUID service layer where name-based network service APIs are supported for diverse "smart applications". Specifically the design of GMap looks to fulfill the following requirements:

Generality: GNRS should make no assumption on network architectures, be independent of network address format, name identity format (e.g., no reliance on hierarchical naming schema like NDN or DNS) or routing protocols (e.g., DMap [41] relies on IP to perform replica placement.) GMap achieves architectural generality by representing a name server with a GUID, just as it does for any other network entity. Replicas are assigned to name servers through direct hashing from a network entity's GUID to a name server's GUID. For clarity, we use the term GUID only to refer to that of a network entity and SID (server identity) to refer to the GUID of a name server. Relying only on the flat identifiers (GUIDs) to perform replica placement, GMap functions independently of network address or naming structures.

Latency: GMap should provide fast lookup and address updates for GNRS users. In addition, agile in-route lookups (i.e., GUID to NA re-binding due to mobility occurring during or after the connection establishment) should be transparent to applications. In GMap, we provide fast processing of lookup queries by geo-locality aware replica placement, which exploits spatial locality patterns generally exhibited by GUIDs. Based on locality and mobility studies [5], [6], [10], [20], [23], [33], we design a three-level hierarchical replica placement scheme to speed up lookups to network objects of varying locality.

Reliability: GMap should address two causes of unreliability: 1) name server overflow due to skewed GUID popularity; 2) the time-varying availability of name servers. GMap engineers the number of replicas K for each GUID to be sufficiently large that the probability of all replica hosts being simultaneously unavailable is negligibly small. The number of replicas for each GUID is also large enough to avoid host server overflow for typical non-hotspot GUIDs. To address the server congestion caused by hotspot GUIDs, small caches are deployed along lookup routing paths to distribute the query workload of hotspot GUIDs away from their replica host servers. Two consistency management mechanisms are applied to the caches to ensure correctness. To be robust to the dynamics of SID availability, availability changes are detected by neighbor SIDs in real time to maintain K available replicas for each GUID. Neighbor SIDs to a SID S_i are the set of SIDs in the leaf set of S_i when all SIDs are organized through distributed hash table (DHT). For non-neighbor SIDs, the availability changes are piggybacked on the GNRS query replies. To avoid high rates of SID churns, admission control is applied at the NCS to exclude volatile ASs from participating in the GNRS as their failure recovery overhead will outweigh the value of their participation.

Scalability: The service maintenance overhead should be reasonably low to support large scale name identities ($\sim 10^{12}$ GUIDs) and name server participation (e.g., millions of name servers world-wide). GMap ensures scalability by avoiding the overhead of per-GUID state information maintenance at servers and state synchronization across all servers. Furthermore, because of the goal of the cache is to serve hotspot GUIDs rather than typical GUIDs, a small cache will be enough. Through our analysis in Section II-F, current commercialized in-memory caches can easily meet the requirements for GMap with 10^{12} GUIDs.

The following subsections elaborate on specific elements of GMap.

A. Replica Placement of GUID to NA Mapping

GMap performs geo-locality aware replica placement of each GUID-to-NA mapping in the following way. For each GUID we deploy K replicas of its GUID-to-NA mapping, consisting of K_1 global replicas, K_2 regional replicas and K_3 local replicas. A replica to name server assignment is performed through *GUID to SID mapping*. For the GUID G_X of a host X, we first place K_1 global replicas at servers whose SIDs are K_1-nearest to G_X. Nearness is defined in terms of a distance metric on GUIDs. In our implementation, we use XOR distance between the bit string values of GUIDs or SIDs. Second, we place K_2 regional replicas at servers whose SIDs are K_2-nearest to G_X and in the same country as X. The global replica servers of G_X are excluded. Finally, we place K_3 local replicas at servers whose SIDs are K_3-nearest to G and in the same city as X, with the global and regional replica servers of G_X excluded.

We define the geo-coverage of a local replica to be a metro area because measurement studies [5], [23] have shown that the local focus is usually mapped to the boundary of a metro area or a town. And we define the geo-coverage of a region replica to be a country for political and operational consistency, as each country may have its own political structures and claims on network content management, which heavily influence network objects' query localities.

With regards to resource management, we argue that the local and regional replica deployment balances the demand and supply of server resources. The reason is that the number of local and regional replicas is determined by the number of GUIDs in the city and country respectively, which generally is in proportion to the population of the area, and thus in proportion with the number of deployed PoPs [36]. Our geo-locality aware replica placement exploits such population-based supply and demand relationships for balanced network resource allocation.

B. GNRS Operation Algorithm

We now look at an example of how updates and lookups work in GMap. Suppose host X with GUID G_X attaches to NA N_X. X first sends an Insertion request, which is captured by the nearest GNRS-enabled router (e.g., a border gateway router) in its AS. The GNRS-enabled router then computes the K_1 global K_2 regional and K_3 local replica host servers through the G_X to SID mapping with N_X as the geo-location of G_X. The GNRS-enabled router then sends the $G_X \rightarrow N_X$ mapping to all replica host servers.

Later, suppose host Y wants to lookup the current NA of G_X. Y sends a lookup request that reaches its nearest GNRS-enabled router, which computes all replicas through the G_X to SID mapping assuming G_X is in the same metro area as Y. This router then selects the nearest host servers in terms of routing distance from a list of all global, regional and local replicas, and sends the lookup request to the selected three host servers. When X and Y are in the same metro area or the same country, Y's lookup request will reach the correct local or regional replica host servers. When X and Y are not in the same country, a locality mismatch occurs in that the local and regional replica servers computed by Y's router will not be serving host X. However, even in the case of locality mismatch, Y's lookup requests will always reach a correct global replica host server because global replica placement is independent of network location.

Simultaneously sending three lookup requests can achieve the fastest reply, but at the cost of increased lookup traffic. In our implementation, we use timeouts to pace the sending of requests. Specifically, Y's nearest GNRS-enabled router sends the lookup request first to the nearest local replica host, waits for a timeout T_1 before sending to the nearest regional replica host, and then waits for a timeout T_2 before sending to the nearest global replica host if it has not received a reply. T_1 and T_2 can be set by each border gateway server based on its estimates of the roundtrip time to routers in the same metro area or country. By selecting T_1 and T_2 to be less than 10 ms, sequential lookups reduce the GNRS query traffic at the cost of a slightly increase in the lookup latency in the case of locality mismatch.

When X changes its network address N_X, it needs to update its mapping by sending out an Update request, which is processed similarly to an Insertion request except that after X's nearest GNRS-enabled router sends the new mapping to all replica host servers, it retrieves the previous network address N'_X to determine any obsolete local or regional replica hosts. If X moves out of a metro region or country, the update request will also notify the obsolete local and regional replicas servers to remove the outdated $G_X \rightarrow N'_X$ mapping.

Total write ordering consistency is enforced by the border gateway router to determine the success of an insertion or update so that the NA changes of a GUID are applied in the same order to all replicas. That is the border gateway router confirms a success insertion or update after receiving the acknowledgements from the majority of host servers.

C. SID Availability Management

In GMap, the correctness of a GNRS operation is ensured by GUID-to-SID mappings. This requires GNRS-enabled routers (i.e., SIDs) have a consistent view of global SID availability. However, the dynamic participation of ASs in the GNRS, either by a newly formed AS joining the GNRS or temporary server unavailability caused by a network failure, may result to inconsistent SID availability if changes have not be synchronized between SIDs. In this section, we address consistency management of SID availability.

By exploiting the following three features of GMap, we implemented a two level-consistency to manage SID availability. First, all name servers are virtually organized as an overlay by their SIDs. The nearest XOR distance mapping of GUID-to-SID follows the object locating algorithm in an overlay; thus, the replacement of a replica host SID also follows the overlay failure recovery by selecting a new replica host that should be in the neighbor set of the failed host [27]. (A neighbor set of SID S_X refers to the set of SIDs closest to S_X.) To ensure fast failure recovery, GMap implements sequential consistency among neighboring SIDs so that a GNRS operation will reach the correct replica host when the request reaches a neighbor of the failed SID.

Second, the churn rate of SIDs is given by the frequency of PoP up/down status changes because an SID represents a unit of AS capacity for providing a share of GNRS. An SID's capability should correspond to average capacity of a stable edge AS, which should be equivalent to at least a major PoP's capacity. Depending on the specific implementation of the SID (such as replication among the clusters of routers that constitute the SID) we argue that local router crash should not affect a SID's availability. As there are no statistics that directly show the churn of PoPs, we predict it to be in the same order as large IP prefix re-announcement or withdrawal, at most dozens per hour [4]. On the other hand, the lookup rate issued by an SID is in proportion to the population in its coverage area, which is on the order of tens of millions (or more) per hour. Assuming GUIDs are uniformly distributed among SIDs, and the number of SIDs is in tens of thousands, this suggests that the communication rate between a pair of SIDs is perhaps thousands per hour. Therefore, the lookup rate between a pair of SIDs is orders of magnitude higher than the rate of SID changes. Taking advantage of the higher communication rate between SIDs and the strong consistency between neighbor SIDs, GMap updates of SID availability can be piggybacked on GNRS operation replies for global synchronization.

Third, as each SID should be certified by an NCS, an admission control can be implemented to exclude volatile ASs (e.g., temporary ad hoc networks) from participating in GNRS for a member AS's participation contribution should outweigh its failure recovery overhead. For name resolution service in volatile ASs, we provide an ad hoc name resolution service, which is outside the scope of this paper. Such ASs can utilize GNRS when they have stable connections to the Internet. In the following, we provide a detailed description of GMap's SID consistency management scheme.

The K_1 global replicas of a GUID is set to be large enough so that the probability of simultaneous failure of all K_1 servers is negligibly small. Given K_1, we must specify the size S_{neigh} of the neighbor set to ensure K_1 live replicas of a GUID. When a regional or local replica host fails, the new host can retrieve the mapping information from the global replica host.

We observe here that $S_{neigh} \geq K_1 + 1$ is sufficient to repair single failures in the replica set. To see this, suppose each GUID has K_1 replicas. Name the closest one to the GUID as the primary host and the other $K_1 - 1$ as assistant hosts. Since the primary host is the SID nearest to the GUID, the $K_1 - 1$ assistant hosts will be in the neighbor set of the primary host. Three types of changes in the replica set can occur. (1) If an assistant host has left, the primary host updates its neighbor set and finds a replacement to maintain K_1 available replicas. (2) If the primary host has left, then the failed primary host is in the neighbor set of the new primary host. The new primary host detects the previous primary's failure and determines that it is the substitute. Thus, the new primary finds a new assistant host to keep K_1 available replicas. (3) If a new SID joins, either it will be a new assistant host or the new primary host, the current primary is in the new SID's neighbor set and is notified that it is now an assistant host. In turn, the previous primary will notify an assistant host to be dropped whose SID is no longer K_1 closest to the GUID.

D. Cache Design

The geo-aware K replica placement described in Section II-A provides fast lookup service in the absence of server congestion. However, the popularity of GUIDs is likely to be highly skewed and projected to follow a Zipf distribution [25], [29], [41]. The popular hotspot GUIDs generate orders of magnitude more queries than the rest of the GUIDs. For example, a hotspot website can have 10^6 times more page hits than an average website [25]. While hotspot GUIDs usually are less than 0.1% of the total GUIDs, these GUIDs make their K replica host servers incur query workloads that are orders of magnitude higher than those of other servers. Such workload imbalance may lead to server overflow.

There are several design alternatives that could improve query workload balance, including per-GUID adaptation of the replica number in response to a GUID's popularity, or cloning a hotspot GUID to multiple GUIDs. However, these alternatives rely on per-GUID popularity tracking.Therefore, we design a cache scheme to distribute the query workload of hotspot GUIDs from their K replica servers to all servers along the routes that the hotspot queries follow to the replica servers. In this way, we mitigate server congestion.

In GMap, a small LRU (Least Recently Used) cache is deployed at each name server. In addition to the K replica host servers, a lookup query may be served by a cache along the route to a replica host server. The purpose of our cache is not to improve the overall query hit rate but only to serve queries for the small collection of hotspot GUIDs. Thus we set the cache size to be small enough (e.g., to cache less than 0.05% of total GUIDs) to reduce overhead.

The challenge is cache consistency maintenance in a highly mobile Internet. Since only the K replicas are directly notified by a GUID Update, cached entries of the GUID become invalid if the cache is not refreshed. We apply two methods to ensure the correctness of a cache entry:

- A predefined TTL is applied to each cache entry.
- A go-through probability P_t is associated with each cache entry such that for a cache hit at time t, the query has probability p_t to go through to the next hop on the route to a replica server.

THE TTL field is intended to provide a baseline consistency for all cache entries. The go-through queries will generate reverse-path replies from a next hop cache or the destination replica server that will update the cached mapping. As a result, cache entries with higher hit rates will have more chances to be refreshed. Thus, the cache entries of popular GUIDs will be frequently updated to ensure its correctness.

Figure 1 shows an example of the go-through probabilistic cache in GMap. Suppose the client of laptop A sends the updated GUID to NA mapping to two replicas (i.e., $K = 2$). When laptops B and D, who seek to connect to A, issue a lookup query for GUID G_A and choose to route their queries to the first replica using shortest path routing. An LRU cache with go-through probability 0.3 is deployed at every name server. While the go-through probability may be varied at each cache and also varied with GUIDs, for simplicity we assume a constant go-through probability in this example. The number shown at each hop denotes the probability that the query of B or D reaches that hop assuming G_A is popular and cached at every hop along the respective routes. For instance, B's query has 0.24% probability to reach the replica host and D's query reaches the replica with 0.81% probability. This example shows that the nearer a cache resides to a replica host, the larger number of queries with smaller probability may reach it, thus, the query workload gets evened out along the route. The query reply of will be routed along the reverse path to refresh cache entries that have been go-through-ed.

E. Cache Structure and Algorithm

Each cache entry records a GUID to NA mapping with the following fields for cache maintenance: remaining TTL, the go-through probability P_t for time period t, an update counter U_t recording the number of NA updates during the period t, a hit counter H_t recording the number of queries for the cached GUID. We set the perceived update counter $C_t = U_t/H_t$ to reflect the update rate and the popularity of the cached GUID. Each cache entry also stores C_{t-1} for adjusting the go-through probability. The go-through probability for a cache entry is

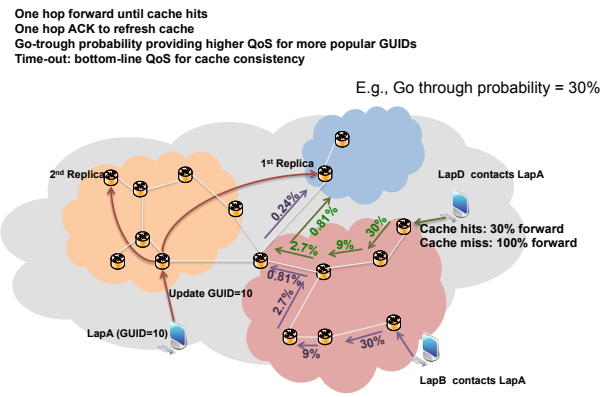


Fig. 1. Cache Assisted Workload Balance Scheme

If receive a query for GUID G_x at time t
  If cache hits
    Forward query with P_t,
      otherwise, reply with cached N_x
    update the status of the cache entry
  Else
    forward query
    create a cache entry with G_x
Else if receive a query ACK for G_x at time t
  If cache hits
    refresh cache entry with N_x in the ACK

Fig. 2. Probabilistic Go-through Cache Algorithm in GMap

initialized to a default value (P_0). Every time period, the go-through probability is adjusted to the changes of perceived GUID's update counter via

$$P_{t+1} = \max(0, \min(P_t + (C_t - C_{t-1}), 1)). \quad (1)$$

An increase in the perceived update counter indicates the cache entry needs a more aggressive go-through probability, while a decrease in the update counter means the go-through probability can be reduced.

The advantage of go-through probability over adaptive TTL for consistency maintenance is that it also reflects the GUID popularity in addition to its update rate. Therefore, we provide stronger consistency for more popular GUID cache entries. A uniform TTL is set as a baseline to time-out any stale cache entry. Figure 2 summarizes the cache algorithm.

F. Cache Overhead Analysis

Since our cache scheme aims for workload balance, we focus on improving the hit rate of hotspot GUIDs rather than that of all requests, as in DNS caches [9], [22]. Thus, a small cache is sufficient to hold hotspot GUIDs while letting the other GUIDs' requests go to the replica servers. The small cache reduces both the storage and lookup latency overhead of our cache scheme. For storage overhead, assume the total number of GUIDs is 10^{12}. And each cache record takes a cache line of 64 bytes, consisting of a 20-byte GUID, up to 5 NAs with total 20 bytes, and allowing for 24 bytes of optional records. In addition, each GUID has a cache index totaling 28 bytes, assuming we take half of a cache line for alignment requests of cache implementation. In total, each GUID takes 1.5 cache lines, and a low cost 64GB cache can accommodate more than 0.05% of total GUIDs, which is enough for GMap to achieve workload balance. For latency overhead, the total cache latency overhead consisting of cache lookup time and memory access time will be around $60 \sim 100$ nanoseconds assuming an implementation with the Cacti cache model and Intel processors [17], [19], which is negligible compared to the network latency.

III. Performance Evaluation

In this section, we experimentally examine the following issues of GMap performance with comparisons to the earlier in-network GNRS scheme DMap [41] and the application layer GNRS scheme Auspice [38]:

- The lookup latency improvement from geo-locality aware replication;
- The server workload balance improvement from cache deployment and cache correctness;
- The robustness to server churns.

A. Experimental Setup

We have developed an event-driven network simulator for GMap evaluation. Descriptions of key components follow.

Simulated event: We simulate two types of events. A network event refers to a GNRS name server leave or join. A GNRS event refers to a GUID insertion, update and lookup.

Network topology: Evaluations of locality aware replication schemes require a large scale global network topology with PoP level geo-information, which is not publicly available. Thus, we developed a topology generator GeoTopo [18] which synthetically generates PoP-level network topologies deployed in given geo-locations. We used Internet measurement data from RocketFuel [39] and current IXP deployment data [3], [8] as input settings for GeoTopo to generate topologies that resemble the global structure and local properties of Internet. The geo-locations where our topology is deployed include all cities that visit the top $10K$ popular websites [1], including more than 80 countries and $20K$ cities worldwide. In our simulations, our topology consists of $20K$ ASs and $300K$ deployed name servers.

Mobility model: In our simulation, we built a mobility model combining both physical and network mobility as follows:

1) For each GUID, we specify its physical mobility level as metro, country, or global.
2) The network mobility model [15] determines the number of network access points a GUID may connect to.
3) Based on the assignment from previous two steps, we determine a "network connection list" of all potential PoPs to which the GUID may connect.
4) For each GUID update event, we simulate the GUID moves between the PoPs in its network connection list.

Based on the statistics in [20], [33] we simulate 60% GUIDs with metro level mobility, 20% with country level mobility and 20% with global level mobility. We define the update rate in a time period as the number of updates over the

number of queries. The default average update rate is set to 0.1, corresponding to one update for every ten queries. We vary the update rate from 0.1 to 0.5 for sensitivity evaluation. To simulate an update, we select a GUID probabilistically in proportion to its update weight which is assigned following an exponential distribution [33] with unit mean.

Workload model: We use a Zipf distribution to model the GUIDs' query popularity with shape parameter 2.04 [25], [41]. To capture the locations of queries to each GUID, we use the exponentially decaying spatial-locality model [5] derived for search engine queries. Given a total location database $L = \{l_i\}$ where l_i denotes a city (location) where PoPs are deployed. For each GUID G_g the probability a query from city l_j is

$$p(g, l_j) = p(F(g))d(l_j, F(g))^{-\beta(g)}, \tag{2}$$

where $F(g)$ is the interest focus point of G_g and $p(F(g))$ is the probability of a query of G_g from its focus point. Note that $d(l_j, F(g))$ is the geo distance from l_j to the focus point $F(g)$ and $\beta(g)$ is the locality coefficient of G_g. As $\beta(g) \to 0$, queries of G_g have no locality (i.e., uniformly spread worldwide); as $\beta(g)$ increases the locality of G_g queries increases. The query workload of GUID G_g is generated as follows: we assign $F(g)$ as current location of G_g, and $p(F(g))$ is $p(K_g)$ where K_g is the popularity of G_g. Following [5], we set the $\beta(g) = 1.1$ for GUIDs of city-wide locality interest (termed as local GUIDs) and $\beta(g) = 0.5$ for GUIDs of country-wide locality interest (termed as regional GUIDs) and $\beta(g) = 0.1$ for GUIDs of global interest (termed as global GUIDs). We normalize the probability $d(l_j, F(g))^{-\beta(g)}$ to 1 over all cities in our simulation database.

B. Experimental Results

Lookup Latency: In Figure 3, we compare the lookup latency of DMap [41], Auspice [38] and GMap by varying the mixture of local, regional and global GUIDs. Figures 3(a)-(c) show the results when (a) 60% of GUIDs are local, (b) 60% are regional, and (c) 60% are global. For fairness, the same number of name servers are deployed for the three schemes. For each GUID, we deploy 7 global replicas in DMap, and 5 global replicas, 1 regional and 1 local replica in GMap. For Auspice, we calculate the dynamic replica placement for each GUID following [38, equation (1)] with the equal total server capacity. Specifically, we record the read count r_i and the write count w_i for each GUID G_i assuming the minimal replica number $F = 3$ following their guideline to calculate β based on

$$\sum_i r_i + 7\sum_i w_i = \sum_i r_i + \sum_i (F + \beta\frac{r_i}{w_i})w_i. \tag{3}$$

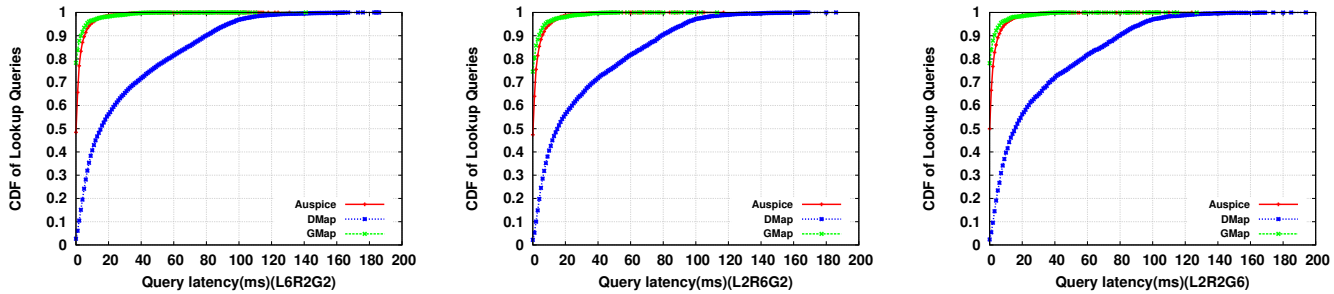
We then record the query count of each GUID from each country (i.e., following their country level region partition) to deploy the $\beta\frac{r_i}{w_i}$ replicas of G_i among the top 10% of countries in which the queries to G_i are most concentrated and $F = 3$ replicas randomly worldwide.

The results in Figures 3(a)-(c) show that GMap and Auspice both provide faster lookup than DMap because of the locality aware replica placement. The 95th percentile latency is improved from around 100ms in DMap to less than 20ms in GMap and Auspice. The results are not sensitive to the varying mixture of local, regional and global query locality combinations. In GMap, we deploy the three layer of local, regional and global replicas for each GUID to accommodate the varied query locality and the per-GUID optimized replica placement in Auspice also adapts well. GMap has more lookups completed within 10ms than Auspice because the region partition in Auspice is at country level while GMap deploys city-wide local replica to serve local queries. Auspice's latency results catch up at 20ms when the regional replicas take effect. GMap has somewhat fewer queries exceeding 120ms than Auspice because GMap has more randomly deployed global replicas for each GUID than Auspice. Overall, GMap provides comparably fast lookup service as Auspice without relying on per-GUID and per-region traffic statistics maintenance and synchronization overhead.

Workload Balance: We compare the workload balance in three cache scenarios: (1) DMap, which has no cache scheme, (2) deploying cache only at the server who issues a query (i.e., source cache), and (3) GMap, which deploys caches along the route from the source issuing a query to the selected replica host (i.e., route cache). Figure 4 shows the normalized GNRS server workload distribution when the cache size is 0.001% of total GUIDs. The normalized workload is the ratio of each server's workload divided by the mean of all servers' workloads. Without cache, the maximum normalized workload of DMap can be over 1000, corresponding to certain servers workload being more than 1000 times the average server workload. The source cache scheme reduces the maximum normalized workload to ∼400 and the route cache in GMap further reduces it to 20. We choose the route cache as a default setting as it makes 99.99% name servers' workloads are within 7 times the mean workload.

Cache Correctness: We measure the error rate as the number of incorrect cache replies (to lookup queries) over the number of issued queries. Figure 5 shows the min, 25^{th} percentile, median, 75^{th} percentile and max error rates by varying the update rate (defined as the average update number over the average query number) from 0.2 to 1. The results are over 50 simulation runs. The default go-through probability is applied when a GUID is newly cached. After each run, each server adjusts the go-through probability for every GUID in its cache based on Equation 1 and local cache statistics. When the update frequency is 0.2, the median successful query rate is over 99.7%. With 0.5 update frequency, the median successful query rate is over 98.5%. Even with the update frequency equal to 1, the median successful query rate is over 96.8%. The cache error can be corrected by "late-binding," where the last hop router re-queries the GNRS for a refreshed location when it discovers current location is invalid.

The overall cache error rate is low because the only GUIDs that can be kept in the small (0.05% of total GUIDs) LRU-based cache are those with extremely high popularity. Moreover, these popular GUIDs typically have below-average update frequencies given their query rates are orders of magnitude higher than those of ordinary GUIDs. Increasing the default go-through probability lowers the error rate to a very



60% local, 20% regional, 20% global (a)

20% local, 60% regional, 20% global (b)

20% local, 20% regional, 60% global (c)

Fig. 3. Lookup latency comparisons with varying mixtures of GUID query locality.

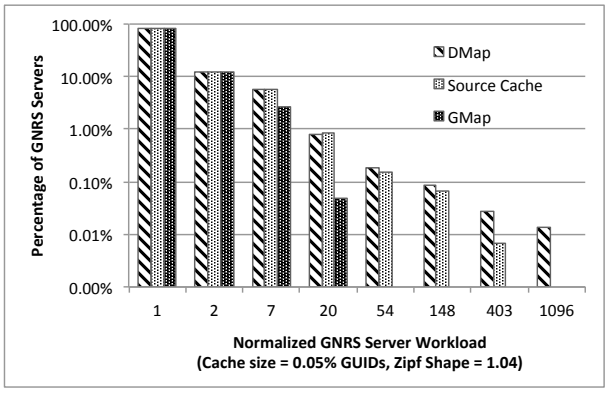


Fig. 4. Workload Balance Comparisons

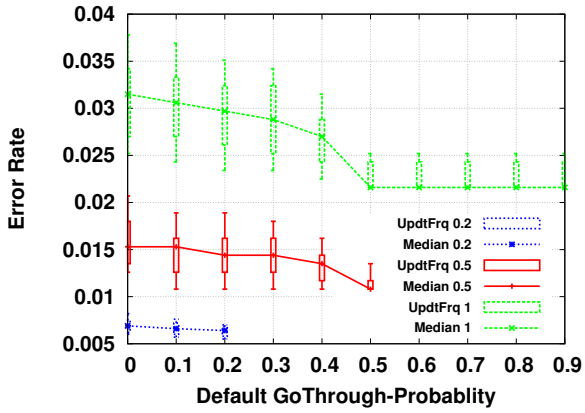


Fig. 5. Cache Error Rate – impacts of Update Frequency and Go Through Probablity

limited extent, thus, we set the default go-through probability to be zero. The adaptive go-through probability cannot keep the error rate when increasing the update frequency because over 99% cache errors occur for newly cached GUIDs when the go-through probability has yet be updated.

Churn robustness: In this experiment we study the impact of GNRS server churn behaviors. As there are new IP prefixes announced or existing prefixes withdrawn by ASes, we expect the GNRS servers have on and off behaviors (i.e., the churn of servers) due to AS capacity fluctuations. To model a churn rate at $x\%$, we set the expected server off time $E[\text{off}]$ and

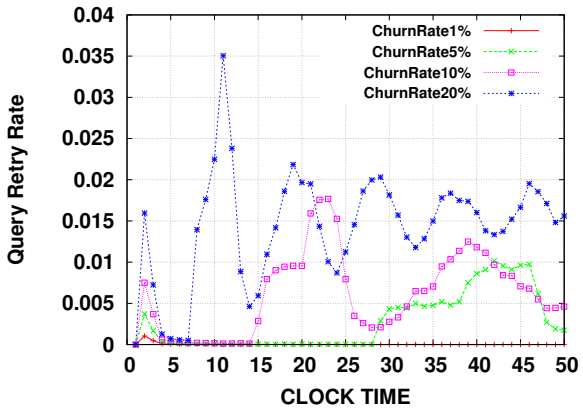


Fig. 6. Query Retry Rate – Varying Churn Rates

expected server on time $E[\text{on}]$ to satisfy $\frac{E[\text{off}]}{E[\text{off}]+E[\text{on}]} = x\%$, and randomly select $x\%$ of servers to be off at the beginning of the experiment. In the course of the experiment, the servers independently cycle on and off following their expected on and off times.

Figure 6 shows the retry rates of queries varying the server churn rate from 1% to 20%, we set the expected server off time to be 2 clock ticks in the experiment. When the churn rate is 1%, the retry rate is negligibly small at less than 0.001. When the churn rate increases from 5% to 10% to 20%, the maximum retry rate increases from 0.01 to 0.018 to 0.035. The overall retry rate is small enough to keep the 99-percentile query latency almost intact as shown in Table I, while the maximum query latency is increased from 158ms to 305ms when the churn rate is 5% due to the retry and to 529ms when the churn rate is 20%, due to multiple retries for a single query. The overall small retry rate shows that our in-network GRNS scheme is robust to server churn behaviors due to sequential consistency maintained among neighbor servers and the piggybacking of server availability updates in the query replies.

IV. SID Capability Analysis

In this section, we provide a simple analysis to shed some light on how an AS can gauge its capacity in order to determine how many SIDs it should apply for participation

TABLE I
QUERY LATENCY VARYING CHURN RATES

Churn	99-perc. Lat.	Max Lat.	Multi-Retry Ratio
0	70 ms	158 ms	0
1%	70 ms	303 ms	0
5%	70 ms	305 ms	0
10%	72 ms	505 ms	0.00001
20%	76 ms	519 ms	0.00005

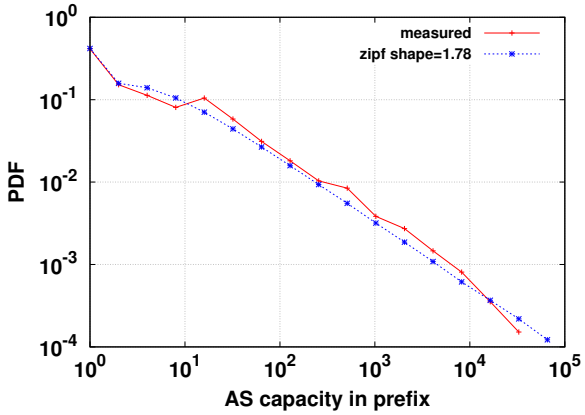


Fig. 7. AS Capacity Distribution

in the GNRS. We use the number of servers an AS is going to dedicate to name resolution as a measure of capacity. First, we benchmark the GNRS throughput of a single server using our prototype implementation on Orbit [31]. We find that the Intel-I7 server, a modest commodity server, can process 110K GNRS requests (updates or queries) per second. Then, we model the AS capacity distribution through IP prefix announcement. We use the measured prefix announcement from DIMES [2] covering around 20K ASs. The AS capacity distribution can be well modeled by a Zipf distribution with shape parameter 1.73 as shown in Figure 7. Finally, we predict the mean number of servers required by an SID in GMap in Figure 8 assuming total 1 trillion GUIDs with the current Internet of 40K ASes and future Internet of 100 ASes, and the requests for each GUID is 3 per second (e.g., 2 queries/sec and 1 update/sec). For example, with 1 trillion GUIDs and 40K ASes, given the mean number of SIDs per AS is 5, the mean number of intel-I7 servers per SID is 12.5, and in total we need 2500K GNRS servers world-wide. And for 100 ASes, the mean number of SIDs per AS is 5, as the mean number of Intel-I7 servers per SID is about 5. Since the Intel-I7 server capacity is very modest, our estimates of server requirements are likely to be conservative.

V. RELATED WORKS

Literature proposals to re-architect the Internet for better mobility support can be categorized into three directions – indirection-based routing, name-based routing, and global name-to-address resolution service. **Indirection based routing** schemes assign a rarely changed location (i.e., home address) to each network entity so that the mobility of the network entity is transparent to other network entities. The home address, which may be an IP address in Mobile IP [34] or a consistent-hash location from a flat identifier in i3 [40], acts as a permanent address for an agent that tracks the current network address of an entity and relays all intended packets to that address. A significant routing stretch is the cost paid for using this indirection as every data packet needs to be routed via the home address. **Name-based routing** schemes handle mobility by routing directly over names assuming the routing update can be propagated globally on the orders of milliseconds. This is a daunting challenge given that the inter-domain routing can take several minutes to converge today. Another fundamental challenge to realize name-based routing is the trade-off of forwarding table size at routers and path stretch without mobility. Theoretical [26] and experimental [7] studies conclude that routing over N flat identifiers imposes a prohibitive $\Omega(N)$ forwarding table size per router in order to achieve a stretch factor around three times that of shortest-path routing. The prohibitive forwarding table size makes scalability come into question. Hierarchical naming structures [21], [24] have been proposed to alleviate the limitations caused by forwarding table size, but this does not accommodate frequent mobility as names move out of their original hierarchical namespace and becomes special cases that need to occupy special entries in the forwarding table, which offsets the savings from the hierarchical naming structure. It has been reaffirmed [15] that high mobility makes routing directly over structured names as challenging as routing over flat names unless indirection routing or a name resolution service is used. **Global name-to-address resolution** service resolves the identity name to its current network address as the first step for establishing network connections. A number of clean-slate designs for re-architecture Internet rely on such global name resolution service to convert an identifier in HIP [28], AIP [4], XIA [16], MobilityFirst [37], LISP [13] or HAIR [14] to either an IP address, a self-certifying network identifier or a hierarchical locator that facilitate routing. Comparisons on the three alternative directions to achieve identifier and locator separation [15], [38] conclude that the provision of GNRS offers the best trade-off between mobility update efficiency, lookup efficiency, and routing performance.

The name resolution scheme Auspice [38] achieves geo-locality aware replica placement by aggregating each GUID's read and write demands in each partitioned geo-region. Its per-GUID state maintenance overhead and periodic redeployment require significant service resources to scale with the increasing number of GUIDs. Extra latency is also introduced for tracing the GUID replica host server redeployment. Another seminal name resolution work CoDoNS [35] uses proactive caching to achieve a pre-defined lookup latency for all GUIDs, based on the total Zipf distribution shape parameter and the popularity rank of each GUIDs. Such proactive caching is not only limited by the per-GUID workload state maintenance overhead but also has problematic update cost for popular GUIDs that are cached by a large number of servers. The in-network GNRS scheme DMap [41], proposed a random K-replica global server placement scheme for identity-to-locator resolution. However, placements of these replica servers were oblivious to the geo-locality of the network entity being served and its lookup query popularity. This caused inefficient lookup

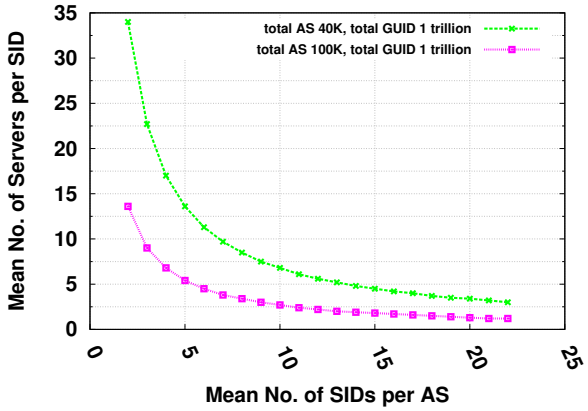


Fig. 8. Server Requirement

query processing and potential name server workload congestion. In addition to these performance problems, DMap's reliance on IP prefixes for replica placement precluded its use in non-IP networks such as name-based networks in [30].

VI. Conclusions

In this paper, we presented an in-network GNRS scheme GMap that achieves fast name-address lookup services and server workload balance with low computing and storage requirements across the network. We deployed geo-location aware replicas to improve lookup latency by exploiting the spatial locality patterns in the lookup queries. A caching technique is designed to distribute hotspot GUIDs' query workload for avoiding server congestion. The experiments show that our geo-location aware replica placement reduces the 95th percentile query latency from roughly 100ms to less than 20ms and our cache deployment mitigates the maximum server workload deviation by more than fifty-fold.

References

[1] Alexa database. http://www.alexa.com/.

[2] The dimes project. http://www.netdimes.org/.

[3] Ager, B., Chatzis, N., Feldmann, A., Sarrar, N., Uhlig, S., and Willinger, W. Anatomy of a large european ixp. In *ACM SIGCOMM* (2012).

[4] Andersen, D. G., Balakrishnan, H., Feamster, N., Koponen, T., Moon, D., and Shenker, S. Accountable internet protocol. In *ACM SIGCOMM* (2008).

[5] Backstrom, L., Kleinberg, J., Kumar, R., and Novak, J. Spatial variation in search engine queries. In *WWW '08*.

[6] Brodersen, A., Scellato, S., and Wattenhofer, M. Youtube around the world: Geographic popularity of videos. In *WWW '12*.

[7] Caesar, M., Condie, T., Kannan, J., Lakshminarayanan, K., Stoica, I., and Shenker, S. Rofl: Routing on flat labels.

[8] Chatzis, N., Smaragdakis, G., Bottger, J., Krenc, T., and Feldmann, A. On the benefits of using a large ixp as an internet vantage point. In *ACM IMC* (2013).

[9] Choungmo Fofack, N., Nain, P., Neglia, G., and Towsley, D. Analysis of TTL-based Cache Networks. Research Report RR-7883, 2012. Voir aussi actes de la conférence : ValueTools - 6th International Conference on Performance Evaluation Methodologies and Tools - 2012 (2012) (http://hal.inria.fr/hal-00760915).

[10] Chung, T., Han, J., Lee, H., Kangasharju, J., Kwon, T., and Choi, Y. Spatial and temporal locality of content in bittorrent: A measurement study. In *IFIP Networking '13*.

[11] Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2013-2018. http://www.cisco.com/.

[12] Evans, D. The internet of things how the next evolution of the internet is changing everything cisco white paper.

[13] Farinacci, D., Fuller, V., Oran, D., Meyer, D., and Brim, S. Locator/id separation protocol (lisp). In *IETF Internet Standard, RFC 6830*.

[14] Feldmann, A., Cittadini, L., Mühlbauer, W., Bush, R., and Maennel, O. Hair: Hierarchical architecture for internet routing. In *ReArch* (2009).

[15] Gao, Z., Venkataramani, A., Kurose, J., and Heimlicher, S. Towards a quantitative comparison of the cost-benefit trade-offs of location-independent network architectures. In *ACM SIGCOMM* (2014).

[16] Han, D., Anand, A., Dogar, F., and et al., B. L. Xia: Efficient support for evolvable internetworking. In *USENIX NSDI* (2012).

[17] HPLab. Cacti an integrated cache and memory access time, cycle time, area, leakage, and dynamic power model. http://www.hpl.hp.com/research/cacti/.

[18] Hu, Y., Zhang, F., Ramakrishnan, K. K., and Raychaudhuri, D. Geotopo: A pop-level topology generator for evaluation of future internet architectures. In *IEEE ICNP* (2015).

[19] Intel. Performance analysis guide for intel processors. https://software.intel.com/.

[20] Isaacman, S., Becker, R., Cáceres, R., Martonosi, M., Rowland, J., Varshavsky, A., and Willinger, W. Human mobility modeling at metropolitan scales. In *MobiSys '12*.

[21] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. Networking named content. In *CoNEXT '09* (2009).

[22] Jung, J., Berger, A. W., and Balakrishnan, H. Modeling TTL-based Internet Caches. In *IEEE INFOCOM 2003*.

[23] Kamath, K. Y., Caverlee, J., Lee, K., and Cheng, Z. Spatio-temporal dynamics of online memes: A study of geo-tagged tweets. In *WWW '13*.

[24] Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K. H., Shenker, S., and Stoica, I. A data-oriented (and beyond) network architecture. In *ACM SIGCOMM* (2007).

[25] Krashakov, S. A., Teslyuk, A. B., and Shchur, L. N. On the universality of rank distributions of website popularity. *Computer Networks 50*, 11 (2006).

[26] Krioukov, D., claffy, k. c., Fall, K., and Brady, A. On compact routing for the internet. *SIGCOMM Comput. Commun. Rev. 37*, 3 (2007).

[27] Maymounkov, P., and Mazières, D. Kademlia: A peer-to-peer information system based on the xor metric. In *IEEE IPTPS'01* (2002).

[28] Moskowitz, R., and Nikander, P. Host identity protocol (hip) architecture. In *IETF Internet Standard, RFC 4423*.

[29] Newman, M. E. J. Power laws, pareto distributions and zipf's law. *Contemporary Physics 46*, 5 (2005).

[30] NSF. Future internet architecture project. http://www.nets-fia.net/.

[31] Orbit. Open-access research testbed for next-generation wireless networks. http://www.orbit-lab.org/.

[32] Pappas, V., Massey, D., Terzis, A., and Zhang, L. A comparative study of the dns design with dht-based alternatives. In *IEEE INFOCOM'06*.

[33] Paul, U., Subramanian, A. P., Buddhikot, M. M., and Das, S. R. Understanding traffic dynamics in cellular data networks. In *IEEE INFOCOM '11*.

[34] Perkins, C. Ip mobility support for ipv4. In *IETF Internet Standard, RFC 3220*.

[35] Ramasubramanian, V., and Sirer, E. G. The design and implementation of a next generation name service for the internet. In *ACM SIGCOMM '04*.

[36] Rasti, A. H., Magharei, N., Rejaie, R., and Willinger, W. Eyeball ases: From geography to connectivity. In *ACM IMC* (2010).

[37] Raychaudhuri, D., Nagaraja, K., and Venkataramani, A. Mobilityfirst: A robust and trustworthy mobility-centric architecture for the future internet. *SIGMOBILE Mob. Comput. Commun. Rev. 16*, 3 (2012).

[38] Sharma, A., Tie, X., Uppal, H., Venkataramani, A., Westbrook, D., and Yadav, A. A global name service for a highly mobile internet. In *ACM SIGCOMM* (2014).

[39] Spring, N., Mahajan, R., and Anderson, T. Quantifying the causes of path inflation. In *ACM SIGCOMM* (2003).

[40] Stoica, I., Adkins, D., Zhuang, S., Shenker, S., and Surana, S. Internet indirection infrastructure. In *ACM SIGCOMM* (2002).

[41] Vu, T., Baid, A., Zhang, Y., Nguyen, T. D., Fukuyama, J., Martin, R. P., and Raychaudhuri, D. Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet. In *IEEE ICDCS* (2012).