**Examining Key Mobility Resources through Denial of Service Attacks on proposed Global Name Resolution Services**

by Colleen T. Rock

S.B., Massachusetts Institute of Technology (2015),

Computer Science and Engineering

Submitted to the

Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Computer Science and Engineering

at the

Massachusetts Institute of Technology

September 2016

© Massachusetts Institute of Technology 2016. All rights reserved.

Author: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Department of Electrical Engineering and Computer Science

May 20, 2016

Certified by: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Karen R. Sollins

Principal Scientist

MIT Computer Science and Artificial Intelligence Laboratory

Accepted by: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Christopher Terman

Chairman, Masters of Engineering Thesis Committee

# Acknowledgements

Karen, parents, Dan, G8

This work was funded in part under NSF Grant 1413973, “NeTS: Large: Collaborative Research: Location-Independent Networks: Evaluation Strategies and Studies”.

Table of Contents

[Acknowledgements 2](#_Toc457772925)

[1. Introduction 5](#_Toc457772926)

[1.1 Example 5](#_Toc457772927)

[1.2 Problem Domain 5](#_Toc457772928)

[1.3 Summary of the Problem 6](#_Toc457772929)

[1.4 Summary of Findings 6](#_Toc457772930)

[1.5 Methodology 6](#_Toc457772931)

[2. Background 7](#_Toc457772932)

[2.1 MobilityFirst 7](#_Toc457772933)

[2.2 Denial of Service Attacks 9](#_Toc457772934)

[3. GNRS System Designs 11](#_Toc457772935)

[3.1 DMap 12](#_Toc457772936)

[3.1.1 Local Replication 13](#_Toc457772937)

[3.1.2 Multihoming 14](#_Toc457772938)

[3.1.3 Edge Cases 15](#_Toc457772939)

[3.2 Auspice 16](#_Toc457772940)

[3.2.1 Replica-Controllers 16](#_Toc457772941)

[3.2.2. Demand Aware Placement 17](#_Toc457772942)

[3.2.3 Paxos 18](#_Toc457772943)

[3.3 GMap 18](#_Toc457772944)

[3.3.1 Asking the Geographically Closest Server 19](#_Toc457772945)

[3.3.2 GMap Uses Caching to Compensate for Popular GUIDs 21](#_Toc457772946)

[3.4 Summary 21](#_Toc457772947)

[4. Hypotheses, Key Resources and Possible Vulnerabilities 24](#_Toc457772948)

[4.1 Possible Attacks in Auspice 24](#_Toc457772949)

[4.2 Possible Attacks in GMap 26](#_Toc457772950)

[4.3 Possible Attacks in DMap 28](#_Toc457772951)

[4.3.1 Malicious AS 28](#_Toc457772952)

[4.3.2 Rainbow Table 29](#_Toc457772953)

[4.4 Summary 29](#_Toc457772954)

[5. Experimental Set Up 31](#_Toc457772955)

[5.1 ORBIT Testbed 31](#_Toc457772956)

[5.2 Availability of GNRS Code 32](#_Toc457772957)

[Abstraction to the Autonomous System Level 32](#_Toc457772958)

[6. DMap Server Attack 33](#_Toc457772959)

[6.1 Recursive Inserts and Updates 33](#_Toc457772960)

[6.2 Attack Setup (in example and “do-ability” in real world) 34](#_Toc457772961)

[6.x Results 36](#_Toc457772962)

[6.X Tradeoffs 37](#_Toc457772963)

[7. Effects of Transport Layer Protocol on GNRS Design 37](#_Toc457772964)

[8. Future Work 38](#_Toc457772965)

[Topology 38](#_Toc457772966)

[Mobility Model 39](#_Toc457772967)

[9. Comparison to Today’s Internet 40](#_Toc457772968)

[10. References 42](#_Toc457772969)

# Introduction

## 1.1 Example

Imagine a post office that provides a service allowing you to send them a postcard saying “I’m staying at my friend’s house for a couple days; please send my mail there.” This allows you to continue receiving your mail while couch-surfing. However, this service that supports frequent movement requires the post office to keep track of all the places where people request that their mail be sent. What if someone mischievously sends many postcards falsely claiming that they moved, or impersonates you saying that you moved? The post office now has to sort through all of these post cards to determine which ones are valid, and although the postman could check to see if you are actually at the new location while delivering mail there, this verification would be very time consuming.

This example[[1]](#footnote-1) illustrates some of the issues that arise when a system attempts to support mobility occurring at a time scale comparable to information delivery. In the Internet, unlike the postal system, your name and address are represented as one value, your Internet Protocol (IP) Address. MobilityFirst (MF) is a design for a Future Internet Architecture to replace the Internet. It supports separating a name, what they call a Globally Unique Identifier (GUID), from a device’s location, called a Network Address.

## 1.2 Problem Domain

*Why is the current internet insufficient? Why does MF exist? Why is MF good? Why can’t MF use DNS? Also mention NDN, etc.*

## 1.3 Summary of the Problem

Since MobilityFirst is a mostly untested, new architecture, I aim to understand its capabilities, strengths and weaknesses. In this thesis I examine this architecture more deeply through the “lens” of denial of service opportunities that arise from its unique characteristics intended to support wide scale and extremely dynamic mobility.

The purpose of studying mobility is to unearth more about the resources that are critical to node mobility, and gain insight informing how to design robust mobile protocols in the current Internet and systematic ways of thinking about mobility.

*\* What are tradeoffs made by name-resolution services trying to support rapid mobility. In what ways can decisions made about these tradeoffs in a particular system be taken advantage of to limit the system’s ability to function as designed?*

## 1.4 Summary of Findings

## 1.5 Methodology

This thesis presents related work in Chapter 2 to set the background for the problem, discussed in Chapter 3, followed by the approach in Chapter 4.

# Background

Some work has been done studying Denial of Service attacks in another Future Internet Architecture, Named Data Networking, but not in MobilityFirst. Examining Denial of Service attacks led to a better understanding of Named Data Networking architecture and influenced a revision of the protocol to include negative acknowledgements [1, 2, 3].

## 2.1 MobilityFirst

MobilityFirst (MF) [4] is a proposed Future Internet Architecture, assuming we could replace the Transport and Internet Layers of the Internet protocol suite. MobilityFirst’s primary design goals are mobility and trustworthiness.

The main aspects of the MobilityFirst architecture are storage aware routing, the management plane, the compute layer, and the GNRS[[2]](#footnote-2). The management plane is federated and provides aggregate network information used to perform accounting and address problems and attacks.

This thesis focuses on the GNRS, which maps GUIDs to NAs. Instead of IP addresses, MobilityFirst has Globally Unique Identifiers (GUIDs) which are like “names”. Like other proposed Future Internet Architectures, MF does not conflate identity and location [4, 5]. In MF, GUIDs are self-certifying and represent interfaces, devices, services, content, human end-users, and groups of GUIDs [4].

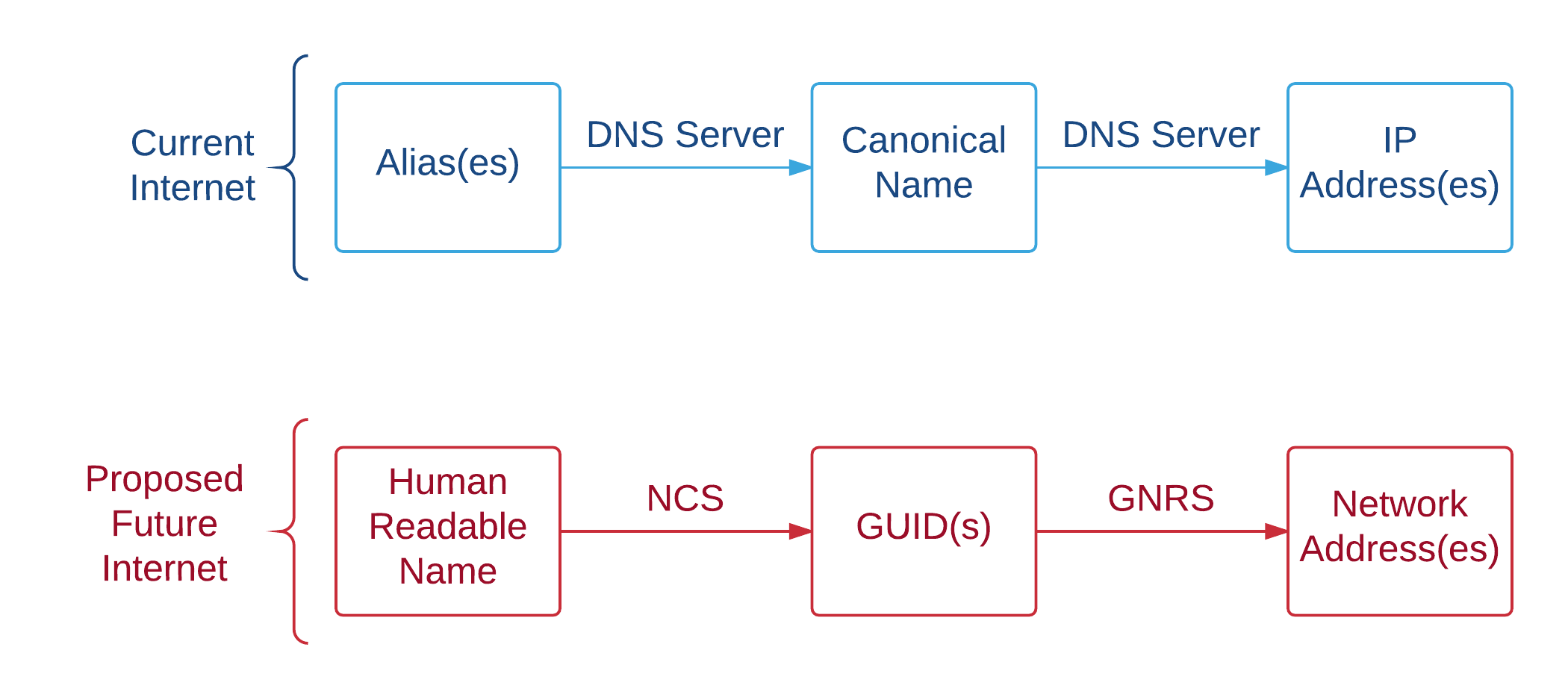


Figure . **GNRS and DNS.** The GNRS provides one of the two resolution tasks that the Domain Name Service handles in the current Internet. The other is performed by a Name Certification Service. Recall from 2.1 that a Network Address is a reference to a sub-network that knows where a GUID is. NAs are used inside the Future Internet to forward traffic to its destination.

MF encourages competition between Name Certification Services (NCSs), which bind “human-readable descriptors” to GUIDs. Unlike IP addresses, GUIDs are not hierarchical. Since they are flat, naively, looking up the location of a GUID can take time on the order of the number of GUIDs. Since routing at the level of GUIDs can be so inefficient, a Global Name Resolution Service (GNRS) is necessary to efficiently map GUIDs to one or more Network Addresses, which correspond to either an Autonomous System or Internet Service Provider that is connected to the GUID. Network Addresses are also self-certifying, which MF authors claim make it easier to create new networks and new Network Addresses on the fly [4]. The GNRS essentially *resolves* a GUID by providing a translation form GUID to NA.

I am looking at the GNRS specifically because it is a key resource everything needs access to: “The GUID-based communication assisted by the GNS forms the ‘narrow waist’ of the MF architecture…” [4]. Another key resource is the GUID and its self-certifying nature.

## 2.2 Denial of Service Attacks

Denial of Service (DoS) attacks aim to deny access to a service or part of a network to an audience. A Distributed Denial of Service (DDoS) attack is a DoS attack that is distributed across many computers, often a recruited botnet. DoS and DDoS attacks illuminate the security and resiliency of proposed architectures by demonstrating how a system can cease functioning [6]. The challenge of trying to prevent an attack without creating an opportunity for a new one can be especially insightful.

“A Taxonomy of DDoS Attack and DDoS Defense Mechanisms” [7] suggests ways to group DDoS attacks: by degree of automation, source address validity, attack rate dynamics and persistence of agent set, possibility of characterization, exploited weakness, victim type, and the impact on the victim.

In their “Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems” Peng et. al. describe four categories of defense against DoS: Prevention, Detection, Source Identification, and Reaction [8]. Prevention mainly uses filtering to limit the ability to spoof IP addresses, including ingress/egress filtering, Router-Based Packet Filtering which basically filters between ASes at border routers, and the Source Address Validity Enforcement (SAVE) Protocol, in which routers keep a table with valid addresses for each interface. The authors ultimately claim that spoofing is happening less frequently in DoS attacks. One reason that DDoS attacks are so hard to detect is that a “flash crowd”, legitimate traffic that can occur when many people suddenly look up the same thing, can appear like a DDoS attack. Denial of Service attacks are tricky precisely because they can look like legitimate traffic; it’s much easier to have false positives than other network security attack types. Defenses for attack detection break into two main categories: DoS attack-specific, which look for known attacks, and anomaly-based, where a traffic is compared to a “normal profile”. The challenge with identifying the source of an attack is that IP routing is stateless and addresses are not verified before they are allowed to send traffic. One suggestion to aid in detection is “Probabilistic IP Traceback” where routers probabilistically append their IP address to packets such that the packets can be “traced back” from where they came from. Overall, Peng et. al. conclude that the decentralization of the Internet and lack of economic incentives to implement defenses, along with the current state of affairs being “good enough” are the main challenges that prevent the implementation of defenses. They suggest that ISPs are a good place to start because they are a preexisting *group* of routers than can work cooperatively without overhauling the entire Internet.

A significant feature of both of these works is that they focus on types of attacks. In contrast, my thesis will focus on the kinds of resources that are attackable, in order to characterize various proposed aspects of MobilityFirst.

# GNRS System Designs

The Internet was not designed for mobility, but with the development and prevalence of wireless technology, mobility has become commonplace. MobilityFirst was designed with the primary goals of mobility and trust. Mobility occurs when a single identity relocates. This includes, but does not always necessitate, a device physically moving. For example, a phone switching between WiFi and 4G LTE changes the network address of the phone. Even though the phone did not physically move, the route a packet takes over the network topology may have changed drastically, resulting in packet loss due to mobility. This mobility can occur rapidly, causing Future Internet Architectures to support the separation of nominal identifiers from location identifiers. “Intuitively, it is easier to work with networking primitives based on identifiers when the locator changes faster than the timescales of the communication session” [9, p. 1].

Just like the white pages allow us to look up someone’s address given their name, the Domain Name Service will translate well-known names, like “google.com” into Google’s IP address for us. Since IP addresses represent both name and location, only one step is necessary. In MobilityFirst, after translating a human-readable name like “google.com” to a corresponding GUID, there is an extra step to find the location of that GUID. This is advantageous because it allows an entity to retain its globally unique identifier, but change its location. This extra step of resolving a GUID to its current location is accomplished by the Global Name Resolution Service. The first step, resolving a human-friendly name to a GUID, is performed by a Name Certification Service. The details related to this resolution are outside the scope of this project because it is outside the core architecture of MobilityFirst. This thesis focuses on the second step of retrieving a Network Address for a GUID, which is done with a Global Name Resolution Service (GNRS).

Adding this location flexibility creates new opportunities for attacks in the global network, as adding new features creates opportunities for bugs and misuse in any system. In the most basic form there are three primary attackable resources in a network: bandwidth, computation, and storage. The proposed ideas have a tradeoff between these three resources. For example, supporting mobility without having a GNRS could require the routers inside the network to obtain and store up to date location information, taxing their storage. If the location information is represented in a resolution service that is invoked prior to the traffic being sent, then the storage, computation and bandwidth resources proposed for the resolution service must be analyzed to determine how they introduce vulnerabilities into the network.

There are three proposed GNRS implementations, DMap, GMap and Auspice. The rest of this Chapter describes and compares the three implementations, and findings on their DoS vulnerabilities is discussed in Chapter 4.

## 3.1 DMap

DMap, a proposed GNRS named after its use of Direct Mapping, is an in-network distributed hash table[[3]](#footnote-3) [9]. DMap distributes *K* global replicas for each GUID to NA mapping among participating routers. When a new device connects, its GUID is hashed into *K* IP addresses using *K* independent hash functions. The servers that announce these IP addresses store a mapping between the GUID and its location(s). The location, called a “Network Address”, is a reference to the Autonomous System that contains the new device. An example with *K*=1 is shown in Figure 2.



Figure . **DMap Insert and Lookup**. GUID GX, located at 58.0.0.1, connects to the network via the server at 45.0.0.5. This server hashes GX with hash function H to obtain the address of the server that will store GX’s location, in this case, 128.5.5.5. GX’s location can be queried from 128.5.5.5, which any server can compute by taking H(GX).

### 3.1.1 Local Replication

In Figure 2, if a different router in AS 1 wants to route to GX, it would first have to hash GX: *H*(GX) = 128.5.5.5. Then , if naively following the DMap protocol, it would ask 128.5.5.5 where GX is located. This could result in an unnecessarily long lookup time, especially if AS 1 and AS 2 are far away or do not have a high bandwidth connection. DMap attempts to deal with this by including local replication. In addition to each of the *K* global copies of the (GX, AS 1) mapping, it will also use a hash function to determine a local server on which to store (GX, AS 1). When servers look up a GUID, they simultaneously send requests to the local server that would know if the GUID is in their Autonomous System and to a “global” copy, using one of the original *K* hash functions. As shown in Figure 3, the server in AS 4 also queries 198.4.4.4 to see if GX is located in AS 4.

Figure . **DMap with Local Replication**. When looking up a GUID, servers use a local hash function, HLOCAL, to simultaneously ask a local server if GX is in the same Autonomous System as the requester. Similarly, when a GUID is inserted, a local hash function is used to place a DMap record in that GUID’s AS.

### 3.1.2 Multihoming

DMap supports multihoming, a device attached to the network with more than one connection. Multiple locations can be sent with an original Insert request. Alternatively, if DMap has already has a location for a GUID and receives another Insert message, the additional address is simply appended to the list of addresses for that GUID. DMap permits each GUID to be associated with up to 5 NAs at once, and to assign expiration times and prioritization weights to each network address. Since a new Insert message does not remove an old network address for a GUID, an Update message is also defined for when the GUID moves from a network address to a new one.

### 3.1.3 Edge Cases

DMap is currently implemented on top of the IP layer, and relies on IPv4 [10]. DMap uses the list of BGP prefix announcements to know which ASs announce which subsets of the IPv4 space in the current version of DMap. An edge case DMap considers is that of hashing to “nonexistent” IP addresses. The IP Hole Problem is the phenomenon that some IP addresses are unclaimed by ASs. If a GUID hashes to an unclaimed network address, it is rehashed (up to *M*-1 times). If, after *M* total hashes, a GUID still hashes to an unannounced IP address, a Deputy AS, an AS with the closest IP distance to the final hashed IP address, will host the GUID-NA mapping. If an AS has new servers join and announces additional IP addresses, it is possible that some already hosted GUID mappings would belong on the new server. The DMap architecture includes a *GUID migration message* so that the new announcing AS can tell the corresponding Deputy AS that it has now joined the network and can host the mapping, prompting the Deputy AS to drop that mapping.

## 3.2 Auspice

Auspice [11, 12] is another proposed global name resolution service (GNRS) that can support current Internet architecture and MobilityFirst Future Internet Architecture. Auspice’s records are more broadly defined than DMap’s GUID to NA mapping. They allow each GUID to be associated with a name record consisting of arbitrarily many key value pairs. This supports “novel network-layer functions such as simultaneous mid-connection mobility and context-aware communication” [11, p. 7]. Auspice allows blacklists and/or whitelists for each that limit write access to each GUID’s name record to a set of GUIDs.

{"occupation":"rocket scientist", "ip address":"127.0.0.1", "name":"frank", "location":"work", "friends":["Joe","Sam","Billy"], "flapjack":{"sally":{"left":"eight","right":"seven"}, "sammy":["One","Ready","Frap"]}}

Figure . **Example Auspice Name Record**. Name records are JSON objects. This is an example from edu.umass.cs.gnsclient.examples.ClientExample in [13].

### 3.2.1 Replica-Controllers

Auspice uses a fixed number of *replica-controllers* for each name, which“dispatch” active replicas for that name. The replica-controllers collect aggregate frequency and location information about requests for their assigned GUID made to its active replicas, and control the placement of active replicas. As shown in Figure 5, active replicas actually store the GUID and its name record, including its network address. A client can determine replica-controllers for a GUID using a hash function, similar to lookups in DMap. After sending a request to a replica-controller, the client receives a list of the current replicas for that GUID and can ask the nearest replica for the GUID’s name record.

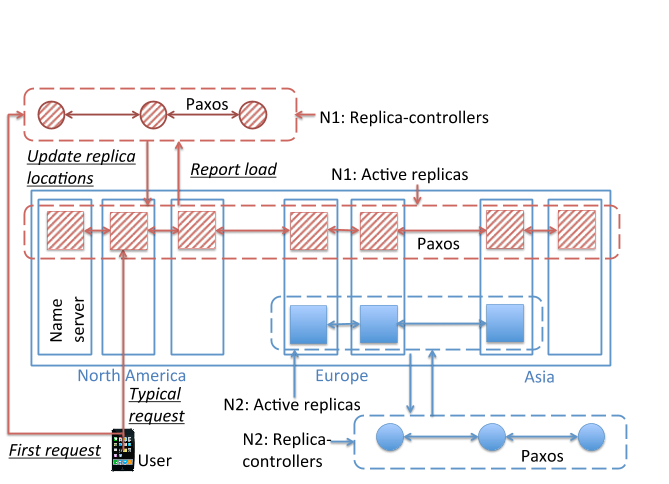


Figure . **Auspice System Overview**. “Geo-distributed name servers in Auspice. Replica-controllers (logically separate from active replicas) decide placement of active replicas and active replicas handle requests from end-users. N1 is a globally popular name and is replicated globally; name N2 is popular in select regions and is replicated in those regions” Figure and Caption from [11, p. 5].

Replica-controllers are contacted “infrequently” by a client or Local Name Server (1) when the Local Name Server (LNS) receives a request for a name it has not seen before, and (2) when the active replica the LNS had previously queried is no longer an active replica for a specific name. The assumption that this is infrequent is discussed in Chapter 4.

### 3.2.2. Demand Aware Placement

Auspice places replicas with the key, value pairs considering both recent demand and update frequency. To keep the “demand aware” placement recent, the placement of replicas is evaluated and adjusted after a pre-determined time period called an epoch. Each epoch, Auspice tunes the number and location of active replicas for each GUID. The system considers the update and lookup rate for that GUID, the geo-distribution of requests for that GUID, and the overall load on the system (across all GUIDs).

Equation . **Demand-Aware Replica Placement.** C is the total capacity of all servers running Auspice. μ is a parameter less than one representing the target utilization. M is a fault tolerance parameter indicating the minimum number of replicas for a GUID. This equation is a reformatted version of Equation 1 in [11, 12].

### 3.2.3 Paxos

Auspice allows a GUID to recursively map to a list of other GUIDs. Appending and truncating the list of GUIDs must be serialized. There are two Paxos instances for each GUID, one for the Replica-controllers and one for the Active Replicas. Total write ordering is guaranteed across updates for active replicas. Stoppable Paxos is used so that when the active replicas are updated, at most once each epoch the Paxos instances can be paused until the switch has been made. The authors of Auspice have designed and implemented their own version of Paxos for Auspice, which they call GigaPaxos [14].

## 3.3 GMap

GMap is an updated version of DMap that also considers geographic location when distributing the GUID-NA pairs [15]. GNRS Servers are also represented by GUIDs, which are called Server Identifiers (SIDs). GMap distributes *K* replicas for each GUID to NA mapping among global replicas, regional replicas and local replicas. Regional replicas are in the same country as the GUID’s NA and local replicas are in the same city or metropolitan area as the GUID’s NA. Assuming the geolocation of the NA and the replicas are known, there is no extra information that must be stored in order to place the GUID-NA mappings at replicas. However, it is unclear what GMap does when a GUID is multi-homed. GMap’s authors argue that updating per GUID as in Auspice leads to scalability problems.

GMap does not have the same extensibility design goals as Auspice, but does mention that “Application-specific policies on the GUID-to-NA mapping are supported by extensible fields of key, value pairs” [15, p. 6]. Unlike Auspice, GMap delegates the computationally expensive task of searching for GUIDs by attributes to the Name Certification Service, although recursively defined groups of GUIDs are still handled by the GNRS.

In Auspice, a server needs to ask a replica-controller which replica is responsible for any given GUID. In GMap, instead of needing to ask a central resource, each sever can compute the SID responsible for a specific GUID. The basic idea is similar to DMap, though the function used to determine the server differs. The authoritative SID for a GUID is defined to be the SID that, when XORed with the GUID, yields the smallest value.

### 3.3.1 Asking the Geographically Closest Server

It makes sense to ask a nearby server where a GUID is, if any nearby servers know. Since there is no way to know a priori whether GUID X is in the current region – we are looking up its location after all – the requesting server assumes it is and asks a server that would be X’s local replica. Computing which server to request the mapping for a GUID becomes complicated when trying to leverage geographic location. When a router receives a lookup request for X with GUID *GX*, it first checks its cache for *GX*. On a cache miss, the router will compute all of the replicas for GX, assuming that X is in the same local area as itself. If X is not in the same local area as the router, then the router will not receive a response from any of the *K3* servers in its city whose SIDs XORed with GX have the lowest values. The server will then send requests to the *K2* servers in its region whose SIDs XORed with GX have the lowest values, which will only respond if X is in the same region as the requesting server. Finally, if X is not in the same region as the requesting server, the server will ask the global replica(s) for X, which are not constrained by any geographic boundary and have the lowest value when XORed with GX. This situation is illustrated in Figure 4.



Figure . **GMap Lookup Requests**. Router Y looks up the NA of GUID GX. It first contacts two servers that do not have any information for GUID GX, since it incorrectly assumes X is in the same city, then region, as Y.

Since the global replica(s) are not dependent on X’s actual location, the computed global replica will always be the actual global replica for GX, and can be considered an “authoritative” server for GX. When X moves enough to change its region, the *K1* authoritative server(s) will still be the nearest XOR to GX, and will just have to be updated, but the *K3* local and *K2* regional servers will need to discard X’s information and the new local and regional replicas will have to be determined and updated. If global, regional, and local replicas are distributed equally, 2/3 of the servers change when X changes regions. Whenever X moves, the cached copies of GX-NAX become stale.

### 3.3.2 GMap Uses Caching to Compensate for Popular GUIDs

In GMap, caching is used to spread out the workload associated with looking up “hotspot GUIDs” [15]. The contents of a GMap cache entry are shown in Figure 7. **GMap Cache Entry**

|  |  |
| --- | --- |
| GUID 🡪 NA |  |
| Remaining TTL |  |
| *Pt* | Go-through probability for time period *t* |
| *Ut* | Number of NA updates during *t* |
| *Ht* | Number of hits for GUID during *t* |
| *Ct* | Perceived update counter *Ct = Ut* / *Ht* |
| *Ct-1* | Perceived update counter for last time period |

Figure . **GMap Cache Entry**

Equation . **GMap Cache Go-through probability**. The go-through probability dictates the frequency with which a cache hit is allowed to “go-through” to the next hop. Allowing some traffic through passively updates the cache entry, keeping it current.

The caching in a GMap server takes into account the local demand for each GUID in the server’s cache. It does not collect aggregate information and statistics about every GUID, and since hits and updates are only recorded and used locally, geographic information is not stored. A server is storing a limited amount of load information for each GUID: keeping two counters, *Ut* and *Ht*, and two static values, *Pt* and *Ct-1*.

## 3.4 Summary

Separate from a Name Certification Service, the main task of a GNRS is to retrieve Network Addresses (NAs) associated with globally unique identifiers (GUIDs).

The proposed Global Name Resolution Services make tradeoffs. DMap is the simplest and most straightforward. In DMap and GMap, there are a fixed number of replicas for each GUID, regardless if it someone’s personal device or a server with much more traffic. GMap uses demand-aware caching to increase the number of copies of mappings for popular “hotspot” GUIDs without keeping global geo-specific statistics on each GUID.

Since Auspice allows recursive GUIDs, it uses a version of Paxos to guarantee consistency. It is unclear how GMap keeps a consistent view of recursively defined GUIDs. DMap makes a best-effort attempt to return the correct list of network addresses for a GUID and does not concern itself with consistency. If a network address does not actually have the desired GUID, the client can request the mapping again. In the current implementation, a random server containing the desired mapping is queried; with more than two replicas, it is likely that a different server will be queried if the mapping is requested again.

There are some cases where it may make sense for a GUID’s address listing to differ based on the location of the mapping. Although Auspice may be best set up to accurately predict which network address may be the most efficient in each location because of its vast collection of statistics, it forces consistency and guarantees that the name record obtained from different active replicas is the same, preventing customization of name records on different active replicas. DMap allows the GUID to set the priorities for different network addresses in the insert message, which is forwarded to the rest of the *K* servers, so the name records should be identical, but it is not guaranteed. Since DMap already uses a separate hash function for storing the GUID’s record locally within its AS, it would be feasible to have that record be distinct from the *K* other copies. GMap is perhaps in the best position to customize name records based on their location, since it already runs different hash functions for each locality (local, regional and global).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Auspice** | **GMap** | **DMap** |
| Location relative to network | Overlaid on top of network | In-network | In-network |
| Algorithm Type | Replicated State Machine | Distributed Hash Table | Distributed Hash Table |
| Record Content | GUID to arbitrarily many key value pairs | GUID to NA(s); GUIDs may be recursively defined | GUID to up to 5 NAs, each with an expiration time and prioritization weight |
| Replica Placement | Geo-located based on requests | Geo-located based on GUID’s physical location | Not Geo-located. One Replica in the GUID’s AS |
| Number of Replicas | Adjusts # of replicas for each GUID based on recent demand and update frequency | Fixed # of replicas for each GUID (each GUID has *K* replicas) | Fixed # of replicas for each GUID (each GUID has *K*+1replicas) |
| Caching | No caching, tries to achieve load balancing by adjusting number of replicas | Caches GUID🡪Network Address Mappings to increase availability of mappings for “hotspot GUIDs” | Future work |

Table . Comparison Table of Auspice, GMap and DMap

# Hypotheses, Key Resources and Possible Vulnerabilities

This chapter identifies key resources in each of the GNRS system designs and forms a preliminary analysis from reading the DMap publication [9], Auspice and GMap technical reports [12, 15], and discussing the GNRSs’ design.

## 4.1 Possible Attacks in Auspice

The key resources in Auspice appear to be the active replicas and the replica-controllers. They need to store a lot of information and be available for clients to look up the location of GUIDs.

Auspice assumes limited contact with replica-controllers, but does not enforce limited contact with them. “In practice, we expect replica-controllers to be contacted infrequently as the set of active replicas can be cached and reused” [11, p. 6]. We suspect that if an entire botnet was directed to request new name(s) at the same time it would overwhelm the targeted replica-controllers.

Although active replicas are intended to handle frequent and voluminous requests,

1. Large-Scale Simultaneous Mobility to overload active replicas
   1. Each member of Botnet A requests a NA for a GUID in Botnet B
   2. Each member of Botnet B requests a NA for a GUID in Botnet A
   3. Each member of Botnets A and B switch network addresses

Note that since LNSs have a list of all active replicas for a name, they can send these requests to all active replicas for a name.

This attack targets the **computation power** of the active replicas.

1. Trick Geo-Distribution

**3.1 Interfere With Demand Estimate**

Goal: Make replica-controllers think there is more demand for a name or subset of names in a region far away from where peak legitimate demand is to worsen or deny service to legitimate demand.

Attack: have botnet in faraway region send many requests for target name/names

**3.2 Interfere with Update Frequency Estimate**

Another way to exploit the distribution, which takes the update rate into account when determining the number of active replicas, is to update a name very frequently such that the replica-controllers limit the number of active replicas. This assumes that an attacker can spoof the target GUID, which is supposed to be self-certifying.

Tricking the geo-distribution is an attack that causes more **bandwidth** to be used per request, increasing latency / response times.

1. Take advantage of the auspice system enforcing total write ordering for updating GUIDs (especially lists). This attack requires more knowledge of Paxos and the “super columns” mentioned in [12]’s Section 3.2.5 Extensibility.

This attack targets system **storage**.

5. Create not useful key, value pairs and fill up the database.

## 4.2 Possible Attacks in GMap

1. Filling up the cache with nonsense and preventing the cache from caching legitimate hotspot GUIDs. This is especially easy since the cache is Least Recently Used and not Least Frequently Used.

This attack targets system **storage**. However, since the caching is intended to decrease bandwidth usage, with a successful attack of this nature, one would expect **bandwidth** usage to increase.

1. Forging an ACK with an incorrect NA for a target popular GUID.

This attack is also a blow to **storage and bandwidth**. It doesn’t overflow the storage, but causes the system to store corrupt data. Since the client does not receive a NA that actually has the GUID, the client will likely submit the request again, using more bandwidth.

1. Flood all GUIDs on a specific router using Rainbow Table

GMap uses a well-known algorithm to assign GUIDs to routers. A Rainbow Table can be created by mapping all GUIDs to their Routers and storing the results. Then, an attacker can send messages to all GUIDs on a target router, flooding it with traffic. This is an attack on the bandwidth to the targeted router, and if requests do get through, the computation power of the router may be strained. Either way, this attack should deny the resolution for all GUIDs on the router.

Auspice mitigates a similar type of attack by using epochs and necessitating system involvement that could easily include gatekeeping. In Auspice, a host (or a server on its behalf) has to ask replica-controllers where a GUID-NA mapping is for new GUIDs. This makes filtering possible at the server and replica-controller levels; they could notice if a specific host is asking for the active replicas for an unusual number of GUIDs. With both rate-limiting/filtering and a shuffle every epoch, Auspice makes building up an accurate Rainbow Table for which active replicas store which GUIDs rather challenging.

1. Attack local GNRS routers via lookup flooding.

The way GMap looks up GUIDs makes it vulnerable to a couple of attacks. When a local GNRS-enabled router receives a lookup request, it checks its cache for that GUID. A local GNRS router must compute all replicas for any GUID GX for which it receives a request and has a cache miss. An attacker can require a local GNRS router to do a lot of computation if they send many lookup requests for unpopular (and therefore not in the cache) GUIDs at once.

Sending many requests from a host Y, requesting GUIDs an attacker knows to be located far away from Y (or nowhere at all), should require RY to send three times as much traffic as the attacker, negatively impacting its available bandwidth for legitimate requests. This attack could also affect RY’s storage, since it must maintain state for the servers it is waiting on. Combining both of these attacks by requesting geographically far GUIDs that are not cached, could significantly bog down a local GNRS router. There are two mitigating factors that work in GMap’s favor against this attack. First, it is easiest to attack nearest the attacker, since host Y attacks its nearest GNRS-enabled router. Unfortunately, this first defense can be easily overcome by the attacker employing a botnet away from themselves and near their target. Secondly, this attack only targets a single router, which is arguably not a core part of GMap. Routers fail all of the time, and traffic gets redirected to other routers. I suspect that, while this attack could be very successful on a specific router, it will only deny service when other routers are not available.

## 4.3 Possible Attacks in DMap

Unsurprisingly, most of the attack opportunities in DMap are related to the edge cases it addresses, discussed in 3.1.3 Edge Cases.

### 4.3.1 Malicious AS

A malicious AS that announces and is therefore assigned a specific range of the name space, can simply not cooperate by storing DMap mappings of GUIDs to NAs. This would require an attacker to control a significant portion of the name space in order to receive mappings. One way around that would be to create a malicious AS that claims to host the addresses in the IP hole and becomes black hole for GUIDs that hash to the IP hole(s).

A malicious AS could also forge GUID migration messages to ASs holding legitimate mappings telling them that they are “deputy” ASs and to drop their mapping. In order for migration messages to work when the deputy AS is actually a deputy, upon receipt of a migration message, the deputy AS would have to either automatically drop the mapping, or check if it is rightful owner of the mapping before dropping it.

This computation-intensive checking could be mitigated by including a Boolean with the mappings on each server, indicating whether that server/AS is acting as a deputy or not. Keeping an additional true/false value would require minimal additional storage but allow the computation to only be performed once, when a GUID is inserted.

* 1. It is unclear how and when servers missing mappings follow the IP hole procedures to attempt to retrieve the mapping from a deputy server. If they do it whenever they are queried for a mapping they don’t have 🡪 can flood with requests for mappings they don’t have.

### 4.3.2 Rainbow Table

Flood all GUIDs on a specific router using Rainbow Table

DMap uses a well-known algorithm to assign GUIDs to routers. A Rainbow Table can be created by mapping all GUIDs to the servers their mapping is stored on and keeping a list of which GUIDs are stored on which servers. Then, an attacker can send messages to all GUIDs on a target router, flooding it with traffic. This is an attack on the bandwidth to the targeted router, and if requests do get through, the computation power of the router may be strained. Either way, this attack should deny the resolution for all GUIDs on the router.

Auspice mitigates a similar type of attack by using epochs and necessitating system involvement that could easily include gatekeeping. In Auspice, a host (or a server on its behalf) has to ask replica-controllers where a GUID-NA mapping is for new GUIDs. This makes filtering possible at the server and replica-controller levels; they could notice if a specific host is asking for the active replicas for an unusual number of GUIDs. With both rate-limiting/filtering and a shuffle every epoch, Auspice makes building up an accurate Rainbow Table for which active replicas store which GUIDs rather challenging.

1. DMap paper says each GUID can only be associated (multi-homed) with up to 5 NAs, but does code enforce this? (I don’t think so).

## 4.4 Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Basic Resource Targeted | Auspice | GMap | DMap |
| Computation | Large-Scale Simultaneous Mobility overloads Active Replicas, which use Paxos | Large-Scale Simultaneous Mobility – have to remove mapping from old local and regional servers and add to new local and regional servers | Large-Scale Simultaneous Mobility would cause inconsistent mappings for same GUIDs |
| Large-Scale Name Lookup overloads Replica-controllers | *Individual routers hash GUIDs to know which replicas to ask* | *Individual routers hash GUIDs to know which replicas to ask* |
|  | Send Insert Requests (with many, many entries) for “fake” GUIDs | | |
| Bandwidth | Interfere with Demand Estimate | Request far-away GUIDs to waste local/regional requests |  |
| Interfere with Update Frequency Estimate |  |  |
| *More central control, easier to filter. Attacker alone can’t determine which GUIDs are where* | Flood all GUIDs on a specific router using Rainbow Table  Is this worse than just asking that router for GUIDs it doesn’t actually have? | |
| Storage | Paxos/Total Write ordering: effect on active replicas **and** replica-controllers |  |  |
|  | Cache Overflow |  |
|  | Forge ACK with incorrect NA for target GUID (possibly different than impersonating GUID depending on structure of ACKs) |  |
| Impersonate GUID, store wrong NA | | |
| Insert GUID multihomed at 1000 addresses | | |

Table Italic = not a significant vulnerability

# Experimental Set Up

As mentioned above, my goal was to discern the features of MobilityFirst’s architecture that allow continued communication during node mobility through examining denial of service attacks in two proposed global name resolution services. After reading the technical reports and predicting where DoS attacks may be possible, DMap and Auspice were run in a testbed and observed under a traffic model for legitimate traffic. Models for network traffic and device movement are necessary to observe the inner workings of the GNRS. A NA, the endpoint of my experiment, maps to our concept of an AS, so I am not concerned with intra-AS topology.

Then, malicious behavior representing attacks was added. Possible preventative measures and their tradeoffs are discussed in Chapter 6.

## 5.1 ORBIT Testbed

The Open-Access Research Testbed (ORBIT) was designed for Next-Generation Wireless Networks and runs on the Internet [16]. ORBIT’s main grid has 200 nodes. It is a realistic way to test an architecture designed to scale as large as the Internet. Since it uses the same links that support the Internet, link behavior is realistic but not customizable. So that tests do not interfere with each other, they had to be scheduled, but ORBIT prevented our attacks from “getting loose” on the currently running Internet. Simulations were used to evaluate Auspice, DMap, and GMap, but each simulation focused on different parts of the architecture that the system designers wanted to evaluate, leaving the simulations incomplete in different ways. Since we wanted to compare the GNRSs, running them all in the same testbed acts as a control in the experiments. ORBIT uses the ORBIT Management Framework [17] to execute experiments.

## 5.2 Availability of GNRS Code

The University of Massachusetts’s Auspice code is available online and information on how to access it is available in [18]. Auspice is currently being ported to ORBIT. There is an Experiment Running Reference [19] available describing how to access the GMap code in ORBIT.

GMap is not currently available on ORBIT, it is only currently implemented in a simulation. DMap is implemented in ORBIT, but not fully. Updates are not included on the client side.

DMap – 1000 \* t nano second delay between client messages – about .01 t ms.

Their solution for the IP hole problem mentioned in section II.B of [9] does not seem to be implemented in the DMap code obtained from [10].

## Abstraction to the Autonomous System Level

In my experimental setup, each ORBIT node is an abstraction for an Autonomous System. [something like the diagram on Karen’s whiteboard]

We don’t care about intra-AS traffic because it is independent of the GNRS. Once traffic gets to an AS, the AS can route it to the specific device in whichever way it sees fit.

# DMap Server Attack

*Suggest and evaluate mitigation strategies, if possible: In this step, I will suggest and analyze possible mitigation strategies, considering the possibility that they may create new vulnerabilities. Promising strategies may be measured in the testbed.*

*Compare GNRS vulnerabilities to current DoS vulnerabilities, focusing in this step on ways that MobilityFirst and the GNRS designs have reduced vulnerability to DoS and DDoS attacks.*

## 6.1 Recursive Inserts and Updates



**Figure 8. DMap Normal Operation with K=2.** Setting K=2 means that GX’s network address is stored on servers corresponding to two independent hash functions. So that the first DMap server knows it must pass the Insert request along to other DMap servers, the request is marked as “recursive” by GX, the device sending its Insert message to 45.0.0.5. Server 45.0.0.5 runs both hash functions, removes the recursive option from the Insert message and forwards it to the servers corresponding to the hashes of GX.

As shown in **Figure 8**, Insert messages coming from end-users running the client version of DMap protocol set a recursive option to True so that the Insert is replicated *K* times. Then, the local DMap server runs the *K* hash functions, stores the GUID’s network address if necessary (because it is one of the *K* servers, for local replication or due to caching), and then forwards a non-recursive version of the request to the remaining servers determined by the hash functions. After sending the remote requests to other servers, the original server, 45.0.0.5 in the figure, keeps info for GX’s information in an “awaiting responses” map. Note that the other servers (128.5.5.5 and 192.5.6.7) do not forward the request to each other since they have a non-recursive version. After 128.5.5.5 and 192.5.6.7 run the hash functions and insert GX’s network address in their storage, they send Insert Response messages back to 45.0.0.5, which in turn sends a message back to GX that its insert was successful.

It is good practice for the default assumption in an Insert message to be recursive, unless the recursive option is specifically set.

There are other possible ways to implement this behavior, for example, having different types (with corresponding Java classes) of insert messages.

## 6.2 Attack Setup (in example and “do-ability” in real world)

Changes made to server code:

* Generate insert requests (usually come from client)
  + Disregard responses, unlike client, don’t waste this server’s memory waiting for responses
  + “attack rate”
  + Can use varying attack rates and not repeat same request and keep track of response rate to measure effectiveness of attack. (Assuming attacker doesn’t have tcpdump of target’s packets or know if other clients are getting their packets through)
* Send recursive requests (usually marks all non-recursive)
* Send to all other BGP/DMap servers

“hashing, rehashing and prefix matching processes are done locally by the border gateway” DMap pg 4

* If attacker can attack a device to a network, can flood local BGP router with requests, fill up queue (because UDP packets just dropped if they never reach the first DMap server)

Does insert fail if any one of the K routers fails? Then they’re not accomplishing their goal of using K replicas to be robust.

Some dropped by queue? Also likely some insert success responses from remote servers dropped by queue.

The possibility and effectiveness of denial of service attacks on GNRS systems was demonstrated in a testbed.

DMap using 128 ASs for all of IPV4 using 1000 random GUIDs, 29 mapped to only 2 ASs. While each server being responsible for 33,554,432 addresses is unlikely, the same machine may announce multiple IPs. If a GUID maps to different addresses on the same machine, DMap still counts the addresses as separate replicas, assuming the GUID-NA mapping is stored on *K* devices when it is actually stored on *K-1* devices. This is one area where GMap improves on DMap. Collisions are significantly less likely, because it ensures that at least 3 (one global, one local and one regional) replicas are on different machines. The only possible machine-collisions are among the replicas at each regional level.

One advantage of DMap is that when a GUID moves within an AS, its mapping (GX, AS 1) does not change, and therefore saves resources. Most movement is within one AS. Unfortunately, the DMap code does not have client-side updates implemented at this time.

DMap Inserts should be signed to prevent an attacker from changing the GUID🡪NA mapping timeout from being really small.

Rainbow Table 128 servers to place mappings on,

Mapped 10,000 GUIDs, 224 of which mapped to less than 3 servers. (about 2%)

100,000

In future iterations of the code this can be addressed by having an additional “backup” hash function, following the IP hole protocol, or simply increasing *K* such that it is irrelevant if a GUID’s mapping is only stored on *K*-1 servers.

## 6.x Results

## 6.X Tradeoffs

DMap servers (co-located with BFG routers) could check that all recursive requests only come from devices in the AS of their BGP router. This implies that the DMap server has to know who all of the first level clients are, and not allow new ones. This would restrict devices moving between ASes, and not meet MobilityFirst’s mobility goal. One could look into requiring a lower level handshake first to establish presence in an AS before being inserted into the GNRS.

Another option is to only allow a fixed number of recursive inserts from each network address in some time period. This would limit how often devices could move. Additionally, it would require additional per-Network Address state on each server, and is likely to be ineffective against the attack given that the attacker was sending inserts at rates three orders of magnitude slower than the client.

# Effects of Transport Layer Protocol on GNRS Design

Problem

Introduce TCP, UDP

Discuss TCP, UDP in context of different GNRS designs

Then, Tie back to attacks

Conclusions / how this affects GNRS designs

# Future Work

The GNRS systems could be evaluated under a more geographically realistic network topology, using GeoTopo [24] to model the network topology. Additionally, one could compare how DMap and Auspice function across multiple topologies, instead of imposing a specific topology on them. Network architecture and network topology can influence each other’s evolution. Similarly, traffic loads and patterns may be different (in the immediate sense and as the Future Internet Architecture changes over time) based on the underlying network architecture.

The attack mentioned in Section 4.3.1 where migration messages are forged could also be tested. It was not tested at this time because Migration Messages are not currently implemented.

## Topology

In the grid on ORBIT, the 200 nodes are in a small room and all within radio range of each other, creating a mesh network, where each node is directly connected to each other node. It is possible to create a multi-hop topology in such an environment by injecting noise as in [20], but this is not functioning on ORBIT at the time of this publication.

Additionally, instead of a constant request rate for sequential GUIDs, a Traffic Model using a Zipf distribution to model request rates for GUIDs would be more realistic.

## Mobility Model

My mobility model need only take into account mobility that would change a GUID’s NA. Since a NA is comparable to an AS in today’s Internet, I use a model derived from observing when users connected and disconnected to different ASs. I am using a Hidden Markov model derived in "Measurement and Modeling Study of User Transitioning Among Networks," which gives a transition probability matrix of a user’s next state based on their current state [21, 22]. They include six states (describe). Their model models mobility among networks in discrete-time chunks of 15 minutes. In order to prevent “batch” movement in the model used for my experiments, whenever UMass’s model dictates that a user changes state during a 15-minute period, when during the 15-minute period is randomly determined to the (millisecond).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Probability of Transitioning to Each State Given Current State | | | | | | |
| **Current State** | 0 new, 0 total | 0 new, 1 total | 1 new, 1 total | 0 new, >1 total | 1 new, >1 total | >1 new, >1 total |
| 0 new, 0 total | 0.88577709 | 9.78E-06 | 0.11012325 | 0 | 0 | 0.00408987 |
| 0 new, 1 total | 0.20181485 | 0.7266925 | 0.02349103 | 0 | 0.04486134 | 0.00314029 |
| 1 new, 1 total | 0.6474509 | 0.2834485 | 0.04665651 | 0 | 0.01936993 | 0.00307416 |
| 0 new, >1 total | 0.0749543 | 0.43327239 | 0.01005484 | 0.42413163 | 0.05393053 | 0.00365631 |
| 1 new, >1 total | 0.17307092 | 0.59418932 | 0.01968135 | 0.15901281 | 0.052796 | 0.00124961 |
| >1 new, >1 total | 0.4380704 | 0.40026076 | 0.03976532 | 0.07887875 | 0.03976532 | 0.00325945 |

Table Transition Probability Matrix [21, 22]

[23] quantitatively compares different network architectures that support mobility and discusses mobility of devices across ASes.

# Comparison to Today’s Internet

In the current Internet, devices that are more mobile, such as personal computers and laptops, tend to have less traffic than static devices. Total mobile data traffic (including smartphones, tablets and mobile PCs) is 5.3 EB per month, while total fixed data traffic is 60 EB per month [25].

However, there are certain instances where an extremely popular, high traffic source is moving. For example consider the Super Bowl, where many people are tuned in to a streamed source moving between many different cameras. Here, there could be a GUID representing the main Super Bowl broadcast that would relocate rapidly.

IP forms a flexible base for today’s Internet, with more features and constraints built on top of it. It is a “best effort” system that is designed to be simple and approximate. As complexity rises, so do opportunities for security breaches and DoS attacks. With a best effort system, one can assume something(s) go wrong, and that’s tolerable in the system. Instead, if one is trying to keep a system perfectly secure and work in every eventuality and edge case, it becomes too cumbersome to be effective. Security-wise, it can turn into an attack/patch war. Similarly, as a basic service of Mobility First, the GNRS should have as few constraints as possible. Key principles of computer system design are simplicity and modularity. Adding additional protocols on top of the GNRS would allow for more advanced features and could incorporate more constraints on how they are used, while leaving the GNRS as a simple primitive.

I think Auspice struggles more in this respect, trying to keep per-GUID statistics and do per-GUID load balancing, which can be computationally expensive and use a lot of storage. I suspect this will require a lot of fine-tuning on the length of epochs. Also, Paxos may be too close to trying to be “perfect” with an all-or-nothing update approach.

A major difference between today’s Internet and MobilityFirst’s network is the target endpoint, an IP address and a Network Address, respectively. “Bringing down” (ie. clogging) a Network Address affects more devices than an IP address, but is harder to do because the link capacity would be greater.

[3] discusses denial of service attacks in the current Internet.

# References

|  |  |
| --- | --- |
| [1] | A. Compagno, M. Conti, P. Gasti and G. Tsudik, "Poseidon: Mitigating Interest Flooding DDoS Attacks in Named Data Networking," in *Annual IEEE Conference on Local Computer Networks*, Sydney, Australia, 2013. |
| [2] | P. Gasti, G. Tsudik, E. Uzun and L. Zhang, "DoS and DDoS in named data networking," *International Conference on Computer Communications and Networks,* pp. 1-7, 2013. |
| [3] | M. Handley and E. Rescorla, Eds., Internet Denial-of-Service Considerations, IETF, November 2006. |
| [4] | A. Venkataramani, J. Kurose, D. Raychaudhuri, K. Nagaraja, M. Mao and S. Banerjee, "MobilityFirst: A Mobility-Centric and Trustworthy Internet Architecture," *ACM SIGCOMM Computer Communication Review,* 2014. |
| [5] | L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang and B. Zhang, "Named Data Networking," *ACM SIGCOMM Computer Communication Review,* vol. 44, no. 3, pp. 66-73, July 2014. |
| [6] | K. R. Sollins, D. D. Clark, W. Lehr, J. Kurose, A. Venkataramani and S. Bauer, "NeTS: Large: Collaborative Research: Location-independent Networks: Evaluation Strategies and Studies," Submission to the National Science Foundation, Solicitation 13-581, November 19, 2013. |
| [7] | J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *SIGCOMM Computer Communications Review,* vol. 34, no. 2, pp. 39-53, 2004. |
| [8] | T. Peng, C. Leckie and K. Ramamohanarao, "Survey of Network-Based Defense Mechanisms Countering the Dos and DDoS Problems," *ACM Computing Surveys,* vol. 39, April 2007. |
| [9] | T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin and D. Raychaudhuri, "DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet," in *IEEE 32nd International Conference on Distributed Computing Systems*, Washington, D.C., 2012. |
| [10] | wontoniii, "MobilityFirst Wiki," GNRS Installation, 7 July 2015. [Online]. Available: http://mobilityfirst.orbit-lab.org/wiki/Proto/cModules/b0GNRS/c0Installation#no1. [Accessed 24 May 2016]. |
| [11] | Sharma, A, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook and A. Yadav, "A Global Name Service for a Highly Mobile Internet," UMASS Computer Science Technical Report: UM-CS-2013-023, 2013. |
| [12] | A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook and A. Yadav, "A Global Name Service for a Highly Mobile Internetwork," *ACM SIGCOMM,* 2014. |
| [13] | v-arun, "MobilityFirst GNS on Github: Getting Started," 5 July 2016. [Online]. Available: https://github.com/MobilityFirst/GNS/wiki/Getting-Started. [Accessed 8 July 2016]. |
| [14] | A. Venkataramani and D. Westbrook, "MobilityFirst/gigapaxos: GigaPaxos is a group-scalable reconfigurable consensus system that can be used to efficiently and easily manage a very large number of independent lightweight replicated state machines.," GitHub, Inc., 14 July 2016. [Online]. Available: https://github.com/MobilityFirst/gigapaxos. [Accessed 14 July 2016]. |
| [15] | Y. Hu, R. D. Yates and D. Raychaudhuri, "A Hierarchically Aggregated In-Network Global Name Resolution Service for the Mobile Internet Technical Report," WINLAB-TR-442, North Brunswick, NJ, March 2015. |
| [16] | "Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT)," [Online]. Available: http://www.orbit-lab.org/. [Accessed 16 December 2015]. |
| [17] | T. Rakotoarivelo, M. Ott, G. Jourjon and I. Seskar, "OMF: a control and management framework for networking testbeds," *ACM SIGOPS Operating Systems Review,* vol. 43, no. 4, pp. 54-59, Jan. 2010. |
| [18] | A. Venkataramani and D. Westbrook, "MobilityFirst/GNS: A Global Name Service for a Highly Mobile and Secure Internetwork," GitHub, Inc., [Online]. Available: https://gns.name/wiki/index.php?title=MobilityFirstGit. |
| [19] | jw898, "ORBIT: GNRS Management," June 2015. [Online]. Available: http://www.orbit-lab.org/wiki/Other/Summer/2015/cMF5. [Accessed 28 November 2015]. |
| [20] | S. K. Kaul, M. Gruteser and I. Seskar, "Creating Wireless Multi-hop Topologies on Space-Constrained Indoor Testbeds Through Noise Injection," in *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, Barcelona, 2006. |
| [21] | S. Yang, J. Kurose, S. Heimlicher and A. Venkataramani, "Measurement and Modeling Study of User Transitioning Among Networks," in *IEEE INFOCOM 2015 conference*, 2015. |
| [22] | S. Yang, Interviewee, *Personal Correspondance.* [Interview]. 20 May 2016. |
| [23] | Z. Gao, A. Venkataramani, J. Kurose and S. Heimlicher, "Towards a Quantitative Comparison of the Cost-Benefit Trade-offs of Location-Independent Network Architectures," in *SIGCOMM*, Chicago, IL, 2014. |
| [24] | Y. Hu, F. Zhang, K. K. Ramakrishnan and D. Raychaudhuri, "GeoTopo: A PoP-level Topology Generator for Evaluation of Future Internet Architectures," in *Proceedings of the 23rd IEEE International Conference on Network Protocol (ICNP)*, 2015. |
| [25] | "Ericsson Mobility Repot: On the Pulse of the Networked Society," Ericsson, Stockholm, Sweden, November 2015. |
| [26] | K. R. Sollins, "TWC: Small: The Interaction Between Mobility and Denial-of-Service Attacks," Solicitation: National Science Foundation, NSF 15-575, SaTC Small, November 18, 2015. |
| [27] | K. Wang, J. Chen, H. Zhou, Y. Quin and H. Zhang, "Modeling denial-of-service against pending interest table in named data networking," *National Engineering Laboratory for Next Generation Internet Interconnection Devices,* July 2013. |
| [28] | I. Stoica, D. Karger, M. F. Kaashoek and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Protocol," in *SIGCOMM*, New York, NY, 2001. |

1. Based on the Mobile Internet Protocol where the post office represents a home agent. [↑](#footnote-ref-1)
2. In the MobilityFirst paper [4], it is called a Global Name Service (GNS), but extended to GNRS (Global Name Resolution Service) in the Auspice and GMap papers. They are the same thing and this document uses GNRS. [↑](#footnote-ref-2)
3. For background information on distributed hash tables, see Chord [28]. [↑](#footnote-ref-3)