GNRS ASSISTED INTER-DOMAIN SERVICES IN MOBILITYFIRST FUTURE INTERNET ARCHITECTURE

BY SUJA SRINIVASAN

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Prof. Dipankar Raychaudhuri

and approved by

_____

_____

_____

New Brunswick, New Jersey

October, 2016

ABSTRACT OF THE THESIS

GNRS Assisted Inter-Domain Services in MobilityFirst Future Internet Architecture

by Suja Srinivasan

Thesis Director: Prof. Dipankar Raychaudhuri

This thesis involves the design and evaluation of an extension to the Global Name Resolution Service (GNRS) of the MobilityFirst Future Internet Architecture (MF-FIA), to support inter-domain services. The GNRS is a scalable distributed system that facilitates name-location separation in the MobilityFirst network. Currently the GNRS stores mappings from a globally unique identifier (GUID) to network addresses alone. The proposed extension makes the GNRS more generic by allowing it to store GUID to mappings of different types. This enhanced framework enables the support for different inter-domain functions.

The extension work was tested and evaluated through both simulations and ORBIT testbed emulations. Two applications, namely multicast and late-binding in Mobility-First, were explored as services enabled by the enhanced GNRS framework. The multicast implementation was integrated into the prototype MobilityFirst software which is based on the open source Click routing platform and evaluated on ORBIT testbed. Late-binding schemes were studied and evaluated through a series of real time trace simulations. Both experiments proved the utility of the extended GNRS framework in realizing advanced inter-domain services in the MobilityFirst network.

Acknowledgements

I would like to express my sincere gratitude to my adviser Dr. Dipankar Raychaudhuri for his invaluable guidance and support through out. Working under him has been a great learning experience. I would also like to thank the rest of my thesis committee: Professor Marco Gruteser and Professor Maryam Dehnavi.

I am thankful to Shreyasee Mukherjee, Francesco Bronzino and Jiachen Chen for all the discussions that helped structure my work. I would also like thank my friends at WINLAB for their wonderful company. Last but not the least, I am thankful to my family for their constant support and encouragement.

Table of Contents

Abstract ii

Acknowledgements iii

List of Figures vi

1. Introduction 1

2. Overview of the MobilityFirst Architecture 3

2.1. Intra-Domain Routing: GSTAR 5

2.2. Inter-domain Routing: EIR 5

3. Extending the GNRS 8

3.1. GNRS Overview 8

3.2. Motivation for extending the GNRS 9

3.3. Design of Extended GNRS 10

4. Proof of Concept 13

4.1. GNRS assisted Multicast 13

4.1.1. Architecture Overview 13

4.1.2. GNRS API support 15

4.2. GNRS assisted Late Binding 17

5. Evaluations 19

5.1. GNRS Extension 19

5.2. Multicast 22

5.3. GNRS assisted Late-Binding 25

6. Conclusion and Future Work 33

6.1. Future Work 33

References 34

Appendices 36

A. GNRS - Updated Protocol Specification 37

List of Figures

2.1. MobilityFirst architecture overview 3

2.2. MobilityFirst stack with GUID as the narrow waist [1] 4

2.3. Representation of the router level topology of ASes in terms of aNodes and vLinks 6

3.1. DMap implementation with K=3 9

3.2. GNRS design modification 11

3.3. GNRS design modification 11

3.4. GNRS Database design modification 12

4.1. Multicast tree structure stored in the GNRS 14

4.2. Multicast Architecture 15

4.3. Mechanism of GUID to NA Binding 17

5.1. Plot of the average insert time vs log of number of inserts 20

5.2. Plot of the average lookup time vs log of number of inserts 20

5.3. Plot of average insert time vs log of number of inserts per second 21

5.4. Plot of average lookup time vs log of number of lookup requests per second 21

5.5. Components of NOMA - GNRS and click routers. Updated modules shown in blue. 22

5.6. 6 node test topology. 23

5.7. Cost comparison for unicast and multicast schemes 24

5.8. Comparison of the control vs the actual data transferred in the 6 router - 15 hosts topology for different sizes of data messages. 24

5.9. Structure of the MAN network 26

5.10. Top level clusters 26

5.11. Plot of transfer time for 15 trials for the different schemes 27

5.12. Hop count time for 15 trials 28
5.13. Plot of average transfer time for different cab speeds 29
5.14. Average hop count at different speeds 30
5.15. Plot of transfer time for 15 trials for the different schemes 30
5.16. Plot of transfer time for 15 trials for the different schemes 31
5.17. Hop count time for 15 trials 31
A.1. Network Address 37
A.2. Common request header 38
A.3. Options 39
A.4. Lookup Request 40
A.5. Insert/Update Request 40
A.6. Common Response Header 41
A.7. Lookup Response 42

Chapter 1

Introduction

Currently, the Internet is witnessing a tremendous increase in the number of mobile end hosts. These include cellular phones, Internet-of-Things (IoT) devices, connected vehicles and many more. As these end hosts move, they encounter frequent transitions as well as disconnections. This makes content delivery while maintaining a seamless experience for the user, difficult. The current TCP/IP internet is based on the conflation of identity and location which is a poor design particularly for end hosts with high mobility. As the device moves, connections break due to a change in the in the network address, requiring application-layer workarounds to provide the mobility support.

Recent efforts have been directed towards doing away with this location-identity conflation. Several future internet architectures [4], [3], [5] have been suggested that separate identity of an end host from its location enabling better support for mobile devices. MobilityFirst [2] is one such solution. It is a mobility-centric architecture that supports large-scale, efficient and robust network services with mobility as the norm. In MobilityFirst, every end host is assigned a global unique identifier (GUID), which is decoupled from its network address or location. A dynamic global name resolution service (GNRS), such as those in [10], [11], [12], [13] is used to track and resolve the mapping between a GUID and its network addresses. MobilityFirst aims to support smooth mobile content delivery by exploiting real-time GNRS updates and queries.

At present, MobilityFirst uses the DMap [10] implementation of the GNRS as the name resolution service. The GNRS stores a mapping between the GUID of a network entity to its network address. This allows for efficient name-address separation. GUID look up requests can be made to the GNRS to get updated information regarding the location of an entity at any given time. The DMap implementation makes the GNRS

a distributed and scalable system.

In this work we look at better utilizing the GNRS infrastructure to offer improved services in the MobilityFirst network. We explore possible inter-domain services that MobilityFirst can support by utilizing the storage provided by the GNRS. For example, consider an island of connected cars moving along a highway. It might make sense to assign a GUID to the group of cars and then send a common message to the group GUID. This can also be to reach a member of the group that does not have internet connectivity, through another member who does. Here, the GNRS would store a list of GUIDs corresponding to all the cars, against a common group GUID. Another example could be storing some of the previous known locations of a user. This might help predict the next possible location of a user or the current location of a disconnected user.

To realize the added benefits of the GNRS, we extend the mapping stored in the GNRS from a simple GUID to location mapping to a more generic framework that stores mapping objects of different types. This framework provides support for mapping types to be defined and introduced as and when new services are developed for the network. As a proof of concept we study two applications - multicast and late-binding. Through results from ORBIT testbed evaluations and simulations we show that GNRS assistance helps to improve inter-domain routing.

Chapter 2 provides a detailed description of the MobilityFirst architecture, its components and routing protocols. We discuss the GNRS extension framework design and implementation in Chapter 3. Chapter 4 describes GNRS assisted multicast and late-binding solutions as proof of the utility of the new framework. Chapter 5 discusses the experiments and results from both actual implementation as well as simulations. Lastly, conclusion and future work are presented in Chapter 6.

Chapter 2

Overview of the MobilityFirst Architecture

MobilityFirst is a clean slate architecture for the future internet. In the current design of the internet, there is a conflation between the identity of an end host and its location. Mobility first eliminates this conflation. As the internet evolves to shift from the fixed host-server model to a one where mobile platforms are the norm, MobilityFirst addresses the challenges of mobility and wireless communication by adding intelligence to the core network, rather than handling them as a last hop problem. MobilityFirst allows applications to communicate with abstract entities including end devices, context and content. It provides inherent support for various services such as multicast, multi-homing, anycast, multi-path and IoT. Figure 2.1 shows the components and routing protocols of the MobilityFirst network.

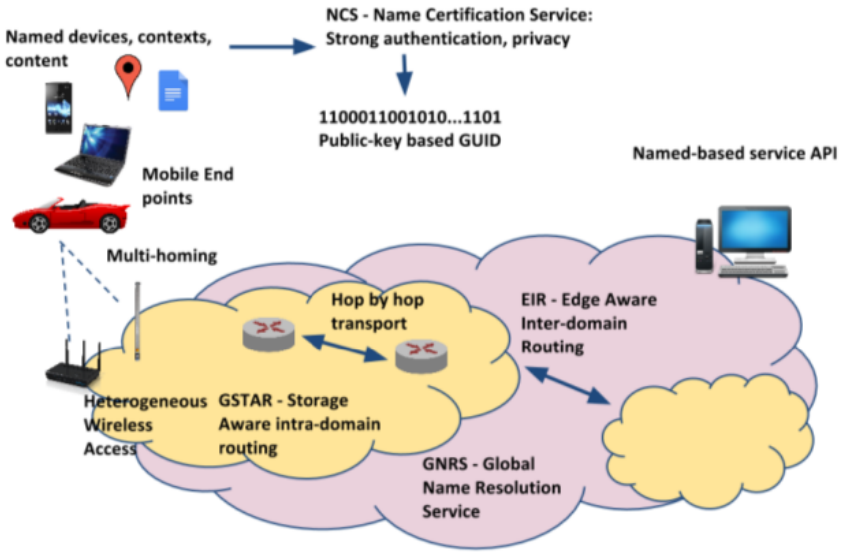


Figure 2.1: MobilityFirst architecture overview

The current IP-based Internet assigns IP addresses to the end hosts that are overloaded to represent the location of the host as well. Every time the host moves to a new location, the IP address has to be updated, changing its identity as well. MobilityFirst achieves a clear separation of the identifier and locator functions through the use of Globally Unique Identifiers (GUIDs).

The GUIDs serve as identifiers for a broad spectrum of objects ranging from a smart phone, service, vehicle, content, group of these objects and even a context. These GUIDs are assigned by a Name Certification Service (NCS) and are derived by hashing the public key. This provides the GUID a self certifying property obviating the need for an external authority for node authentication. GUIDs are dynamically mapped to a set of network addresses that corresponds to the physical attachment points or locators corresponding to the current attachment points to the internet, for the network objects. These mappings are stored in logically centralized Global Name Resolution Service (GNRS). This results in a scalable system in which, packets can be routed based on GUIDs of the end host which can automatically be resolved to a NA or a set of NAs based on where the end device is located. Details of the GNRS design and implementation are discussed in chapter 3. Just like IP is narrow waist for the current Internet, Figure 2.2 shows the GUID layer as the narrow waist of the MobilityFirst protocol stack.

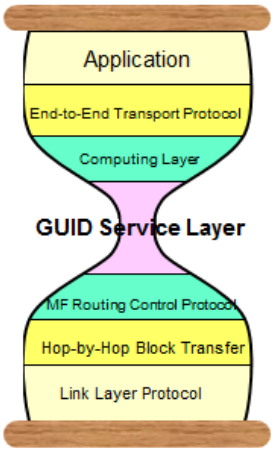


Figure 2.2: MobilityFirst stack with GUID as the narrow waist [1]

2.1 Intra-Domain Routing: GSTAR

Generalized storage aware routing or GSTAR [7] is the routing protocol used in MobilityFirst for intra-domain routing. It addresses the mobility related challenges directly at the networking layer. It is primarily a storage aware, hop-by-hop, link state routing protocol that enables routers to store data in case of network problems and transfer them later. GSTAR achieves high performance across a wide range of mobile environments such as wireless mesh, ad-hoc, DTN(disruption tolerant network) as well as relatively stable wired networks [6].

GSTAR uses both GUIDs and network addresses cooperatively for routing within the domain. The data packets called chunks, carry both the GUID as well as the address. At the first router, the GNRS is queried for the network address corresponding to the GUID and this is attached to the chunk. When the chunk reaches the destination network, if the destination has moved away from this network, the GNRS can be queried again for the updated location of the GUID. Hence, the GUID acts as the most authoritative piece of information for routing and must always be present as part of the chunk. Hop-by-hop transport protocol as described in [9] is used in GSTAR.

Each node participating in the GSTAR protocol uses two topologies to take routing decisions. The first, called intra-partition graph, is maintained via flooded link state advertisements. F-LSAs, as they are called, carry fine grained information about links in the domain. The second, DTN graph, is obtained via epidemically disseminated link state advertisements. The D-LSAs carry connection probabilities between the nodes. When a chunk arrives at a node, it looks at the intr-partition table first. If an entry exists, the chunk is forwared provided the link quality is good. Else, it is stored till the quality improves. If no such path is found, the DTN table is used to for a probabilistic view of the network to push out the chunk.

2.2 Inter-domain Routing: EIR

Edge-aware Inter-domain routing or EIR [8] is the routing protocol used to communicate between different domains of the network. EIR satisfies the basic inter-domain routing

protocol requirements of scalability, robustness, and support for flexible routing policies similar to BGP, the current inter-domain routing protocol of the Internet. Additionally, EIR also supports emerging use-cases of network-mobility, multihoming, multicast and anycast-services.

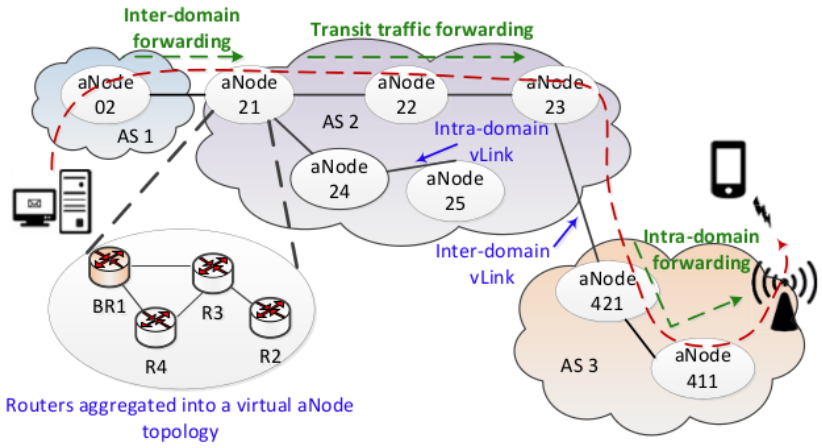


Figure 2.3: Representation of the router level topology of ASes in terms of aNodes and vLinks

In EIR, the autonomous systems are defined in terms of abstractions - aNodes and vLinks as shown in Figure 2.3. This enables networks to optionally express their internal structure and state in terms of aggregated virtual nodes and links each associated with key parameters such as bandwidth, latency and availability.The aNode and vLink abstractions are inspired by the Pathlet routing protocol [14] and are intended to provide a mechanism for autonomous systems (ASes) to express some details about their internal graph and wireless edge properties. Network State Packets (NSPs) encapsulate the information about the aNodes and vLinks and are flooded through out the network in a telescopic manner. This telescopic flooding provides fast updates to nearby networks while eventually providing all networks with a global view of the network topology. Each packet carries a service identifier (SID) in the header that allows support for multiple routing policies by serving to identify the service intent such as multi-homing, multicast or anycast. Late binding and in-network storage allow in-transit packets to be rebound to a new destination network address using the GNRS.

The protocol has been shown to provide improved support and flexibility for routing to wireless devices, network-assisted multipath routing, routing to multiple interfaces (multihoming) and service anycast. With increased expressiveness of network structure and node/link properties, the protocol is designed to have reasonably small overhead via telescopic dissemination of the nSPs while providing good service level performance in highly mobile scenarios.

Chapter 3

Extending the GNRS

3.1 GNRS Overview

The GNRS is one of the key components of MobilityFirst that allows for identifier and location separation. The current Internet provides poor support for mobility. As the end user changes his location, connections have to be re-established with every new network address. Application level support or protocols like mobile IP is needed to provide a seamless experience. Besides MobilityFirst, the idea of separating names from locators has been supported by a number of architectures such as HIP [3],XIA [5] and NDN [4]. This separation allows inherent support for mobility and multihoming.

The current GNRS implementation uses the Direct Mapping(DMap) scheme to achieve a scalable distributed system for shared hosting of the GUID to NA mappings. To perform the mapping the flat GUID is hashed K times produce K values in the NA space. The GUID to NA mapping is then stored in the GNRS of the corresponding ASes. The DMap scheme leverages on the existing routing infrastructure to spread the mappings all over the network. Thus, each AS hosts the shared mapping along with those of the GUID present in that AS.

Figure 3.1 shows the working of the DMap scheme where NAs are IP addresses and K=3. An insert from user A causes his GUID to be stored at the local AS as well as distributed to 3 other ASes. When another user wishes to contact A, the GUID is hashed to produce the IP addresses as before. Having the routing information base, the closest replica is selected to fetch the mapping.

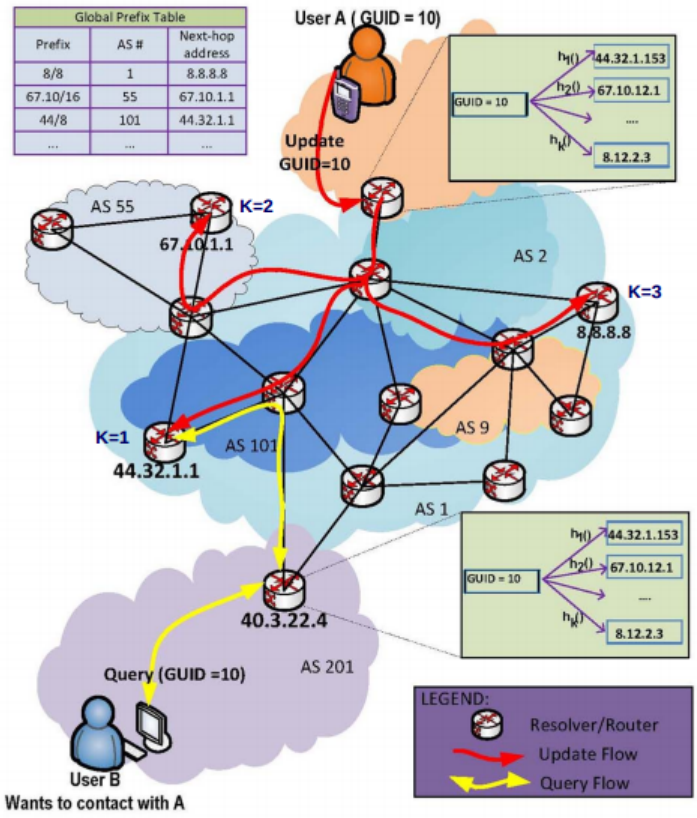


Figure 3.1: DMap implementation with K=3

The DMap implementation proves to be a simple and efficient scheme. It uses the existing routing protocol and requires no additional state information to be stored in the routers. It results in a single overlay hop for all requests. Storing the mappings at different ASes does not cause latency issues as these can be done in parallel. The look up latency is kept minimum by fetching the mapping from the closest AS. DMap scheme has been shown to achieve low latency with a mean value of 50 ms and 95th percentile value of 100 ms which is good for supporting mobility in the gloabal internet [10].

3.2 Motivation for extending the GNRS

As discussed above the GNRS is a fast, in-network scheme for globally distributing and storing GUID to NA mappings. Having this scalable efficient infrastructure motivated us to think what more could be achieved with the GNRS. This work proposes storing more information in the GNRS besides the GUID to NA mappings. Below we discuss

some of the possible advantages of storing more information in the GNRS.

Dynamic Network Formation and Mobility: Vehicular networks are a good example of dynamic networks. Disconnected islands of cars can form a network and peer with edge networks as they move. In such a case a GUID can be assigned to the group which in turn maps to the GUIDs of the individual vehicles.

Virtual Networks: The GNRS can be used to store topology information about dynamic virtual networks. A 'virtual' GUID can store a list of 'virtual' GUIDs that form that network. These in turn map to the actual router GUIDs.

Last Location: The GNRS can store the last known location of a GUID at all times. In case the GUID is currently disconnected, the packet can be sent to the previous location. Intuitively, the user would probably be closer to his previous location. Sending the packet to this location will be better than holding the packet at the sender. A re-look up could also be done closer to the destination network too see if there is an update regarding the destination location. These schemes would improve the delivery time and hence the overall network performance.

These scenarios motivated the idea of extending the GNRS to allow storing several types of mapping besides the simple GUID to NA mapping. This will facilitate development of various services in the network using the distributed scalable service of the GNRS.

3.3 Design of Extended GNRS

The original design of the GNRS stores a mapping record between a GUID to list of NA. It supports update, insert and lookup requests. Update and insert messages store a single or a list of NAs corresponding to a GUID. Lookup messages are used to fetch the NAs corresponding to a GUID.

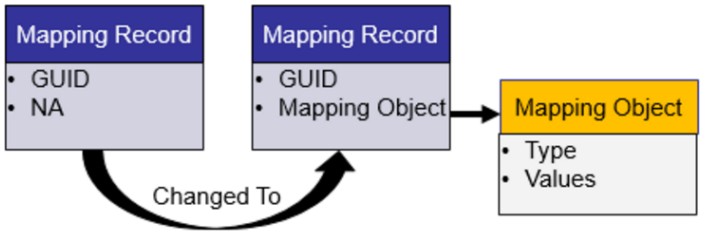


Figure 3.2: GNRS design modification

This original design is modified as shown in figure 3.2. Instead of a mapping record of a GUID to list of NA, the record now stores a mapping between a GUID and a generic mapping object. The structure of the object is determined as needed by the service the GNRS has to support. However, the generic structure of this object is a type and list of values. Some examples of this are shown in figure 3.3.

Use case	<GUID, Type, Value>
NA Mapping	<$GUID_x$, 1, NA_x>
Previous NAs	<$GUID_y$, 2, [NA_1,NA_2...]>
Virtual Networks	<$GUID_z$, 3, [$VGUID_1$,$VGUID_2$,...]>

Figure 3.3: GNRS design modification

The requests insert, update and lookup messages now change in format. The insert/update messages now contain, not only the the GUID and corresponding mapping object, but also the type of the mapping object. A single message can insert/update multiple mapping objects, each identified by their individual type. A look up message can either provide a GUID and a type or just a GUID. In case both a GUID and type are provided, that particular mapping object value is returned. On the other hand if only a GUID is specified then all the mapping objects of various types corresponding to the GUID are returned. The database design is modified to support both primary and secondary key. The primary key is a concatenation of the GUID and type to be fetched. The secondary key is the GUID alone. A search by the secondary key returns mapping objects of all types currently stored in the GNRS.

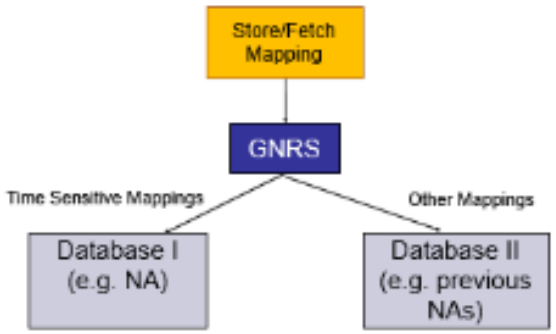


Figure 3.4: GNRS Database design modification

A proposal to ensure that the storing of mappings of different kinds does not affect the latency for queries to the basic GUID to NA, is shown in figure 3.4. Instead of a single database, we propose two databases. One that will store the time sensitive mappings(e.g. GUID to NA) and the other one to store the remaining types(e.g. Virtual GUIDs, Previous NAs). Currently since there are limited types, a single database is used. This design proposal needs to be evaluated and verified once sufficient types are identified and likely to degrade the original database performance.

Chapter 4

Proof of Concept

To demonstrate the benefits of using the GNRS extension to improve routing in MobilityFirst, two services were implemented and studied as described below.

4.1 GNRS assisted Multicast

The GNRS extension facilitates the development of a novel inter-domain multicast scheme enabled by name-identity separation. The proposed named-object multicast (NOMA) scheme [16] uses the GNRS to store the network address based multicast tree. The GNRS extension provides the API for inserts and lookups required by NOMA. As the multicast packet travels through the network, the GNRS provides fast in-network address look ups at branching points for delivering data to the members of the multicast group.

4.1.1 Architecture Overview

Several applications like video streaming, online gaming and social networks require sending the same data to multiple entities located in different parts of the network. The entity groups vary in size and longevity. Also, mobility of the end hosts makes this environment more dynamic. Several multicast protocols exist currently. Many of these rely on unicast based solutions using overlay networks without in network support for multicast. Solutions like PIM-SM [17] and MOSPF [18] are based on in-network support but have not been very successful in inter-domain multicast with issues in scalability and co-ordination across domains [19]. They also do not handle mobility scenarios well. Each time the a group member moves, it has to re-join the group and the multicast tree has to be modified which generates a lot of distributed control traffic.

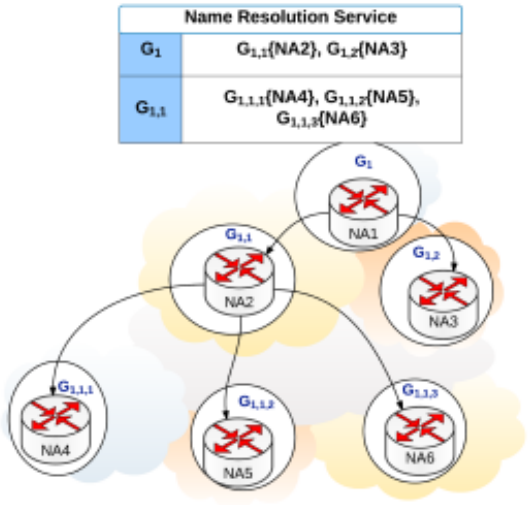


Figure 4.1: Multicast tree structure stored in the GNRS

GNRS assistance enables NOMA to be designed as an in-network multicasting solution based on named objects that are dynamically resolved to routable network addresses. A multicast group is identified by a unique GUID. The corresponding multicast tree topology is represented using name recursion and is stored in the GNRS as shown in 4.1. Each branching point is identified by a GUID. A look up for this GUID returns the next set of net work addresses to forward data to.

The multicast tree management has two main components - membership, and building and managing the multicast tree. Both these are achieved in NOMA by using the GNRS by two forms of indirection as shown in figure 4.2. A fisrt distinct GUID (GMng) is used for node membership. Any entity interested in joining the multicast group can insert its GUID against GMng in the GNRS. When a multicast message reaches a gateway router closest to the source, it builds the multicast tree. Recursive mappings are then used to express the tree graph by assigning to each branching router a name that exclusively identifies it in the context of the given multicast flow. Then we recursively follow the tree structure. As shown in figure 5.5 the root of this tree is identified by the multicast flow unique name mapping to the first branching router (GMulti Gr11). This router then maps to its children in the tree (Gr11 Gr21, Gr22). This continues

until the leaves of the tree are reached, where we identify the leaves as the destination nodes.

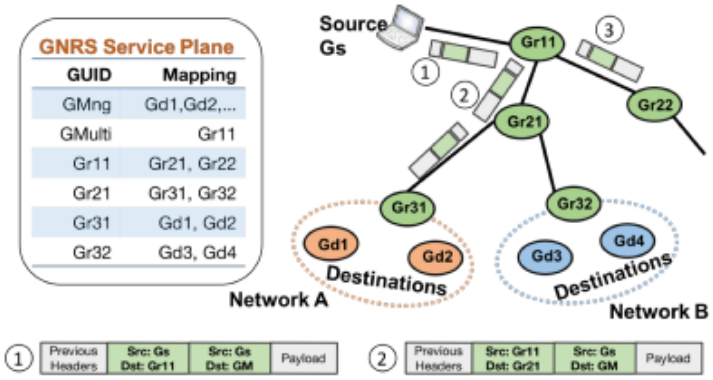


Figure 4.2: Multicast Architecture

Shortest path tree (SPT) algorithms are used for generating the multicast tree topology. The first algorithm considered is the Longest Common Path. In this scheme packets are forwarded along the longest-common path (LCP) to all the destinations, as a single copy, until the branching point is reached, where the packet is copied and delivered towards multiple destinations. This allows all destinations to receive multicast packets across the shortest path from the source. Another heuristic is the look-ahead longest-common path (LA-LCP) algorithm. Unlike LCP, which branches whenever there a divergence of shortest paths to multiple destinations, LA-LCP, compares the overall network cost of branching from the current node and branching from each of the possible next hops, and decides to branch downstream if the cost is lower for the latter, thereby deviating from the SPT. This reduces the overall packet hops in the network, with slight increase in computation complexity [16]. The LA-LCP algorithm is used for the evaluation of NOMA implementation.

4.1.2 GNRS API support

The GNRS extension provides support for the multicast tree insertion as well as the recursive look ups from the branching points of the tree. A new type of mapping object

corresponding to multicast is introduced in the GNRS which supports the storing of a list of (NA, GUID) pairs corresponding to the NA of a branching point and the GUID that should be then queried from that point. The following section details the interactions between the GNRS and the different components involved in the NOMA architecture.

- GNRS and interested entities: The end hosts that are interested in joining a particular multicast group, can request to insert their GUIDs against the multicast group GUID.

- GNRS and the gateway router: On receiving a multicast packet for a group, the gateway router queries the GNRS. The GNRS resolves the GUIDs of all the entities that belong to this multicast group and return a reduced NA list that the end hosts are currently connect too. Having the gateway router bulid the tree enables the tree computation to be topology- aware, as unicast path information of the NAs is available at the gateway. The tree is then inserted into the GNRS. The insert message of type 'multicast' has a the the following structure:
  GUIDM : [(NA1, GUIDNA1), (NA2, GUIDNA2) ... (NAn, GUIDNAn)]
  GUIDNA1 : [(NA11, GUIDNA11), (NA12, GUIDNA12) ... (NA1x, GUIDNA1x)]
  GUIDNAn : [(NAn1, GUIDNAn1), (NAn2, GUIDNAn2) ... (NAny, GUIDNAny)]
  The multicast tree root is identified as NA1 which has the mapping to the next hop NA along with the corresponding GUID for each child of the root. These in turn map to their children in the multicast tree. Once a tree is computed, it is updated in the GNRS such that downstream nodes do not need to recompute the tree again.

- GNRS and branching nodes: Each branching point duplicates the multicast packet that is receives for each child that it has to forward the packet to. The node queries the GNRS with the GUID it receives in the multicast message, with type set as 'multicast'. For the above example, node of address NA1 will query the GNRS for GUID GUIDNA1. This will return the next hops NAs as NA11, NA22 and so on. The node will attach the corresponding GUIDs(GUIDNA11, GUIDNA22) to

the corresponding duplicated packets. The packet are then sent out to the NAs.

The NOMA framework presents an efficient multicast architecture that scales well to fit medium and large scale trees and handles client mobility with disconnections [16]. The GNRS extension helps this architecture by providing the API to store the multicast tree as well as help in the delivering of packets across the tree via look ups at every branching point.

4.2 GNRS assisted Late Binding

The separation of the client identifier from its location in the network opens the door for in-network late binding. At an access router, the GNRS is queried for the NA corresponding to the destination GUID. This NA can be appended to the packet. This allows routers down the path to forward the packet based on the NA alone. This is referred to as early binding or 'fast path' routing. However network states that a network observes from far away could be obsolete during the transit of the packet and hence result in routing failure. Packets are then re-binded to the correct NA.

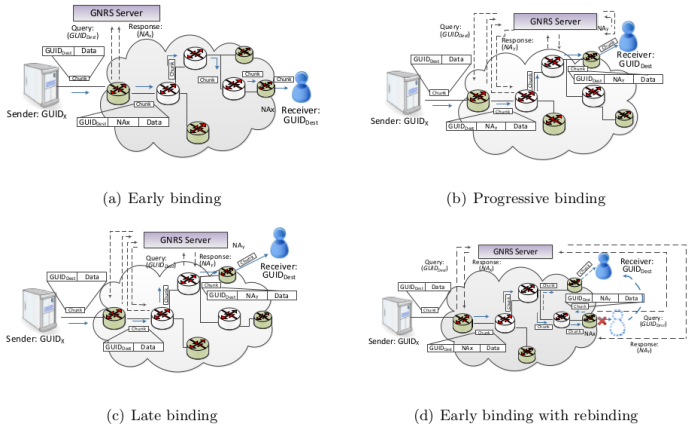


Figure 4.3: Mechanism of GUID to NA Binding

To overcome this, there is an option for routers en-route to do a subsequent GNRS look up for the updated location of the client, which may have changed due to mobility or temporary disconnection. This is known as 'slow path' routing. Slow path routing allows for implementation of late binding algorithms in the network to decide the NA

once the packet reaches closer to the destination. This enables the network to perform better even if the routing protocols are unable to keep up with the pace of client mobility.

Prior work on late binding involved determining the point for performing a re-lookup and evaluation of the same. It compared the scenarios of late binding at a junction router with a high degree, to that with re-binding the packet to the correct destination upon routing failure. Results showed an improved path stretch using late binding as compared to rebinding. This motivated us to look into further improving the network efficiency through late binding. In particular we look at how the GNRS can help late binding algorithms perform better.

At any given time, the GNRS provides the current location of a client that is connected to the network. As the client moves to a different network, the corresponding network address get updated into the GNRS. As the association records are logged by the GNRS, it naturally facilitates predicting the user location at any given time. In the simplest case, in the event of a disconnected user, we forward the packet to the last known location. Intuitively, if a sender is located fairly away, then re-binding the packet from the last known location would be faster than re-sending it all the way from the sender again.

The GNRS extension allows the routers to retrieve the last known location of GUIDs. Also, based on the location logs in the GNRS a separate prediction module can be added to the GNRS that would at anytime predict what the possible location of a GUID would be. Algorithms can combine using the GNRS prediction module and previous work on deciding when to query the GNRS for an updated location. This would lead to an improved network performance efficiency.

Chapter 5

Evaluations

5.1 GNRS Extension

The DMap version of the GNRS is implemented in Java. Changes were made to this implementation to accommodate storing mapping objects of different kinds. Modifications were also made to the GNRS API, that is used by the routers (implemented on the click routing platform) to communicate to the GNRS. We anticipated that the changes to the original design of the GNRS would increase the query latency marginally due the additional filtering by type and more complex mapping object structure. However, we need this increase to be as minimum as possible.

We compare the performance of the original DMap implementation to the extended framework. The experiments are performed on a linux machine that has a single client and a single instance of GNRS running. For this first experiment, a query (insert or lookup) is sent to a new unused instance of the GNRS and once the response is received the next message is sent. In the figure 5.1, we plot the average RTT (round trip time) against the log (base 10) of the number of inserts. We see that the initial set up time is high due to several initialization routines (e.g. setting up the database tables) occurring for getting the GNRS up and running. However, over time the average RTT reduces. This is because the initial time's effect is reduced by the increasing number of samples considered. Figure 5.2 shows a similar plot for the lookup query times.

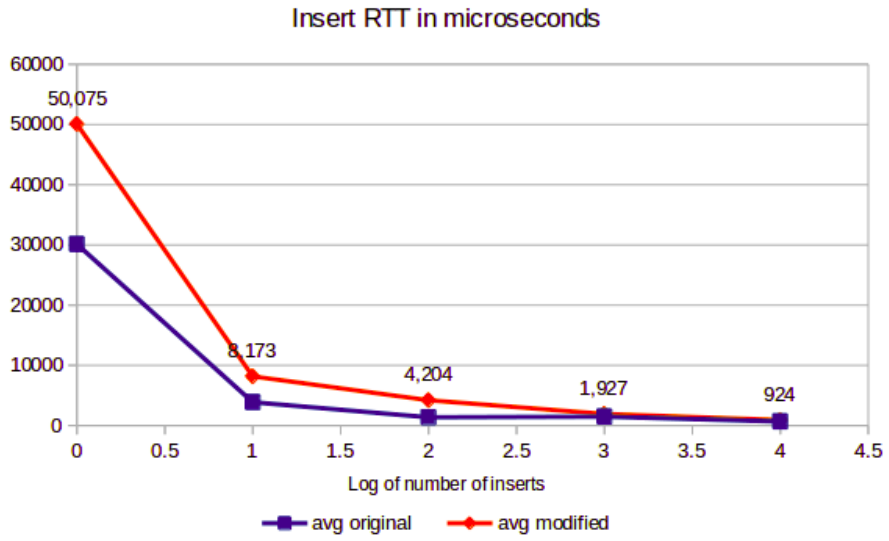


Figure 5.1: Plot of the average insert time vs log of number of inserts

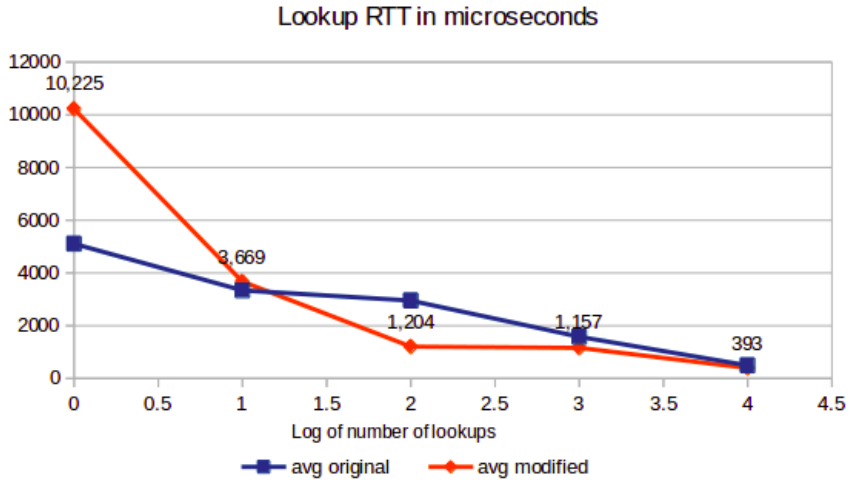


Figure 5.2: Plot of the average lookup time vs log of number of inserts

Another experiment was done to study the performance of the framework where in we studied the average RTT for inserts/lookups per second. For this experiment the we initialize the GNRS with around 1000 GUID to NA mappings.This is done by using an existing database with 1000 mappings. In this case Figures 5.3 and 5.4 show a plot of the average RTT as the number of queries per second are increased from 1 to a 1000. We see that the initial set up time is not that high now as the database initialization is already done. Though the RTT values for the modified GNRS are higher than the

original implementations, they are still within acceptable limits for good performance of the distributed infrastructure.

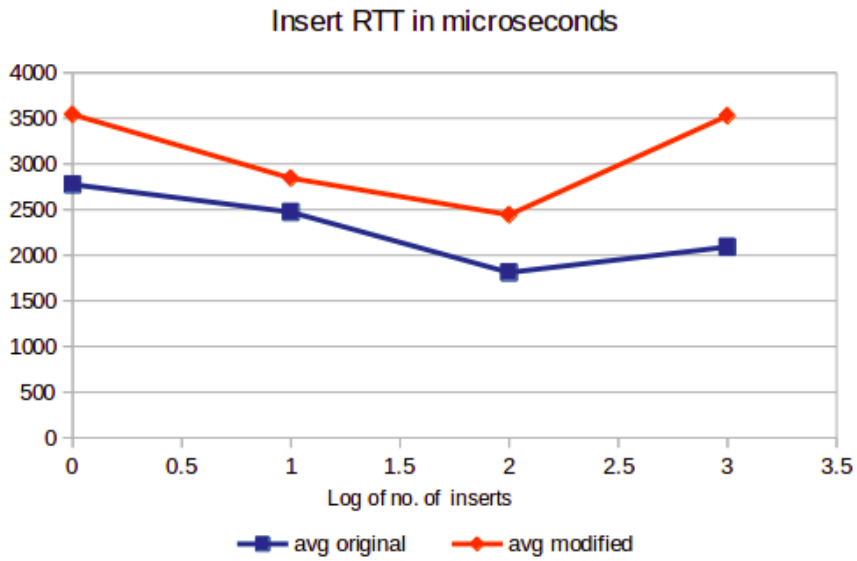


Figure 5.3: Plot of average insert time vs log of number of inserts per second

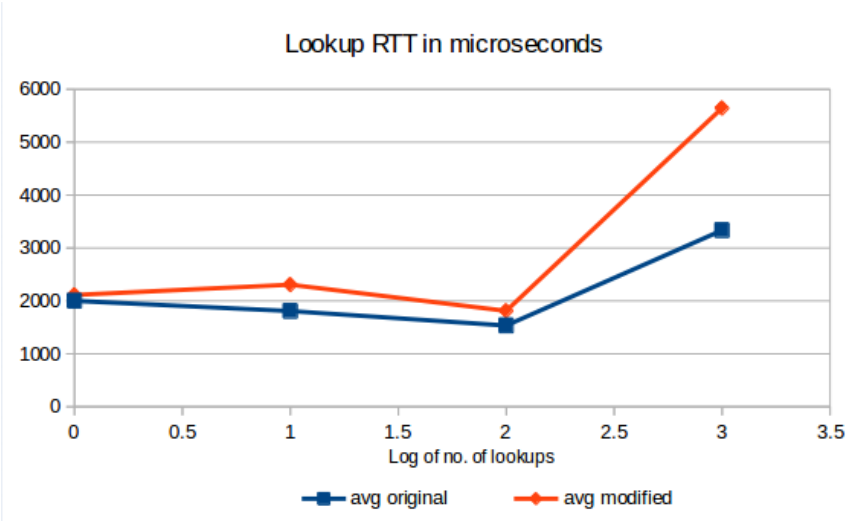


Figure 5.4: Plot of average lookup time vs log of number of lookup requests per second

We see that the extension framework increases the query latency by a small value. This is attributed to the relatively more complex nature of the mapping object stored and corresponding processing. This is a trade off for the benefits of using this extended framework of the GNRS to support inter-domain services in the MobilityFirst network.

5.2 Multicast

Performance evaluation based on large scale analytical modeling as well as fine-grained packet-level simulation on network simulator (NS3) can be found in the NOMA paper [16]. Here we evaluate the performance of NOMA on the ORBIT [20] testbed.

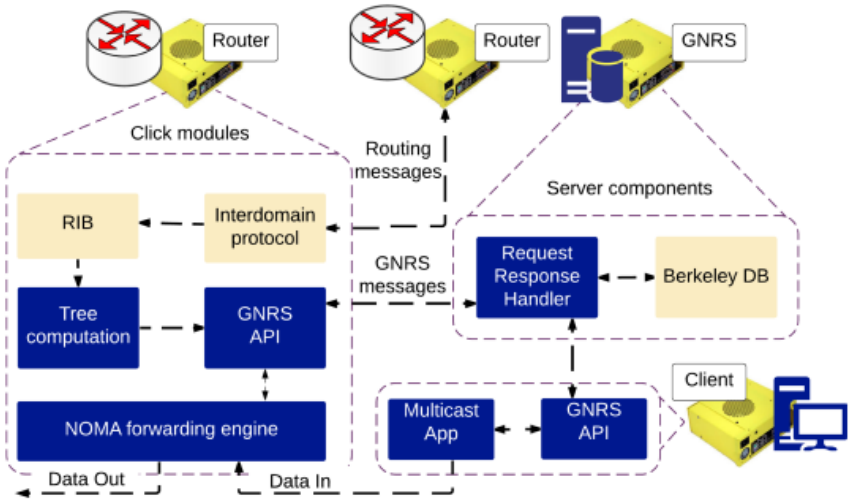


Figure 5.5: Components of NOMA - GNRS and click routers. Updated modules shown in blue.

The NOMA design is implemented over the existing MobilityFirst infrastructure. Network nodes are implemented on the Click modular router software. The GNRS extension framework provides the updated API for inserts and lookups during the multicast tree creation and packet forwarding. Figure 5.5 shows the overall system diagram with the components modified for this implementation highlighted in blue.

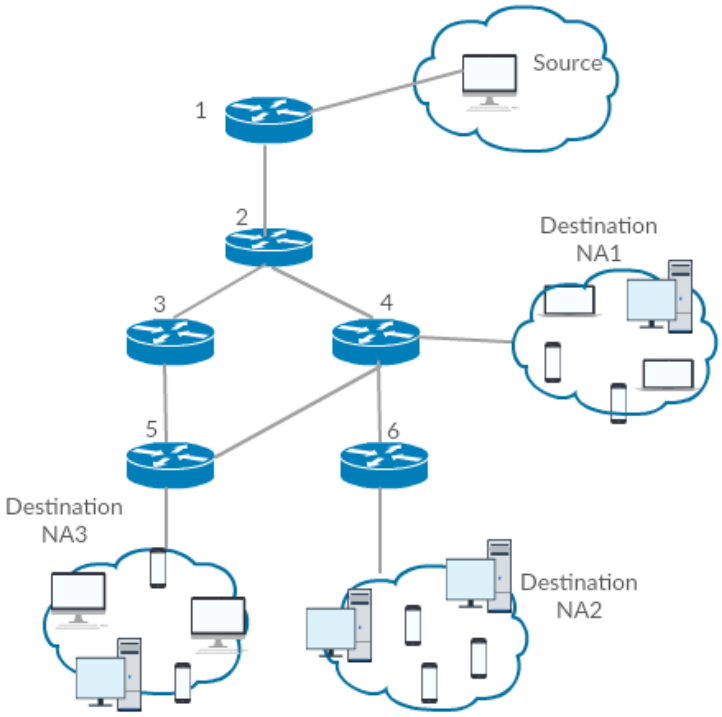


Figure 5.6: 6 node test topology.

For the purpose of evaluation on the ORBIT testbed a topology is used as shown in figure 5.6. The 6 click routers are connected as shown. The source is connected to router 1. There are 15 end hosts that are part of the multicast group. These send association messages to the GNRS to join the multicast group. These are distributed over three ASes connected to via border routers 4, 5 and 6. The source sends a multicast message to Router 1. Router 1 queries the GNRS to know the end network addresses. Router 1 builds a tree for NA1, NA2 and NA3. The LA-LCP algorithm results in the multicast tree containing routers 1,2,4,5 and 6. Without the LA-LCP algorithm, the multicast packet would be duplicated at node 2. Node 2 would be a branching point and the packet would reach NA3 via node 3. However, the LA-LCP algorithm prevents the branching at node 2. It calculates a lower cost for branching at node 4. At each branching point a GNRS lookup is performed (as explained in Chapter 4) to fetch the next points to send the multicast packet. We study the hops count and control overhead for this topology.

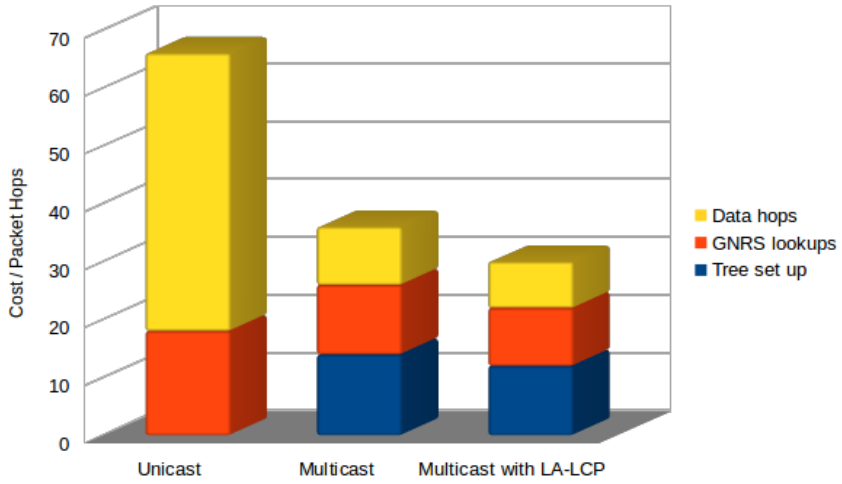
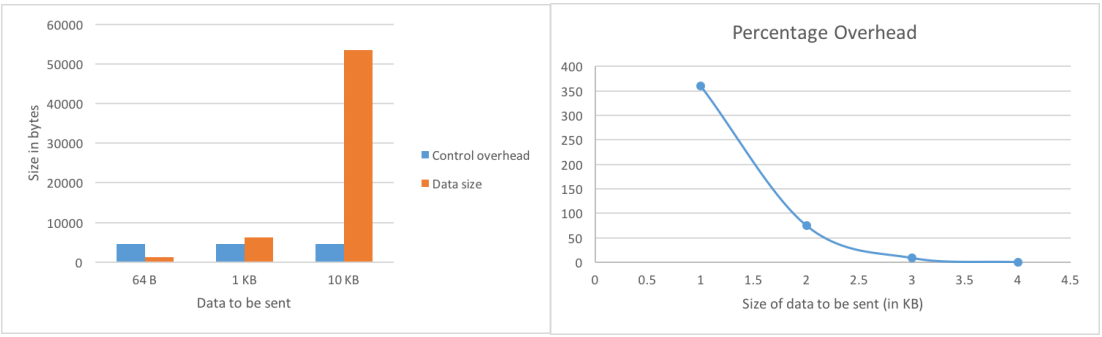


Figure 5.7: Cost comparison for unicast and multicast schemes

Figure 5.7 shows a the cost in terms of packet hops to send a message to 15 end hosts in the 6 router topology. We see that though multicast schemes involve a cost for setting up of multicast trees and insert it into the GNRS, the over all cost is much lesser than sending the message via unicast. Also, we see that the LA-LCP performs better than the just LCP. While the LCP algorithm causes packet duplication and branching at node 2, LA-LCP branches only further at node 4, totally avoiding node 3. This reduces the total number of packet hops.



(a) Plot of overall control and actual data (b) Percentage overhead for different data sizes

Figure 5.8: Comparison of the control vs the actual data transferred in the 6 router - 15 hosts topology for different sizes of data messages.

We also study the control overhead in terms of the total number of bytes that need to be transferred over this topology. The NOMA paper [16] shows a comparison of the control packet overhead for the tree set up between NOMA and PIM-SM. For tree sizes around 50 nodes and above, NOMA performs much better than PIM-SM. In figure 5.8 we study the control overhead in terms of the total number of control bytes transferred when compared to the actual data being transmitted. The control overhead includes interest messages from the end hosts to join the multicast group, the tree insert by the border router (Router 1 for 5.6 in this case) and the GNRS lookups at the branching nodes. The message to be sent from the source is varied from 64 bytes to 1 MB. For very small message sizes, the control overhead is large when compared to the actual data flowing through the network. For moderate sizes like 1KB we see that the control overhead is around 75%. However, the percentage overhead reduces significantly as the message size increases as shown if figure 5.8b. Further, if we consider that GNRS lookup messages are cached on the branching routers, the overhead will be further reduced. Also, as the multicast tree will likely be used for longer than one message, the percentage overhead is further reduced.

Overall, from these ORBIT evaluations as well as from the simulation results from [16], we can conclude that the NOMA architecture is an efficent and scalable multicast solution for inter-domain networks. The GNRS plays a crucial part of the solution. The extension framework allows the GNRS to store the recursive multicast tree and support lookups at the multicast branching points. Thus the GNRS extensions effectively helps facilitate minter-domain multicast in the MobilityFirst network.

5.3 GNRS assisted Late-Binding

In this section we study through simulations the benefits of using the GNRS in improving existing late binding infrastructure in MobilityFirst. For the simulation we use a Caida data set from 2012 [15] that provides the location of 22,000 access points(APs) in San Francisco, California. We also have the location of a cab that is moving through the city. By calculating the closest AP to the cab, we generate a trace for the cab as it connects to different APs during its trips around the city. We then simulate sending

data to the cab and record the transfer time and hop count for analysis.

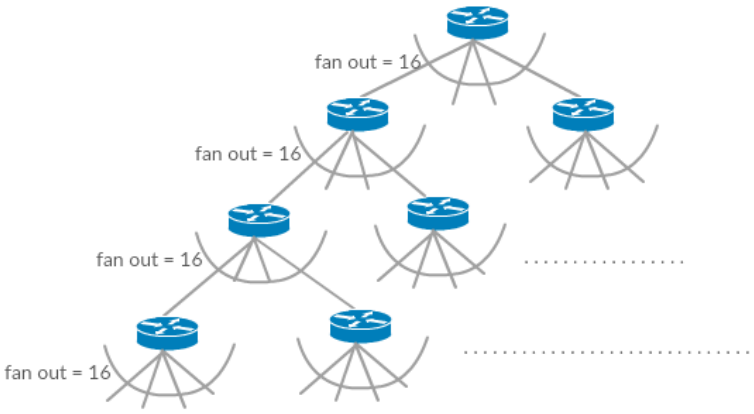


Figure 5.9: Structure of the MAN network

To have a structure more similar to aNodes in MobilityFirst, we group close located APs of the 22,000 from the Caida data set together to reduce the number of APs to 4096. We then consider these in a Metropolitan Area Network(MAN) architecture. The tree structure has four levels with a fan out of 16 at each level as shown in figure 5.9. We use a clustering algorithm to create the tree structure. This clustering is then repeated in each group to create further levels of the MAN tree. Figure 5.10 shows the top level clusters after one round of clustering.

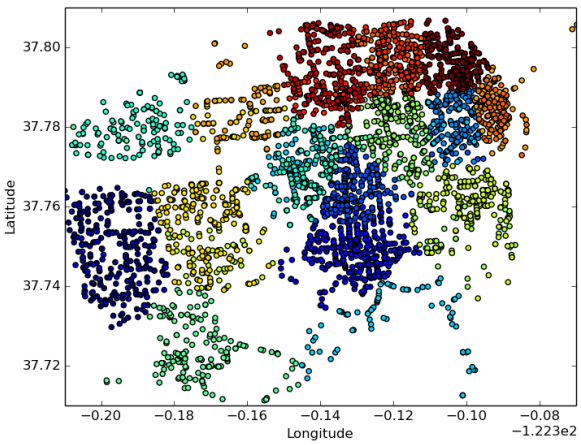
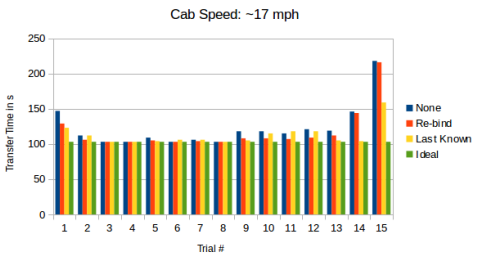


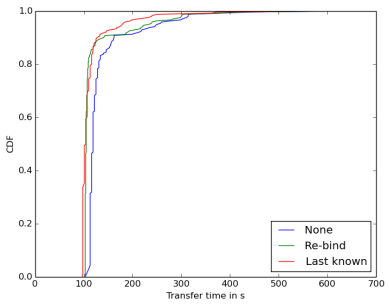
Figure 5.10: Top level clusters

For each instance from the cab's log, we calculate the AP that the cab is closest to. This gives us a trace of the APs that the cab connects as it moves through the city. We simulate sending 1 GB files to the cab. The sender is located 5 hops away from the top node of the MAN network. We assume the link speeds to be 10 MBps.
Four different scenarios are used for the simulations:

- **None**: This is the simple case where on delivery failure, the packet is re-sent from the original sender's location.
- **Re-bind**: When a delivery fails, the GNRS is queried for an updated location and the packet is forwarded from the current network address, instead of the original sender's location.
- **Last Known**: This is an extension to the 're-bind' case. The main difference is observed when the user is disconnected and his current location is not available in the GNRS. In such a case while the 're-bind' scheme holds the packet, waiting for a location update, the 'last known' scheme forwards the packet to the last know location in the GNRS. We expect the user to be closer to his previously known location when compared to the location of the sender.
- **Ideal**: This scheme represents best possible scenario. Using prediction schemes with the information available in the GNRS can enable us to get closer to the performance of the ideal case. This scheme serves as a guideline to measure the other schemes.



(a) Plot of transfer time for 15 trials



(b) CDF of the transfer time

Figure 5.11: Plot of transfer time for 15 trials for the different schemes

The two plots above show the time to transfer 1 GB files. In figure 5.11a we conduct 15 trials of sending the 1 GB file. For the trials like 3 and 4 we see there is little or no difference between transfer time for the different cases. This is due to the fact that the cab was largely located in the same AS. During trials 1, 14 and 15, large number of transitions and disconnections are observed. We observe that for most of the trials, the 'last known' case performs better than the 're-bind' or 'none' cases. However, in some cases the transfer time is more. This is due to the fact that the packet suffers multiple re-bindings due to rapid movement of the cab. Figure 5.11b is the CDF plot of the transfer time of over 200 trials. It shows that the 'last known' case performs better than the 're-bind' which in turn is better than the 'none' scheme.

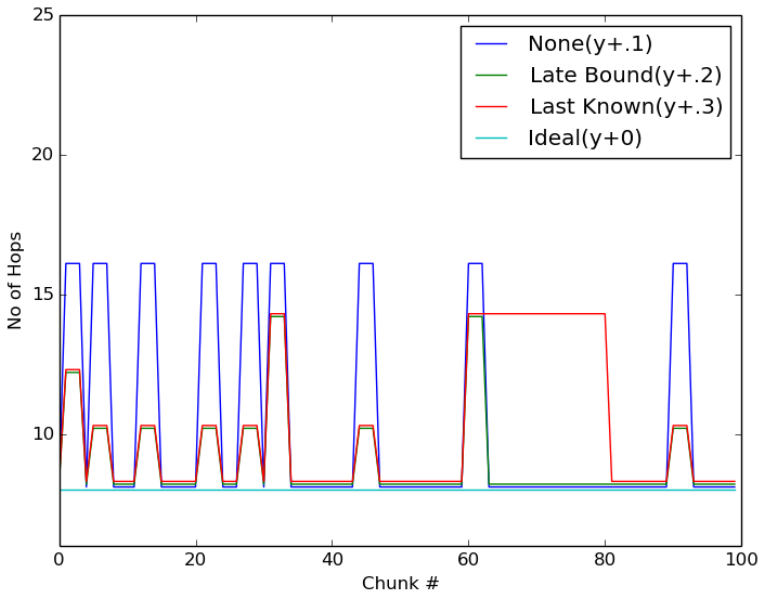


Figure 5.12: Hop count time for 15 trials

Figure 5.12 shows the number of hops encountered by each chunk during the first trial. We see that number of hops are lesser for both the 're-bind' and 'last known' case when compared to the 'none' case. Between the 60th and 80th chunk we can see a marked difference between 'last-known' and 're-bind' case. This can be attributed to the fact that during the transmission of these chunks the cab experienced a lot of disconnections. For the 'last known' sceme this causes the chunks to get transmitted to

the last known location, which then have to be re-binded to the correct location. The normal 're-bind' case holds the packets. It does not transmit packets when the cab is disconnected and the user location is unknown. It transmits them only after the cab connects to an AP and its location is updated in the GNRS. Hence the hop count for these chunks is not much. However, even though the number of hops is larger, we see from figure 5.11 that the overall transfer time for the 'last known' case is lesser than the 're-bind' case.

The average speed of the cab in the trace was 17mph. As disconnections and transitions will be more when a cab is moving faster, the experiment is repeated for higher speeds, 32 mph and 65 mph. Figure 5.13 and 5.14 show a comparison of the transfer time and hop count plots for the three speeds.

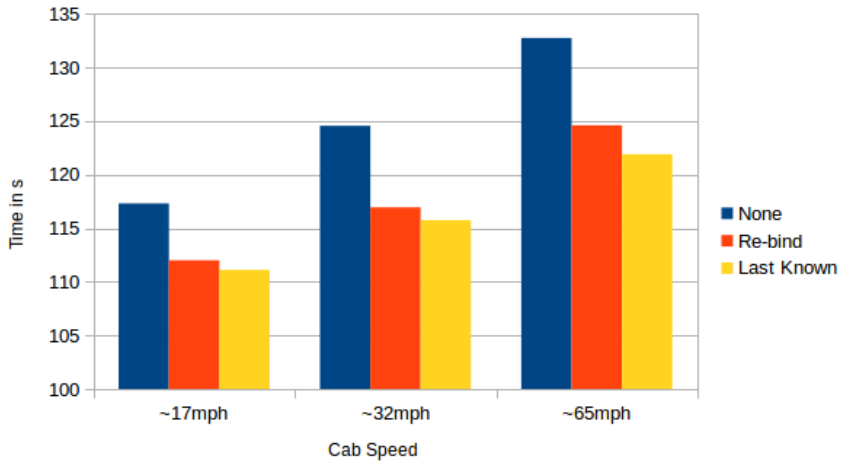


Figure 5.13: Plot of average transfer time for different cab speeds

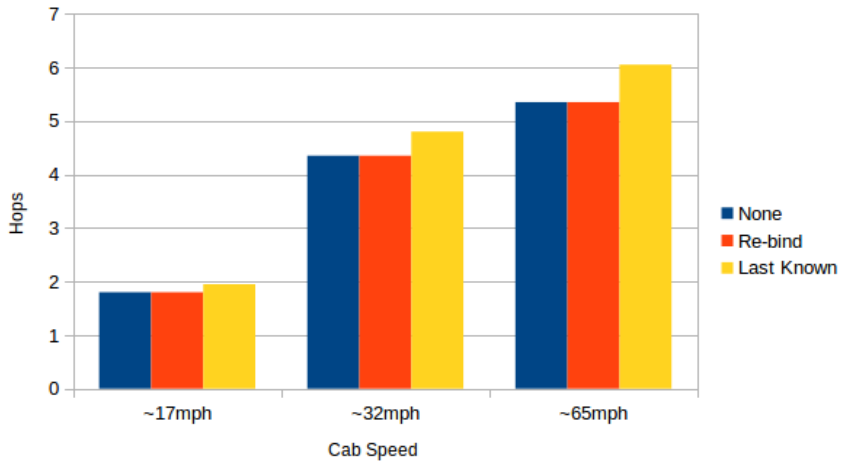
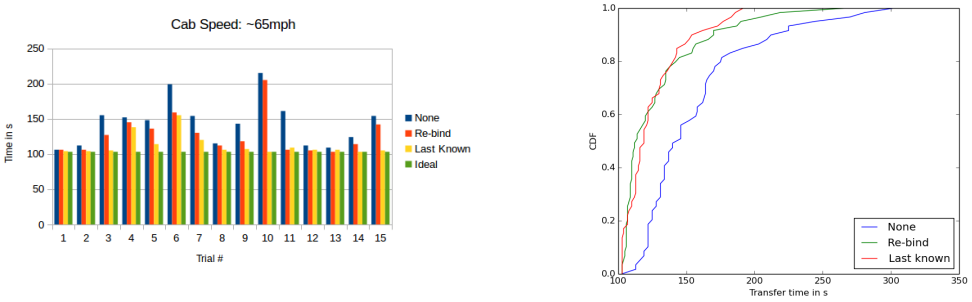


Figure 5.14: Average hop count at different speeds

We see that average transfer time increases significantly for case with no rebinding. For the 'none' case a 11.25% increase is seen. The 'last-known' case performs the best with an increase of 9.72%. We see that the average hop count is always more for the 'last known' case. As explained before, this is attributed to the fact that the packets have to be rebounded from the last know location to the current location. Transfer time for 'last known' is lesser than the other two schemes.



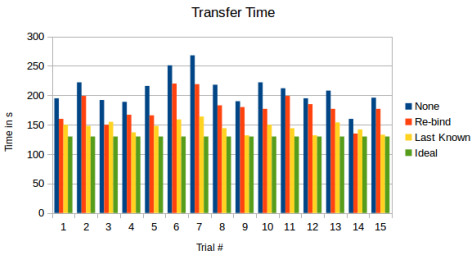
(a) Plot of transfer time for 15 trials

(b) CDF of the transfer time

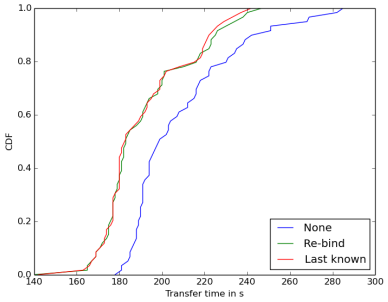
Figure 5.15: Plot of transfer time for 15 trials for the different schemes

In order to study what happens at higher speed, we repeat the first experiment with the cab speed at 65 mph. We see from figure 5.15 that there are more transitions and disconnections in this case. Figure 5.15b shows a marked improvement for late binding

schemes when compared to the one without any late binding. At higher speeds, as there are more disconnections and transitions, late binding schemes fare better. For the base case, any failure will result in re-sending the packet all the way from the original source and hence be more expensive.



(a) Plot of transfer time for 15 trials



(b) CDF of the transfer time

Figure 5.16: Plot of transfer time for 15 trials for the different schemes

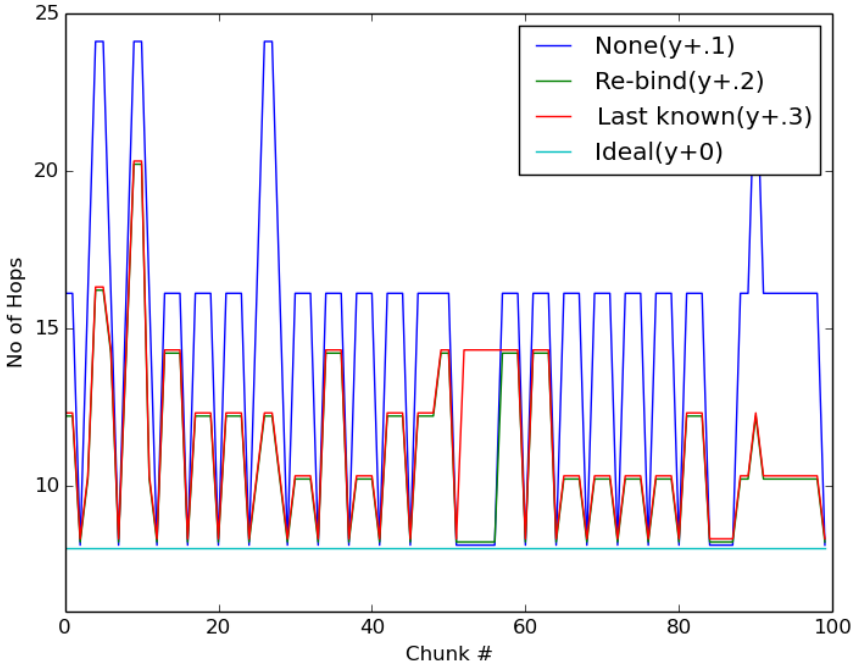


Figure 5.17: Hop count time for 15 trials

Another experiment conducted to study the transfer time and hop count characteristics was to modify the trace such that the cab would not be connected to any AP for more 30 seconds. From figure 5.16 we can see that transfer time is much higher. Figure 5.17 also shows higher number of hops when compared to the previous experiments. Since the association is limited to 30s, a large number of chunks get delivered to a network address after the cab has moved away from that location. This causes the chunks to be re-transmitted. We see that the late binding schemes out perform the scheme that does not use late-binding. The number of hops as well as the overall transfer time is higher when late binding is not used.

From the above studies we can see that re-binding along with sending to the last known location improves the network routing performance. Using the last known location from the GNRS has resulted in a reduced transfer time. Even if in certain cases the number of hops traveled by the packets is higher, the transfer time is less. The CDF plots clearly show an improvement in the transfer time when using late binding schemes as opposed to plain routing.

These results motivate the fact that using the logging information from the GNRS to assist in inter-domain routing will be beneficial to the network. One such idea is to develop a prediction module that takes end node mapping information from the GNRS and at any point returns the predicted location of the end node. The packets can be sent directly to the predicted location. Previous work on content caching and prefetching at the edge [21], has shown the benefits of such a prediction module with the GNRS. Future work can be done to research various algorithms and mechanisms to use the logging information from the GNRS to make accurate predictions for more effective late binding and data delivery.

Chapter 6

Conclusion and Future Work

The extension to the GNRS allows the support for different services in the Mobility-First network. The change increases the query responses slightly but within acceptable limits. We studied two applications of this extension. The GNRS assisted multicast solution enables efficient inter-domain push multicast. The multicast tree is stored in the GNRS and through recursive look-ups, a scalable multicast solution is achieved. The second application is late binding in inter-domain routing. Late binding with additional schemes such as sending to the last known location, provides significant performance improvements for mobile clients that change location frequently.

6.1 Future Work

The extension to the DMap implementation of the GNRS can be used to support further services in the MobilityFirst network. The extension provides a generic framework for supporting new services. As and when these services are added to the MobilityFirst network, the corresponding support can be added to the GNRS implementation with minimal code changes.

Also more work can be done to develop a complete and efficient late-binding algorithm. Prior work has been done to determine the en-route router to perform the late-binding. This work looks at utilizing simple information from the GNRS to improve the late-binding performance. Future work can involve developing a prediction module in the GNRS to help estimate the location of the destination at any time. This will further help reduce the time taken to reach destinations and improve the network efficiency.

References

[1] MobilityFirst project, http://mobilityfirst.winlab.rutgers.edu/.

[2] Dipankar Raychaudhuri, Kiran Nagaraja, and Arun Venkataramani. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2012

[3] Robert Moskowitz, Pekka Nikander, Petri Jokela, and Thomas Henderson. Rfc 5201host identity protocol. 2008.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *CoNEXT*, 2009.

[5] Ashok Anand, Fahad Dogar, Dongsu Han, Boyan Li, Hyeontaek Lim, Michel Machado, Wenfei Wu, Aditya Akella, David G Andersen, John W Byers, et al. XIA: An Architecture for an Evolvable and Trustworthy Internet. In *HotNets*, 2011.

[6] Nehal Somani, Abhishek Chanda, Samuel C. Nelson, Dipankar Raychaudhuri. Storage aware routing protocol for robust and efficient services in the future mobile internet. *IEEE International Conference on Communications (ICC)*, 2012.

[7] S. C. Nelson, G. Bhanage, and D. Raychaudhuri. GSTAR: Generalized storage-aware routing for MobilityFirst in the future mobile Internet. In Proceedings of *MobiArch*, pages 19-24, 2011.

[8] Shreyasee Mukherjee, Shravan Sriram, Tam Vu, Dipankar Raychaudhuri. EIR: Edge-Aware Inter-Domain Routing Protocol for the Future Mobile Internet

[9] S. C. Nelson, G. Bhanage, and D. Raychaudhuri. GSTAR: Generalized storage-aware routing for MobilityFirst in the future mobile Internet. In Proceedings of *MobiArch*, pages 19-24, 2011.

[10] Tam Vu, Akash Baid, Yanyong Zhang, Thu D. Nguyen, Junichiro Fukuyama, Richard P. Martin, Dipankar Raychaudhuri. DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet. In *IEEE ICDCS, 2012*

[11] Yi Hu, Roy D. Yates, Dipankar Raychaudhuri A hierarchically aggregated in-network global name resolution service for the mobile internet.

[12] X. Tie, A. Sharma, and A. Venkataramani. A global name service for a highly mobile internetwork. Technical report, UMASS, Tech. Rep., 2013.

[13] Arun Venkataramani, Abhigyan Sharma, Xiaozheng Tie, Hardeep Uppal, David Westbrook, Jim Kurose, Dipankar Raychaudhuri. Design requirements of a global name service for a mobility-centric, trustworthy internetwork. In *IEEE COMSNETS*, 2013.

[14] P. Godfrey, I. Ganichev, S. Shenker, and I. Stoica. Pathlet Routing In *ACM SIGCOMM Computer Communication Review* , vol. 39, no. 4. ACM, 2009, pp. 111–122.

[15] The CAIDA UCSD Internet Topology Data Kit. Available: `http://www.caida.org/data/topology-data-kit`.

[16] Shreyasee Mukherjee, Francesco Bronzino, Suja Srinivasan, Jiachen Chen and Dipankar Raychaudhuri. Achieving Scalable Push Multicast Services Using Global Name Resolution. *IEEE GLOBECOMM*, 2016.

[17] Dino Farinacci, C Liu, S Deering, D Estrin, M Handley, Van Jacobson, L Wei, Puneet Sharma, David Thaler, and A Helmy. Protocol indepen- dent multicast-sparse mode (pim-sm): Protocol specification. 1998.

[18] John Moy. Rfc 1584multicast extensions to ospf. *SRI Network Information Center*, 1994.

[19] Christophe Diot, Brian Neil Levine, Bryan Lyles, Hassan Kassem, and Doug Balensiefen. Deployment issues for the ip multicast service and architecture. *Network, IEEE*, 14(1):7888, 2000.

[20] Dipankar Raychaudhuri, Ivan Seskar, Max Ott, Sachin Ganu, Kishore Ramachandran, Haris Kremo, Robert Siracusa, Hang Liu, and Manpreet Singh. Overview of the orbit radio grid testbed for evaluation of next- generation wireless network protocols. In *In Wireless Communications and Networking Conference, 2005 IEEE*, volume 3, pages 16641669. IEEE, 2005.

[21] Feixiong Zhang, Chenren Xu, Yanyong Zhang, K. K. Ramakrishnan, Shreyasee Mukherjee, Roy Yates, Thu Nguyen. EdgeBuffer: Caching and prefetching content at the edge in the MobilityFirst future Internet architecture. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, IEEE 2015.

Appendices

Appendix A

GNRS - Updated Protocol Specification

The GNRS supports three types of requests from a client: insert, update and, lookup or query request, where the operations may be thought equivalent to the basic functions on a map data structure storing key/value pairings. The difference between the insert and update requests is that the former equals a 'set' operation wherein any previous value mapped to the key is replaced by the new bindings. In an update operation, the new values are appended to the existing ones. The extended GNRS stores GUID/value bindings of different types. The client can send a lookup request for a particular type, or he can request for the all the mappings of a particular GUID. The sections below detail common GNRS objects and message protocols.

Network Address: The Network Address is a network-routable identifier acting as a communication endpoint (source, destination) within GNRS. A network address is represented in GNRS as the triple (Type, Length, Value). GUIDs may be bound to multiple Network Address values that actually identify the same network endpoint in different formats.

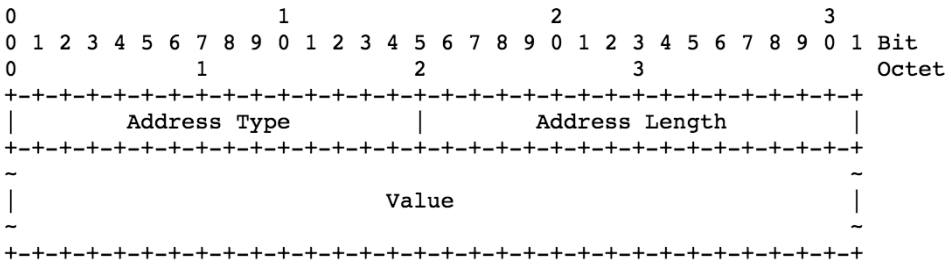


Figure A.1: Network Address

- Type - 16-bit value identifying the type of network address represented.

- Length - 16-bit unsigned integer value identifying the length of the network address in bytes.
- Value - Variable-length binary value. Contains the raw (binary) form of the network address, dependent on the type and length.

Common Request Header: This is the header for any kind of request sent to the GNRS.

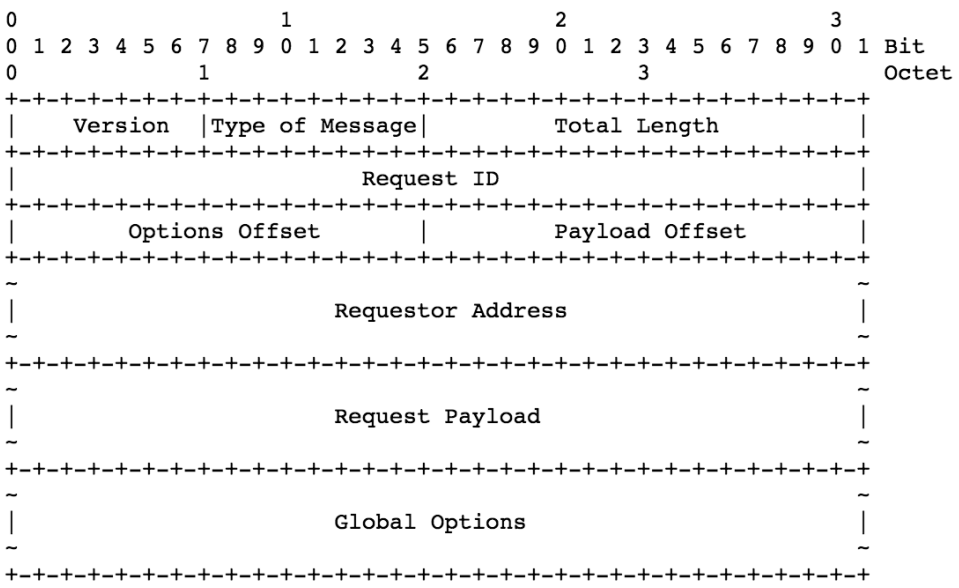


Figure A.2: Common request header

- Version - Protocol version. Currently 0 for development.
- Type - The type of message that follows (insert/update/lookup).
- Total Length - The total length of the message, including the header fields, in bytes.
- Request ID - Identifier for this request from the requestor. The pair (Request ID, Requestor Address) should be unique within a reasonable period of time (hours or days).
- Options Offset - Byte offset of the beginning of the options fields. An offset value of 0 indicates no options.

- Payload Offset - Byte offset of the beginning of the message payloads.
- Requestor Address - Network Address of the original sender (originator) of the request message.
- Requestor Payload - Message-specific request payload, including options. See message types below. Global Options - A set of global options for the query.

Options: There are two types of options - Global and Local. Global options which are appended to the message as whole (e.g. Recursive Option). Local options are associated with each binding that is stored in the GNRS (e.g. Expiration Option). Both types are encoded in the same format. Options are encoded as a (Type, Length, Value) 3-tuple. Unsupported options can be ignored by the receiving host, but should be preserved when stored or forwarded so that other hosts have the opportunity to interpret them. The block of options are located either before or after the payload of the message. The highest bit of the type field is a reserved flag to indicate the final option for the message. The range of values for an option Type is 0x00-0x7F, and if the highest bit is set (i.e., (TYPE AND 0x80) == 0x80), then no additional options will follow.

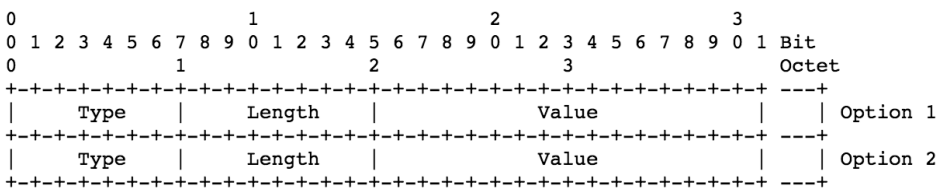


Figure A.3: Options

Lookup Request: The lookup request is used to retrieve the binding(s) for a GUID. It includes the query GUID and optional parameter 'type'. If a type is specified, only that particular binding is fetched. Multiple types can be included in a single lookup request. If no type is specified, all the bindings for that GUID are returned. Its' format is as follows:

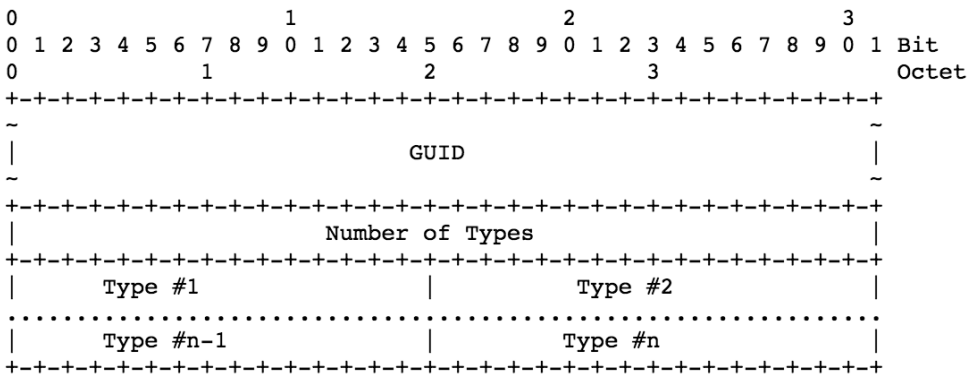


Figure A.4: Lookup Request

- GUID - The GUID being queried in this message.
- No of Types - The number of types being fetched. Value of 0 fetches all the stored bindings for the GUID.
- Type - Type of binding

Insert/Update Request:

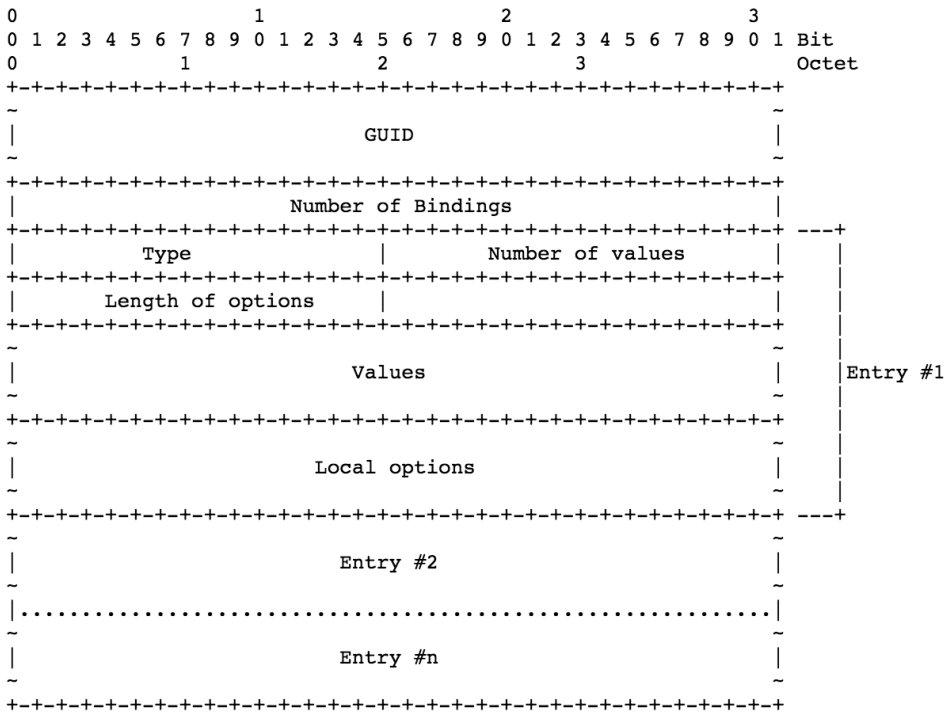


Figure A.5: Insert/Update Request

- GUID - The GUID value being !inserted/updated.
- Number of Bindings - Number of bindings being inserted/updated.
- Entry - Binding of a type
- Type - Type of the binding being inserted/updated
- Number of values - Number of values of the type.
- Length of options - Length of local options.
- Values - The values of the type. Parsing will be according to the type (e.g. Network Address).
- Local Options - The options associated with the values.

Common Response Header: This is the format of the common response header for insert, update and lookup query responses.

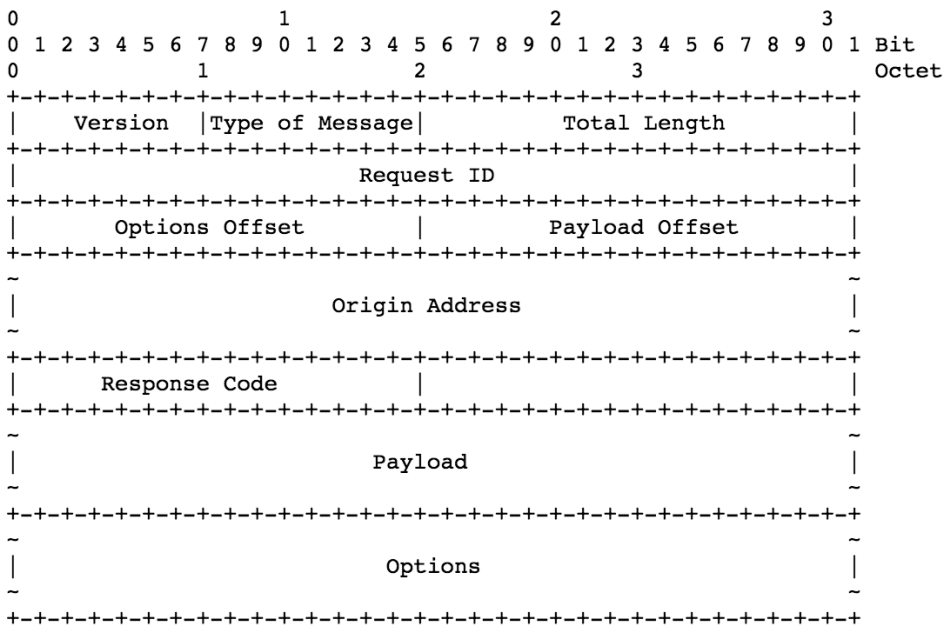


Figure A.6: Common Response Header

- Version - Protocol version. Currently 0 for development.
- Type - The type of message that follows.

- Total Length - The total length of the message, including the header fields, in bytes.
- Request ID - Identifier for this request from the requestor. The pair (Request ID, Requestor Address) should be unique within a reasonable period of time (hours or days).
- Options Offset - Byte offset of the beginning of the options fields. An offset value of 0 indicates no options.
- Payload Offset - Byte offset of the beginning of the message payloads.
- Origin Address - Network Address of the original sender (originator) of the reply message.
- Payload - The type-specific payload of the reply message, including options.
- Options - A set of options for the query, limited to the set described above.

Lookup Response:

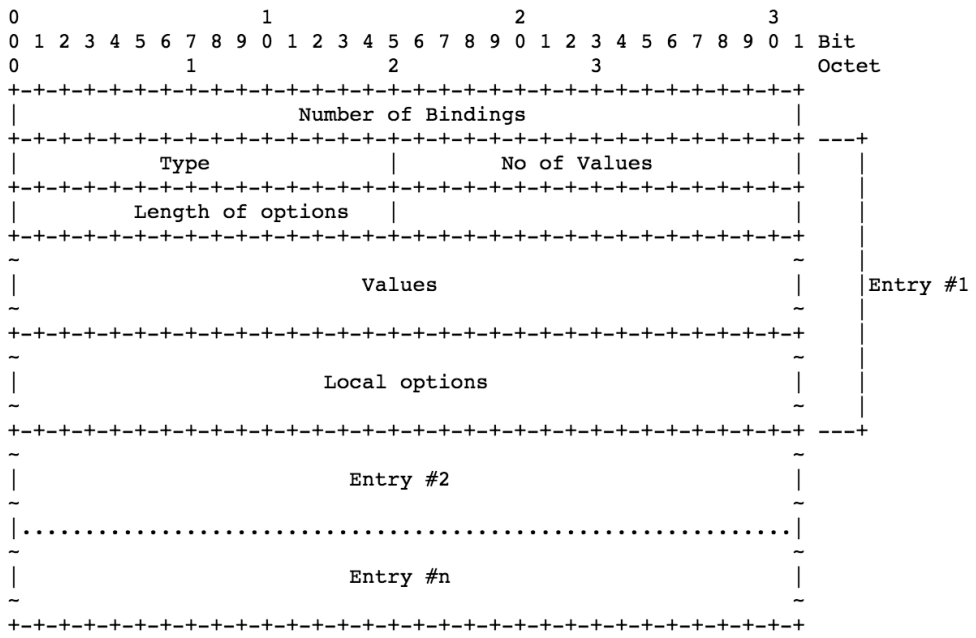


Figure A.7: Lookup Response

- GUID - The GUID value being inserted/updated.
- Number of Bindings - Number of bindings in the response
- Entry - Binding of a type
- Type - Type of the binding
- Number of values - Number of values of the type
- Length of options - Length of local options.
- Values - The values of the type. Parsing will be according to the type (e.g. Network Address).
- Local Options - The options associated with the values

Insert/Update Response: The payload portion of Insert/Update response messages is empty.