SIMULATION-BASED ROUTING PROTOCOLS ANALYSES

A Thesis
Presented to
The Academic Faculty

by

Talal Mohamed Jaafar

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2007

SIMULATION-BASED ROUTING PROTOCOLS ANALYSES

DEDICATION

To my wife Angie and to my parents for their encouragement and support.

To my uncles Mohamed and Hussein Mereby for their unconditional support and advising for the last nine years.

PREFACE

ACKNOWLEDGEMENTS

In the name of Allah, the Most Beneficent and the Most Merciful.

By Allah's will the completion of this work has become a reality. I start by thanking Allah for assisting and guiding me in this work. I thank Him for giving me the confidence and patience to finish this work.

I thank Dr. George Riley for being my advisor and guide in the PhD program. His continuous efforts and support were a big contribution to this work. I would like to thank all the MANIACS members (labmates) for their support and friendship. I like to offer special thanks to Dr. ElMoustapha Ould-Ahmed-Vall for his feedback on my thesis.

I would like to thank my beloved wife Angie for her support and commitment in the last year of this work. I woulk like to thank my parents for their unconditional support throughout the years.

Finally, I am sure that there are alot of friends/relatives that I did not mention, but their friendships are far to be forgotten.

TABLE OF CONTENTS

DEDICATION iii

PREFACE iv

ACKNOWLEDGEMENTS v

LIST OF TABLES ix

LIST OF FIGURES x

SUMMARY xi

I INTRODUCTION 1

1.1 Routing Overview 2

1.2 Routing Protocols Simulations 4

1.3 Thesis Contributions 5

1.3.1 Ghost Nodes: An Enabling Technique for Distributed Network Simulations 5

1.3.2 BGP-Anycast Routing Simulation Analysis 6

1.3.3 EIGRP Simulation Model and Seamless Mobility Using Route Updates 6

1.4 Dissertation Outline 7

II GHOST NODES: AN ENABLING TECHNIQUE FOR DISTRIBUTED NETWORK SIMULATIONS 9

2.1 Motivation 9

2.2 Distributed Space-Parallel Network Simulations 10

2.3 Network Simulators 18

2.4 Ghost Nodes in *GTNetS* 19

2.5 Experimental Results 30

2.6 Conclusion 34

III BGP-ANYCAST ROUTING SIMULATION ANALYSIS 36

3.1 Motivation 36

3.2 Background 38
3.2.1 BGP 39
3.2.2 Anycast Routing 41
3.3 Measurement Studies 43
3.4 Simulations 45
3.4.1 GTNetS BGP++ 46
3.4.2 Simulating AS Topology 46
3.4.3 Simulating DNS Client/Servers 46
3.4.4 Distributed Simulations 47
3.4.5 Failures 47
3.4.6 Measurement Methodology 48
3.5 Experimental Results 48
3.5.1 Tier-1 topology Simulation 49
3.5.2 Tier-1,Tier-2 Topology Simulation 50
3.6 Conclusion 56
IV EIGRP SIMULATION MODEL AND SEAMLESS MOBILITY USING ROUTE UPDATES 57
4.1 Motivation 57
4.1.1 EIGRP 58
4.1.2 Seamless Mobility 58
4.2 Related Work 62
4.2.1 EIGRP Studies 62
4.2.2 Host-Mobility Studies 62
4.3 Simulation Framework 64
4.3.1 EIGRP 64
4.3.2 Wireless Handoff 65
4.4 Experimental Results 67
4.5 Conclusion 78

V CONCLUSIONS AND FUTURE WORK 79

5.1 Ghost Node: An Enabling Technique for Distributed Network Simulations 79

5.2 BGP-Anycast Routing Simulation Analysis 81

5.3 EIGRP Simulation Model and Seamless Mobility Using Route Updates 82

REFERENCES 83

VITA 88

LIST OF TABLES

1 Star Topology Memory Usage 31

2 MILNet Memory Usage 33

3 BGP Performance 54

4 EIGRP Overhead During a Mobility Speed of 4.4ft/sec 77

LIST OF FIGURES

1 Simple Space–Parallel Topology 11
2 Difficult Space–Parallel Topology 12
3 Simple Sequential Script 20
4 Distributed Simulation Script 21
5 Node Implementation 24
6 Adding Applications 27
7 Ghost Node Topology 29
8 Simple Star/Ring Topology 30
9 Milnet Topology 32
10 Typical Anycast Setup 40
11 Hierarchical Anycast Setup 42
12 A Geoplot Visualization of Tier-1 Simulated Topology 45
13 Load Distribution of F-Root Servers (44 nodes) 49
14 Load Distribution of F-Root Servers (5476 nodes) 51
15 Load Distribution of J-Root Servers (5476 nodes) 52
16 DNS Response Time without Failures (5476 nodes) 54
17 DNS Response Time with Topology Failures (5476 nodes) 55
18 Mobile Node Operation 59
19 Mobile IP Operation 60
20 Campus Buildings Layout 68
21 Campus Building Network 68
22 EIGRP Convergence Time for 300ft Radio 72
23 EIGRP Convergence Time for 200ft Radio 73
24 EIGRP Convergence Time with Data Flows 74
25 EIGRP Convergence Time with No Data Flows 75
26 TCP Performance for a MS 76
27 TCP Performance for a MS with On-campus Connection 76

SUMMARY

The Internet is a global communication network, organized into regions called Autonomous Systems (ASs) that are interconnected. An AS is typically an Internet Service Provider (ISP), a corporate, or a university network consisting of a group of routers under the control of a single administrative entity. The routing infrastructure of the Internet is vast and is governed by two types of routing protocols: interior (i.e. OSPF, EIGRP), used within an AS, and exterior (i.e. BGP), used to connect ASs together.

These routing protocols, coupled with routing policies, pose significant challenges in understanding their performance and behavior. They are sophisticated distributed algorithms, and are deployed in medium to large-scale networks. Studying real time behavior of these protocols is either unfeasible or limited to data extracted through probes from few locations, which may not characterize the performance of these protocols throughout the Internet.

This research increases the knowledge on some routing protocols by contributing an enabling technique for large-scale network simulation, and simulation testbeds to analyze two routing protocols.

A new approach to federated network simulations that eases the burdens on the simulation developer in creating space-parallel simulations is presented. Difficulties that arise from the need for global topology knowledge when forwarding simulated packets between federates is overcome by utilizing a topology partitioning methodology that uses Ghost Nodes. A ghost node is a simulator object in a federate that

represents a simulated network node that is spatially assigned to some other federate, and thus that other federate is responsible for maintaining state information associated with that node. However, ghost nodes do retain topology connectivity information with other nodes, allowing all federates in a space-parallel simulation to obtain global information concerning the network topology. Experimental results show that the memory overhead associated with ghosts is minimal relative to the overall memory footprint of the simulation.

The second contribution of this thesis is a detailed simulated Anycast testbed. In this project, we implemented IP Anycasting in BGP++ which is a detailed BGP simulator based on Georgia Tech Network Simulator (GTNetS). The simulator supports longest-prefix-match routing based on routes discovered through BGP. We use simulations to study the advantages of using Anycast for DNS root servers, and to assess the impact of topology failures on the performance of BGP during Anycast deployment. We employ topology data available from RouteViews project and from CAIDA (Cooperative Association for Internet Data Analysis) data sets to build realistic topologies to simulate the supporting Internet backbone topology. The experimental results show that Anycast indeed provides higher availability and smaller user latency for DNS requests. No load balancing was observed using the local-global Anycast deployment mode. In addition, the BGP churn associated with a topology failure is reduced when Anycast is deployed.

The third contribution of this thesis is a scalable EIGRP simulation model and a new approach for host mobility within an AS. In this effort, we developed a detailed simulation model of EIGRP, and we used it to evaluate EIGRP performance under a very dynamic network. Also, we discuss an approach for seamless mobility and continuous connectivity for users of mobile wireless devices as they move within an AS. The solution is to allow end systems to retain a fixed IP address as those systems move across subnet boundaries, and to use route advertisement updates (by EIGRP)

to inform routers of new or revised routes to reach the mobile hosts as they migrate. Simulation results show the ability of EIGRP to update routing tables in a timely fashion, usually within a single TCP timeout period.

CHAPTER I

INTRODUCTION

The Internet infrastructure is composed of Autonomous Systems (ASs) that are interconnected together. These ASs exchange routing information through routing protocols. Common protcols are such as Open Shortest Path First (OSPF) [38], Enhanced Interior Gateway Routing Protocol (EIGRP) [3], and Border Gateway Protocol (BGP) [45]. These protocols provide the algorithms and communication protocols for determining routes to every destination. In addition to routing protocols, there are a number of routing policies and services that enhance routing in the Internet. IP Anycast [41], for example, is an internetwork service that allows multiple hosts to be configured to accept traffic on a single IP address. Mobile IP [44] is a service that allows transparent routing in the Internet of IP packets to mobile nodes. However, the interactions of these policies and services with other elements of the Internet infrastructure is not well understood.

This research provides detailed simulation analyses of some routing protocols and services. The simulated IP Anycast and EIGRP testbeds offer new capabilities for future research. They are implemented in Georgia Tech Network Simulator (GTNetS) [46]. GTNetS features a detailed and scalable model of BGP and features a unique technique that simplifies simulations of large-scale complex network topologies.

The rest of this chapter is organized as follows. An overview of Internet routing is discussed in Section 1.1. Section 1.2 motivates the need for routing protocols simulation. Thesis contributions are introduced in Section 1.3. Section 1.4 provides the outline of the thesis.

1.1 Routing Overview

The Internet is a global communication network that is organized into regions called Autonomous Systems (ASs). An AS is typically an Internet Service Provider (ISP), a corporate, or a university network that consists of a group of routers under the control of a single administrative entity. Each AS constitutes a distinct routing domain. ASs need to exchange routing information in order to effectively forward packets to remote destinations. Routing is the process of moving packets from one network to another, and routing protocols provide the algorithms and communication methods for determining routes to every destination. The routing infrastructure of the Internet is vast and is governed by two types of routing protocols: intra-domain and inter-domain.

Intra-domain routing protocols, also known as Interior Gateway Protocols(IGP), enable routers within an AS to communicate with each other. There are a number of common IGP protocols such as Open Shortest Path First (OSPF), Routing Information Protocol (RIP), and Enhanced Interior Gateway Routing Protocol (EIGRP). The purpose of these protocols is to ensure that the internal routers have a coherent and up-to-date view of how to reach each remote destination within the AS.

Reaching a remote destination across ASs is accomplished through inter-domain protocols, also known as Exterior Gateway Protocols (EGP). The details of how the network is configured or what happens within an AS are hidden from other ASs, giving the AS the freedom to choose any IGP protocol. However, to connect ASs together reliably, ASs have to run the same EGP to exchange reachability information between two routers in a network of ASs. The only EGP protocol in use today is BGP, and is considered the glue that is interconnecting all of the ASs together to form a unified communication infrastructure. Within an AS, IGP and EGP routers communicate in order to exchange information about internal and remote destinations, forming a global network of ASs.

In addition to routing protocols, there are a number of routing policies and services that play a key role in the Internet infrastructure. For example, IP Anycasting is an internetwork service that allows multiple hosts (i.e. servers) to be configured to accept traffic on a single IP address. The motivation for Anycast service is to simplify for the user the process of finding the closest server for a particular service. In this context “closer” means with respect to network metrics and may not be necessary the closest one geographically.

IP Anycasting is easy to implement and has recently been deployed by many operators. Current global-scale IP Anycasting deployment is the anycasting of Domain Name System (DNS) root servers [23, 2], AS-112 servers [1], and Distributed Denial of Service (DDoS) sinkholes [21]. The discovery of the Anycast servers is done through BGP routing table updates. However, using a shared /24 prefix address advertisement, the effects of IP Anycast technique on other elements of the Internet infrastructure (BGP) is not well understood.

Mobile IP is another routing service that allows transparent routing in the Internet of IP packets to mobile nodes. In IPv4, a point of attachment to the Internet is uniquely identified by its IP address. When a mobile device moves outside the network identified by its IP address, all the packets destined for that node are lost. In the Mobile IP approach, as the mobile host moves away from its home network, it gets associated with a different IP address that has the subnet prefix of the new (foreign) link. Then, it binds this new address with a router (agent) at the home network. Later, all the packets destined to the mobile host are routed through its home network to the foreign network. However, this service and other mobility approaches have different deployment requirements.

Modeling and understanding the interactions of these services with the routing protocols are of central interest in this research.

1.2 Routing Protocols Simulations

The interior/exterior routing protocols are sophisticated distributed algorithms. A deep understanding of their performance is very complex as they are deployed in medium to large-scale networks. Reliable evaluation tools are needed for testing current routing protocols or to assess proposed enhancements and new architectures. Current evaluation tools are of three types: theoretical analysis, laboratory testbeds, and simulation experiments.

Theoretical analysis works well for simple protocols. However, as the protocol of study gets complex, the model abstraction for mathematical analysis becomes very hard. For example, the inter-domain routing protocol BGP cannot be analyzed in a mathematical model as it is too complex to formulate mathematically. Its implementation is discussed in many RFCs.

Laboratory testbeds are usually small-scale prototypes. They are unable to perform experiments at a meaningful scale because of the need for extensive resources, and the inability to develop an experimental setup that reflects the commercial nature of the Internet. Therefore, the laboratory experiments produce intitial results, and should not be used as basis for large-scale deployment because they may not reveal the limitations of the protcol of interest if deployed at large scale.

Simulation has become the method of choice for many networking research problems. As new protocols are designed and tested, computer based simulations are used to validate the correctness of the new protocol, and are used to measure the performance of the new protocol under a variety of experimental conditions. Current network simulators such as ns-2 [35], OPNet [6], SSFNet [10], and GTNetS are common platforms for computer network research. However, as the size and capacity of modern networks have increased, the ability to simulate such networks has decreased. Simulation of large-scale network is very difficult, if not impossible, due to the excessive requirements of both memory and CPU time. Distributed network simulations have been implemented in few simulators such as Parallel/Distributed ns (pdns) [50], Dartmouth SSF (DASSF) [31], and GTNetS. However, insuring correct packet forwarding between the federates is still a difficult problem. In a space–parallel network simulation the model for the entire simulated network is divided logically into k sub–models, where k is the number of federates in the distributed simulation. With this approach, each federate is responsible for approximately $1/k^{th}$ of the entire topology, and instantiates simulation objects to represent its own portion of the network. Since a given federate has no responsibility for the remaining $(k-1)/k$ portion of the network, no simulation objects are created and thus the federate has no knowledge of the remaining topology. Simulating large-scale networks and analyzing routing protocols/services is a central part of this dissertation.

1.3 Thesis Contributions

In this Section, we summarize the contributions of this dissertation.

1.3.1 Ghost Nodes: An Enabling Technique for Distributed Network Simulations

First, we introduce a new approach to federated network simulations that simplifies space-parallel simulations. Previous approaches have difficulties that arise from the need for global topology knowledge when forwarding simulated packets between federates. In all but the simplest cases, proper packet forwarding decisions between federates requires routing tables of size $O(mn)$ where m is the number of nodes modeled in a particular federate, and n is the total number of network nodes in the entire topology. Further, the benefits of the well–known NIx-Vector [48] routing approach cannot be fully achieved without global knowledge of the overall topology.

We overcome these difficulties by utilizing a topology partitioning methodology that uses *Ghost Nodes*. A *ghost node* is a simulator object in a federate that represents a simulated network node that is spatially assigned to some other federate, and thus that other federate is responsible for maintaining state information associated with that node. However, ghost nodes do retain topology connectivity information with other nodes, allowing all federates in a space-parallel simulation to obtain global information concerning the network topology. We show experimentaly that the memory overhead associated with ghosts is minimal relative to the overall memory footprint of the simulation.

1.3.2 BGP-Anycast Routing Simulation Analysis

In this part of the research, we implement IP Anycasting in BGP++ [14] which is a detailed BGP simulator based on Georgia Tech Network Simulator (GTNetS). The motivation for this research is that an increasing number of DNS Root Server operators are using IP Anycasting techniques to improve availability and load balancing. This setup might be susceptible to a wide variety of failures, which most measurement studies may not be able to capture.

We use detailed BGP simulations to develop a simulated Anycast testbed. This testbed is used to study the effect of IP Anycasting on BGP, mainly BGP convergence time and BGP churn (exchanged update messages). We replicate a real world topology (Tier-1 and Tier-2 ASs) in a simulated environment, and analyze the impact of failures on DNS and BGP infrastructure with Anycasting.

We find indeed that Anycast provides higher availability and better user latency for DNS queries than using a single DNS server. We also simulated all 3 modes of Anycast deployment, discussed later.

1.3.3 EIGRP Simulation Model and Seamless Mobility Using Route Updates

The last part of the research is two-fold. The first goal is to evaluate the Enhanced Interior Gateway Routing Protocol *EIGRP* via packet simulations. To this end, we developed a detailed simulation model of EIGRP (publicly available), and we used it to evaluate EIGRP performance under a very dynamic network.

Another part of this research is the introduction of a new approach to supporting host mobility within an AS. As wireless devices move across coverage boundaries for a given access point, present mobility solutions require that the device be assigned a new IP address within the address range assigned to the new access point. This address reassignment leads to a number of difficulties for applications requiring uninterrupted connectivity, such as peer–to–peer file transfers, real–time stock quotes, and streaming multimedia. Proposals to enable continuous connectivity in the presence of mobility exist for both IPV4 [44] and IPV6 [12], although neither have seen widespread deployment. We discuss an approach for seamless mobility and continuous connectivity for users of mobile wireless devices as they move within an autonomous system. Since interior routing protocols such as EIGRP are well equipped to adapt to routing changes within a subnetwork, we find that indeed, mobile wireless devices can maintain continuous connectivity across access point handoffs while at the same time maintaining a single IP address.

We present simulation results showing the ability of EIGRP to update routing tables in a timely fashion, usually within a single TCP timeout period. The fast convergence of EIGRP enables the support of applications requiring uninterrupted connectivity.

1.4 Dissertation Outline

The rest of the thesis is organized as follows. Chapter 2 discusses a new technique to federated network simulations. This technique is used later on in the research to enable large-scale simulation. Chapter 3 provides a simulation analysis of IP Anycast. This is done via the extension of a detailed BGP simulator to include IP Anycast. In Chapter 4 we describe our implementation efforts to provide a simulation model of EIGRP, and a new approach for mobile computing. Chapter 5 outlines the conclusions drawn from the research throughout this doctoral thesis.

CHAPTER II

GHOST NODES: AN ENABLING TECHNIQUE FOR DISTRIBUTED NETWORK SIMULATIONS

This chapter presents the *Ghost Node Technique*, a new approach to federated network simulations that eases the burdens on the simulation developer in creating space-parallel simulations. It is an enabling technique for large-scale simulations where there is a need for global topology knowledge when forwarding simulated packets between federates (*Federate* is synonymous with *simulator instance* and *simulator*).

Section 2.1 discusses the space-parallel approach for distributed simulation and motivates the need for the new technique. The difficulties associated with the existing distributed-simulation techniques are discussed in Section 2.2. Current distributed network simulators are described in Section 2.3. Section 2.4 gives the basic design of our *Georgia Tech Network Simulator* with emphasis on the ghost node implementation. Section 2.5 presents memory usage statistics comparing the ghost node approach to traditional approaches. Finally, Section 2.6 describes conclusions from this work.

2.1 Motivation

One approach to creating network simulation models for large–scale topologies is to use a space–parallel partitioning methodology, coupled with distributed simulation methods. In a space–parallel network simulation the model for the entire simulated network is divided logically into k sub–models, where k is the number of federates in the distributed simulation. With this approach, each federate is responsible for approximately $1/k^{th}$ of the entire topology, and instantiates simulation objects to represent its own portion of the network. Since a given federate has no responsibility for the remaining $(k-1)/k$ portion of the network, no simulation objects are created and thus the federate has no knowledge of the remaining topology. This approach is relatively easy to implement, and is the method used in space–parallel distributed network simulators such as *Parallel/Distributed ns* (*pdns*) [50, 49] and the *Georgia Tech Network Simulator* (*GTNetS*) [46, 47].

Further, this method has very good scalability, since each federate need only be concerned with its assigned network elements, and thus only need to allocate memory for a fraction of the entire set of network elements. However, as we shall show this approach introduces a number of difficulties that must be addressed in order to insure correct packet forwarding between the federates.

2.2 Distributed Space-Parallel Network Simulations

To illustrate the concepts and issues regarding space–parallel network simulations, we present two simple examples. Consider the simple topology shown in Figure 1, consisting of four subnetworks. Each subnetwork has four hosts, two intermediate routers, and one gateway router. Each of the four gateway routers is connected to each of the other three gateway routers, forming a fully connected mesh. All of the simulated nodes for a subnetwork have a common 24 bit network address prefix, such as 192.168.0.x for subnetwork 0 as shown.

Now suppose that, due to resource constraints in our simulation environment, we cannot model more than seven network nodes in a given federate without running out of memory on the computing platform in use. Clearly, such limited resources are not realistic, but are used here for illustrative purposes. Even with these resource constraints, we can still create a simulation of the four subnetwork topology by using space–parallel distributed simulation.

We create four different federates, each running on a separate hardware platform.

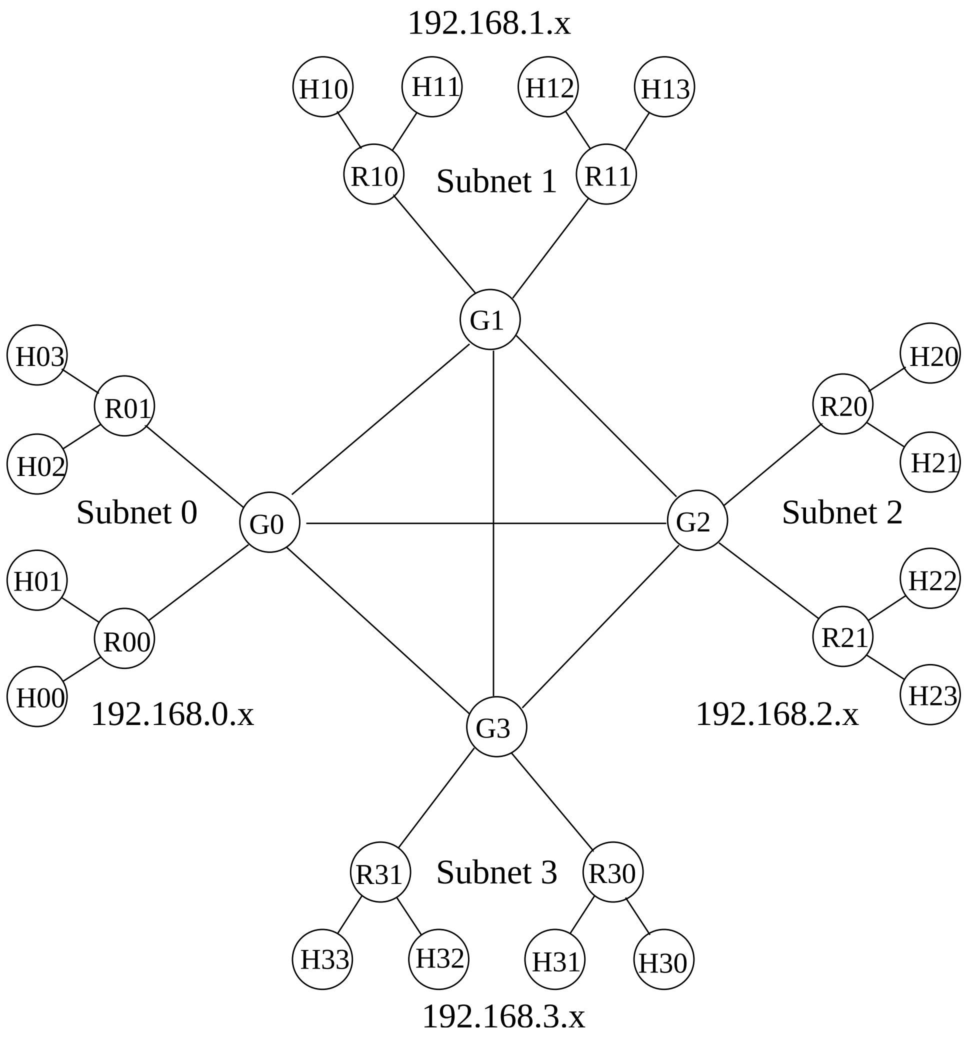

Figure 1: Simple Space–Parallel Topology

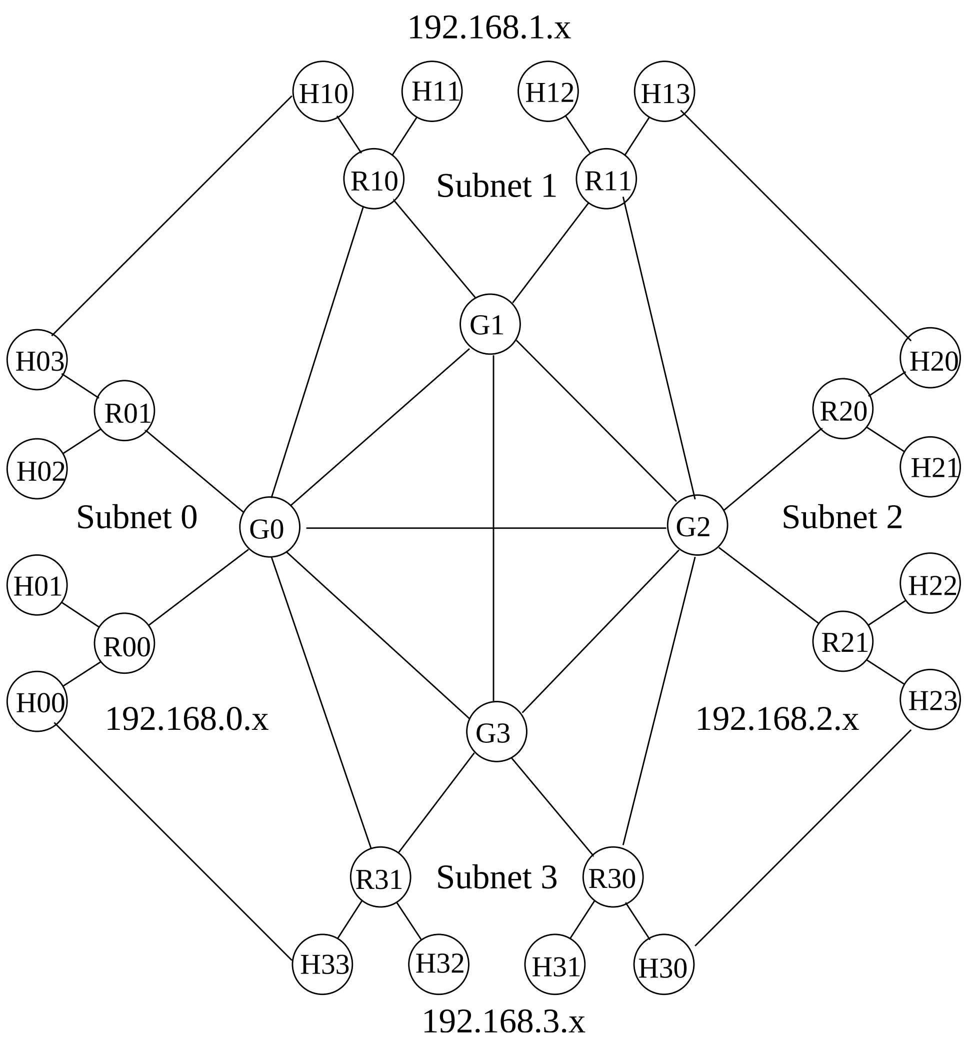

Figure 2: Difficult Space–Parallel Topology

Each of the four federates instantiates models for the seven nodes in a single subnetwork. For example, federate 0 would model the seven nodes in subnetwork 0, federate 1 would model the seven nodes in subnetwork 1, etc. The simulation environment must have some way to describe simulated links between federates (such as the link from $G0$ to $G1$). Links that span federate boundaries are called *remote links*, or *rlinks.* Issues such as time management and event distribution between federates can easily be solved using one of several available *Runtime Infrastructure* packages, such as the Georgia Tech *Federated Simulation Developers Kit* (*FDK*) [17], or the *DMSO RTI* [4]. The end result is that we are able to model twenty–eight network nodes, using four instances of a simulator that can only model seven nodes each, using the space–parallel methodology. The following paragraphs discuss some of the issues that arise when determining correct packet routing in this type of simulation.

Default Routes. In this simple example, the routing of packets between federates is nearly trivial. Suppose host $H00$ sends a packet to host $H23$. Federate 0 can easily determine that the destination node ($H23$) is not modeled locally[1]. Since in this example the destination node is defined and managed on federate 2, federate 0 must make a routing decision based on incomplete knowledge of the overall topology. In this case however, gateway node $G0$ is the only way that packets can leave or enter subnet 0 (and hence federate 0), $H00$ simply forwards the packet to node $G0$ (through node $R00$) for further processing. In *pdns* and *GTNetS* this is known as a *Default Route*, and works well when there is a single simulated node responsible for all packets entering and leaving the portion of the network topology mapped to that federate.

[1]Details of how this is done are dependent on the implementation, and are not important for this discussion

Inter–Federate Route Aggregation. Route aggregation is a method used in Internet routers to reduce routing table size. If all of the routing table entries for a set of destination addresses are identical, and the destination set has a common *address prefix*, then this entire set of routes can be stored with a single entry.

Using route aggregation, once the packet arrives at gateway node $G0$, the routing decision is again easy and requires little memory. Although gateway node $G0$ has three *rlinks*, the routing decision can easily be made based on the destination *IP Address*. The *rlink* from $G0$ to $G1$ is the correct path for any *IP Address* starting with `192.168.1`, since all nodes with that prefix are defined in federate 1. Thus, using route aggregation, only three routing table entries (one for each *rlink*) are sufficient for federate 0 to make correct routing decisions in all cases. Both *pdns* and *GTNetS* provide commands to specify this type of aggregated routing entries for the remote links. In this simple case, assuming the use of NIx-Vector routing within each federate, we need routing state in each of the gateway nodes of size $O(f)$, where f is the number of federates in the distributed simulation.

A More Complicated Example. It appears from the above discussion that the problem of inter–federate routing in space–parallel network simulations is easily solved. However, consider the slightly modified topology shown in Figure 2. This topology is nearly identical to the previous example, except the addition of two more inter–subnet links, connecting certain hosts to hosts in a neighboring subnet, and two extra links from gateway nodes $G0$ and $G2$ to neighboring interior routers. With this topology, the simplifying assumptions observed for the previous example no longer hold, and inter–federate routing of packets becomes much more difficult to manage.

First, the notion of the *default route*, indicating that all packets not destined to a local *IP Address* should be routed to a common gateway, can no longer be used. Thus, each node in each federate will need a routing table (potentially with $O(n)$ entries, where n is the total number of nodes in the simulation) to select which inter–federate gateway node is the best path to remote nodes.

Secondly, the clean and simple route aggregation method that worked nicely on the previous example may no longer work. Now, gateway node $G0$ has four *rlink*s, two to subnet 1 and two to subnet 3. The assignment of the *IP Addresses* to nodes in subnets 1 and 3 will affect how well the route aggregation will work for the *rlink* routing entries. In the best case, we can still use a single aggregate entry for each *rlink*, but in the worst case we need routing entries for *every node* in subnets 1 and 3 in the routing table for $G0$. The end result of both of these problems is that we still need routing state of size $O(mn)$, where m is the number of nodes managed in each federate, and n is the total number of nodes in the global topology. We point out that the $O(mn)$ memory requirement is the worst case, and in practice one can still expect some saving from route aggregation.

Using NIx-Vector Routing. An efficient source–routing methodology called NIx-Vector routing is discussed in [48]. With this method, a route between a source and a destination is calculated only when needed, and is cached at the source for later re–use. Further, the calculated path from the source to the destination is stored *in the packet* using the compact NIx-Vector format, that allows intermediate routing decisions to be made without the use of routing tables. However, this approach requires global information concerning the topology from the source to the destination. Clearly, in the space–parallel methodology, this global topology knowledge is not available. However, we can provide additional routing information at the *rlink*s that allows a partial NIx-Vector to be calculated within a federate.

Suppose host $H00$ is sending a packet to host $H13$. Since host $H13$ is managed by federate 1, federate 0 lacks global knowledge of the topology to calculate a NIx-Vector to $H13$. However, if each *rlink* in federate 0 has routing information specifying *those* *addresses that are reachable from this link*, and *the number of hops to reach each destination*, a NIx-Vector that routes the packet to the appropriate gateway can be calculated using a modified *Breadth First Search* (*BFS*) algorithm. In our example, the *rlink* from $G0$ to $G1$ will have a routing entry indicating it can reach $H13$ in three hops, and the *rlink* from $H03$ to $H10$ will indicate it can reach $H13$ in five hops. The modified *BFS* algorithm will calculate that the shortest path from $H00$ to $H13$ should use the *rlink* from $G0$ to $G1$, and calculates a NIx-Vector from $H00$ to $G1$ (the last hop is the *rlink* from $G0$ to $G1$).

This method has some of the benefits of NIx-Vector routing, in that no routing tables are needed at any node excepting those with *rlink*s to other federates. The memory requirements are still $O(kn)$ (k is the number of inter–federate *rlink*s, and n is the total number of nodes in the simulation). Further, in all cases except the simplest topologies, the calculation of these inter–federate routes can be time consuming. For example, we computed inter–federate routes for the million–node *MILNET* topology defined by Liljenstam et. al[29]. The off–line computation took 4 hours on a 866Mhz desktop processor, resulted in more than 5 million inter–federate routes, and required over 500MB of disk space to store the computed routing information.

Using Pre–Computed Routes. A simple approach to intra–federate routing is to use *Pre–Computed Routes.* In this approach, an off–line program creates a simplified and memory efficient representation of the topology. Once this complete topology model is created, routing information can be computed for all nodes, giving paths to all other nodes. An advantage of this approach is that the time–consuming route computation step can be performed once, and used repeatedly in the simulation runs. The obvious disadvantage of this method is the $O(n^2)$ memory requirements for the all–pairs routing tables. For example, if the entire topology consists of one million nodes, the resulting pre–computed routing tables would consist of 10^{12} entries, consuming unreasonably large amounts of disk space and memory. This approach is used by the distributed memory Dartmouth *SSF* (*DaSSF*) simulator[31] described in [30].

At the same time, one can argue that if combined with the information on the traffic designation in simulations, the method using precomputed routes actually can significantly reduce memory usage by encoding only those routes required by the simulation. However when using large topologies, the source and destination pairs are often chosen randomly (for example as in the web–browsing model of *GTNetS*), and thus route precomputation is not feasible without a complete all–pairs computation (or at least all pairs that *might* make a random connection).

Using Routing Protocols. Another approach to inter–federate and intra–federate routing in network simulations is the use of simulated *Routing Protocols* within the simulation. For example, we could include a model of the widely–used *Border Gateway Protocol* (*BGP*) on each node with inter–subnet connections. In the example in Figure 2, this would be nodes $G0$, $G1$, $G2$, $G3$, $H00$, $H03$, $H10$, $R10$, $H13$, $H20$, $H23$, $H30$, $R31$, and $H33$. Further, we could use an *Interior Routing Protocol*, such as *EIGRP*[3, 19] or *OSPF*[39] on interior routers within a subnetwork (such as nodes $R00$, $R01$, $R11$, $R20$, $R21$, and $R30$ in our example). This approach is used by the *SSFNet* simulator[11, 9], and results in an easy to use space–parallel simulation. Additionally, this method inherently deals with dynamic topology changes, such as reacting to link or node failures. When creating the simulated topology, the user need not be concerned about routing information, since the routing protocols will compute the best routes using routing message exchanges between federates. Further, these routing protocols use route aggregation techniques to reduce the size of the resulting routing tables as much as possible. However, this approach still requires simulator memory to hold the routing tables calculated by the routing protocol, which can be excessive.

Our solution to these difficulties is to introduce the notion of a *Ghost Node*. A ghost node is a simulator object that acts as a placeholder for nodes that are assigned to other federates. The ghost node object has none of the complex and memory intensive state needed for real nodes (such as queues, routing tables, port maps, and applications). Rather, a ghost node simply contains topology connectivity information about links and neighbors. Thus, using ghosts, a federate is afforded a global view of the simulated topology, without the memory overhead of maintaining unneeded state for the ghosts.

2.3 Network Simulators

There are several network simulation tools available that use a space–parallel approach to distributed simulation. However, most of these simulators introduce difficulties when insuring correct packet forwarding between the federates.

Parallel/Distributed ns (*pdns*) by Riley[49, 50] (based on the venerable *ns2*[35] simulator) has used the space–parallel approach from its outset. However, simulating large and highly connected topology is challenging as routes cannot be aggregated.

The *SSFNet* simulator was initially designed for parallel simulation in a multi–threaded shared–memory environment, but has since been adapted by Liu and Nicol[30] to support distributed memory platforms. However, SSFNet makes use of simulated routing protocols to ensure correct packet forwarding, which requires simulator memory to hold the routing tables calculated by the routing protocol, which can be very excessive.

The Dartmouth *SSF* (*DASSF*) simulator[31] also has been adapted for a distributed simulation with a space–parallel methodology. Their approach is to use pre-computed routes, routing information is computed off-line for all nodes, giving paths to all other nodes. However, this approach has an $O(n^2)$ memory requirements.

Wu et. al[55] report some limited success in adapting the commercial *OPNet* simulator[6] to operate in a distributed environment, using a space–parallel approach.

The concept of using limited state objects (ghosts) as place–holders for remote objects is not new. In the Distributed Interactive Simulation (*DIS*) community, tools such as SIMNet[37] often use dead reckoning or other approximation methods to estimate the state of objects that are managed in remote federates. In a battlefield simulation for example, a federate may report the position and velocity of a tank object at a particular point in time. Other federates will maintain the tank's location by assuming a constant velocity until informed otherwise. Ferenci[16] discusses the use of *Proxy Objects* in distributed simulations, which are conceptually similar to our ghosts. However, Ferenci's proxy objects exist primarily to facilitate inter–federate message routing, and do not in fact represent any global state. Additionally, Ferenci discusses his approach in the context of optimistic simulations, where we deal exclusively in the conservative environment. To our knowledge, we are the first to apply the limited–state object method to represent the global topology information in space–parallel network simulations.

2.4 Ghost Nodes in GTNetS

In this Section, we discuss the basic design of the space–parallel distributed simulation support in *GTNetS*, with particular attention to the ghost node approach. A *GTNetS* network simulation is created by writing a *C++* main program that instantiates objects representing the network topology (nodes, links, queues, etc.), and the applications that create data flows, such as web servers, web browsers, and FTP clients. Also each *GTNetS* simulation instantiates a single `Simulator` object that controls the simulation (maintains the pending event list and schedules events).

```
#include "simulator.h"
#include "node.h"
#include "linkp2p.h"

int main()
{ // Simple sequential simulation
  Simulator s; // Sequential simulation
  Node* n1 = new Node(); // Node 1
  Node* n2 = new Node(); // Node 2
  // Define a link object
  Linkp2p link(Rate("1Mb"),
               Time("10ms"));
  // Add the link from n1 to n2
  n1->AddDuplexLink(n2, link,
      IPAddr("192.168.1.1"), Mask(32),
      IPAddr("192.168.1.2"), Mask(32));
```

Figure 3: Simple Sequential Script

GTNetS supports both sequential, single–process simulations as well as space–parallel distributed simulations. The majority of *GTNetS* simulations will use sequential execution, so the distinction between sequential and distributed execution was made as simple as possible. To this end, two versions of the object constructor for the `Simulator` object are provided, one with no parameters and one with a single *Distributed Simulation Identifier* parameter. For sequential simulations, the default constructor without arguments is specified by the user, in which case none of the distributed simulation support functions are called, and the complete topology is assumed present in the single address space. See Figure 3 for a simple code snippet. The remainder of this Section will focus on the distributed simulation methods.

To specify a distributed simulation, the `Simulator` object is instantiated with a single integer argument, assigning an instance identifier to this simulator that is unique within the federated simulation. If there are to be k federates in the distributed simulation, the instance identifier is in the range of $0 \dots (k-1)$. When the

```
#include "simulator.h"
#include "node.h"
#include "linkp2p.h"

int main(int argc, char** argv)
{ // Simple distributed simulation
  // Get instance id from arguments
  int myId = atol(argv[1]);
  Simulator s(myId); // Distributed sim
  // n1 is managed by simulator 0
  Node* n1 = new Node(0); // Node 1
  // n2 is managed by simulator 1
  Node* n2 = new Node(1); // Node 2
  // Define a link object
  Linkp2p link(Rate("1Mb"),
               Time("10ms"));
  // Add the link from n1 to n2
  n1->AddDuplexLink(n2, link,
      IPAddr("192.168.1.1"), Mask(32),
      IPAddr("192.168.1.2"), Mask(32));
```

Figure 4: Distributed Simulation Script

`Simulator` object is constructed in this manner, *GTNetS* will call the necessary distributed simulation support functions in the *Run-Time Infrastructure (RTI)*, such as initialization functions, data distribution, and time management requests. Further, the instance identifier is used to determine if node objects are to be *real* nodes or *ghost* nodes, as discussed in the following paragraphs.

The next action needed in the distributed simulation script is to identify, for every node in the topology, which federate is responsible for that node object. This is accomplished by providing a node object constructor with a single argument that specifies an instance identifier. If the specified instance identifier matches that specified on the `Simulator` object constructor, then this federate is responsible for the node object, and a *real* node object is created. Otherwise, a *ghost* object is created.

See Figure 4 for a simple code snippet showing a distributed simulation instance. Note that the only differences (other than command line argument processing) are the `myId` parameter passed to the `Simulator` constructor, and the single integer arguments passed to the `Node` object constructors. In this simple example, one simulator process is initiated with the command line argument "0", and the second is initiated with the command line argument "1". Notice that when node objects `n1` and `n2` are created, the `Node` constructor is called with the arguments 0 and 1 respectively, indicating that node `n1` is to be modeled on federate 0, and `n2` is to be modeled on federate 1. In federate 0, node `n1` is a *real* node and `n2` is a *ghost*. In federate 1, node `n1` is a *ghost* node and `n2` is *real*.

There are two important points to be seen from this simple example. First, there is little difference from the users' perspective between the sequential and the distributed execution. The only differences are the presence of the *instance id* parameter on the `Simulator` constructor, and the *responsible instance id* parameter on the `Node` constructor. Excepting a few other minor differences discussed later, the remainder of the script is identical. Secondly, each federate runs *exactly the same script*. Using this method, the same *GTNetS* main program can be used for each federate in the distributed simulation, distinguished with command line parameters.

Ghost Node Implementation. From the above discussion, it is clear that in *GTNetS* the `Node` objects come in two flavors, *real* nodes and *ghost* nodes. Equally clear is that the *API* for the two node types (i.e., the set of member functions available to object owners) must be identical or nearly identical. If this were not the case, there would be many conditional checks in the simulation script to take different actions depending on the *real* or *ghost* status of the nodes. Note for example the call to `AddDuplexLink` for `Node` object `n1` in the above example. While the actions taken in this method are different for *real* and *ghost* nodes, the *API* is the same. In fact, all `Node` methods are identical for *real* nodes and *ghost* nodes. Finally, the *ghost* node implementation must be memory efficient, utilizing as little state as possible. If this were not the case, the advantage of exploiting multiple processors to simulate larger networks would be lost, since the entire topology is required on every federate.

We achieve these goals by using a simple one–level method indirection as shown in Figure 5. The basic `Node` object has all the *API* methods needed by *GTNetS* to manage nodes, but only has a single *Implementation Pointer* state variable. This implementation pointer points to an object that is a subclass of class `NodeImpl`. The `NodeImpl` class defines the required set of methods needed for nodes, but only has state common to *ghost* nodes and *real* nodes. The only common state between *ghost* nodes and *real* nodes is the *IP Address* and a vector of *Interfaces*. In this context, the word *Interface* refers to a simulation model of a hardware link interface (such as a *NIC* card) in a router or end system. Finally, there are two classes that are subclasses of `NodeImpl`, namely `NodeReal` and `NodeGhost`. Objects of class `NodeReal` have all the state associated with *real* nodes, such as port maps, routing information, animation size and color, and location information.

When a node is created in a distributed simulation, the `Node` constructor checks whether the system identifier specified in the constructor argument matches that specified in the `Simulator` constructor. If so, this node is real, and a new object of class `NodeReal` is created and pointed to by the implementation pointer. If the system identifiers do not match, the node is a ghost, and a new object of class `NodeGhost` is created.

We mentioned previously that both real and ghost nodes maintain a list of *Interfaces* that model the link interfaces in nodes and routers. This seems at first glance to be inefficient in terms of memory usage, since these interfaces have a substantial amount of state (for example a packet queue) that is not necessary for ghost nodes. We solve this problem by defining two `Interface` subclasses, `InterfaceReal` (which has the state needed to model an interface), and `InterfaceGhost` (which does not). When a new `Interface` object is needed by a node object, real nodes create a real

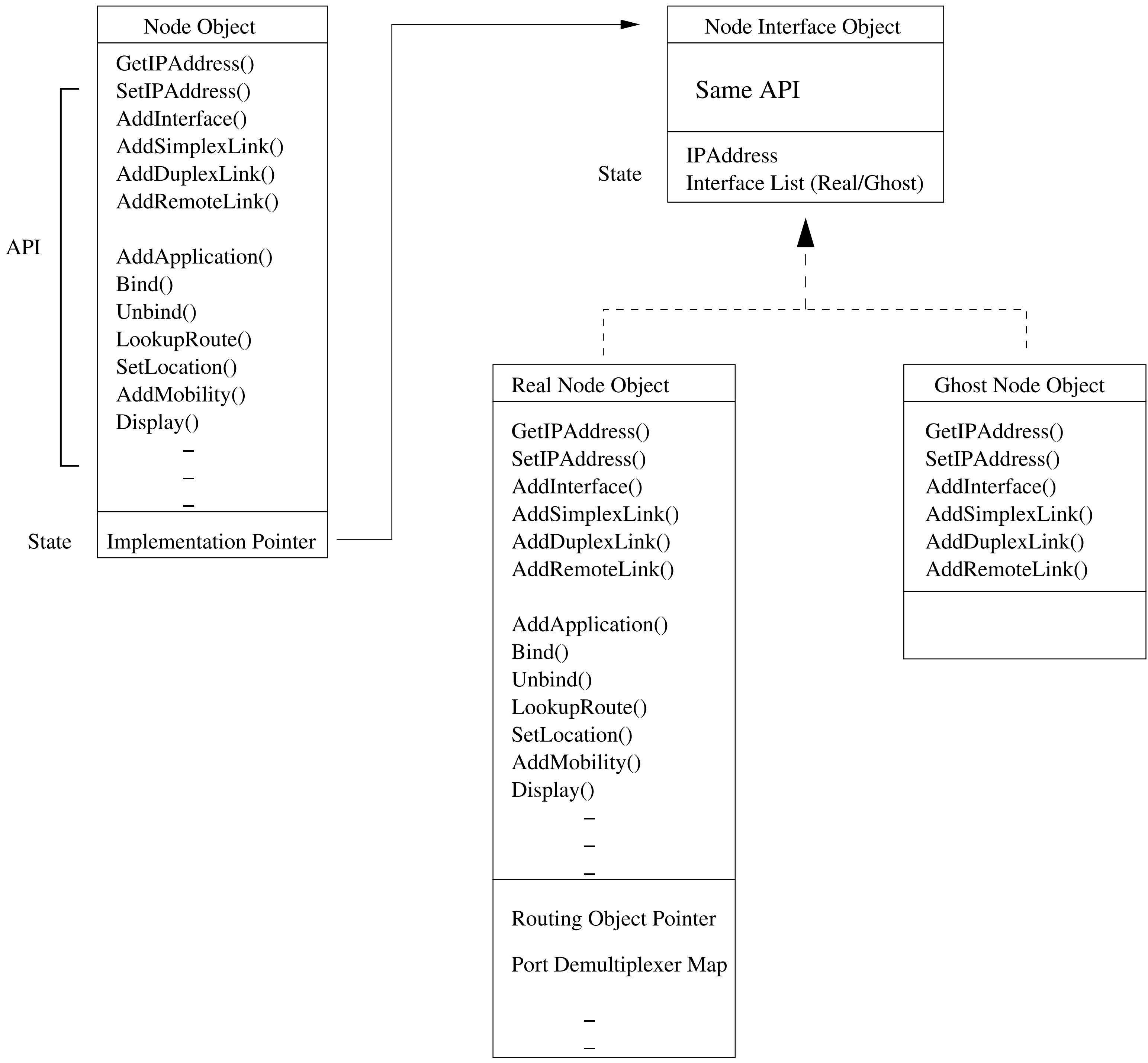

Figure 5: Node Implementation

interface, and ghost nodes create a ghost interface. Similarly, we use real and ghost `Link` objects for the same purpose. The key point is that the *API* is common across real and ghost objects, allowing any owner of these objects to call the defined methods without regard to whether the object is real or ghost.

We did an analysis by inspection of the memory used by real and ghost nodes and their associated state, and confirmed this analysis by using the C++ `sizeof` operator. The total memory cost of a real node is given by `(244 + 266`x`)` bytes and the total memory cost of a ghost node is given by `(40 + 44`x`)` bytes where x is the number of interfaces for that particular node. Using this memory analysis, we can show quantitatively the real vs. ghost memory used as a function of number of federates for a given topology (memory cost for both real and ghost nodes is dependent on the number of interfaces for that node). For instance, using the topology shown in Figure 8, and for a star size of 3500, the memory cost of a real star (all star nodes are real) is 2.716 MB, and the memory cost of a ghost star (all star nodes are ghost) is 0.448 MB. Therefore, running that topology on 8 federates will require a total of 3.136 MB of memory overhead for the ghost nodes. Once the number of federates involved in the simulation is known, the memory cost for the ghost nodes can be computed, and the simulation can be noted as feasible or not based on the memory available at every federate.

Using this technique of real and ghost objects, each federate in the distributed simulation has a complete picture of the simulated topology, and can compute NIx-Vector routing information from any source to any destination. We show in the next Section that the overhead incurred by ghost objects is likely to be small compared to the overall memory footprint of the simulation.

There is one case where the behavior of a ghost node and a real node requires the simulator user to be aware of whether the node is real or not. All of the previous discussion has focused exclusively on the topology generation part of the simulation script. In a network simulation, we also need to define the flow of data between the nodes in the topology. In *GTNetS* this data flow is defined using *Application* objects. *GTNetS* presently has defined application models for thirteen different application behaviors, including web browsers, web servers, Gnutella clients, and others. However, we do not use the concept of ghost applications. Since applications are added to nodes using the `AddApplication` method for node objects, a simpler method is to design ghost nodes to ignore any request to add an application. Since the semantics of the `AddApplication` method are that it returns a pointer to the newly added application object, the design is that ghost nodes simply return a *NULL* pointer instead. The user simulation scripts simply check for a *NULL* return from the `AddApplication` call, and if so skips further application initialization. See Figure 6 for a code snippet illustrating this point. While the script does not directly determine whether an application is being added to a ghost node or a real node, it detects the *NULL* return to differentiate between the actions performed by the two node types.

Consistent Topology View. It is apparent that, for the ghost node approach to work properly, all federates must have a consistent view of the global topology being modeled. While this seems easy to achieve, there are two instances that can cause problems with this requirement.

First is the use of randomly generated topologies, using a tool such as the *Georgia Tech Internet Topology Modeler* (*GTITM*)[57]. In our *GTNetS* simulator, a single *C++* object can represent an arbitrarily large network, generated randomly based on the *GTITM* technique. Thus, different federates could randomly generate different topologies, thereby violating our consistent view constraint. In this case, care must be taken to insure the random number generators are seeded in a deterministic way to insure each federate generates identical random topologies.

The second issue is the modeling of link or node failures in a network. If a

```
int main(int argc, char** argv)
{ // Simple distributed simulation
  // Get instance id from arguments
  int myId = atol(argv[1]);
  Simulator s(myId); // Distributed sim
  // n1 is managed by simulator 0
  Node* n1 = new Node(0); // Node 1
  // n2 is managed by simulator 1
  Node* n2 = new Node(1); // Node 2
  // Add WebServer application to n1
  WebServer* svr = n1->AddApplication(
        TCPServer());
  if (svr) {
    // Application added
    svr->EnableGCache();
  }
  // Add WebBrowser to n2
  WebBrowser* br = n2->AddApplication(
        WebBrowser( ... ));
  if (br) {
    // Added, configure and start
     br->ConcurrentConnections(4);
     br->Start(0.1);
  }
```

Figure 6: Adding Applications

given federate has a *real* node representation of a given network node, and generates a random node failure event, other federates must be made aware of this failure. Although not implemented in our simulator, it is straightforward to use state update events between federates to achieve these notifications. Further, these state update events need *not* be sent between federates with zero simulation time advance, since node and link failures cannot be detected in a network any faster than packets can flow through the network. In a real-life network, topology changes are not propagated instantaneously. In fact, the farther a network element is from the location of topology change, the longer it takes for the topology-change information to be propagated to the element. This is important because it is well known that zero time update events between federates leads to poor performance in conservative distributed simulation environments.

Summary. The design of *GTNetS* leads to an easy to use, low overhead way to manage a space–parallel network simulation. The ghost nodes give the simulation the necessary topology information to calculate NIx-Vector routing information, while at the same time use little memory. Figure 7 shows the difficult topology presented in the previous Section, from the perspective of federate 0. All of the nodes in subnet 0 are real nodes indicated with solid lines, and all other nodes are ghosts, indicated with dotted lines. Further, all links in the other subnets are ghost links (again with dotted lines), excepting those *rlink*s connecting those real nodes in subnet 0 to ghost nodes in other subnets. These cross subnet links are special links, called *RTILinks*, which use the services of the runtime–infrastructure to transfer packets between federates. Finally, a large–scale network simulation should take into account the effects of policy–based routing as defined by the *Border Gateway Protocol* (*BGP*). By offering a global view of the topology, the ghost approach gives a more complete picture of the overall network, and will enable efficient implementation of policy-aware routing decisions.

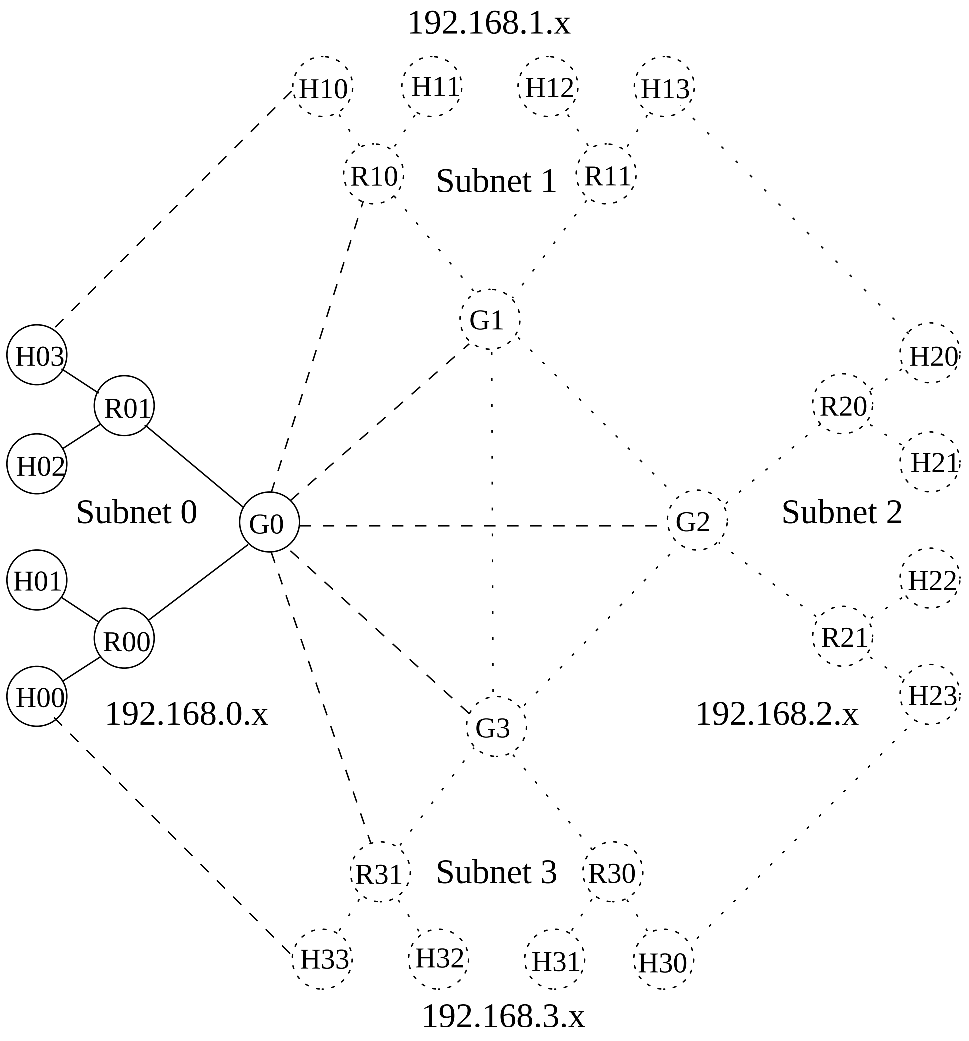

Figure 7: Ghost Node Topology

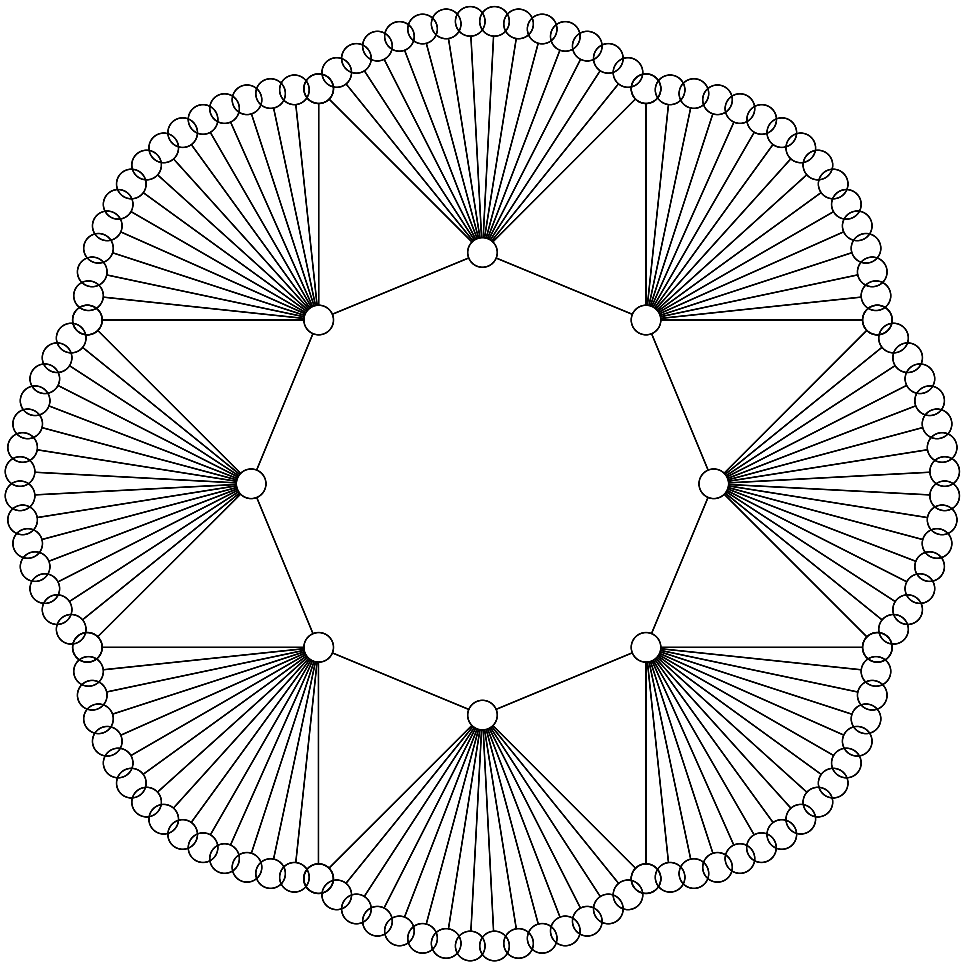

Figure 8: Simple Star/Ring Topology

2.5 Experimental Results

To demonstrate the effectiveness of the ghost node approach, we ran two sets of experiments to measure the memory usage of the space–parallel network simulations, using both the traditional approach with manually specified inter–federate routing and our new ghost node approach. The first set of experiments uses a simple topology similar to that shown in Figure 8. This topology consists of k subnetworks (k is eight in the Figure shown), each with n nodes arranged in a star topology (n is sixteen in the example). Each of the subnetworks is connected to its neighbors, forming a ring. This topology was chosen since it is the best possible case for the traditional approach. Each of the leaf nodes in the star subnetwork can use the simple *default route* method

Table 1: Star Topology Memory Usage

Method	Star Size	Node Count/Federate			Memory	Execution Time
		Real Nodes	Ghost Nodes	Total Nodes	(MB)	(sec)
Traditional	1,501	1,501	0	1,501	45	156
Ghost	1,501	1,501	10,507	12,008	47	150
Traditional	3,501	3,501	0	3,501	53	188
Ghost	3,501	3,501	24,507	28,008	57	152
Traditional	7,501	7,501	0	7,501	68	271
Ghost	7,501	7,501	53,507	60,008	77	150
Traditional	15,001	15,001	0	15,001	95	331
Ghost	15,001	15,001	105,007	120,008	114	228

to reach the single gateway node. At each gateway, the *route aggregation* method can easily and efficiently specify which addresses are reachable on each of the *rlink*s.

The simple topology was run on eight federates, with varying numbers of leaf nodes per subnetwork. The experiments were performed using both the traditional approach and the ghost node approach. One hundred and fifty *TCP* flow endpoints were assigned to leaf nodes, and 1MB transfers were simulated. The memory usage and execution time of each simulation is shown in Table 1. Since in this experiment, all federates model an identical subnetwork, results are only shown for federate zero. As can be seen, the memory footprint for the ghost node approach is only slightly larger than the traditional method. For the 120,008 node case (the largest we performed, 8 connected stars), there were 15,001 real nodes and 105,007 ghost nodes per federate. The ghosts required a total of 19 MB of memory, representing a 20 percent increase. As mentioned before, the number of federates involved in the simulation determines the feasibility of the experiment as the overhead incurred from the ghost nodes can be excessive when the number of federates becomes large.

Interestingly, the overall execution time for the ghost node approach is less than the traditional approach. Using ghosts, we pay a one–time cost for the calculation of the NIx-Vector, but gain an $O(1)$ routing decision at each hop in the path. Without

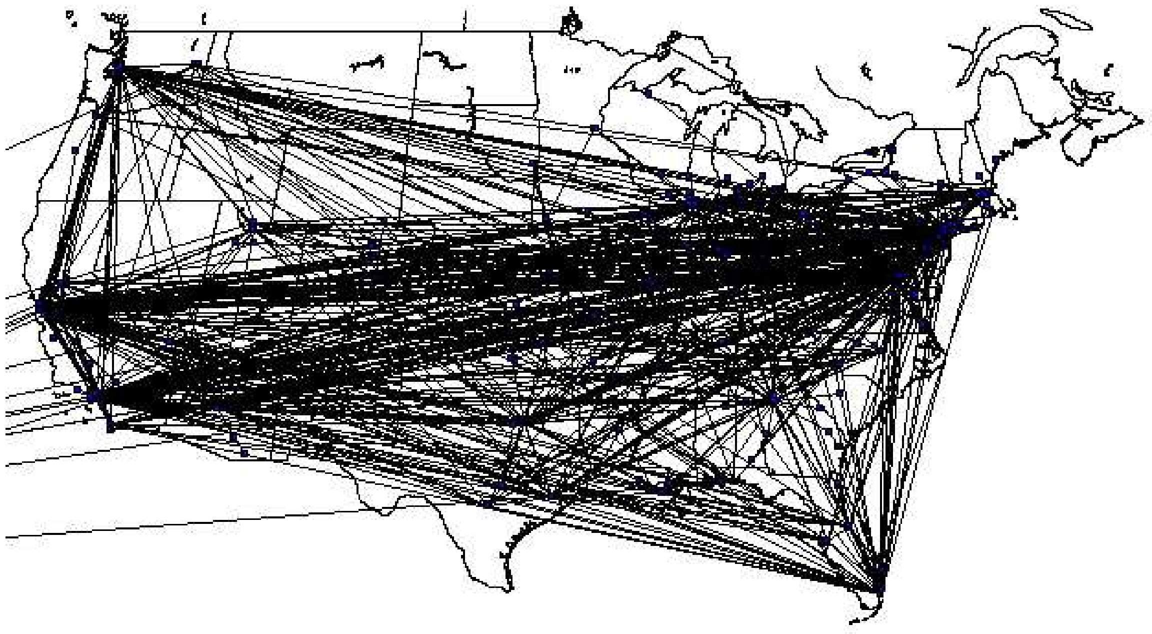

Figure 9: Milnet Topology

NIx-Vectors, the routing at gateway nodes and hubs requires an $O(k)$ computation (k is the number of *IP Address*s assigned to the node) to determine if the packet has arrived at the destination. We are examining a less burdensome way to make this decision, to reduce this overhead to $O(\lg k)$.

The second set of experiments used a large and complex topology known as the *MILNet* defined by Liljenstam et. al[29]. This topology consists of a backbone network containing over three thousand routers and eleven thousand links, that roughly models the backbone network for United States military bases. Connected to the backbone are 163 subnetworks of various sizes from five hundred to nine thousand nodes each. The entire network exceeds one million nodes. A graphical representation of the *MILNet* backbone is shown in Figure 9.

The results from the *MILNet* experiments are shown in Table 2. The *MILNet* topology was divided among 8 federates, with federate zero modeling the high–speed

Table 2: MILNet Memory Usage

Federate	Method	Node Count/Federate			Memory	Execution Time
		Real Nodes	Ghost Nodes	Total Nodes	(MB)	(sec)
0	Traditional	3,070	0	3,070	73	692
	Ghost	3,070	1,095,558	1,098,628	216	299
1	Traditional	181,268	0	181,268	704	699
	Ghost	181,268	917,360	1,098,628	833	299
2	Traditional	144,769	0	144,769	568	696
	Ghost	144,769	953,859	1,098,628	699	299
3	Traditional	141,421	0	141,421	557	695
	Ghost	141,421	957,207	1,098,628	688	299
4	Traditional	150,060	0	150,060	588	699
	Ghost	150,060	948,568	1,098,628	720	299
5	Traditional	151,465	0	151,465	593	697
	Ghost	151,465	947,163	1,098,628	724	299
6	Traditional	171,224	0	171,224	671	705
	Ghost	171,224	927,404	1,098,628	798	299
7	Traditional	155,351	0	155,351	606	698
	Ghost	155,351	943,277	1,098,628	737	299

backbone and the other federates modeling approximately equal parts of the remaining nodes. In the traditional (Non–Ghost) approach, we used an off–line routing computation program that required more than 4 hours of CPU time and computed more than 5 million inter–federate routes. This routing information was then used to populate the inter–federate routing information in the remote links. In contrast, the ghost node approach uses the on–demand NIx-Vector routing method and thus no precomputation is needed. The results show that the memory used for ghosts is considerable, but in most cases a small fraction of the total memory usage. The exception is for federate zero, which models only 3,070 of the high–speed backbone routers of *MILNet*. This federate has more than a million ghosts, using 143MB of memory, which is 200 percent increase of the overall memory. In all other federates, the ghosts take between 100MB and 150MB, accounting for around 16 percent of the total.

Interestingly, the initialization time for the ghost node simulation is less than half of that of the traditional method. In this experiment, the entire topology is specified in a large *XML* file, which must be read in its entirety by both approaches. However, the traditional approach also requires the population of the inter–federate routing information. As mentioned, this information consists of over 5 million individual routes, which take considerable time to read and process, as evident by the larger initialization times.

2.6 Conclusion

This chapter showed that the ghost node approach is a viable method to achieve efficient and easy–to–use space–parallel network simulations. The memory required for the ghosts is small relative to the overall memory footprint of a large–scale network simulation. Using ghost nodes, no precomputation of routing information is needed and the memory–efficient NIx-Vector routing method can be used. The implementation of ghost nodes in *GTNetS* allows the same simulation script to be used for all federates, with simple command line parameters identifying node mapping.

While it is demonstrated here that the memory overhead associated with our ghost node approach is small, there is still the potential for excessive overhead when the number of federates becomes large. For example, if the same experiments were run on a thousand node supercomputer cluster, such as the Pittsburgh Supercomputer, the overhead would likely to be unmanageable. If we used one thousand federates, for every real node defined in a federate there would be approximately 999 ghost nodes. Even with the relatively small memory footprint of a ghost, the total memory for ghosts would be substantial. Future work could investigate the use of *distributed graph algorithms*, such as those described in [7], to allow a complete NIx-Vector calculation from any source to any destination in the presence of incomplete topology information. This should result in substantial memory savings at the expense of additional overhead for each NIx-Vector calculation.

CHAPTER III

BGP-ANYCAST ROUTING SIMULATION ANALYSIS

Increasing number of DNS Root Server operators are using IP Anycasting techniques to improve availability and load balancing of Root Servers. However, IP Anycast interaction with other elements of the Internet infrastructure is not well understood. This chapter presents a simulated Anycast testbed that could be used to study the advantages of IP Anycast and its impact on the main factor of the Internet infrastructure, BGP. The need for such a simulation framework is motivated in Section 3.1. A brief background about BGP and IP Anycasting is described in Section 3.2. The recent IP Anycast studies are discussed in Section 3.3. Section 3.4 provides details of our simulator . Section 3.5 describe our experimental setup and discusses the results. The chapter is concluded in Section 3.6

3.1 Motivation

Even though the Border Gateway Protocol (BGP) is the de facto Inter-domain routing protocol of the Internet, the survival of the Internet is critically dependent on the performance of the Domain Name System (DNS). DNS is a hierarchical distributed database which maps names and addresses on the Internet, and this mapping service is an important element for the Internet infrastructure. The root of the hierarchy is referred to as DNS Root Server. Currently, there are 13 Root Server operators worldwide. To ensure high availability of the DNS service, some of the Root Servers are replicated or mirrored in various locations. This was achieved through the deployment of IP Anycasting. In IP Anycasting, a group of servers operated by a particular organization share the same unicast address. Packets destined to this address will be routed to only one of the servers. IP Anyasting is a means to locate and communicate with one of a set of distributed servers within a network.

The main goals of IP Anycasting are to facilitate robust distributed system operation, ensure availability, and to reduce latency observed by the service user. The DNS system has been lately a prime target for DDoS attacks. The most recent DDoS attack against Root Servers was on 6 February 2007. As reported by The Internet Corporation for Assigned Names and Numbers (ICANN) [25], a DDoS attack on six or more of the Internet's root servers only damaged two of the servers. The attack had a very limited impact since several Root Server operators had implemented IP Anycasting since the previous DNS attack on 21 October 2002. Hierarchical deployment of Anycast servers helps in segragating traffic into regions, thus the impact of an attack is limited locally.

Since the development of IP Anycasting, a number of studies has been conducted to characterize the advantages of anycasting the IP prefix of the Root Servers [5, 8, 52]. The studies have shown that the availability of the Anycast prefixes is improved, and the end-to-end latency perceived by the user is decreased. Nonetheless, these studies have shown a few failures which can make the Anycast prefixes unavailable for several minutes. This problem has been attributed to BGP convergence. Also, one of the studies showed that the strict BGP flap-damping policies resulted in the withdrawal of a prefix by the routers even though the server was available for the entire period of the study.

The studies mentioned above are among the detailed analysis case studies which were performed on Anycasting. They provide a wide range of results about real-time behavior of the system. However, all of the studies are limited to data extracted through probes from a few locations. In addition, there is a need to evaluate the performance of Anycast under certain failure cases which may not occur during the observation period of the study. Therefore, we need a way to characterize the performance benefits of Anycast on a large scale, and need to have a good understanding of how the system performs globally. We designed a simulation platform that can be used in a controlled environment to achieve these goals.

In this research, we have developed a detailed BGP anycast simulator based on Georgia Tech Network Simulator [46]. The BGP simulator has capability to incorporate inferred AS relationships to better reflect routing policies that might be used in reality. The simulation also supports longest-prefix- match routing based on routes discovered through BGP. Using this setup, we have created an Anycast testbed to simulate a realistic large-scale topology that include providers to F-,J-,K-, and M-Root Servers.

We have performed a number of simulations to evaluate the effect of Anycast on DNS and BGP performance. Some of the interesting evaluations we have incorporated is comparison of different modes of IP Anycast deployment. There are three different modes that can be used for IP Anycast deployment, but only two of them are actually deployed. Also, we have incorporated evaluations for two different kinds of failures: silent link failure, and explicit route withdrawal from an Autonomous System (AS). As later discussed, we find that these two failures have different impact on the downtime of a prefix. We will release the simulator to the research community as a tool that can be used by Root Server operators to judge performance impact of different placement options for future Anycast servers.

3.2 Background

DNS is a critical infrastructure service for the Internet. Several DNS root server operators are using Anycast to ensure availability, load balancing, and to reduce latency perceived by the end user. The Anycast mechanism used is IP level Anycasting, which is naturally supported by the existing BGP infrastructure. In the next few sections, we give a brief background on both protocols and how their behaviors directly affect each other.

3.2.1 BGP

BGP is the de facto inter-domain routing protocol of the Internet [45]. It is currently the only routing protocol used to maintain connectivity between ASs. BGP routers in different ASs form peering sessions to exchange network reachability information. Such sessions run over a reliable transport protocol (TCP), which ensures transport reliability and eliminates the need for BGP to handle retransmissions. BGP is a path-vector protocol where each router selects best routes to destinations based on advertisements from neighboring routers/peers. BGP messages are used to exchange information and help maintain states between the routers participating in the peering session. There are four types of messages:

1. Open: Used to start a BGP session (request to open a BGP session over an existing TCP/IP session)

2. KeepAlive: Used to keep the peering session running when no update messages are exchanged. They are exchanged between the BGP peers to let each other know that they are still alive (running). In case a BGP peer does not receive a KeepAlive message from its peer, it will remove all the routes learned from that peer from its Forwarding Information Base (FIB).

3. Notification: Used to send an error message (i.e. received a corrupted update message).

4. Update: Used to transfer routing information between the BGP peers. It contains the actual route updates; the information included in this message can be used to construct a graph describing the relationships between ASs.

Overall, the update messages carry routing information while the other messages carry session management information. BGP is not strictly a standard routing protocol in the sense that it includes commercial relationships configured in its routers

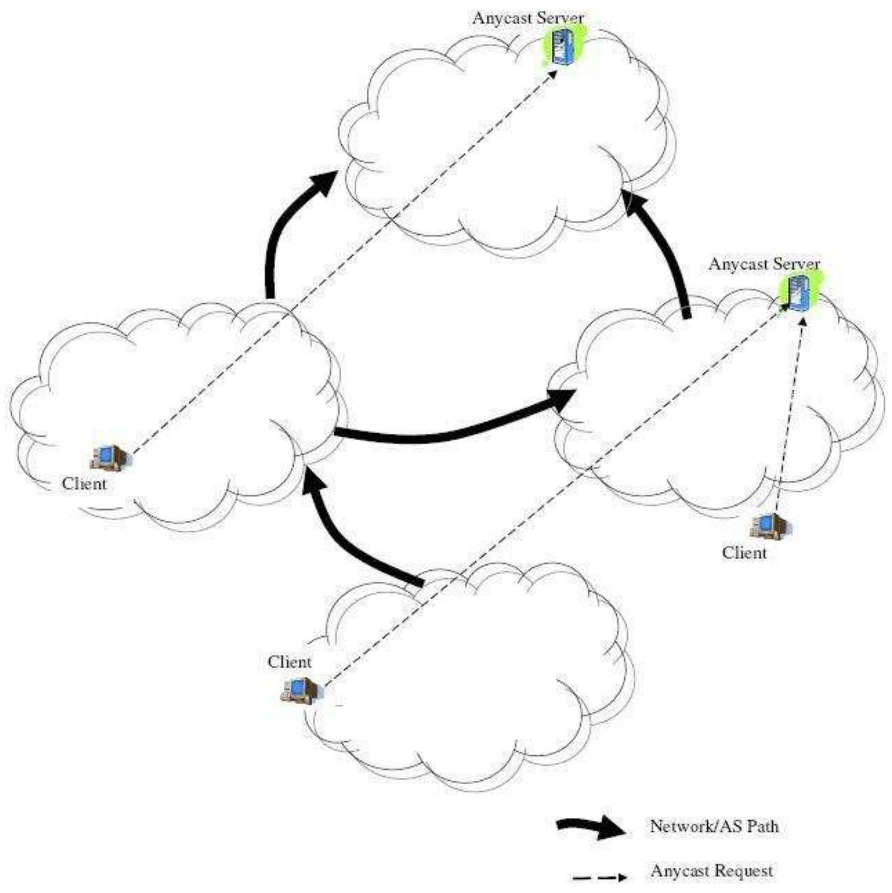

Figure 10: Typical Anycast Setup

policies, which are applied when selecting the routes during the Decision process.

Empirical measurements have shown that there can be a considerable delay in BGP convergence after routing changes. Labovitz et al. [28] show that BGP routers in the Internet may take tens of minutes to converge (have a consistent view of the topology). They relate the delay in convergence to temporary routing table oscillations formed by the BGP path selection process. Similary, BGP path exploration and Flap Dampening would have an impact on the stability of other implemented protocols/services (i.e. IP Anycast).

3.2.2 Anycast Routing

Anycast is neither unicast, where a single host receives all traffic, nor multicast, where many hosts receive (all) traffic to multicast group. Rather, it is a mechanism whereby multiple nodes are configured to accept traffic on a single IP address. IP Anycast is described in RFC 1546 [42] as a mechanism which simplifies service discovery and achieves load balancing. In this mechanism, a set of servers providing a service share the same Unicast IP address. A client wishing to use the service sends requests to the specific Unicast address (termed as Anycast address). The actual node that receives a packet is determined by routing, and the packet is not guaranteed as it could be dropped like any other packet. Being an IP layer service, IP Anycasting provides best effort to deliver packets destinated to a server of the anycast group. However, being a network layer service means that sequential packets may be delivered to different anycast nodes. Therefore, the anycast service is best used for single request/response type protocols such as DNS. At a broad level, this mechanism is supposed to achieve coarse load balancing as clients get directed to servers near them. Figure 10 shows a typical Anycast deployment scenario.

To achieve IP Anycasting, servers located in different ASs advertise the same Anycast prefix through BGP. Client ASs receiving the advertisements, choose the shortest AS length route (or another route which is preferred due to BGP policy settings) and direct queries to server residing in that AS. IP Anycasting can be viewed as ASs using multi homing, meaning that a client sees multiple different paths to the same destination.

IP Anycasting can be deployed in different modes. Some servers in the Anycast deployment might have limited bandwidth or limited server resources and might not be able to support traffic at a global level. In such cases, the advertisement of such servers is scoped. The scoping is achieved by using BGP No-Export community field. This community field helps in restricting the radius of advertisement to 1 AS hop.

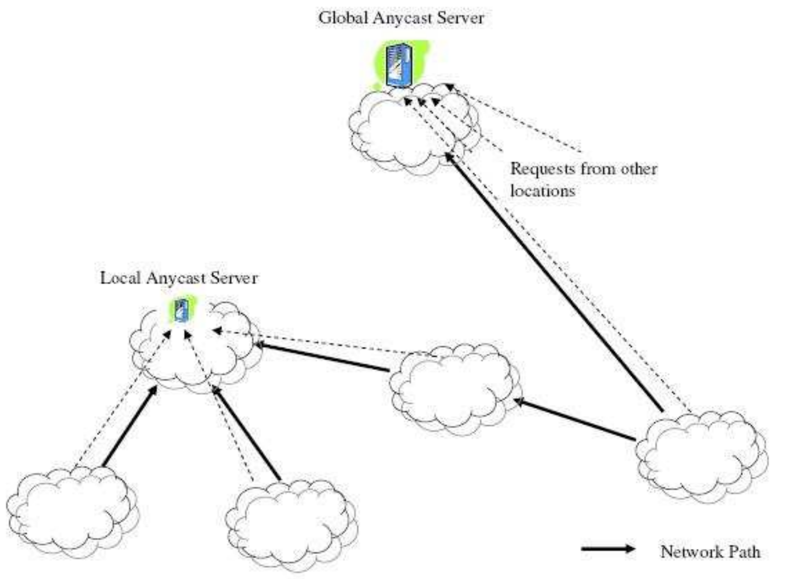

Figure 11: Hierarchical Anycast Setup

Nodes whose advertisement is thus scoped are called "Local" Nodes. The rest of the nodes whose advertisement is not scoped are referred to as "Global" Nodes. A deployment in which there is a mix of Local and Global Nodes is termed "Hierarchical" setup. Deployments which make no such scoping are referred to as "Flat" setup. Figure 11 shows a Hierarchical deployment setup.

Typically, the DNS servers run a BGP daemon (Zebra bgpd for example) which advertises the BGP Anycast prefix used by the server to the BGP routers of the AS. The BGP daemon withdraws the prefix whenever a server failure is detected. IP Anycasting has been used for supporting DNS servers for several years. F-, K-, J-, and M-Root Server operators have had substantial deployments. Anycasting has also been used for Top Level Domains (TLDs), .org and .info maintained by UltraDNS. The F-, K-, J-, and M-Root Anycasting is Hierarchical whereas the UltraDNS deployment is Flat.

3.3 Measurement Studies

In [8], Colitti provided some early measurement results about K-Root Anycast performance. K-Root had 2 Global nodes deployed and there were many local nodes. The key observations of the study were:

1. Anycast provides good latency to clients.
2. K-Root Anycast Deployment is quite stable.
3. Local nodes do not take much load off the global nodes, hinting at greater load imbalance than expected.

One of the reasons why local nodes do not get much traffic is due to the BGP policy settings. Normally, global nodes have better paths and hence preferentially chosen by BGP. However, with the hierarchical structure we expect local nodes to be chosen. Local nodes could advertise a more specific prefix and thus force the traffic through a local node; however, this is not realistically feasible as most ISPs will not advertise a /32 prefix.

Another major study regarding DNS Anycast measurements has been reported by Sarat et al. [52]. They provide a detailed analysis of Anycast behavior. Their measurements focus on different modes of deployment comparing Hierarchical F-,K-Root deployments with Flat Ultra DNS setup. They make use of PlanetLab nodes sending queries to the Anycast IP Addresses. The key findings of their study can be summarized as:

1. Improved latency and stability.
2. Few outages which last more than 100 seconds observed.
3. Flat DNS deployment was less stable than Hierarchical.

Ballani et al. [5] have reported similar results regarding DNS Anycast performance. They also observe that with IP Anycasting, the closest server is not necessarily chosen. They attribute this to specific BGP policies between interacting ASs. BGP promises shortest AS path selection only when all policy implications are the same.

Though these studies do assure that the client latency and system stability is improved with Anycasting deployed, they fail to answer some important questions:

1. How many clients get affected when outages are observed?
2. What is the impact of Global Node failure vs. Local Node failure?
3. What is the BGP churn for deploying IP Anycasting?
4. What is the impact on response time and prefix availability due to topology failures or changes?

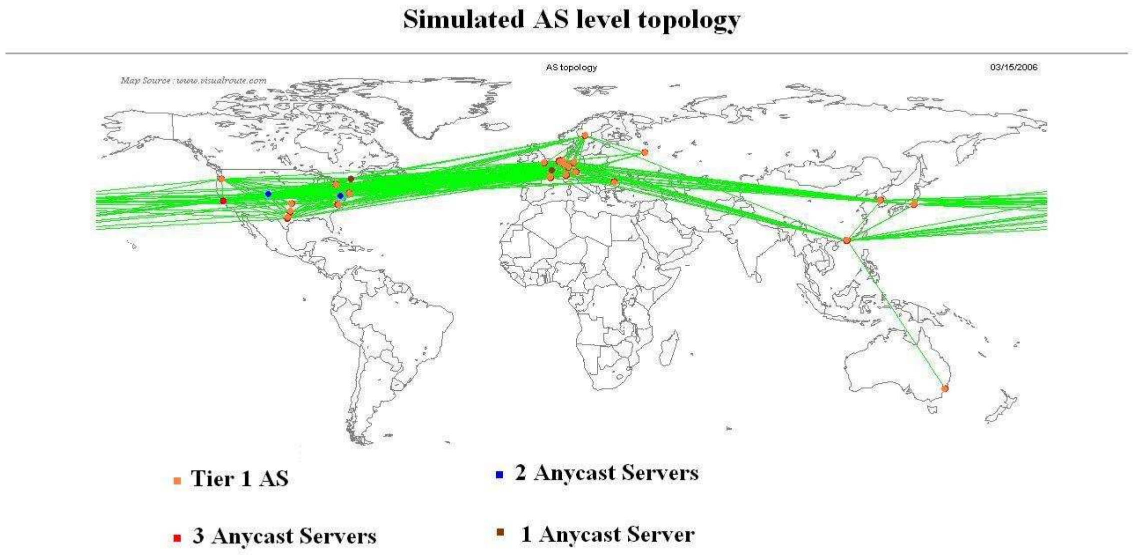

Figure 12: A Geoplot Visualization of Tier-1 Simulated Topology

To the best of our knowledge, there have been no other detailed simulation-based studies for DNS Anycasting. Many simulation studies have focused on studying BGP convergence properties [22, 40]. In this research we try to infer global effect of BGP convergence on the total downtime of a prefix. A client might have visibility to the server even if the network has not converged as the intermediate paths might still be valid.

Mao et al. [34] describe a strategy to find out shortest policy path in an AS graph obtained from RouteViews BGPTables. They use this technique to effectively measure the Anycast Servers that the clients will choose and thus estimate the load on each server. However, in our simulations we are interested in measuring impact of topology failure on Anycast downtime and BGP convergence which cannot be statically inferred.

3.4 Simulations

In this section we present details of our simulator features and capability.

3.4.1 GTNetS BGP++

GTNetS is a discrete event based packet level simulator developed in Georgia Tech. GTNetS shares similarities with ns-2, but exposes a richer programming interface to the user. The GTNetS implementation of BGP protocol is termed BGP++. The BGP++ implementation is based on Open Source Zebra BGP code. Zebra BGP was chosen as it is a production BGP implementation compliant with the RFC. Many of the techniques used to port Zebra code to ns-2 were used in porting Zebra to BGP++. A number of changes were made to the underlying Zebra data-structures which helped in simulating a large number of nodes on a single workstation. A discussion about the BGP++ implementation can be found in [15].

3.4.2 Simulating AS Topology

For our simulations, we chose to model the real AS topology as opposed to synthetically generated topologies. Since we wished to analyze DNS Anycast deployment in a realistic topology, the choice was fairly straight forward. The AS topology was inferred based on BGP routing tables as observed by the RouteViews Project [36]. We also made extensive use of AS link adjacency data available on CAIDA websites for generating the topology. Finally, we used a modified technique [13] based on Gao's [18] AS level inference strategy to infer the provider-customer, peer-peer relations between ASs.

3.4.3 Simulating DNS Client/Servers

We have a very simple simulation of DNS Clients and Servers. Each client sends a UDP request to the Anycast IP address every t seconds (discussed later). The client encodes data about originating node ID and time of the request. The UDP server responds to a request and returns the original message with details about the serving Anycast node ID. On receipt of the message, the client calculates the latency and logs the entire UDP message. The logs are parsed to gather statistics about latencies and server load.

3.4.4 Distributed Simulations

Simulations of up to 1000 nodes can be run on a modern workstation. However, the goal of our simulations is to scale to as many ASs as necessary for a simulation of a realistic topology. This requires us to use distributed simulations. GTNetS supports a parallel distributed mode which allows simulations to be run on multiple machines in a Local Area Network (LAN). We use the Chaco and Metis [24, 26] graph partitioning tools to split the simulation topology between different machines. The partitioning ensures minimum interconnections between participating machines thus ensuring efficient simulation.

3.4.5 Failures

For our analysis, we simulated two kinds of failures:

1. Silent link failure: Link failures could be on any segment of the chosen best AS path, but we restrict it to a failure in the last hop AS link. Silent link failures would rely on BGP hold-time timers to get triggered for detection. In this failure mode, the client could still communicate with the same server (if it is up) through a different path (if one exists). Such failures are expected to have a long downtime but less network churn as fewer updates about the failures are exchanged.

2. Explicit withdrawal of an Anycast prefix: This failure would mean that the server is unreachable and hence the prefix is withdrawn by the advertising AS. The Anycast client would switch to a different server in this failure case. Explicit withdraw is expected to have shorter downtime but involves potentially more network churn as many updates can be exchanged.

Our simulation also logs the number of BGP updates that get exchanged due to failures. This can be used to measure the BGP churn resulting from Anycast server instability.

3.4.6 Measurement Methodology

All of our simulation experiments consisted of several steps as follows.

1. The simulation of the topology is started, and BGP is allowed to converge.

2. Multiple Anycast servers are started in the ASs which advertise the Anycast prefix (these servers are the DNS Root Servers), and multiple clients are started in different ASs (these are the DNS clients).

3. The clients send an UDP packet to the server every t seconds, which is timestamped and returned to the client by the server.

4. Failures are induced into the system using one of the failure modes discussed earlier.

5. The simulation stops at a pre-determined time. As previously mentioned, the update messages logs, client requests/responses logs, and other statistics will be parsed for the metrics of interest.

3.5 Experimental Results

We divided our simulations into two parts. In the first part, we ensured the functionality of the simulator by using a small topology of 44 nodes. This consisted of the 44 Tier-1 topologies as inferred from the CAIDA AS ranking data. Of the 44 ASs, we found 10 ASs to provide service to the F-Root AS. For all of our simulations, we chose to place the Anycast servers at the provider nodes. The remainder of the AS nodes (34) were chosen to be DNS clients, and each client is sending one DNS request per second. The 44 Tier-1 ASs are connected through 467 links. The simulation time for

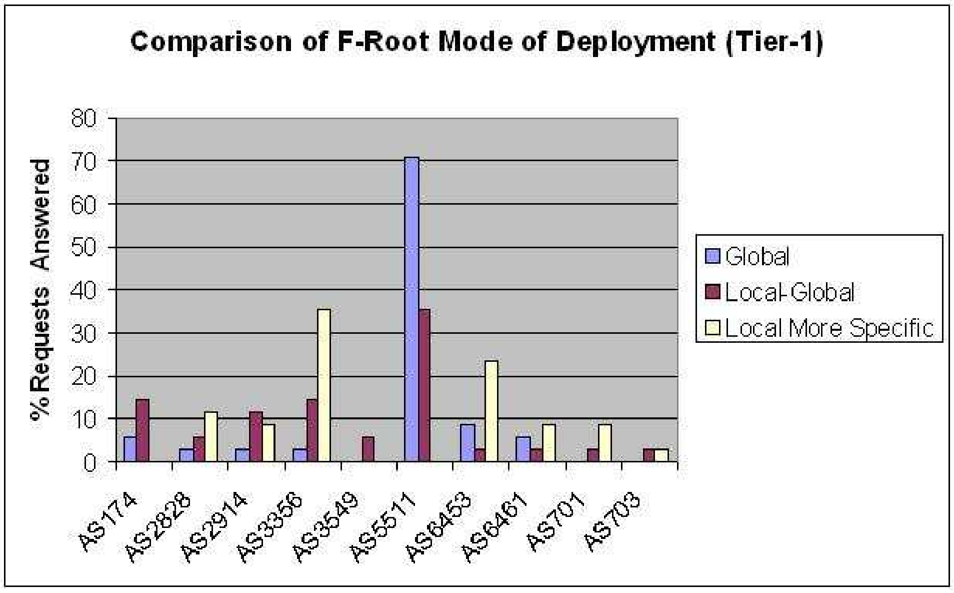

Figure 13: Load Distribution of F-Root Servers (44 nodes)

stage 1 experiments is 2000 seconds. BGP routers start and are allowed to converge. At 1000 seconds, the DNS clients start sending requests, and at 1200 seconds failures are induced into the topology. A visualization of this topology is available in Figure 12. The figure shows the number of Anycast servers that are located at overlapping sites.

3.5.1 Tier-1 topology Simulation

The 44-node setup can be run on a single machine. There are a total of 10 Anycast server instances. The first set of experiments entailed simulation of all three modes of Anycast deployment: flat, local- global hierarchy, and local-global hierarchy with more specific prefix. A flat Anycast deployment means that all anycast servers are global, their catchment area is global (global load of client requests). In the local-global hierarchy deployment, the local anycast servers handle only the requests from its local area. The adverstisement of the local prefixes is scoped. In the third mode of deployment, the anycast prefixes advertised by the local servers are /32 compared to /24, which will force BGP to choose the local servers as the closest servers for the client requests.

We ran the experiments and did not observe any load distribution patterns for flat or hierarchical deployment. We attribute this to the BGP decision process and to the small topology of study. However, as expected we did notice that when advertising more specific prefixes, the global nodes tend to receive a very low percentage of DNS clients requests, as show in Figure 13. This is due to the fact that BGP decision process will favor /32 prefix over any other metric (local preference).

We ran another experiment, using local-global hierarchy mode, where we induced the failures mentioned earlier. In the link down failure case, we found that there is a significant loss of DNS requests, whereas the withdraw case shows better performance. The experimental results show that it took 123 seconds for the network to realize that a link failure has occured. As expected, silent link failures would rely on BGP hold-time timers to get triggered for detection, resulting in larger convergence time. As a result a lot of DNS client requests were lost, 1307 requests (3.84 percent). In addition, the measurements for the withdraw case reveal that the effective downtime of the prefix is less than one second. We attribute this to quick convergence of BGP as the graph is very strongly connected.

3.5.2 Tier-1,Tier-2 Topology Simulation

After verifying the correct functionality of our simulator, we moved to the second part of simulation experiments where we expanded the number of nodes in our topology by including Tier-2 ASs. This increased the total number of nodes in our system to 5476. These nodes are interconnected by 14,468 links. We used distributed simulations with 16 federates for this topology. Thus each federate modeled about 350 nodes. In this stage, we evaluated the following metrics of interest:

1. BGP Convergence: After inducing the failures in the topology, we measure how long it takes BGP to reach a steady state both in Anycast and non-Anycast

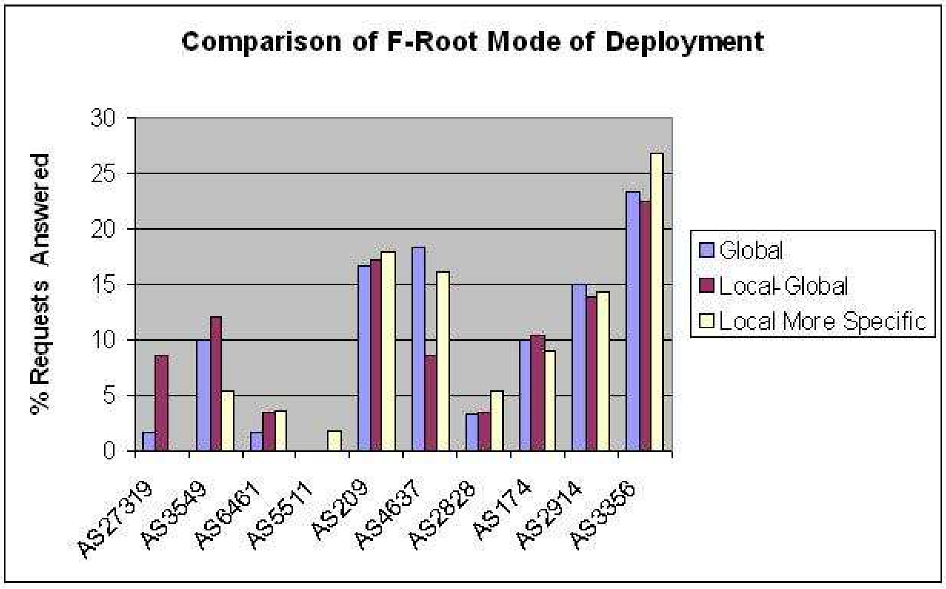

Figure 14: Load Distribution of F-Root Servers (5476 nodes)

deployment.

2. BGP Churn: The failures will cause an exchange of update messages, and we quantify this exchange both in Anycast and non-Anycast deployment.

3. End-to-End User Latency: One of the advantages of Anycast deployment is the decrease of latency perceived by the enduser. We verify this in multiple experiment scenarios.

4. Anycast Load Balancing: We measure the load distribution of DNS client requests among the DNS Anycast servers.

We have extracted information about ASs that provide connectivity to the F-, J-, K-, and M-root sites. In all, we have 5,476 ASs and 14,468 interconnecting links. We found in the topology 10 ASs that provide service to the F-root, 6 ASs that provide service to the J-root, 17 ASs that provide service to the K-root, and 4 ASs that provide service to the M-root. Initially, we had each node advertising one prefix each, but experiments showed that the memory requirements for this network would

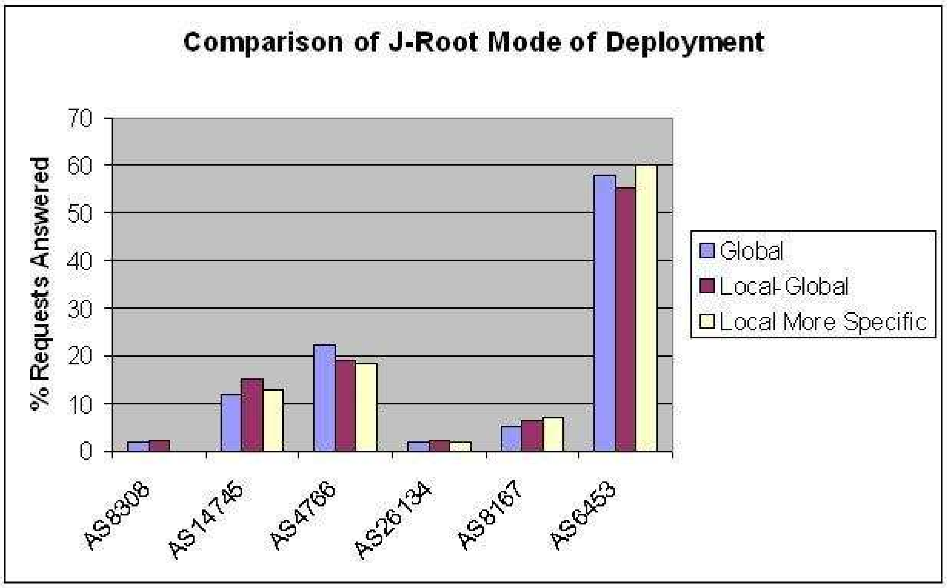

Figure 15: Load Distribution of J-Root Servers (5476 nodes)

be quite high. Thus, we changed the BGP configuration files so that only Anycast prefixes get advertised. We verified using a smaller topology that advertising the non-anycast prefixes does not have any measurable effect on the metrics of our study. This is because the failures are only induced on the Anycast server nodes, which only advertise Anycast prefixes.

In order to make our simulation experiments more realistic, we collected DNS request statistics to drive our simulations. We are mainly interested in the sending rate of DNS requests to the root servers. The data was retrieved from the DNS Statistics project by CAIDA, which has data only for F- and K-Root. This data showed average requests per second, 6699.18 and 7449.65. As for the J- and M-Root servers, we assumed a rate of 6000 requests per second.

In total, we conducted 8 experiments. The first three experiments entailed evaluating the load distribution among all root servers under the three different modes of deployments. We find that the load distribtion under the local-global mode is not significantly different from that of flat, or all-global deployment. This is illustrated in Figure 14 and Figure 15. However, as seen in the first stage of simulations, the pattern of hierarchical deployment (with more specific prefix) is redundant in the second stage of experiments. The global nodes tend to receive very low number of DNS requests. For example, in the case of F-Root servers, global node AS3549 receives around five percent of the DNS queries, while global node AS27319 receives none. Similarly for global node AS8308 in the J-Root servers.

The second three experiments were all run with the hierarchical mode of Anycast deployment (advertising /24 prefixes for both local and global nodes). All experiments have a simulation time of 2000 seconds. The first experiment does not have any failures. In the second experiment, we take down one of the interfaces of an F-Root server node (AS27319) at 1300 seconds of simulation time. Next, in the third experiment, we further take down interfaces of J-,K-, and M-Root servers at 1300 seconds of simulation time. As previously stated, we are interested in the robustness and effectiveness of Anycast deployment. The experimental results show a loss of 16,321 DNS requests (0.24 percent) destinated to AS27319 due to taking one of its interfaces down. As predicted, taking J-,K-, and M-Root interfaces down did not have any effect on the loss rate of DNS request destinated to the F-Root Anycast address. However, the convergence time in the latter case was longer, 214 seconds compared to 152 seconds.

The last two experiments were all run with non-Anycast deployment, with traditional unicast deployment using one-to-one mapping between DNS servers and IP addresses. The only difference between these two experiments is that F-, J-, K-, and M-Root interfaces go down at 1300 seconds in the second experiment, as was done in the earlier experiment with the Anycast deployment. The results of this experimental setup will enable us to compare the performance of both Anycast and non-Anycast deployment, as well as their effect on BGP. In other words, we will measure BGP convergence, amount of BGP churn associated with both cases of deployment, in addition to the DNS response time for the F-Root DNS server.

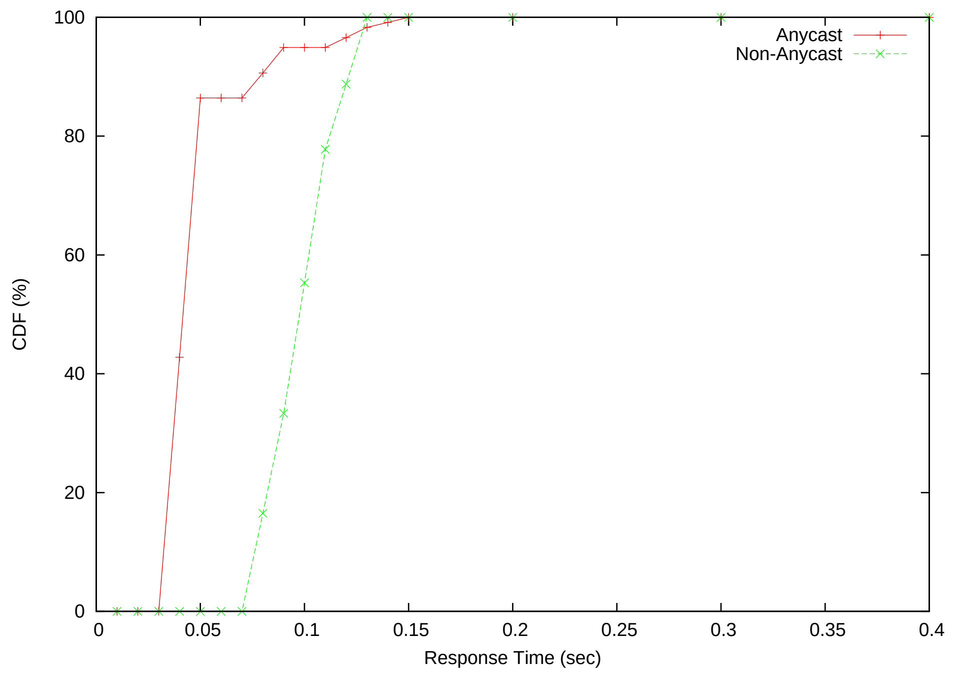

Figure 16: DNS Response Time without Failures (5476 nodes)

Table 3: BGP Performance

	Anycast	Unicast
BGP Convergence Time	211 sec	178 sec
BGP Churn (Update Messages)	12,019	30,978

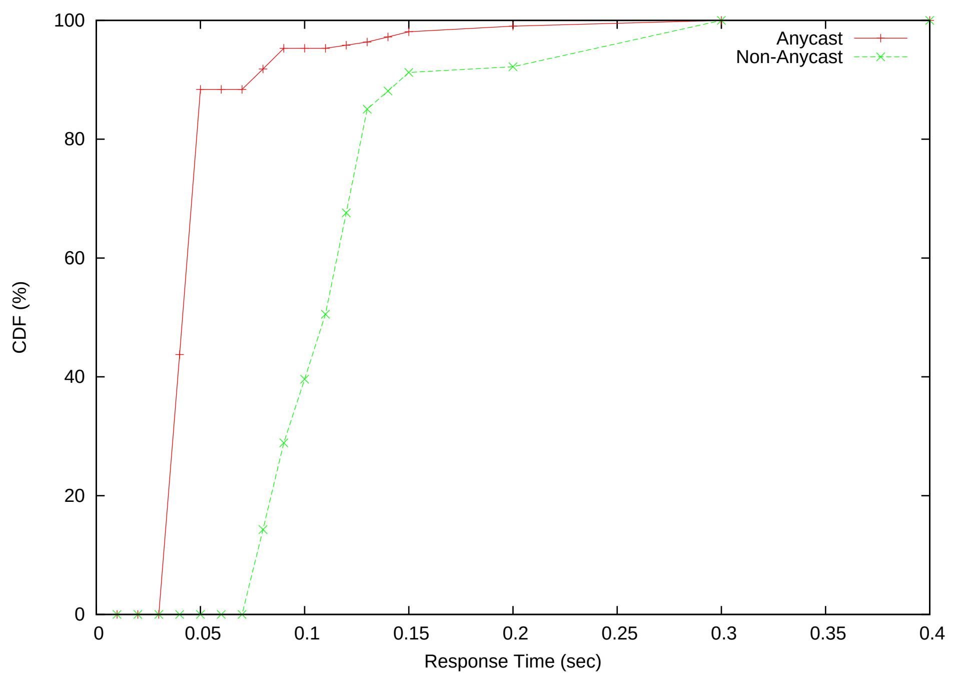

Figure 17: DNS Response Time with Topology Failures (5476 nodes)

The experimenal results show clearly the advanages of IP Anycasting. As seen in Table 3, the failures induced in the topology (taking root servers interfaces down) cause BGP churn of 30,978 update messages in the non-Anycast deployment compared to 12,019 update messages with Anycast deploment. This is due to the fact that with using Anycast, the updates only propagates to the affected routers and other routers best path will remain the same. However, the convergence time due to failures in the case of Anycasting is slightly longer than the unicast-case. This is not a major drawback as new AS path could be selected before BGP completely converges.

As mentioned earlier, the DNS response time is measured by noting the time of each DNS request and the time each response is delivered back to the client. Figure 16 and Figure 17 shows the comparison of DNS response time between both Anycast and non-Anycast deployment with and without topology failures. It is easy to see in the CDF of Figure 16 that 85 percent of the DNS requests got answered at 0.05 seconds with Anycast compared to 0.115 seconds using unicast prefixes for the F-Root servers. Furthermore, introducing topology failures has a direct effect on the response time in the unicast case compared to the Anycast case. The response time for 85 percent of the request was around 0.13 seconds, an increase of 13 percent. With Anycast, the probability to find a best path after a topology change is higher than that in the unicast case.

3.6 Conclusion

In this work we have built a simulation framework that will allow us to analyze the performance of Anycast Server deployment. Our framework supports detailed BGP simulations and routing based on the routes learned through simulated BGP. Our simulations allow realistic simulation of topologies as observed through RouteViews project. We have used our simulation to analyze performance of different Anycast deployment modes. We have also analyzed the impact of different failure modes for the deployment scenarios. We find that most current deployments do not achieve good load balancing. We find that when Local nodes advertise a more specific prefix, this reduces the load on Global nodes. Also, our experimental results show higher availability of the prefix and decreased latency using IP Anycasting. Our comparison of IP Anycasting to the traditional approach shows that IP Anycasting causes less BGP churn when failures occur.

Future work could incorporate more failure models to determine their impact on IP Anycasting and BGP. Many different scenarios should be evaluated and we expect that using our simulations the community will gain a better understanding of the advantages of DNS Anycasting.

CHAPTER IV

EIGRP SIMULATION MODEL AND SEAMLESS MOBILITY USING ROUTE UPDATES

This chapter presents a scalable simulation model of Enhanced Interior Gateway Routing Protocol (EIGRP) and a new approach for host mobility within an Autonomous System (AS). EIGRP is widely deployed, and our simulator can be used as a framework to analyze the performance and behavior of EIGRP in different scenarios. In addition, this chapter shows that host mobility can be supported without the deployment of new protocols, special configuration, or support from any end applications.

Section 4.1 motivates the needs for an EIGRP simulation model and a new host mobility approach. Existing EIGRP studies and current mobile computing approaches are discussed in Section 4.2. Section 4.3 discusses the simulation framework and our implementation efforts of EIGRP and wireless-handoff in Georgia Tech Network Simulator to support route updates for mobile nodes. Our experimental setup and results are presented in Section 4.4. We conclude in Section 4.5.

4.1 Motivation

The motivation for this research is two-fold. First is the implementation of an EIGRP simulation model which can be used to evaluate the performance of EIGRP in a variety of scenarios. Next is the implementation and evaluation of the new mobile computing appproach. Even though host mobility is used to create a highly dynamic topology for EIGRP test-case scenarios, each part of this work has its own motivation.

4.1.1 EIGRP

Since its development, EIGRP has been shown to converge as quickly as a link-state protocol while at the same time being loop free. The EIGRP designers [3] state that many medium-scale network studies were performed and EIGRP proved to be a robust and reliable intra-domain routing protocol. As of 2000, network architects [43] state that EIGRP and Open Shortest Path First *(OSPF)* [38] are being implemented in approximately half of the networks. EIGRP is not only an enterprise-oriented routing protocol, but also a protocol that can be used in service-provider environments because it has fewer topology limitations than OSPF [43]. However, OSPF has seen more deployment in the service-provider market because most new service-provider oriented technologies such as MPLS/VPN (Virtual Private Networks based on Multi-Protocol Label Swapping) are first implemented within the framework of OSPF. Although EIGRP is widely deployed, there have been few published studies measuring its performance.

These routing protocols are sophisticated distributed algorithms and a deep understanding of their performance and behavior is difficult as they are deployed in medium to large-scale networks. Simulation tools have been typically used in computer network systems study to evaluate architectures or perform systems tuning. However, there is usually a trade-off between accuracy and scalability. A full detailed model of the system will require a large amount of memory and CPU power, and sometimes lengthy execution time. On the other hand, an abstract model will result in better scalability at the expense of less accuracy.

4.1.2 Seamless Mobility

Wireless local-area networks are becoming increasingly common among university and corporate campuses. A 2004 survey [20] representing 516 two- and four-year public and private colleges and universities across the United States reported that

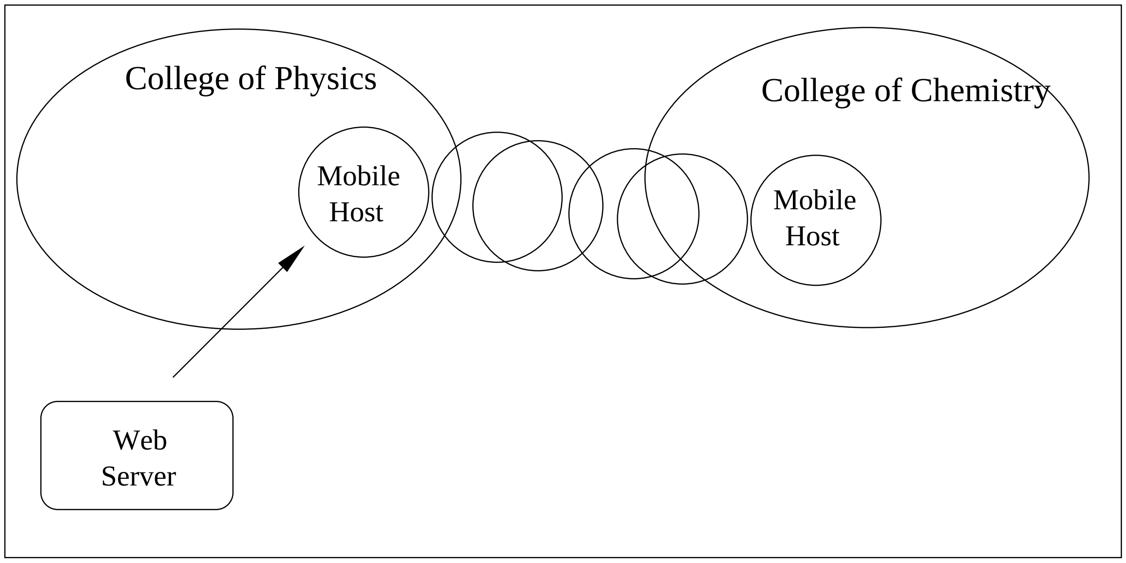

Figure 18: Mobile Node Operation

19.8 percent of the colleges participating in the survey indicate that full-campus wireless networks are up and running at their institutions as of fall 2004, compared to 14.2 percent in 2003. Also, the survey reported that wireless networks are available in more than 35.5 percent of college classrooms. This revolutionary change in network technology development and deployment indicates the need for mobile networking research. An important feature of wireless networks is the ability of end hosts to move within the physical region covered by the subnetwork. However, this mobility action leads to difficulties in handling host IP addresses and forwarding packets within the subnetwork.

To illustrate the issues regarding IP mobility, let us consider the scenario as shown in Figure 18. A wireless user is browsing a web page while he is sitting or moving within the College of Physics building. As long as he stays within range of the wireless base station of Physics, the packets are delivered to the mobile host using the IP address that was assigned to it by DHCP, as long as there are no changes in the session identifiers. If the user starts moving away from the College of Physics to a different building, the College of Chemistry in our example, his wireless device

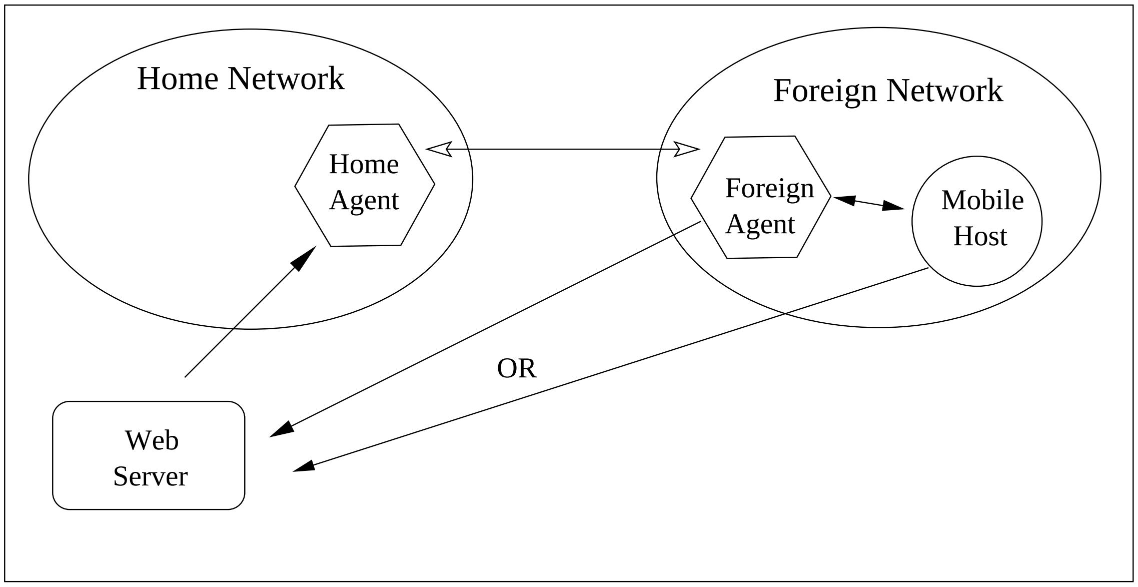

Figure 19: Mobile IP Operation

could be in one of two states. First, it could get disassociated from the current access point, resulting in a broken link which then results in packets being dropped at the access point. In the second case, the device could get associated with another access point that is on a different network and receives an new IP Address from a different DHCP process. In either case the remote endpoint of the session is unaware of the new address and continues sending the packets to the old IP address, resulting in packets being dropped at the old access point. In the next section, we discuss the basic concepts and operations of Mobile IP in both IPv4 and IPv6. We also discuss the difficulties and overhead arising from these approaches.

In IPv4, a point of attachment to the Internet is uniquely identified by its IP address. So, when a mobile device moves outside the network identified by its IP address, all the packets destined for that node are lost. In Mobile IP, as the mobile host moves away from its home network, it gets associated with another IP address that has the subnet prefix of the new (foreign) link, and it binds this new address with an agent on a local router at the home network. Then, all the packets destined to the mobile host are routed through its home network to the foreign network. Figure

19. illustrates the basic operations of Mobile IP.

The mobile agent of every network identifies its presence by sending agent advertisement using an ICMP router advertisement. A mobile host uses Agent Discovery to determine if it is connected to its home network or a foreign network. If the mobile host is connected to its home network, the packets that are destined to it are delivered using standard Internet routing mechanisms. Otherwise, the mobile node gets associated with a new IP address offered by the foreign network, known as the care-of-address. The correspondent node for a TCP connection with the mobile node cares only about the home IP address since it is the packet header destination address. Therefore, the mobile node must inform its home network with the IP address (care-of-address) using a process known as Registration. The care-of-address will be registered with the home agent, which will forward all the packets destined to the home IP address to the new location of the mobile node.

For example, suppose initially the mobile node is located on its home network, and has a TCP connection with a web server. As the mobile node moves to a different network as shown in Figure 19, the packets sent from the web server will arrive at the home network via standard IP routing. Then, the home agent will intercept all of these packets, and it will tunnel them to the foreign agent or the mobile node itself. The packets are tunneled to hide the mobile host home address from routers along the path from the home to foreign network. Next, the packets will be de-capsulated and delivered to the mobile node. Note that the packets originated from the mobile node could be delivered to the web server directly from the foreign agent without going through the home network.

The main difference between Mobile IPv4 and Mobile IPv6 is that mobility support is integrated in IPv6. The basic binding and registration processes are similar, but in Mobile IPv6 the mobile node can inform its correspondent node (the other end point of the connection) with its current location by registering its care-of-address with the correspondent node. Thus, any new packet addressed to the mobile node is sent directly to the new care-of-address by checking the correspondent cache bindings. When sending a packet from the mobile node to the correspondent node, the mobile node stores its home address in a new *Home Address* destination option in the IP header. In addition to the support of IPv6 in both networks, local routers at the home network are required for at least one registration of the new care-of-address. This is required since the first packet addressed to the mobile node has to be routed via the home network routers. After that the packets are routed directly to the new care-of-address.

4.2 Related Work

Many studies have been done in both areas of research.

4.2.1 EIGRP Studies

Several simulation studies have been done to evaluate the performance of interior routing protocols for new applications/architectures. However, most of these studies have used OSPF models. For example, OSPF is implemented in NS-2 [35], SSFNet [10], and GLOMOSIM [58]. One of the reasons could be that EIGRP is proprietary. Nevertheless, Opnet [6] has developed an EIGRP model which has been used in such studies, and a simulation of HP backbone yielded good performance of EIGRP [3]. Still, one can notice that there are few analysis studies on EIGRP.

4.2.2 Host-Mobility Studies

Several solutions for seamless continuity of applications and sessions during mobility have been proposed. They differ depending on the layer of the OSI model at which they are implemented. Mobility solutions have been proposed for: link layer, application layer, and network layer.

In link layer mobility solutions such as [51], GSM or 802.11 handle all the mobility and the IP/network layer is unaware of the changes of points of attachment to the Internet. However these solutions are access technology–specific, since integrating heterogeneous access media at the link layer becomes very complex to deploy. Another link layer mobility solution is to use virtual wireless LANs using switches and forming layer 2 subnets. The APs will be connected to switches, and once a mobile host associates or disassociates, the mapping table of the switch will be updated. There are some disadvantages of this approach as well. First, the outstanding packets will be lost as there is no queuing in switches, and secondly the network has to be configured for virtual LANs which can be difficult to install and maintain.

Another viable approach to mobility is to move the burden of managing the session and the underlying changes at the IP layer to the application layer. However, rebuilding all the applications to support mobility and be backward compatible is not viable, as it is very complex and expensive. NetMotion Wireless Inc. developed a driver that sits between the application layer and the transport layer. This mobility approach requires a server (a proxy for the mobile device) as well as a software installed on the mobile device. In a similar work, Snoeren [53] suggested an architecture that uses modified transport layer protocols at the end hosts without any changes to the IP layer.

As mentioned earlier, Mobile IP by Perkins [44] is a modification to IP which allows mobile nodes to receive their data packets wherever they happen to be attached to the Internet. Mobility is solved at the network layer by hiding the changes in IP address from upper layers. This approach is presently considered the most developed and deployed mobility solution.

Zhuang [59] proposed a mobility solution ROAM that is built on top of the Internet Indirection Infrastructure (*i3*). *i3* is implemented as an overlay network consisting of a number of servers across the Internet which introduces an extra support overhead.

4.3 Simulation Framework

The simulated network had to be very dynamic to effectively test EIGRP limits. This was accomplished by allowing the wireless Access Points (APs) to behave as EIGRP routers (or running EIGRP agents), and access points. Also, the end systems were allowed to retain a fixed IP address while those systems move across subnet boundaries. This way, as the mobile hosts move across network coverage, the wireless handoffs between the mobile hosts and the APs will trigger the EIGRP agents to send route advertisements to inform routers of new or revised routes to reach the mobile systems. The following sections describe our implementation efforts of EIGRP and the wireless handoff mechanism into *GTNetS*.

4.3.1 EIGRP

EIGRP [3] is an intra-domain routing protocol that leverages the strong points of both distance-vector and link-state protocols: it converges quickly while remaining loop free at all times. This is achieved by using a system of diffused computation where every route calculation is computed in a coordinated fashion among multiple routers. EIGRP is based on the Diffusing Update Algorithm (DUAL) which is used to compute shortest paths in a distributed manner and without ever creating routing-table loops or incurring counting-to-infinity behavior. Simulation studies [56] have shown that DUAL's average performance after a topology change such as link failure, or link-cost increase or decrease, is significantly better than the Distributed Bellman-Ford (DBF) algorithm used in Routing Information Protocol (RIP), and it is similar to the performance of an ideal link-state algorithm with much less CPU overhead.

EIGRP's updates are similar to a distance-vector protocol, as they are vectors of distances transmitted only to directly connected neighbors. However, the updates are partial, non-periodic, and bounded. They are partial since the updates contain only the changed routes, and not the entire routing table. They are only sent whenever a metric or topology change occurs (non-periodic), and they are sent to the affected routers only (bounded). EIGRP has shown to provide loop freedom and quick convergence in medium-scale networks[3]. Also, a simulation of HP backbone yielded good performance of EIGRP [3]. However, a true analysis and diagnosis of EIGRP protocol at a large scale has not been undertaken. This analysis is essential taking into perspective the number of deployed EIGRP-enabled Cisco routers.

We developed a scalable simulation model for EIGRP. The protocol is not ported, but rather implemented in a high quality software network simulator (*GTNetS*). Also, we have implemented a subset of EIGRP functionality, since for our performance analysis we only need link failure, link restoration and link-metric change.

4.3.2 Wireless Handoff

As mentioned earlier, in our simulation the wireless handoffs are the events which trigger EIGRP to send routing updates. In *GTNetS* we have implemented a fairly complete subset of handoff mechanisms based on the 802.11 MAC protocol. A handoff mechanism essentially illustrates the basic steps which must be taken when a mobile station disassociates with the current access point and associates with the new one.

The wireless communications of mobile devices are vulnerable to communications interception to some degree, and thus there needs to be a control of such communications to protect the information while in transit. In our area of study, a number of security attacks could be exploited to either disrupt the functionality of the implemented protocols or to gain access to sensitive information. For example, the EIGRP updates are triggered by the wireless handoffs of the mobile nodes. Therefore a malicious user could send fake association or disassociation messages to disrupt the routing while a mobile host may or may not be already associated with an access point (the network does not converge). In addition, an adversary machine could advertise itself as an existing mobile station and associate with an access point and start receiving all packets destined to that mobile station. These packets could be very sensitive such as online banking transaction, secure access session, etc.

Due to the above reasons, we use an authentication mechanism between the APs and the mobile hosts. Once a mobile host associates with an AP, a shared key will be generated and that key will be propagated with the EIGRP updates informing the peer EIGRP routers about the new host. Thus, if a malicious user tries to send fake association or disassociation messages to disrupt the routing while a mobile host is already associated with an AP, the association/disassociation procedure will fail. This will happen because the current AP that the mobile host is associated with or any other AP has the secret key of the host being faked and knows that the message being advertised is not valid.

The wireless layer design in *GTNetS* allows for stations to be designated as Access Points (APs) or Mobile stations (MSs). The APs are connected to the wired network. Our design assumes that the APs have the role of EIGRP routers as well as access points, but this functionality can be decoupled without any effect on the proposed routing scheme. In the current scheme, we have additional local state information in the form of a last-heard timer at each MS and AP. While the AP needs to maintain one such timer for each associated MS, the MS has to maintain only one for its currently associated AP.

The handoff scheme we use is slightly different from the one specified in the 802.11F. The mobile stations always listen to the periodic beacons sent from the Access Points (typically, every 0.1 seconds). Depending on the received signal strength (RSS) and other factors, the MS determines if the incoming beacon's transmitter is a more appropriate access point than its current association. To prevent oscillating associations we chose a threshold margin, which is the difference in signal strength that the MS must see between the current association and the incoming beacon's signal. Voluntary disassociations are initiated by sending an association request to the new AP and a disassociation message to the currently associated AP. The disassociation message is sent only after the new association has been acknowledged by an association response message by the new AP.

Both the APs and the MSs need to mutually know when each has left the operating range of the other. This mechanism is implemented by running a last-heard timer for each of the associated mobile hosts. The mobile stations send reassociation messages every 5 seconds to the access point. The receipt of a reassociation message resets the corresponding last-heard timer. A timeout of this last-heard timer means that the MS has disassociated involuntarily. On the other hand, at the MS's end, the last-heard timer is reset at the reception of a beacon from the associated AP. If the MS does not hear beacons from its currently associated AP, its last-heard timer will timeout, at which point it will assume it is no longer associated. This would make the MS try to associate with any other AP from which it hears a beacon.

The wireless MAC layer notifies the EIGRP layer of any associations and disassociations that have occurred. These notifications trigger the EIGRP diffusing computations that adjust the routing tables appropriately. For instance, when an AP receives a disassociation message from a MS (moving out of range), it sets the link metric for that MS to infinity. On the other hand, when an AP receives an association message from a MS (moving in range), it sets its link metric to a certain value (this value is the same for all MSs). We chose the value of 100, but it could be any reasonable value. Also, the secret key will be propagated with the EIGRP updates informing the peer EIGRP routers about the new MS.

4.4 Experimental Results

The test environment is a simulation of a subset of *Georgia Tech* campus which consists of 7 buildings, in an area of 120 acres as shown in Figure 20. All 7 buildings are connected to the backbone routers with a mix of 1Gb and 100Mb links. The

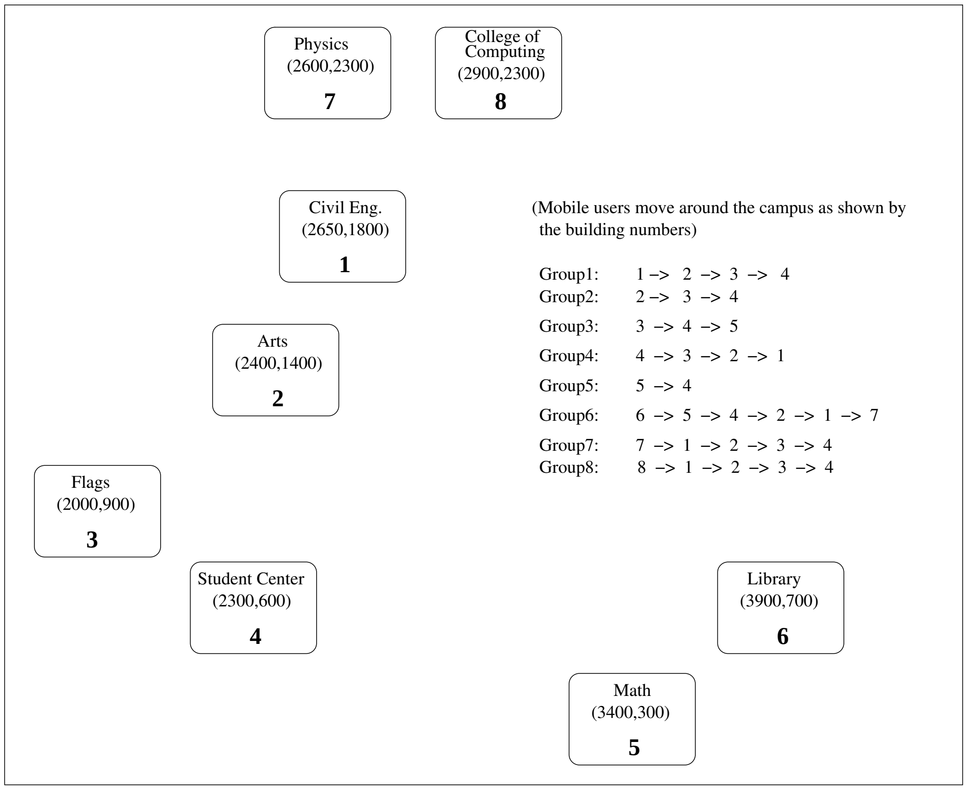

Figure 20: Campus Buildings Layout

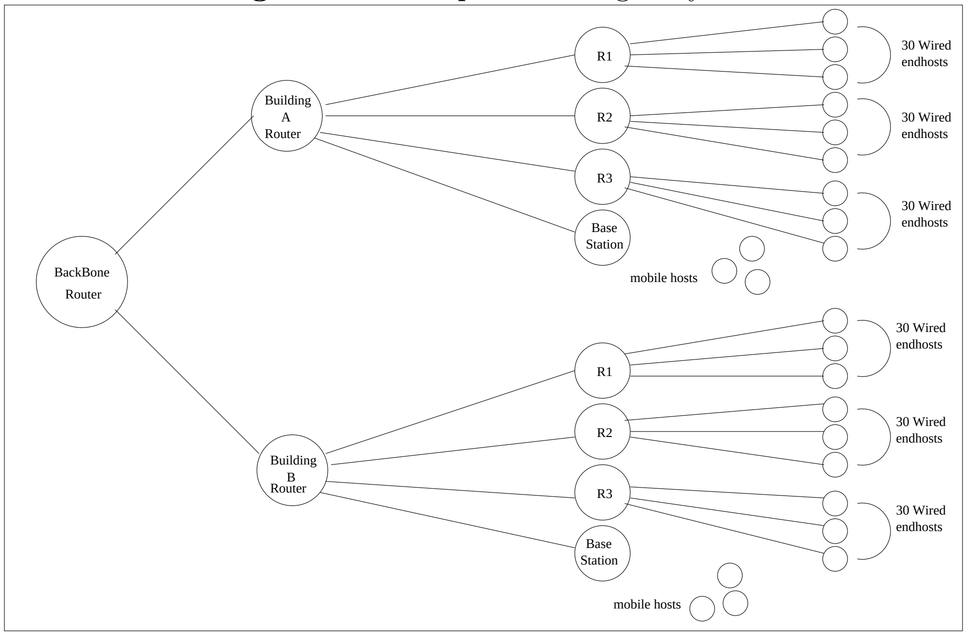

Figure 21: Campus Building Network

experimental network topology was constructed as follows. Every building network includes a wired and a wireless network. The wired network consists of 3 subnets with 30 end hosts each connected through 10Mb links as shown in Figure 21, while the wireless network is made of a single access point and 9 mobile stations per access point on average. The total topology has 720 wired end hosts, 72 mobile stations, and 42 routers including the 8 access points. The choice of buildings was made to include worst-case scenario for EIGRP convergence. The EIGRP agents are triggered by the handoffs of the APs, and the handoffs come in two flavors: live-handoff, and dead-handoff depending whether there is an overlap in wireless coverage or not.

In [27], Kotz reported that 53 percent of the traced wireless traffic was web browsing and the rest included data-backup, peer-to-peer file sharing, file transfer, etc. However, we believe that with today's development in wireless technology, 54g wireless cards, mobile users will tend to do most of their work through wireless media wherever possible, which leads us to believe that most or all of them would have one or more long–lived active TCP connection.

We modeled two types of mobile users. First, those who start a TCP connection and remain stationary. The second are those who start a TCP connection, and then move around the campus while the connection is active. Our experiments included both types of users, as this would be the more realistic model and would also clearly show EIGRP convergence capability when mobile devices move across subnetwork boundaries.

All of our experiments have background traffic which includes web browsing and data-backup/file transfer traffic running on the wired end hosts. There are 700 web browsers on 200 wired end hosts that randomly connect to a group of web servers(located outside the campus), each handling a large number of simultaneous requests. The size of the individual web object requests, the size of the replies, and the time delay between the requests is modeled based on empirical measurements described by [33]. There are 75 file transfers between 150 wired end hosts with a uniformly distributed size between 20 and 80 MB. The wireless users traffic is composed of 72 long-term on-campus/off-campus TCP connection uniformly distributed between 20 and 50MB each. As mentioned previously, all of the simulations were performed using our *Georgia Tech Network Simulator* (*GTNetS*), enhanced to include our detailed model of the EIGRP protocol and realistic wireless handoff models. The simulations were run for 400 simulation seconds, which resulted in a number of wireless handoff actions and routing convergence computations as reported below.

To illustrate the robustness and reliability of EIGRP, we ran four sets of experiments to collect a set of metrics. Some metrics were also chosen to demonstrate the feasibility of the mobile computing within an AS with the deployment of EIGRP. The chosen performance metrics are the following:

1. ***EIGRP Convergence Time*** is the period of time that takes the routing protocol to converge and the routing tables to reach a steady state. This metric determines the overall performance of TCP connections, since with long convergence times active TCP connections might experience substantial packet losses and several timeouts resulting in reduced performance. Figure 22 and Figure 23 shows the EIGRP convergence time (log scale) throughout the simulation time for two experiments. Both experiments have the same topology shown in Figure 20, the only difference being the radio range for both wireless devices being 300 feet (resulting in overlapping coverage between most of the buildings) in first experiment and 200 feet (resulting in several dead zones) in the second.

   Initially, all EIGRP routers started randomly between 0 and 20sec, after which the mobile devices start moving according to a specific waypoint model. We see in the figures that EIGRP has a maximum convergence time of around 0.7 milliseconds for the 300ft radio range, and 9.0 milliseconds for the 200ft radio range. Using 200ft radio range, there will be more dead zones resulting in mobile hosts disassociating from APs (triggering EIGRP updates). Next, while EIGRP is converging, mobile hosts associations will trigger new EIGRP updates thus extending the convergence process. The convergence period is acceptable even for wireless users, as the packets destined to the MS need to be forwarded to the new AP as soon as the MS get associated with it.

It is true that the mobility model (number of users, walking patterns and speed) has a substantial effect on the EIGRP convergence time. In the worst case, handoffs would be so frequent that the protocol would never converge since during the convergence the topology has again changed. However, we believe that our experimental results are scalable to any reasonable mobility pattern and reasonable network topology size. We considered the normal walking speed of 4.4 feet per second as the general speed within the campus, and we defined specific waypoint models that best represent the mobility patterns for the campus users. Since the longest measured convergence time was less than 10 milliseconds, we would have to experience more than 100 handoffs per second to overrun the convergence process with update actions. One of the reasons that our convergence times are fast is that our experiments are limited to a single AS within a small geographic region, leading to a very small propagation delay. In our wired topology, we used one microsecond for the propagation delay on all wired links. This is certainly reasonable for most moderate sized AS subnetworks.

In addition, the reason for the extra spikes in convergence time shown in Figure 23 is the number of dead zones. A dead zone occurs when a user leaves the coverage area of his existing access point association before coming into coverage range of another access point. As soon as the MS is disassociated from the AP, an EIGRP update event is triggered and results in subsequent DUAL computations. When it comes into range with a new AP, another EIGRP update event

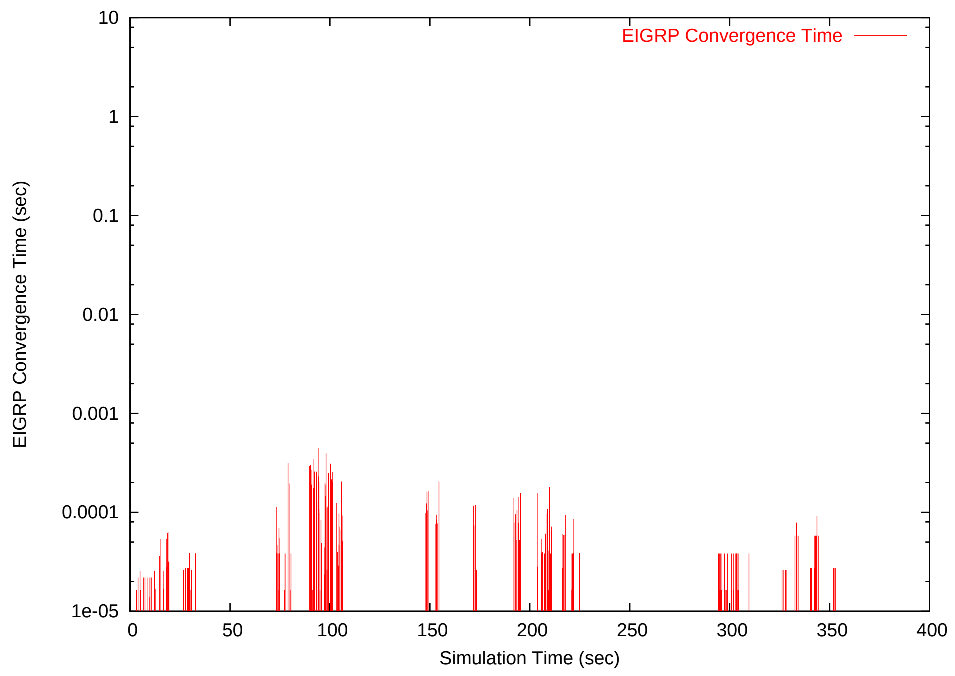

Figure 22: EIGRP Convergence Time for 300ft Radio

is triggered and EIGRP has to converge again.

Since most EIGRP packets are configured to have the same priority as any other packets in the network, a heavy load on the network might cause a longer convergence time for the routing protocol, due to increased queuing delay on the congested links. This is illustrated in Figure 24 and Figure 25. However, even with a large load on the network, the convergence time of EIGRP in this environment is still extremely small and acceptable for our applications.

2. ***TCP Performance*** measures the amount of data sent by an active TCP connection per unit time. In this environment, the EIGRP convergence time is one of the main factors that impacts TCP performance. If the network experiences excessively long EIGRP convergence times, active TCP connections would endure heavy packet loss and numerous timeouts. Having shown with previous experiments that EIGRP in fact converged quickly due to host mobility, we expected that the TCP connections over moving wireless media with handoffs

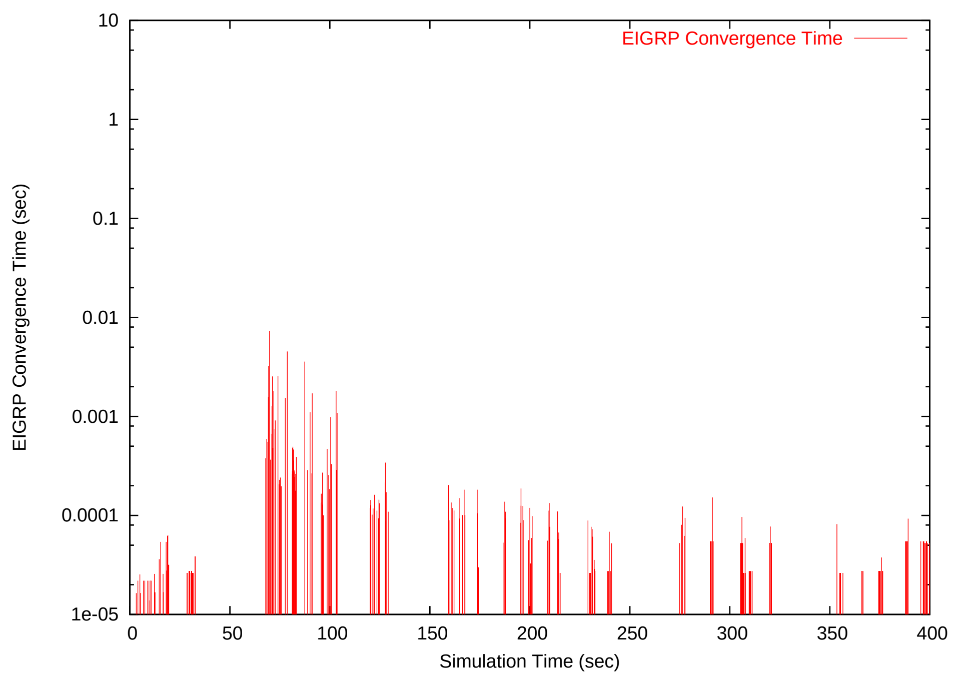

Figure 23: EIGRP Convergence Time for 200ft Radio

to behave as similarly as the stationary wireless media. We point out that TCP over wireless media has several other issues which are outside the scope of this paper.

The same experiments that were used to compute EIGRP convergence time resulted in the TCP performance measurement illustrated in Figure 26 and Figure 27. The figures show the TCP sequence number transmitted as a function of time. Clearly, a higher slope indicates better throughput. As mentioned earlier, we have 72 mobile devices, with each one having an active TCP connection. The flow that is shown in Figure 26 is one of the 72 wireless flows. We chose this particular one as it represents the TCP performance for a mobile station as it experiences handoffs. For this particular flow, the MS visits three networks during its mobility pattern. The handoff actions occurred at approximately 100, 200, and 300 seconds. One can notice that the first handoff was clean, with few retransmissions. However, the connection during the second

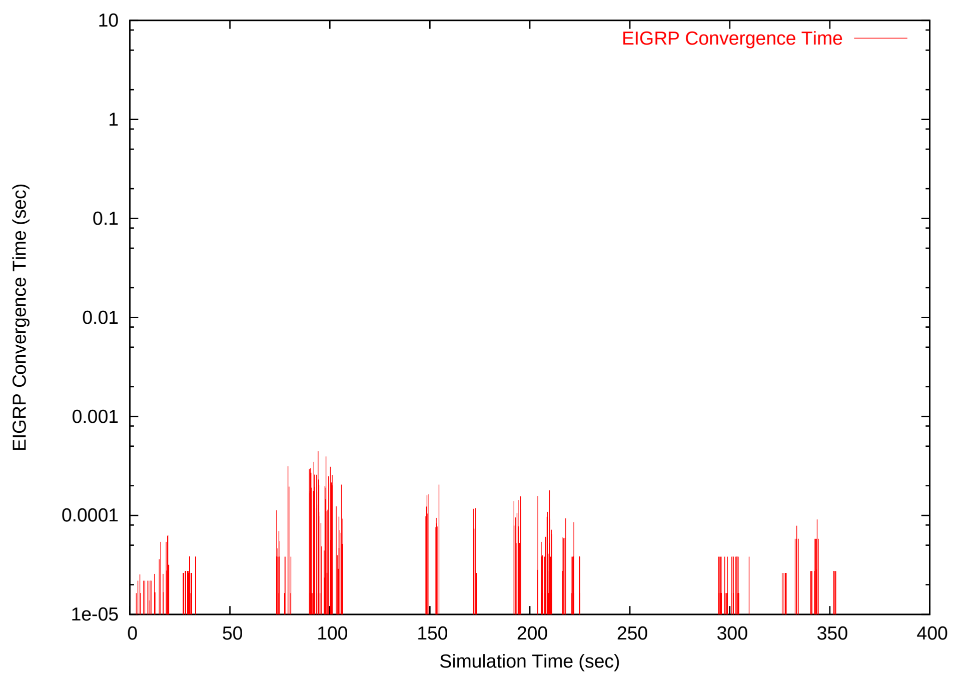

Figure 24: EIGRP Convergence Time with Data Flows

handoff had no activity for short intervals. This was not a dead zone handoff; rather it was a live handoff. In cases when the mobile is at the boundary of overlapping access points, we are bound to see oscillations because of the wireless characteristics and the CSMA/CA properties. The multiple associations and disassociations that we see when the mobile station move through such a region are an artifact of this.

The results in Figure 26 and Figure 27 are for a 300ft radio range experiment with and without mobility. The MS shown in the bottom curve had a total of 11 handoffs. In most of our results, the stationary MS throughput was higher than the mobile one as expected. However, some of the mobile users (actively moving) had similar throughput as the stationary users as shown in Figure 27. This was for the on-campus connection since its short RTT (Round Trip Time) enables the wireless user with mobility to perform better. When a MS gets associated with a new AP, it resets its transmit queue. Again, the experimental

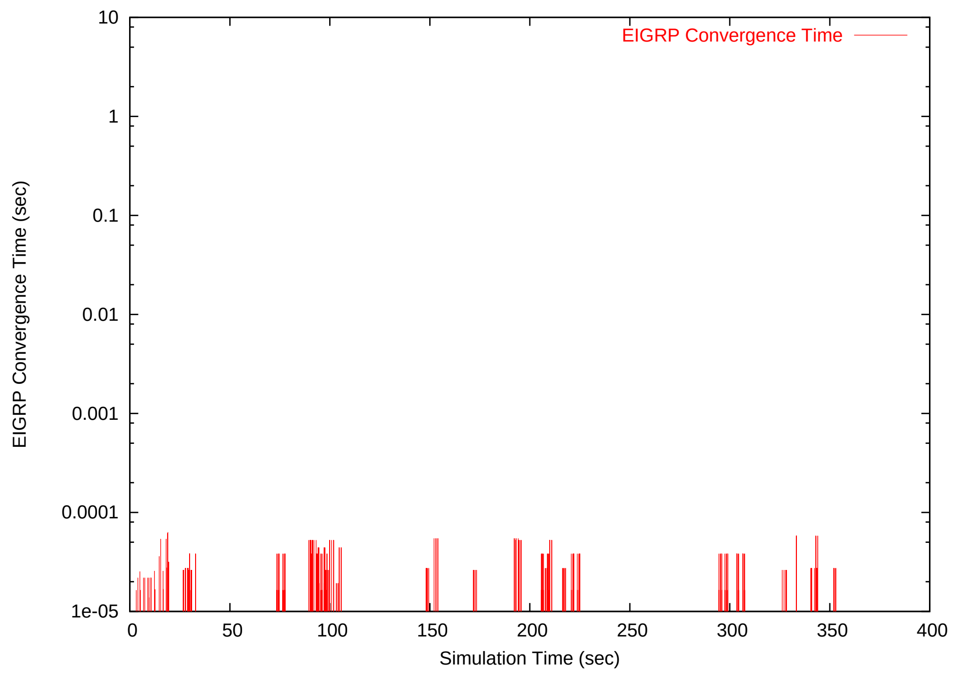

Figure 25: EIGRP Convergence Time with No Data Flows

results are based on the wireless users which incur a lot of dynamic network changes, thus representing worst-case scenarios for EIGRP performance.

3. ***EIGRP Overhead*** is the overhead incurred in the routing protocol due to a handoff from one access point to another. In a traditional wired network, when the EIGRP routers start they will exchange their routing table information with neighboring routers, causing many update and reply messages. After some period of time, the EIGRP protocol converges to a steady state with each EIGRP speaker having the same view of the overall network topology. As long as there is no router failure, link failure or cost metric change, there is only the low overhead of the periodic EIGRP Hello packets. However, in our experimental setup every AP is running an EIGRP protocol instance, the dynamic changes in the network due to end host mobility induce a number of EIGRP messages as the protocol recomputes the optimal routing paths. Therefore, we measured the total number of EIGRP messages to show the effect of the additional network

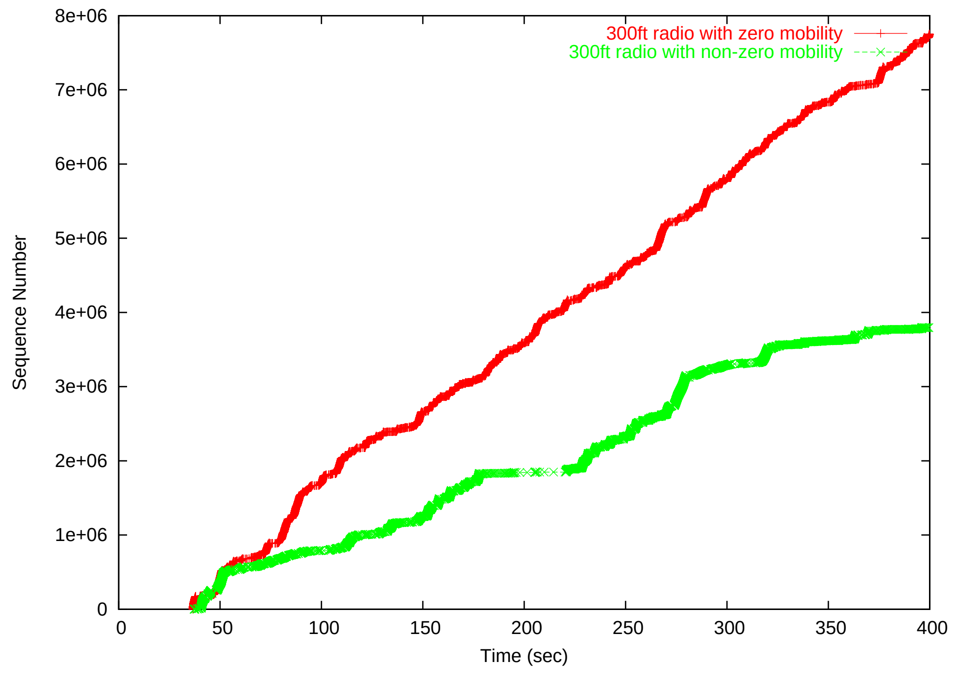

Figure 26: TCP Performance for a MS

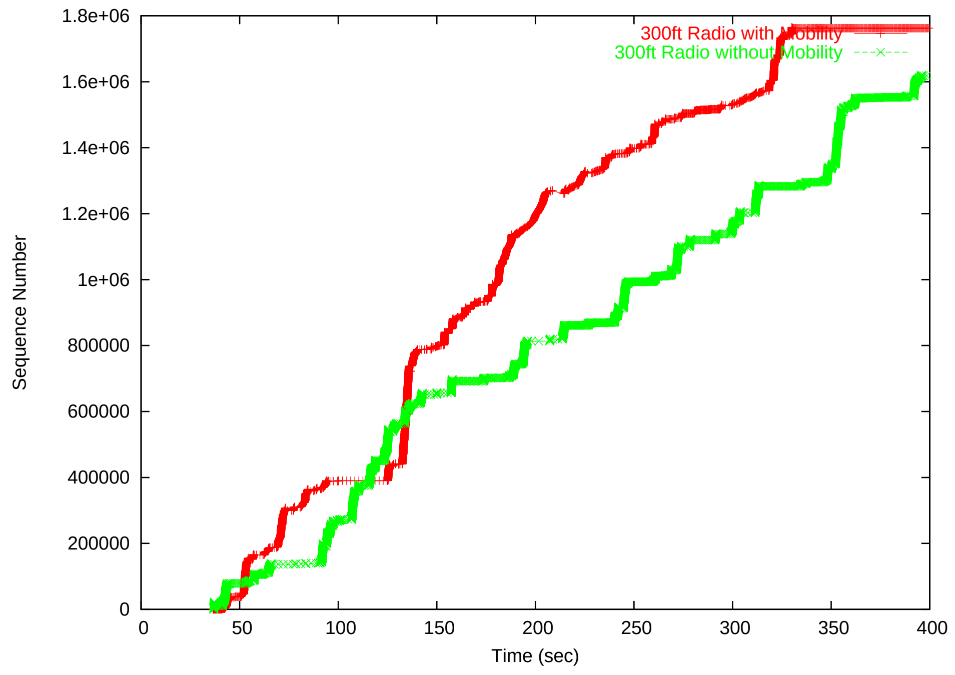

Figure 27: TCP Performance for a MS with On-campus Connection

load due to mobile handoffs and routing reconvergence.

Table 4: EIGRP Overhead During a Mobility Speed of 4.4ft/sec

	EIGRP Events During	
	Startup	Mobility
Updates	1,897	4,346
Queries	0	4,662
Replies	0	4,662

Table 4 shows the overhead of EIGRP measured when running an experiment with normal walking speed of 4.4 feet per second for mobile devices, and using a 300 foot radio range APs and MSs across the campus subset shown in Figure 20. When the network starts up, the EIGRP routers have to exchange the routing tables, which triggers 1,897 EIGRP update packets. After the network converges and all routing tables have reached a steady state, the MSs start their TCP connections and begin to move between the campus buildings. All MSs in our experiments follow a specific waypoint model designed to realistically model a user walking on campus. The mobility in this particular experiment resulted in an EIGRP overhead of 4,346 update packets, 4,662 query packets, and 4,662 reply packets. This may seem substantial, but recall that the EIGRP protocol uses partial updates rather than full routing table exchanges. Further, these several thousand updates were spread over a period of 400 seconds throughout our simulation execution. Since, there are a total of 40 MS in motion and on average each moves between four buildings (according to our specific waypoint model), the 300ft radio range experiment resulted in a 284 handoffs. Some of the handoffs (a small percentage) were due to the coverage overlap, and this resulted in some oscillations.

4.5 Conclusion

We have developed a scalable and detailed simulation model of EIGRP that is publicly available for researchers in the computer networks community. Also, our performance analysis of the protocol has shown its robustness and capability to adapt quickly to a very dynamic network. In addition, we have shown that the host mobility using route updates is a viable method to achieve seamless mobility and continuous connectivity for users of mobile wireless devices as they move within an AS. The EIGRP overhead incurred from mobility is minimal as all of EIGRP query and reply messages are small. Using our approach, there is no need to deploy new hosts or agents, make special configuration, or request support from any end points. We do need instances of the EIGRP protocol running on the APs, or an interface between the AP and existing EIGRP routers to inform the routing protocol of associations and disassociations. Also, an authentication mechanism between the APs and the mobile hosts is needed to prevent session hijacking.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

This dissertation provides contributions to the field of conducting detailed large-scale realistic IP Anycast (coupled with BGP) and EIGRP simulations. We first developed a new technique to federated network simulations that enables large-scale simulations regardless of the complexity of the network topology being simulated. The second contribution was the extension of a detailed BGP simulator (BGP++) to include IP Anycast service. The third contribution was the implementation of EIGRP into a scalable network simulator, and the introduction of a new approach to host mobility within an AS. Route updates are used to convey the new point of attachment of the mobile node.

5.1 Ghost Node: An Enabling Technique for Distributed Network Simulations

In Chapter 2 we introduced a new approach to federated network simulations. One way to creating network simulation models for large-scale topologies is to use a space-parallel partitioning methodology, coupled with distribued simulations methods. In this approach the simulated network is divided into k sub-models, where k is the number of federates in the distributed simulation. With this approach each federate is only responsible for approximately $1/k^{th}$ of the entire topology, and instantiates simulation objects to represent its own portion of the network. Since a given federate has no responsibility for the remaining $(k-1)/k$ portion of the network, no simulation objects are created and thus the federate has no knowledge of the remaining topology.

However, difficulties arise in order to insure correct packet forwarding between the federates especially when the simulated network is well interconnected. We overcome these difficulties by introducing a new mechanism that provides full topology knowledge at every federate. We utilize a topology partitioning methodology that uses Ghost Nodes. A ghost node is a simulator object in a federate that represents a simulated network node that is spatially assigned to some other federate, and thus no other federate is responsible for maintaining state information associated with that node. The ghost node acts as a placeholder for nodes that are assigned to other federates. It has none of the complex and memory intensive state needed for real nodes (such as queues, routing tables, port maps, and applications). Rather, it simply contains toplogy connectivity information about links and neighbors. Therefore, using ghosts, a federate is affordable a global view of the simulated topology, without the memory overhead of maintaining unneeded state for the ghosts.

Experimental results of small (15,000 nodes) and large (over 1 million nodes) networks showed that the ghost node approach is a viable method to achieve efficient and easy-to-use space-parallel network simulations. The memory required for the ghosts is small relative to the overall memory footprint of a large-scale simulation. The implementation of ghost nodes in GTNetS allows the same simulation script to be used for all federates, with simple command line parameters identifying node mapping.

Even with the ghost node approach, the simulation user must still specify the mapping of node objects to federates. In all but the simplest cases, determining a suitable and efficient mapping is challenging and requires considerable analysis of the traffic patterns between the simulated network elements. Liu and Chien[32] describe an automated method to partition networks used in their *MicroGrid*[54] emulation tool. These results seem promising, and investigating their applicability to the ghost node approach for space–parallel network simulation can be a future research work.

5.2 BGP-Anycast Routing Simulation Analysis

In Chapter 3 we provided a simulated IP Anycast [41] testbed that is implemented in a detailed BGP simulator. This testbed was used to replicate the real world topology in a simulated environment in order to analyze the performance and limitations of IP Anycast. Also, we investigated the impacts of IP Anycast on BGP.

Our experimental analysis were all based on DNS Anycast deployment, because an increasing number of DNS Root Server operators are using IP Anycasting techniques to improve availability and load balancing of root servers. We wanted to simulate DNS Anycast deployment that is as close as the real deployed network. Thus, we inferred a large realistic Tier-1 and Tier-2 topology (5476 ASs) based on BGP routing tables as observed by the RouteViews Project [36]. Also, we made extensive use of AS link adjacency data available on CAIDA websites for generating the topology.

In addition, we used a modified technique [13] based on Gao's [18] AS level inference strategy to infer the provider-customer, peer-peer relations between ASs. We used this experimental setup to reflect the commercial nature of the Internet. Also, our simulations entailed two kinds of topology failures: silent link failure, and prefix withdrawal. These failures were induced in the topology after BGP converged to evaluate the impact of BGP convergence on the response time of DNS Root Server. The simulation of the 5476 node topology was only made available through the use of the Ghost technique introduced in Chapter 2.

Our study showed higher availability of the prefix and reduced latency using IP Anycasting. Furthermore, our comparison of IP Anycasting to the traditional approach (using a single server per service) showed that BGP incurs less overhead when IP Anycast is deployed. The BGP churn was measured after topology failures were induced. Like other studies, we found that most current Anycast deployments do not achieve good load balancing. However, in the case where local nodes advertise a more specific prefix, the load on global nodes is reduced.

To our knowledge, this is the first detailed BGP simulator coupled with IP Anycast service. We are expecting this simulator to be used as a framework for future analysis of IP Anycast. Future work could incorporate more failure models to determine their impact on IP Anycast and BGP. Also, it could be used to futher investigate the effects of BGP policies on load balancing.

5.3 EIGRP Simulation Model and Seamless Mobility Using Route Updates

In Chapter 4, we introduced a simulation model of EIGRP in a scalable simulator GTNetS. Also, we presented a new approach for host mobility within an AS. The chapter discusses the simulation framework and our implementation efforts of EIGRP and wireless-handoff in GTNetS.

In short, we developed a model of EIGRP protocol in GTNetS. The protocol was not ported, but rather implemented from publicly available specification. We implemented a subset of EIGRP functionality, specifically link failure, link restoration and link-metric change. Also, our new host mobility approach can be summarized as follows. The wireless Access Points (APs) are allowed to behave as EIGRP routers (or running EIGRP agents), and access points. Also, the mobile hosts are allowed to retain a fixed IP address while those systems move across subnet boundaries. This way, as the mobile hosts move across network coverage, the wireless handoffs between the mobile hosts and the APs will trigger the EIGRP agents to send route advertisements to inform routers of new or revised routes to reach the mobile systems.

Our experiments showed EIGRP's robustness and capability to adapt quickly to a very dynamic network (a network composite of wired and wireless hosts with high mobility). In addition, we have shown that host mobility using route updates is a viable method to achieve seamless mobility and continuous connectivity for users of mobile devices as they move within an AS. The results showed that EIGRP converges faster than a single TCP timeout in most cases.

REFERENCES

[1] "As112 project." http://www.as112.net, March 2006.

[2] Abley, J., "Hierarchical anycast for global service distribution." http://www.isc.org/tn/isc-tn-2003-1.html, 2003. ISC Technical Note ISC-TN-2003-1.

[3] Albrightson, B., Garcia-Luna-Aceves, J. J., and Boyle, J., "Eigrp – a fast routing protocol based on distance vectors," in *Proceedings of Networld/Interop*, May 1994.

[4] Bachinsky, S. T., Mellon, L., Tarbox, G. H., and Fujimoto, R. M., "Rti 2.0 architecture," in *Proceedings of the 1998 Spring Simulation Interoperability Workshop (SIW'98)*, Mar 1998.

[5] Ballani, H., "Some random pnp sh*t-note." unpublished!

[6] Bertolotti, S. and Dunand, L., "Opnet 2.4: an environment for communication network modeling and simulation," in *Proceedings of the European Simulation Symposium*, October 1993.

[7] Bertsekas and Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice–Hall, 1989.

[8] Colitti, L., "Effects of anycast on k-root.some early results." http://ripe.net/ripe/meetings/ripe-51/presentations/pdf/ripe51-anycast-k-root.pdf, 2005.

[9] Cowie, J., Liu, H., Liu, J., Nicol, D., and Ogielski, A., "Towards realistic million-node internet simulations," in *International Conference on Parallel and Distributed Processing Techniques and Applications*, June 1999.

[10] Cowie, J., Ogielski, A., and Nicol, D., "The SSFNet network simulator." Software on-line: http://www.ssfnet.org/homePage.html, 2002. Renesys Corporation.

[11] Cowie, J. H., Nicol, D. M., and Ogielski, A. T., "Modeling the global internet," *Computing in Science and Engineering*, January 1999.

[12] D. Johnson, C. P., "Mobility support for ipv6." Network Working Group, June 2004. Internet RFC 3775.

[13] Dimitropoulos, X., Krioukov, D., Huffaker, B., Claffy, K., and Riley, G., "Inferring as relationships: Dead end or lively beginning?," in *Proceedings of Fourth Workshop on Efficient and Experimental Algorithms (WEA'05)*, May 2005.

[14] Dimitropoulos, X. and Riley, G., "Creating realistic bgp models," in *Proceedings of Eleventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'03)*, pp. 64 – 69, Oct 2003.

[15] Dimitropoulos, X. and Riley, G., "Large-scale simulation models of bgp," in *Proceedings of twelfth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'04)*, Oct 2004.

[16] Ferenci, S., Perumalla, K., and Fujimoto, R., "An approach to federating parallel simulators," in *14th Workshop on Parallel and Distributed Simulation*, May 2000.

[17] Fujimoto, R., Ferenci, S., Loper, M., McLean, T., Perumalla, K., Riley, G., and Tacic, I., "Fdk users guide." Georgia Institute of Technology, March 2001.

[18] Gao, L., "On inferring automonous system relationships in the internet," *IEEE Transactions on Networking*, vol. 9, pp. 733–745, December 2001.

[19] Garcia-Luna-Aceves, J. J., "Loop–free routing using diffusing computations," *IEEE/ACM Transactions on Networking*, vol. 1, Feb 1993.

[20] Green, K. C., "The 2004 campus computing survey," in *Proceedings of EDUCAUSE*, October 2004.

[21] Greene, B. and McPherson, D., "Isp security: Deploying and using sinkholes." http://www.nanog.org/mtg-0306/sink.html, 2003. NANOG TALK.

[22] Hao, F. and Kippol, P., "An internet scale simulation setup for BGP," *Computer Communications Review*, vol. 33, July 2003.

[23] Hardy, T., "Distributing authoritative name servers via shared unicast addresses," April 2002. Internet RFC 3258.

[24] Hendrickson, B. and Leland, R., "The chaco user's guide," 1994.

[25] ICANN, "Root server attack on 6 february 2007." http://www.icann.org/announcements/announcement-08mar07.htm, March 2007.

[26] Karypis, G. and Kumar, V., "Metis: Unstructured graph partitioning and sparse matrix ordering system."

[27] Kotz, D. and Essien, K., "Analysis of a campus-wide wireless network," in *Proceedings of ACM MobiCom'02*, Sept 2002.

[28] Labovitz, C., Ahuja, A., Bose, A., and Jahanian, F., "Delayed internet routing convergence.," in *Proceedings of ACM SIGCOMM '00*, Aug 2000.

[29] Liljenstam, M., Liu, J., and Nicol, D. M., "Development of an internet backbone topology for large–scale network simulations," in *Proceedings of the 2003 Winter Simulation Conference (WSC'03)*, Dec 2003.

[30] Liu, J. and Nicol, D., "Learning not to share," in *16th Workshop on Parallel and Distributed Simulation*, 2002.

[31] Liu, J. and Nicol, D. M., "DaSSF 3.1 user's manual," April 2001.

[32] Liu, X. and Chien, A. A., "Traffic–based load balance for scalable network emulation," in *Proceedings of the ACM Conference on High Performance Computing and Networking*, November 2003.

[33] Mah, B. A., "An empirical model of http network traffic," in *Proceedings of IEEE INFOCOMM*, pp. 592–600, 1997.

[34] Mao, Z. M., Qiu, L., Wang, J., and Zhang, Y., "On as-level path inference." ACM SIGMETRICS, 2005.

[35] McCanne, S. and Floyd, S., "The LBNL network simulator." Software online: http://www.isi.edu/nsnam, 1997. Lawrence Berkeley Laboratory.

[36] Meyer, D., "Oregon routeviews database." http://www.antc.uoregon.edu/route-views/, March 2006. University of Oregon Advanced Network Technology Center.

[37] Miller, D. C. and Thorpe, J. A., "SIMNET: The advent of simulator networking," *Proceedings of the IEEE*, vol. 83, pp. 1114–1123, Aug 1995.

[38] Moy, J., "Internet RFC1131: Ospf specification." Network Working Group, Oct 1989.

[39] Moy, J., "Internet RFC2328: Ospf version 2." Network Working Group, April 1998.

[40] Nykvist, J. and Carr-Motyckova, L., "Simulating convergence properties of bgp," in *Proceedings of IEEE International Conference on Computer Communications and Networks ICCCN'02*, 2002.

[41] Partridge, C., Mendez, T., and Milliken, W., "Host anycasting service," November 1993. Internet RFC 1546.

[42] Partridge, C., Mendez, T., and Milliken, W., "Host anycasting service." http://d.root-servers.org/october21.txt, 1993. Internet RFC 1546.

[43] Pepelnjak, I., *EIGRP Network Design Solutions.* Cisco Press, 2000.

[44] Perkins, C., "Ip mobility support for ipv4." Network Working Group, January 2002. Internet RFC 3220.

[45] Rekhter, Y. and Li., T., "RFC 1771, border gateway protocol 4," March 1995.

[46] Riley, G. F., "The Georgia Tech Network Simulator," in *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, pp. 5–12, ACM Press, 2003.

[47] Riley, G. F., "The Georgia Tech Network Simulator." Software on-line: http://www.ece.gatech.edu/ research/ labs/ MANIACS/ gtnets.htm, 2003.

[48] Riley, G. F., Ammar, M. H., and Fujimoto, R. M., "Stateless routing in network simulations," in *Proceedings of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, August 2000.

[49] Riley, G. F., Fujimoto, R. M., and Ammar, M. H., "A Generic Framework for Parallelization of Network Simulations," in *Proceedings of Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'99)*, October 1999.

[50] Riley, G. F., Fujimoto, R. M., and Ammar, M. H., "Parallel/Distributed ns." Software on-line: www.cc.gatech.edu/ computing/ compass/ pdns/ index.html, 2000. Georgia Institute of Technology.

[51] Ritter, M., Friday, R. J., Garces, R., Filippo, W. S., Nguyen, C., and Srivastava, A., "Mobile connectivity protocols and throughput measurements in the ricochet mcdn system," in *Proceedings of ACM MobiCom'01*, July 2001.

[52] Sarat, S., Pappas, V., and Terzis, A., "On the use of anycast in dns." ACM SIGMETRICS, 2005.

[53] Snoeren, A. and Balakrishnan, H., "An end-to-end approach to host mobility," in *ACM/IEEE Mobicom*, August 2000.

[54] Song, H., Jakobsen, D., Bhagwan, R., Zhang, X., Taura, K., and Chien, A., "The MicroGrid: a scientific tool for modeling computational grids," in *IEEE Supercomputing Conference (SC'2000)*, November 2000.

[55] Wu, H., Fujimoto, R., and Riley, G., "Experiences parallelizing a commercial network simulator," in *Proceedings of the Winter Simulation Conference*, Dec 2001.

[56] Zaumen, W. and Garcia-Luna-Aceves, J., "Dynamics of link state and loop-free distance-vector routing algorithms," in *Journal of Internetworking*, December 1992.

[57] Zegura, E. W., Calvert, K., and Bhattacharjee, S., "How to model an internetwork," in *Proceedings of IEEE Infocom 96*, 1996.

[58] Zeng, X., Bagrodia, R., and Gerla, M., "GloMoSim: a library for parallel simulation of large-scale wireless networks," in *Proceedings of the 12th Workshop on Parallel and Distributed Simulation*, May 1998.

[59] Zhuang, S., Lai, K., Stoica, I., Katz, R., and Shenker, S., "Host mobility using an internet indirection infrastructure," in *Proceedings of MobiSys'03*, May 2003.

VITA

Talal Mohamed Jaafar was born and grew up in Lebanon. He received his B.S. in Computer Engineering from Georgia Tech in 2001. He received his Master of Science and Doctor of Philosophy degrees in Electrical and Computer Engineering for Georgia Institute of Technology in 2004 and 2007, respectively.

Between January and August 2004, Talal worked as an intern with Optimi (Atlanta). In summer 2005, he worked as an intern with Dell (Austin), and in summer 2006 he worked as an intern with Cisco Systems (San Jose). He will start with Cisco in June 2007 as a full-time employee.