Reactive Routing in HIDRA Networks

A Senior Project
presented to
the Faculty of the Computer Engineering Program
California Polytechnic State University, San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Engineering

by
Scott Michael Marshall
March 2011

Abstract

In recent years, the Internet has grown so large that the future scalability of the Internet has become a major concern. The two primary scalability concerns are the size of the forwarding table and the ability for BGP to converge while distributing hundreds of thousands of routes.

HIDRA [5, 9] is a new Internet routing architecture that is backwards-compatible with existing routing technologies and protocols that focuses on feasibility-of-implementation. HIDRA remedies the first Internet scalability concern by proposing a means to reduce the number of entries in the default-free zone (DFZ) forwarding table.

This project extends HIDRA by designing a complete reactive routing implementation. This addresses the second Internet scalability concern by reducing the number of routes maintained by BGP. Adding reactive routing still allows for incremental deployment of HIDRA and also provides a means for finer route control with support for load-balancing and host mobility.

This paper provides background information on HIDRA, discusses reactive routing design details, and validates complete HIDRA behavior though a series of experimental tests and results.

Contents

1 Introduction **1**

2 Project Background **2**
2.1 Requirements 3
2.2 Personal Goals 3

3 HIDRA Background **4**
3.1 Overarching Project Goals 4
3.2 Fundamental Concepts 4
3.3 Addressing Scheme 5
3.4 Proactive Routing 5
3.5 Encapsulation 6
3.6 Decapsulation 6

4 Design Overview **6**
4.1 Related Work 7
4.2 Reactive Routing 8
4.3 ENCAP Records 9
4.4 The Circular Dependency 10

5 Design Details **11**
5.1 Failure Detection and Route Failover 11
5.2 Failure Recovery 12
5.3 Load Balancing 13
5.4 IP Mobility 14

6 Implementation **14**
6.1 Software Architecture Overview 14
6.2 Network Testbed 16

7 Experiments and Results **17**
7.1 First-Packet Latency Comparison 17
7.2 Link Failure 18
7.3 Load-Balanced Throughput 19
7.4 Failure Recovery 21
7.5 IP Mobility 22

8 Conclusion **22**

Appendices

A Glossary **23**

B References **24**

C Source Code **25**

D Analysis of Senior Project Design **28**
D.1 Summary of Functional Requirements 28
D.2 Primary Constraints 28
D.3 Economic 29
D.4 Environmental 29
D.5 Manufacturability 29
D.6 Sustainability 29
D.7 Ethical 30
D.8 Health and Safety 30
D.9 Social and Political 30
D.10 Development 30

List of Figures

1 HIDRA software design overview and packet processing flowchart. . . 15
2 Testbed network topology diagram 16
3 CDF of observed first-packet *roundtrip* latencies for “legacy” (*i.e.*non-HIDRA), proactive HIDRA, and reactive HIDRA network configurations within the testbed. A few reactive data points at 850 ms (starting around the 97^{th} percentile) are truncated to make the graph more readable. 17
4 Ping latencies between a host in AS1 and AS4 before and after link failure. The link between AS2 and AS4 was unplugged at time 30. Connectivity is restored, using the path through AS3, at time 36 in the legacy network and time 37 with reactive mapping. Missing data points indicate packets never reached their destination. 19
5 Comparison of HIDRA network throughput with and without reactive route load-balancing. 20
6 Reactive failure and recovery detection fall back on a slower route when the faster primary route becomes unavailable due to link failure. The left and right edges show traffic moving along the faster path, while the middle section shows traffic being forwarded along the slower path due to link failure. Transitions from one network path to another are nearly seamless. 21

List of Tables

1 Summary and brief description of the fields in the new DNS ENCAP resource record. 9
2 Summary of source code properties of complete HIDRA software stack. 25

1 Introduction

The Internet has shown considerable growth in recent years, so much so that the future scalability of the Internet has become a major concern. Most scalability concerns surround two major areas: the number of entries in the default-free zone (DFZ) forwarding table (FIB) and BGP [13]. More specifically, as the Internet expands, the number of entries in the DFZ FIB increases, which complicates routing policy and increases the cost of routers [3]. The convergence, performance, and stability of BGP is the second major area of concern because BGP must exchange an ever-increasing set of less-stable routes [15].

HIDRA [5, 9], a Hierarchical Inter-Domain Routing Architecture, is a new Internet routing architecture that was designed at Cal Poly. The goal of HIDRA is to reduce the size of the DFZ FIB. This is accomplished by splitting the Internet into a multi-level hierarchy using IPv4 encapsulation to move network packets between levels of the hierarchy. Unlike many other proposed Internet routing architectures, HIDRA's primary focus is real-world deployability. So, as a result, HIDRA is compatible with current network hardware, existing protocols such as IPv4 and BGP, existing network number allocation policies, and existing business policies and constraints. Furthermore, HIDRA is backwards-compatible with the current Internet routing architecture, which allows HIDRA to be deployed incrementally. Finally, tests have verified that HIDRA does in fact substantially reduce the number of entries in the DFZ FIB as well as curb the DFZ FIB's growth rate.

This project extends HIDRA by designing a complete reactive routing implementation, which adds reactive route mapping for all levels in the HIDRA network hierarchy below the root. Reactive routing addresses the second Internet scalability concern by reducing the number of routes BGP advertises by an order of magnitude. Reactive routing also enables additional complex routing features, such as route load-balancing and host mobility. This makes HIDRA a complete packaged solution to today's Internet scalability concerns. The majority of this report is focused on the design details of this new reactive routing implementation.

The addition of reactive routing to HIDRA preserves HIDRA's primary focus: real-world deployability. As with the initial HIDRA design, reactive routing does not require router hardware upgrades nor changes to router firmware. Rather, the reactive mapping protocol utilizes the existing DNS hardware and software infrastructure and management policies to distribute route mappings. This means that the only change required to enable reactive routing is adding the new route mappings to existing

DNS servers hosting the new mappings. Also, as with the initial HIDRA design, reactive routing can be deployed incrementally, as HIDRA fully supports operation on networks with BGP-based proactive routing, DNS-based reactive routing, or both.

While IP mobility was not a focus of this project, HIDRA's reactive routing implementation does provide a means for zero-stretch IP mobility. Therefore, the relationship between reactive routing and mobility is briefly discussed.

Several facets of the new reactive routing design have been tested on the HIDRA network testbed. These experimental tests include functional tests for normal behavior, route failure, and route recovery. Performance tests have also been completed to validate route load-balancing and to compare reactive routing to existing routing technologies.

The remainder of this report is organized as follows: Section 2 provides additional background information for this senior project, Section 3 provides the necessary background context for HIDRA to understand this project, Section 4 introduces reactive routing, Section 5 discusses the reactive routing design in detail, Section 6 details the overall HIDRA software architecture and the HIDRA network testbed, Section 7 shares experimental test results, and Section 8 concludes. Furthermore, subsequent appendices include a glossary, bibliography, source code, and analysis of this senior project design.

2 Project Background

Some insight into how this senior project began may be useful before delving into the technical details of HIDRA and the reactive routing design. When this senior project was started, HIDRA was operating correctly with proactive (*i.e.*BGP) route distribution, and some of the initial frameworks for reactive routing were in place from previous thesis work at Cal Poly [11]. More specifically, the new ENCAP resource record type was designed, modifications to the ISC BIND DNS server [4] had been completed, and basic resolver functionality was added to the core of the HIDRA software stack. However, there were areas of the implementation that were incomplete, which meant that were were limitations in the resolver, lack of complete support for IP mobility, no support for link failure and recovery, nor support for route load-balancing. Additional details on each of these technical areas will be provided later in this paper as needed.

2.1 Requirements

As indicated in the introductory paragraph to this section, some of the initial reactive routing components were already present within the HIDRA software stack when this senior project was started. The majority of this senior project focused on adding support for dynamic network changes to HIDRA's reactive routing stack. Therefore, the high-level requirements for this project were to:

- Revise HIDRA's packet routing engine and DNS resolver to remove reliance on a hard-coded root ASN and to move to "mixed-mode" HIDRA networks with reactive routing bootstrapped by a single proactive (*i.e.*BGP) route.
- Add support for dynamically detecting and routing around failures in the network to the reactive routing stack.
- Enhance reactive routing by adding support for quick detection and notification of route recovery.
- Fully utilize reactive routing parameters by adding support for load-balanced routes in HIDRA networks.

2.2 Personal Goals

Completing this senior project extended far beyond the technical goals listed in the previous section. Throughout the senior project process, I set several personal goals for myself:

- Learn about the research process in preparation for pursuing a M.S. Computer Science degree in graduate school.
- Become familiar with cutting-edge network technology and networks-related research.
- Learn how to develop and debug Linux kernel modules, and become familiar with the Linux kernel's networking stack.
- Understand the conference paper writing process and submit a conference paper on HIDRA [9].
- Enhance my LaTeXdocument writing skills.

As subsequent sections of this report indicate, I have accomplished all of these personal goals throughout the lifespan of this project.

3 HIDRA Background

The reactive routing mechanisms designed here are not specific to HIDRA, but they were developed within the contex of HIDRA. So, providing basic background information regarding HIDRA provides a context and groundwork for the remainder of this paper. Readers are encouraged to read the original HIDRA paper [5] for additional details.

3.1 Overarching Project Goals

As indicated in the Introduction, HIDRA was designed to solve one of the two primary concerns regarding Internet scalability: the ever-increasing number of entries in the DFZ FIB. However, in proposing a new routing architecture for the Internet, it was critical to resolve the second major scalability concern: the ability for BGP to converge as larger numbers of routes are advertised and exchanged. One solution to the BGP scalability problem is to move away from proactive BGP routing to reactive routing, which is the approach taken here.

3.2 Fundamental Concepts

HIDRA is a hierarchical network architecture with a minimum of two levels, with Level 0 (designated L_0) at the top of the hierarchy. ISPs are typically members of both L_0 and the level below, Level 1 (designated L_1). As members of both levels, ISPs are responsible for forwarding packets across levels of the hierarchy and transmitting packets across L_0. In contrast, end-site networks (both stubs and multi-homed) are typically members of L_1 only, and reach L_0 through their upstream provider(s).

To move data up the hierarchy (*e.g.*L_1 to L_0), a packet is encapsulated, wherein an additional IPv4 header is added to the front of the packet. So, a packet originating in one L_1 network destined for another L_1 network will have two IPv4 headers as it crosses the L_0 segment of the network, with the outermost header designating the destination L_0 network that contains the destination L_1 address. Upon reaching the L_0 destination ISP, the packet is decapsulated, such that the outer IPv4 header is discarded. At this point, the packet is forwarded to the L_1 destination. This process can be extended for a hierarchy with more than two levels by adding additional encapsulation and decapsulation steps. This general network architecture, called "map-encap" [12], is not new, and has been discussed in other papers [17, 6].

3.3 Addressing Scheme

In a simple two-level configuration, the network hierarchy can be seen as having a location-identifier split [6], with L_0 as the *location* and L_1 as the *identifier*. Understanding this basic split is critical to the approach taken for L_0 addressing in HIDRA.

HIDRA uses IPv4 as the L_0 addressing protocol and must correctly map a destination L_1 address into the encapsulating IPv4 L_0 address. To determine a node's location in the network (*i.e.*its L_0 address), HIDRA uses the autonomous system number (ASN). ASNs are already allocated and managed by the existing Internet Assigned Numbers Authority (IANA) infrastructure, so using ASNs leverages existing number allocation policy.

To form a L_0 IPv4 address, a well-known /8 prefix is combined with the 24 least-significant bits of the ASN. Outright, this choice yields two benefits. First, all L_0 traffic can be easily identified, filtered, and routed using a single rule for the well-known /8 prefix. Second, it is easier to map arbitrary L_1 addresses to L_0 addresses, since the mapping process must simply map a L_1 address to an ASN.

3.4 Proactive Routing

Proactive routing distributes the L_1-to-ASN mapping table *before* any of the mappings are needed, and continues to do so in response to network topology changes (*e.g.*link failures). The primary benefit of proactive mapping is minimal first-packet-latency to a new L_1 destination since the mapping is already available in DRAM as a quick table lookup. The greatest drawback of proactive mapping is greater memory requirements to store the entire mapping table, even for mappings that are never used. Furthermore, proactive mapping results in increased network traffic due to the ongoing need to exchange route information.

HIDRA uses unmodified BGP [13] as the proactive routing protocol to exchange L_1-to-ASN mapping information. As with many other decisions made while designing HIDRA, BGP was selected for backwards-compatibility, particularly in regards to route advertisements and routing policy. Unmodified BGP is well-suited for HIDRA because it already exchanges all of the necessary information: the destination network prefix (*e.g.*L_1 addresses) and the ASN (*e.g.*L_0 address). HIDRA stores all proactive routing information in the routing table (RIB), but does not add it to the forwarding table (FIB), thereby reducing the number of DFZ FIB entries.

3.5 Encapsulation

HIDRA requires that an encapsulation device be present in each L_1 network to encapsulate packets before they traverse L_0. Undoubtedly, the first encapsulation device must be reachable by a packet's L_1 source without requiring any additional encapsulation. Furthermore, to avoid potential scalability problems, the encapsulation device should be as close to the originating host as possible. If the host encapsulates its own packets, this is termed "host-based encapsulation".

When proactive routing is used, "host-based encapsulation" is not feasible, since each host would require active BGP sessions with nearby routers to obtain the L_1-to-ASN mappings. In these scenarios, a network-based encapsulation device is required, and in the future, potential router firmware upgrades could allow border routers to act as HIDRA encapsulation devices.

If the reactive routing technology described in this paper is used, encapsulation devices need not run BGP sessions, and instead, use simple DNS [10] queries to obtain L_1-to-ASN mappings. This makes "host-based encapsulation" a realistic and wise implementation choice.

3.6 Decapsulation

Decapsulation is a far simpler process in comparison to encapsulation, since a lookup operation is not required. Rather, the outermost IPv4 header is removed and discarded, leaving the L_1 header intact. Note, however, that the decapsulation process must occur when the packet enters the L_1 network's nearest upstream provider (the "last-hop" L_0 network) and not within the L_1 network itself. This allows the resulting packet to travel the best L_1 path to its final destination.

It is worth noting that this decapsulation plan does result in an asymmetry in the network. More specifically, encapsulation is done close to the source host, but decapsulation is done at the destination ISP.

4 Design Overview

With the foundations of HIDRA in place, the discussion can now turn to the core focus of this senior project: reactive routing. This section provides details of the existing reactive routing framework present in HIDRA before this project began. Upon completion of this section, the reader will have enough context to sufficiently

understand the design details for route failure, route recovery, and load-balancing in HIDRA networks that are presented in Section 5.

4.1 Related Work

There are a number of research papers that address various aspects of reactive routing, and many of these aspects are similar to those added to HIDRA. However, unlike many of the routing architectures proposed in the existing research papers, HIDRA strives to provide a realistic, incremental migration strategy from the current Internet routing architecture to the proposed alternative. This goal separates HIDRA from the existing works.

Feamster *et. al.* [7] identify many of the benefits of and motivation for using reactive routing. However, their work primarily focuses on peer-to-peer networks and is not directly applicable to HIDRA.

In [19], Yang *et. al.* describe NIRA, A New Inter-Domain Routing Architecture. The proposed reactive routing scheme uses rate-limited ICMP messages as a means to notify network nodes of a change in route availability. This approach is similar to that used in HIDRA. Adoption of NIRA may be impeded due to the new proactive and reactive routing protocols proposed.

In the paper "Architectural Principles of the Internet" [2], Carpenter documents several principals for the Internet. One of the principals, "Circular dependencies must be avoided", supports HIDRA's use of "mixed-mode" networking with a proactive route bootstraping reactive routing to avoid circular dependencies.

The Tunneling Route Reduction Protocol (TRRP) [16] does propose the use of DNS [10] as a means to globally distribute gateway addresses. Doing so would provide end-users reactive routing capabilities via a reverse DNS lookup on the destination IP address. However, the use of a generic DNS TXT record rather than a more-specific resource record type may impede adoption. Another proposal [1] also proposes using DNS TXT records to distribute routing information. HIDRA's creation of the new ENCAP resource record encompasses these ideas.

In Hierarchical Architecture for Internet Routing (HAIR) [8], Feldmann *et. al.* describes concepts similar to those used in HIDRA, such as a multi-level hierarchy. However, HAIR's use of IPv6 and an alternate reactive mapper make it different than HIDRA.

4.2 Reactive Routing

In contrast to proactive routing, which was explained in Section 3.4, reactive routing does not distribute any encapsulation mappings in advance. Rather, a query operation for the encapsulation mapping is performed the first time a route is used, and it is subsequently cached for future use. The benefit of reactive routing in comparison to proactive routing is that memory is not wasted storing routes that are never used. The downside to reactive routing is greater first-packet-latencies as the initial query operation is performed. However, since the mappings are cached, latencies of subsequent packets are equal in reactive and proactive scenarios.

Long-term, every host should have reactive mapping software installed to enable "host-based encapsulation". Also note that some advanced reactive routing features, such as IP mobility, may require "host-based encapsulation" to be functional. This allows HIDRA to still be deployed incrementally without requiring immediate adoption of reactive routing.

DNS [10] is used to store and distribute destination-specific encapsulation mappings. There are innumerable benefits from choosing DNS for this duty, ranging from administrative policies, to existing wide-spread deployment, to existing use in establishing most Internet connections, to security.

A major roadblock in proposing a new Internet routing architecture is preserving an organization's ability to have control over their own resources. In the case of DNS, administrative control is already delegated to organizations by a central authority. Organizations can change their DNS entries freely, which results in delegated and distributed autonomous control. If a new reactive routing protocol were proposed, it would have to meet similar administrative requirements.

A second benefit for using DNS is that it is already widely deployed and is well-understood by network administrators. Since DNS is already critical for most Internet activities and is used in establishing most Internet connections, it is easy to "piggy-back" encapsulation mappings onto existing DNS server responses. Furthermore, since local DNS servers cache query results for low latency response times (even for response types they do not understand), the performance of local encapsulation devices will benefit from these existing fast, local caches.

Finally, DNS can be made secure through DNSSEC [18]. DNSSEC allows records to be signed, which prevents against spoofing attacks.

Reactive routing in HIDRA could use a storage and distribution system other than DNS, but there is no reason to design an entirely new system that would need to meet the same set of requirements.

One assumption made in the preceding paragraphs was that a forward DNS query would always be issued for a L_1 destination, and that the result for that same query would also include the required encapsulation mapping. It is worth noting that HIDRA does also support returning encapsulation mappings in response to a reverse lookup on the destination L_1 address, so a forward lookup is not required.

4.3 ENCAP Records

While some previous proposals [16, 1] recommend using the DNS TXT resource record type to store encapsulation mappings, HIDRA defines a new DNS resource record type called the ENCAP record to store and distribute encapsulation mappings. As noted in Section 4.2, ENCAP records are returned in response to both forward and reverse DNS queries. The six fields of the ENCAP record are summarized in Table 1.

Field	**Size**	**Description**
Priority	1 Byte	Expresses the order in which ENCAP records are used.
Preference	1 Byte	When multiple ENCAP records exist with the same priority, traffic is load-balanced between them weighted by the preference value.
Level	1 Byte	Level of encapsulation. Used to obtain ENCAP records for all levels in the hierarchy with a single query.
Flags	1 Byte	Reserved for future use.
Type	2 Bytes	The type of the encapsulation data.
Data	Variable Length	Encapsulation data. Specific format determined by type field.

Table 1: Summary and brief description of the fields in the new DNS ENCAP resource record.

The first two 1-byte fields, *Priority* and *Preference*, provide network administrators the ability to customize encapsulation decisions based on route failover and load-balancing. When several ENCAP records are available for the same encapsulation level, the record with the highest *Priority* is used first. If there are several records of equally-high *Priority*, then the *Preference* field is used as a weight for load-balancing.

The third 1-byte field, *Level*, defines the level in the hierarchy for which the ENCAP record applies. The 1-byte *Flags* field is reserved for future use.

The 2-byte *Type* field designates the type and size of the data stored in the variable-length *Data* field. The *Data* field specifies the encapsulation address and is of variable length because encapsulation addresses can be specified several ways: 4 bytes for an ASN or IPv4 address, or 16 bytes for an IPv6 address.

4.4 The Circular Dependency

A circular dependency is created by HIDRA's reactive routing system. A reactive lookup requires a DNS query packet to be sent to a remote DNS server, which is typically in a different L_1 network. This is usually the case because most ASes do not run their own *root* DNS server. However, to reach the remote DNS server, the original DNS query packet must be encapsulated. To perform the encapsulation by mapping the remote DNS server's L_1 address into a L_0 address, yet another DNS query for encapsulation data must be issued. This completes the cycle that causes the circular dependency.

Surprisingly, DNS already has a similar circular dependency. A local DNS resolver needs to know the IP address of the *root* nameserver before any lookups can be performed. To avoid this problem, local resolvers use a "hints" file which provides this information. Once the *root* nameserver has been contacted, it can provide the "glue" address data for subsequent DNS servers down the name resolution chain.

There are two ways to avoid the circular dependency in HIDRA. The first option is to include ENCAP records alongside the A records for the *root* nameservers inside the "hints" file. Since this is likely too impractical to deploy, a second solution is proposed. HIDRA already functions correctly with routes distributed proactively via BGP. The goal is to reduce the number of routes advertised by BGP, but not to eradicate BGP completely. So, BGP can be used to proactively distribute the encapsulation state for, and forward packets to, the *root* nameservers. The *root* nameservers then provide the encapsulation "glue" for subsequent DNS servers, so that all other encapsulation mappings are obtained reactively. This is termed a "mixed-mode" network, where the critical infrastructure routes are handled proactively and all other routes are handled reactively. "Mixed-mode" networking breaks the circular dependency because the mapping for the *root* nameserver's L_0 address does not depend on a DNS lookup.

5 Design Details

Completing HIDRA's reactive routing implementation involved several small design changes and three major feature additions. This section solely focuses on the design details for the three major feature editions: dynamic failure detection, dynamic failure recovery, and route load-balancing.

5.1 Failure Detection and Route Failover

Detecting failures in networks using reactive routing is significantly more difficult than in networks using proactive routing. In networks with BGP-based proactive routing, link failures are automatically detected and routed around within tens of seconds[1]. In contrast, networks using reactive routing typically rely on timeouts of cached encapsulation mappings as a means to route around failures. When a cached mapping expires, it is discarded. If another packet for the same destination is encapsulated, a reactive lookup is performed and a new mapping is used. To achieve the same failure detection performance as networks using proactive routing, cache timeouts can be set small. Unfortunately, this leads to poor performance since the benefits from caching are lost.

The failure detection design for reactive routing in HIDRA networks does not rely on timeouts and has performance on par[2] with networks using proactive routing. Fast failure detection times are achieved by imposing an additional requirement on encapsulation devices in a HIDRA network. More specifically, encapsulation devices are required to notify the L_1 source of any packet destined for a L_1 network that is no longer accessible. To accomplish this, each encapsulation device must store a small amount of additional state information. Encapsulation devices already know the set of *present* L_1 routes through the iBGP peering session(s) they establish with their L_1 router(s). Now, encapsulation devices must also keep track of *past* L_1 routes (L_1 routes that were present at one time but are no longer available) in a *stale routes table*. Before a packet is encapsulated, the encapsulation device checks the packet's source and destination L_1 addresses. If the source address is not part of the encapsulation device's *present* L_1 network **and** the destination address is a part of a *past* L_1 network, then a "Stale L_1 Route" ICMP Redirect [14] packet is returned to the L_1 source of the packet.

[1]Exact response times will vary depending on the BGP settings in use on the routers responsible for detecting the failure and propagating the update message(s).

[2]Quantitative results are available in Section 7.2.

The L_1 source address of the ICMP packet is the encapsulation device that detected the problem. The L_1 destination is the L_1 source of the packet that triggered the redirect. The ICMP packet contains the ASN of the encapsulation device that detected the problem. A new ICMP Redirect subcode is used to differentiate HIDRA's "Stale L_1 Route" redirect from the other redirects currently in use.

When a node receives a "Stale L_1 Route" ICMP Redirect packet, it marks the encapsulation mapping as unavailable. The unavailable mapping is identified by examining the ASN within the ICMP Redirect message payload and the destination address found in the IP header within the payload of the ICMP packet. In accordance with ENCAP record (Section 4.3) policies, if the newly-unavailable mapping is the only remaining mapping with the highest *Priority*, the next highest priority set of mappings is used. If, after marking the current entry as unavailable, there are more mappings with the same *Priority*, only the remaining valid mappings are considered.

5.2 Failure Recovery

Quickly handling link recovery situations in networks with reactive routing is as difficult as quickly detecting failures. Like failure detection, typical recovery detection algorithms rely on timeouts. This, again, leads to a trade-off between responsiveness and network overhead.

The link recovery algorithm designed for reactive routing in HIDRA networks expands upon the ideas in the failure detection design described in Section 5.1. The recovery algorithm places additional requirements on encapsulation devices and requires significantly more state information to be stored in *recovery notification table*. Unlike the failure detection algorithm, which must store one entry for each *past* L_1 route in the *stale routes table*, the recovery algorithm must store pairs of <*past* L_1 route, L_1 address of transmitter>in the *recovery notification table*.

It is important for an encapsulation device to distinguish between *new* L_1 routes and *restored* L_1 routes. When an encapsulation device receives a route advertisement via iBGP, it checks to see if the advertised route is in the list of *past* L_1 routes recorded in the *stale routes table*. If so, then the encapsulation device knows that the route has been *restored*, and the route is removed from the *stale routes table*. Otherwise, the encapsulation device knows that the route is a *new* route. Since the L_1 route notifications are maintained via iBGP, route change detection (both failure and recovery) occurs shortly after link states change.

The recovery algorithm works as a two step process. First, the encapsulation

device saves a list of transmitters it must notify of a route recovery. The encapsulation devices do this by adding a record for the transmitter to the *recovery notification table* each time the device notifies a new transmitter that a L_1 route is unavailable with the "Stale L_1 Route" ICMP Redirect packet (Section 5.1). Then, once the L_1 route is *restored*, the encapsulation device notifies each transmitter in the *recovery notification table* associated with that L_1 route that the route is available by sending a "Recovery Notification" ICMP Redirect packet.

The L_1 source address of the ICMP packet is the encapsulation device sending the route recovery notification. The L_1 destination address is the L_1 address of the node that was sent the original "Stale L_1 Route" ICMP Redirect in response to a route failure. The ICMP packet payload contains the ASN of the encapsulation device that detected the problem and the L_1 address that is now available. A new ICMP Redirect subcode is used to differentiate HIDRA's "Recovery Notification" redirect from both HIDRA's "Stale L_1 Route" redirect and from the other redirects currently in use.

When a node receives a "Recovery Notification" ICMP Redirect packet, it marks the encapsulation mapping as available. The mapping is identified by examining the ASN and L_1 address within the ICMP Redirect message payload. In accordance with ENCAP record (Section 4.3) policies, if the newly-available mapping has a higher *Priority* than that of the existing mapping in use, the new mapping is used. If, after marking the current entry as available, there are more mappings with the same *Priority*, all valid mappings with the same *Priority* are considered.

5.3 Load Balancing

The design of the ENCAP record (Section 4.3) for HIDRA's reactive routing implementation opens the door to complex features and technologies not always seen in traditional routing, such as load balancing. Given that the design for the ENCAP record was established before this senior project was started, many of the fundamental pieces for load balancing were already in place.

The load balancing design used by HIDRA uses a simple weighted preference algorithm. When a reactive lookup is performed and several ENCAP records with equally-high *Priority* values are available, load balancing is used. More specifically, HIDRA's packet forwarding engine uses the *Preference* field from each ENCAP record to assign weights to each encapsulation mapping. Then, as packets are encapsulated and forwarded, the set of available encapsulation mappings are cycled through according to their weights.

5.4 IP Mobility

Enhancing HIDRA's route selection engine to support load balancing also corrected existing support for IP mobility. IP mobility was not a focus of this senior project, so it will not be discussed in great depth here. Readers interested in additional details regarding support for zero-stretch IP mobility in HIDRA are encouraged to read the original thesis on mobility or the most recent HIDRA paper [11, 9].

A critical requirement for successful IP mobility is that a host must be addressable by the same identifier, independent of its location within the network (in the same, or different, AS). HIDRA supports zero-stretch mobility through the addition of a third level to the routing and addressing hierarchy. At the inside of the encapsulation hierarchy is L_2, the mobile node's constant identifier. The L_1 address is assigned by the network to which the mobile host is currently connected, and serves as a part of the mobile node's location identifier. The outermost layer is L_0, the network address for L_1, which completes the mobile node's location identifier.

As the mobile node moves from network to network, the L_2 address remains constant, while L_1 (and thus, L_0) will change. As detailed in Section 3, HIDRA already supports efficient packet forwarding to L_1 addresses, satisfying basic requirements for IP mobility.

6 Implementation

All of the software additions and changes required to complete the reactive routing implementation and features described in this report were applied to the existing HIDRA prototype software stack. The primary goal of the HIDRA prototype is to validate correct functionality from proposed theory and to gain basic insight into the overhead involved in encapsulation and reactive mapping. It is worth remembering that since the work is a prototype, the intent is not to optimize for top performance.

6.1 Software Architecture Overview

This section provides a high-level overview of the entire HIDRA software stack. For listings of source code for this senior project, see Appendix C.

Figure 1 details the overall architecture and packet flow of the current HIDRA prototype software stack. HIDRA includes both a Linux kernel module and a user-space daemon. The kernel module is primarily used for fast encapsulation, decapsulation, and packet forwarding. The kernel module processes packets quickly by leveraging

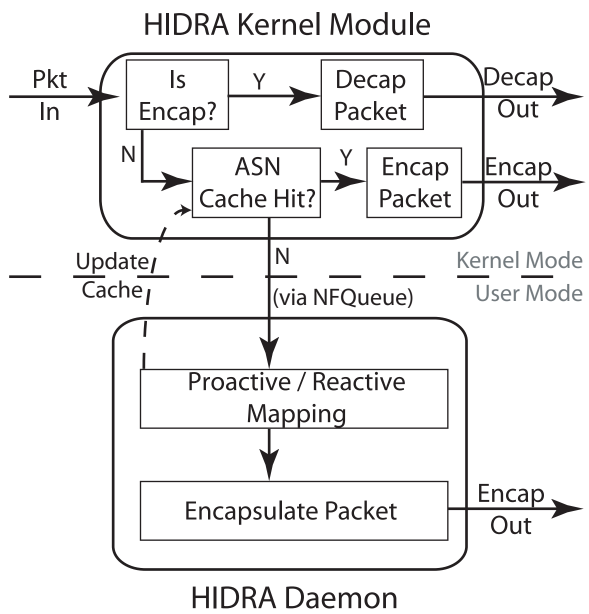

Figure 1: HIDRA software design overview and packet processing flowchart.

cached encapsulation mappings that have been obtained by the user-space daemon. Once a mapping is cached, future packets sent to the same destination need not make the slow kernel→user-space trip. The daemon is responsible for establishing iBGP peering sessions with local L_1 routers, obtaining encapsulation mappings from DNS in the form of ENCAP records (Section 4.3), and managing the list of *present* and *past* L_1 routes for recovery notifications. To control packet flow in and out of a Linux host running the HIDRA software stack, the daemon creates a series of `iptables` rules.

The HIDRA kernel module and daemon interact three ways: through `IOCTL` calls, shared memory, and packets via NFQUEUE. The `IOCTL` calls are used to check the status of the HIDRA kernel module, to delete cached encapsulation mappings, and to manage the *recovery notification table*, which is stored in shared memory. NFQUEUE is used to push packets from the kernel module to user-space for additional processing, such as the "Stale L_1 Route" and "Recovery Notification" ICMP Redirects (Sections 5.1 and 5.2).

A DNS server needed modifications to support the new ENCAP resource record, including returning the appropriate "glue" records in response to NS queries. As a part of previous HIDRA work [11], ISC BIND [4] was modified to include these features.

To add complete reactive routing support to HIDRA, several substantial changes were required. The existing resolver present in HIDRA's user-space daemon was

modified to support "mixed-mode" configurations, so that reactive routing functions correctly if BGP only advertises a single /32 route for the *root* nameserver. The resolver and packet forwarding engine were also modified to support multiple hierarchy levels as well as multiple mappings for the same L_1 destination to support load balancing (Section 5.3) and IP mobility (Section 5.4).

To support quick failure and recovery detection, the user-space daemon was modified to track the list of *past* L_1 routes. To enable the appropriate triggers to send the "Stale L_1 Route" and "Recovery Notification" ICMP Redirects (Sections 5.1 and 5.2), `iptables` rules using the `MARK` operation were designed. Finally, both the kernel module and daemon were modified to appropriately respond to the new ICMP Redirects by adjusting the set of available encapsulation mappings appropriately.

6.2 Network Testbed

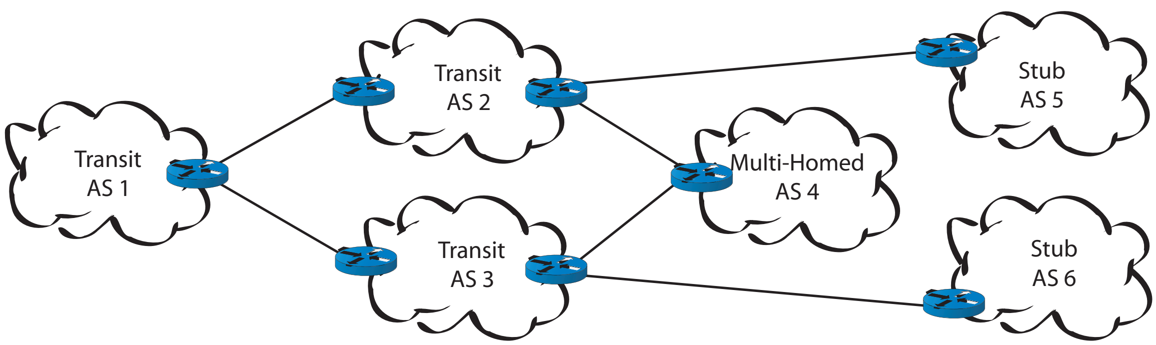

Figure 2: Testbed network topology diagram

The HIDRA network testbed is composed of six ASes, using Cisco routers, Cisco switches, and Linux workstations as the encapsulation devices and test hosts. An AS-level network topology diagram for the HIDRA network testbed is shown in Figure 2. Network connectivity between devices is either 100Base-T or 1000Base-T. HIDRA operates using IPv4 addressing for both L_0 and L_1, though separate IPv6 connectivity is available as a control plane. This network is private and does not play a role in routing traffic across the commodity Internet.

A DNS server acting as the *root* nameserver is located in AS1, while two authoritative top level domain (TLD) servers are located in AS2 and AS3. No redundant DNS servers are present. The network testbed does operate in a "mixed-mode" fashion, with a single proactive /32 IPv4 encapsulation mapping **only** for the *root* DNS server advertised via BGP. All other mappings were only distributed using reactive routing.

7 Experiments and Results

Several experiments were run on the HIDRA network testbed (Section 6.2) to test for correct functionality and to gain insight into the overhead caused by encapsulation and reactive routing.

7.1 First-Packet Latency Comparison

The first test performed compares the first-packet latency of three network configurations: "legacy" networks (*i.e.*the current Internet architecture), HIDRA networks with proactive routing (*i.e.*BGP), and HIDRA networks with reactive routing.

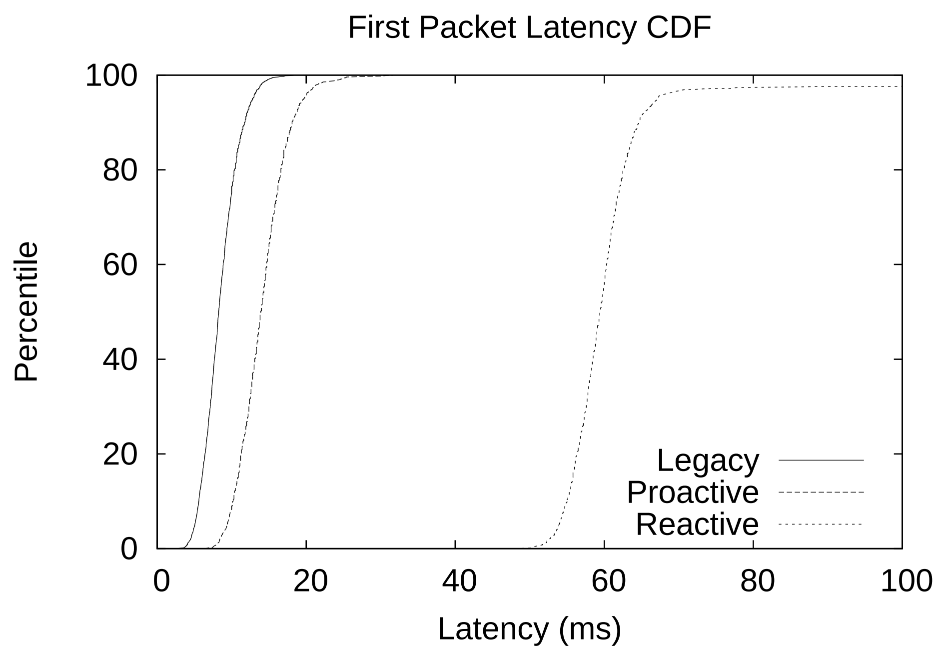

Figure 3: CDF of observed first-packet *roundtrip* latencies for "legacy" (*i.e.*non-HIDRA), proactive HIDRA, and reactive HIDRA network configurations within the testbed. A few reactive data points at 850 ms (starting around the 97^{th} percentile) are truncated to make the graph more readable.

In this experiment, 2000 ICMP Echo [14] (*i.e.*ping) packets were sent to a host in AS5 from a host in AS6. All encapsulation and decapsulation operations were performed by hosts in AS2 and AS3. Before each ping packet was sent, all caches (ARP, DNS server, DNS resolver, HIDRA kernel module, etc.) were cleared on each of the hosts. Figure 3 is a CDF plot comparing all 2000 *roundtrip* first-packet ping latencies.

The difference in packet latency for the "legacy" and proactive HIDRA network configurations is roughly *5.5ms*. This difference is attributable to both the longer path traveled by the packet (though the encapsulation/decapsulation hosts) as well as CPU time spent on the encapsulation and decapsulation operations. The added *45.5ms* latency between the proactive and reactive network configurations is a consequence of the reactive lookup and mapping process, which involves issuing recursive DNS queries to resolve L_0 destination addresses. It is worth pointing out that some of the added first-packet latency is caused by **two** sets of lookup/encapsulate operations: those for the original ping packet **and** those for the recursive DNS query and response packets themselves. Also, note that *one-way* latencies account for slightly more than half of the *roundtrip* latencies due to caching of encapsulation mappings as the ping packet traverses the network.

Only relative time values are meaningful, as absolute time values can vary greatly depending on the CPU power of encapsulation/decapsulation hosts, the capabilities of the network equipment in use, and the size of the network.

7.2 Link Failure

This test verifies that HIDRA's reactive routing implementation can properly route around link failures that cause changes to encapsulation mappings. The test also quantifies how quickly HIDRA responds to these failures.

To test HIDRA network operation in spite of failures, a steady stream of ping packets, one per second, was sent from AS1 to AS4. The *Priority* field of the ENCAP records used in this experiment were adjusted so that under normal network conditions, the packet stream would traverse the path AS1→AS2→AS4. After the 30^{th} ping packet was sent and the response was received, the network cable between the routers in AS2 and AS4 was unplugged. The consequences of this failure are seen as packet loss (*i.e.*no data points) in Figure 4 starting at time 31. As the figure shows, packet loss continues until the network repairs itself, at time 36 for the "legacy" network and at time 37 for the reactive HIDRA network. This additional delay is inherent in reactive routing, as the "Stale L_1 Route" ICMP Redirect (Section 5.1) packet cannot be sent until *after* BGP acknowledges the L_1 route changes.

The higher latency of the 37^{th} reactive packet is a direct result of HIDRA's design. As indicated by the packet flow annotations in Figure 1, under normal network conditions, all encapsulation operations are handled by the HIDRA kernel module via cached encapsulation mappings. However, when the ICMP Redirect packet is

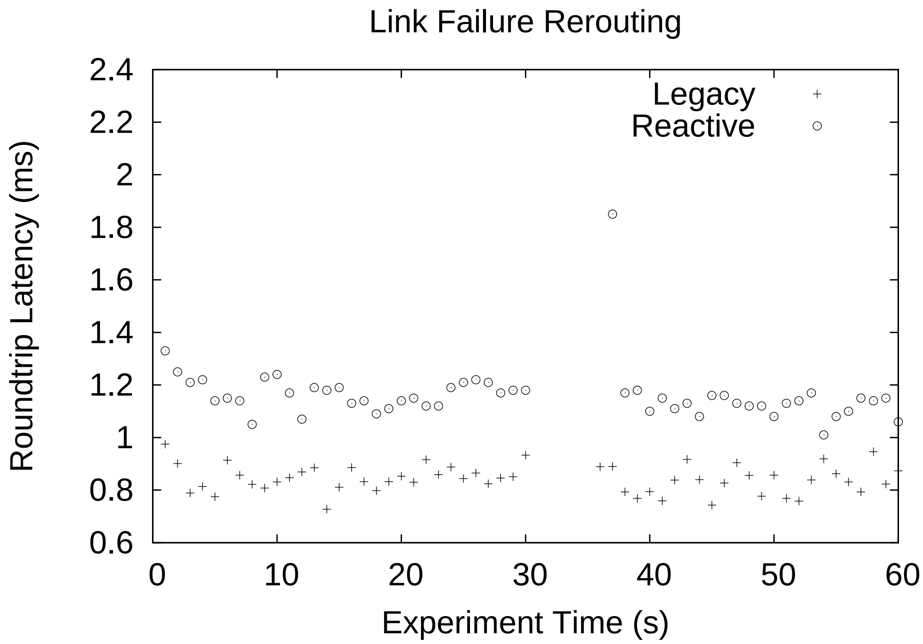

Figure 4: Ping latencies between a host in AS1 and AS4 before and after link failure. The link between AS2 and AS4 was unplugged at time 30. Connectivity is restored, using the path through AS3, at time 36 in the legacy network and time 37 with reactive mapping. Missing data points indicate packets never reached their destination.

received, the cached encapsulation mapping is flushed, causing the next packet that requires encapsulation to the same L_1 destination to be processed by HIDRA's user-space daemon. This additional packet travel and processing time causes the higher latency of the 37^{th} packet. Once the packet is processed by the daemon, a new encapsulation mapping is stored in the kernel cache, reducing the latency of subsequent packets.

Also, note that since there was a lower *Priority*, but still valid, ENCAP record (for encapsulation via AS3) returned in response to the original DNS query issued by the HIDRA daemon, an additional DNS query was not required to determine the new encapsulation mapping for the L_0 destination of the 37^{th} packet. Had the DNS query been necessary, the latency of the packet would have been even higher.

7.3 Load-Balanced Throughput

The purpose of this experiment was to validate the new route load-balancing feature. For this test, the AS2$\longleftrightarrow$AS4 and AS3$\longleftrightarrow$AS4 network links were configured to operate at only 10 Mbps. This configuration change was made because the current

HIDRA network testbed cannot fully saturate a 100 Mbps link due to the CPU overhead involved with packet encapsulation and decapsulation.

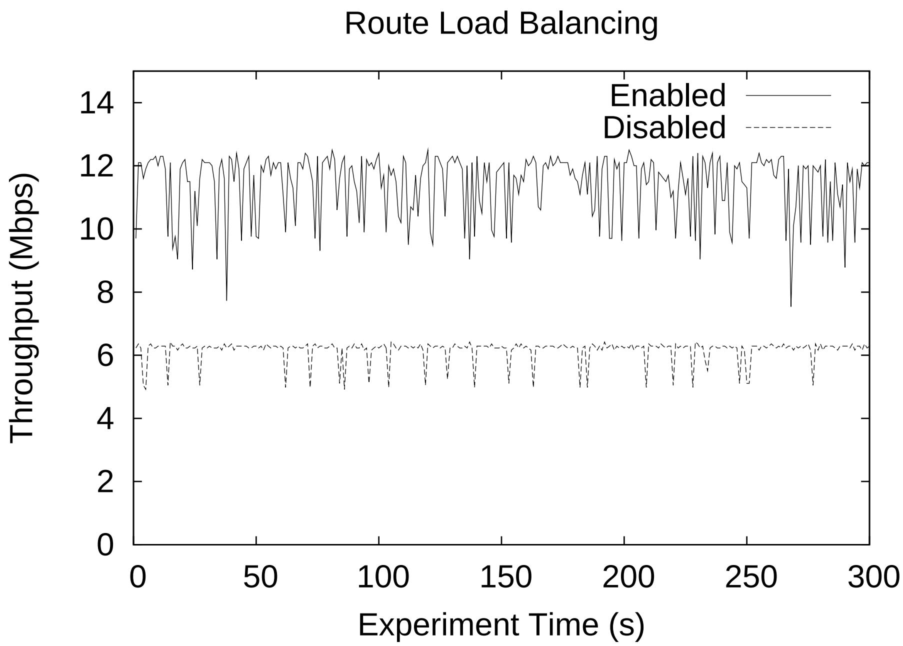

Figure 5: Comparison of HIDRA network throughput with and without reactive route load-balancing.

To measure throughput with and without load-balancing, the `iperf` tool was used to send a data stream across the network from a host in AS1 to a host in AS4 for 300 seconds. First, network throughput with load-balancing disabled was measured. To achieve this, the *Priority* field was adjusted to always route traffic along the AS1→AS2→AS4 path. To enable route load-balancing, the *Priority* fields of the two encapsulation mappings for the host in AS4 were set equal, so that packets would travel via AS2 or via AS3. However, the *Preference* fields were set to two different values. This is the case because "return traffic" originating from AS4 in the HIDRA network testbed is configured to always follow the path AS4→AS2→AS1 and never the path AS4→AS3→AS1. To ensure that the links via AS2 and AS3 were utilized equally, the *Preference* values were set with a ratio of 1:2, so that twice as many "outbound" packets from AS1 would travel the AS1→AS3→AS4 path as the AS1→AS2→AS4 path. These throughput results are presented in Figure 5.

The experimental data shows that enabling load-balancing has a significant impact on network performance. When load-balancing was disabled, network throughput averaged *6.17 Mbps*, while when load-balancing was enabled, network throughput

averaged *11.4 Mbps*. This equates to roughly an **85% increase** in network throughput.

7.4 Failure Recovery

A novel approach was taken to verifying failure recovery in reactive routing. For this experiment, the AS2⟷AS4 network link was configured for 100 Mbps operation, while the AS3⟷AS4 was configured for 10 Mbps operation. The *Priority* fields were adjusted to route traffic along the faster AS1→AS2→AS4 path by default.

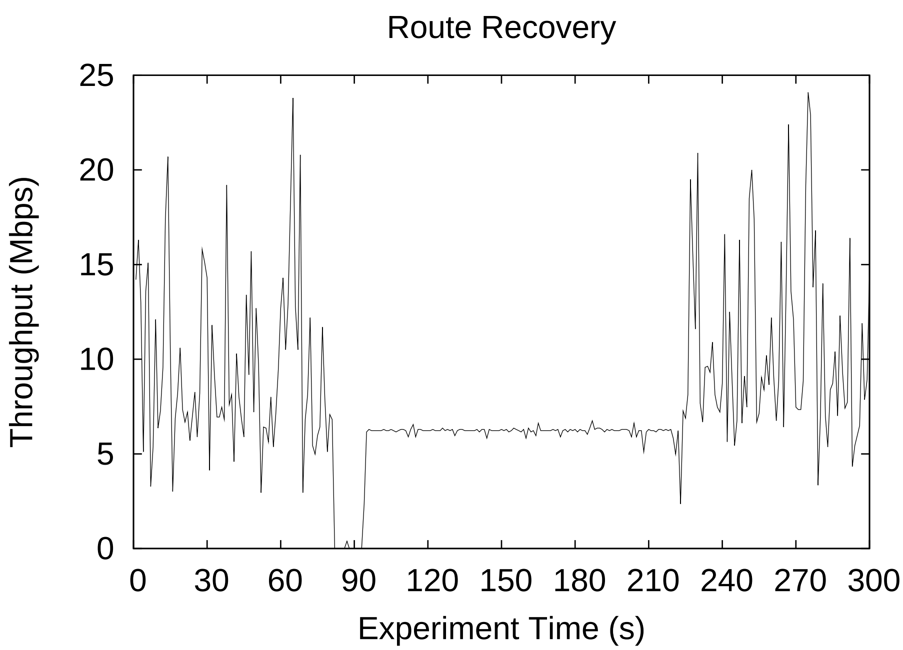

Figure 6: Reactive failure and recovery detection fall back on a slower route when the faster primary route becomes unavailable due to link failure. The left and right edges show traffic moving along the faster path, while the middle section shows traffic being forwarded along the slower path due to link failure. Transitions from one network path to another are nearly seamless.

The `iperf` tool was used to generate a steady data stream from a host in AS1 to a host in AS4. Initially, when all links were up, packets traveled the 100 Mbps AS1→AS2→AS4 path, as expected. This can be seen on the left side of Figure 6. At approximately 80 seconds into the test, the AS2⟷AS4 network link was taken down. From 80 seconds to 93 seconds, no packets reached AS4 while HIDRA's failure detection algorithm (Section 5.1) worked to route around the link failure. At 93 seconds, traffic began to flow once again, now along the slower 10 Mbps AS1→AS3→AS4

path. This is seen in Figure 6 as the middle segment with significantly lower average throughput. Once a total of approximately 180 seconds had passed from the test start, the AS2⟷AS4 network link was brought back up. Roughly 42 seconds later, at time 222 seconds, the failure recovery was complete and the reactive mappings switched back to the faster 100 Mbps AS1→AS2→AS4 path that was issued a higher *Priority* value. This is seen on the right side of Figure 6.

7.5 IP Mobility

Complete zero-stretch IP mobility support for HIDRA networks using reactive routing was enabled as a part of the software development efforts involved with this senior project. However, as previously mentioned, IP mobility was not a focus of this senior project. Readers interested in IP mobility metrics are encouraged to read the original thesis on mobility or the most recent HIDRA paper [11, 9].

8 Conclusion

Adding complete reactive routing to HIDRA was an interesting, yet challenging, senior project. The skills acquired and lessons learned span a wide range of areas, including: kernel debugging techniques, avoiding race conditions when shared memory is used, developing complex `IOCTL` interfaces, designing informative experiments to verify claims made in theory, and writing a conference paper. The tests and results presented in Section 7 indicate that each and every one of the project requirements detailed in Section 2.1 have been met.

HIDRA has become more than just a new architecture for Internet routing with a goal of reducing the number of entries in the DFZ FIB. The addition of reactive routing to HIDRA now allows it to address the second Internet scalability concern: the ability for BGP to converge when increasing numbers of less-stable route advertisements are being made. HIDRA's support for existing network hardware (without the need for firmware upgrades), an incremental deployment strategy, and "mixed-mode" networks makes it a plausible and viable replacement for the Internet routing technologies in use today.

Appendices

A Glossary

This glossary provides a brief listing of the technical terms and acronyms used throughout this paper.

AS Autonomous System: A collection of network devices and hosts managed by a single entity with established business and routing policies.

ASN Autonomous System Number: The unique identifying number assigned to each Autonomous System on the Internet.

BGP Border Gateway Protocol: A proactive routing protocol used to advertise routes between routers.

DFZ Default-Free Zone: The group of Internet autonomous systems whose routers do not require a default route to forward any packet to any destination.

DNS Domain Name System: A hierarchical Internet naming system used to store mappings between names and IP addresses.

FIB Forwarding Information Base (Forwarding Table): A table present in routers that provides quick access to packet forwarding decisions.

HIDRA Hierarchical Inter-Domain Routing Architecture: The proposed Internet routing solution to present-day Internet scalability concerns.

"Host-Based Encapsulation" When individual network nodes issue reactive lookups and encapsulate their own traffic.

ISP Internet Service Provider: A company that provides Internet connectivity to consumers and businesses.

"Mixed-Mode" A HIDRA network configuration wherein reactive and proactive routing are used together to avoid the circular dependency (Section 4.4).

RIB Routing Information Base (Routing Table): A table present in routers that stores route information made available by routing protocols, such as BGP.

Stretch A ratio of the actual network path length compared to the optimal network path length.

B References

[1] RFC 1383 (rfc1383) - an experiment in DNS based IP routing. http://www.faqs.org/rfcs/rfc1383.html.

[2] RFC 1958 - architectural principles of the internet. b. carpenter, ed.. http://rfc.sunsite.dk/rfc/rfc1958.html.

[3] A proposal for scalable internet routing & addressing. http://tools.ietf.org/html/draft-wang-ietf-efit-00.

[4] ISC Bind. https://www.isc.org/software/bind.

[5] Bryan Clevenger, Daniel Nelson, and John M. Bellardo. Hidra: Hierarchical inter-domain routing architecture. In *Proceedings of the 2010 Information and Communication Technology Conference*, 2010.

[6] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/ID separation protocol (LISP). http://tools.ietf.org/html/draft-farinacci-lisp-12.

[7] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek. Measuring the effects of internet path faults on reactive routing. In *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 126–137. ACM of New York, NY, USA, 2003.

[8] Anja Feldmann, Luca Cittadini, Wolfgang Mühlbauer, Randy Bush, and Olaf Maennel. Hair: hierarchical architecture for internet routing. In *Proceedings of the 2009 workshop on Re-architecting the internet*, ReArch '09, pages 43–48, New York, NY, USA, 2009. ACM.

[9] Scott M. Marshall, John M. Bellardo, Daniel Nelson, and Bryan Clevenger. Reactive encapsulation mappings in hidra. In *Proceedings of the 2011 Internet and Multimedia Systems and Applications Conference*, 2011.

[10] P. Mockapetris and K. J. Dunlap. Development of the domain name system. In *Symposium proceedings on Communications architectures and protocols*, SIGCOMM '88, pages 123–133, New York, NY, USA, 1988. ACM.

[11] Daniel Nelson. Internet scale reactive routing and mobility. 2009.

[12] RFC 1955 -new scheme for internet routing and addressing (encaps) for ipng. http://tools.ietf.org/html/rfc1955.

[13] RFC 4271 - a border gateway protocol 4 (BGP-4). http://tools.ietf.org/html/rfc4271.

[14] RFC 792 - INTERNET CONTROL MESSAGE PROTOCOL. http://tools.ietf.org/html/rfc792.

[15] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica. Towards a next generation inter-domain routing protocol. In *Proceedings of Third Workshop on Hot Topics in Networks (HotNets-III)*, 2004.

[16] TRRP. http://bill.herrin.us/network/trrp.html.

[17] P. F. Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. *ACM SIGCOMM Computer Communication Review*, 18(4):35–42, 1988.

[18] various. Collection of rfcs that define dnssec. http://www.dnssec.net/rfc.

[19] X.. Yang, D.. Clark, and A.W. Berger. Nira: A new inter-domain routing architecture. *Networking, IEEE/ACM Transactions on*, 15(4):775 –788, August 2007.

C Source Code

A summary of properties for the HIDRA sources is listed in Table 2. The complete source code for the HIDRA software stack is nearly 8000 lines of code and will not be included here. However, select fragments of the source code written as part of this senior project follow.

Component	**Lines of Code**	**Language**
Kernel Module	1241	C
User-Space Daemon	6720	A mixture of C and C++

Table 2: Summary of source code properties of complete HIDRA software stack.

Three code fragments critical to reactive routing in HIDRA are included here. The first fragment is a function that is called to filter out invalid ENCAP records from a reactive lookup, the second fragment shows user-space handling of the HIDRA ICMP Redirects, and the third shows kernel-level handling of the HIDRA ICMP Redirects.

- ENCAP Record Filtering

```
void reactive_dst_callback(void *response, int num_responses, void *data)
{
   PacketInfo *pkt = (PacketInfo*)data;
   const struct sniff_ip *ip = (struct sniff_ip*)pkt->payload();
   struct encap *resp = (struct encap*)response;
   int i;
   PacketInfo::LookupStatus status;
   int curr_strata, curr_prio;

   status = pkt->dstLookupStatus;
   curr_strata = curr_prio = -1;
   pkt->numStrata = 0;

   if (0 == num_responses) {
      printf("NO reactive response for ip dst %s!\n", inet_ntoa(ip->ip_dst));
      status = PacketInfo::ERR;
   }
   else {
      for (i = 0; i < num_responses; i++) {
         /* Immediately skip over encap records for stale routes */
         if(isStaleRoute(ip->ip_dst, resp[i].data)) {
            continue;
         }

         /* Not stale. Dump if equal strata but different prio */
         if(resp[i].strata == curr_strata && resp[i].prio != curr_prio) {
            continue;
         }

         /* Now, either strata differ, or quanta are same AND prio are same. */
         pkt->destAddrs.push_back(resp[i]);

         /* If strata differ, update trackers and inc packet's numStrata counter */
         if(resp[i].strata != curr_strata) {
            curr_strata = resp[i].strata;
            curr_prio = resp[i].prio;
            (pkt->numStrata)++;
         }
```

```
      }

      if ((resp->type) == ENCAP_ASN_TYPE) {
         pkt->asnDst = htonl(resp->data);
      }
      else {
         assert(0);
      }
      if(pkt->destAddrs.size() > 0) { // Need at least one dest to be valid
         status = PacketInfo::DONE;
      } else {
         status = PacketInfo::ERR;
      }
   }

   pthread_mutex_lock(&pkt->statusMutex);
   pkt->dstLookupStatus = status;
   int savePkt = check_and_transmit(pkt);
   pthread_mutex_unlock(&pkt->statusMutex);
   if (!savePkt)
      delete pkt;
}
```

- ICMP Redirect Handling in User-Space

```
if(icmp->icmp_code == ICMP_REDIR_HIDRA_STALE_ENCAP) {
   // Ensure this route is no longer used during future encap ops
   struct sniff_ip *msg = (struct sniff_ip*)((char *)icmp +
     sizeof(struct sniff_icmp));
   struct in_addr msg_dst = msg->ip_dst;
   // Add to stale routes
   addStaleRoute(msg_dst, *msg_asn);
   }
else if(icmp->icmp_code == ICMP_REDIR_HIDRA_L1_RECOVERY) {
   // Remove from stale routes
   uint32_t *staleL1 = msg_asn + 1;
   struct in_addr staleL1_addr;
   staleL1_addr.s_addr = *staleL1;
   delStaleRoute(staleL1_addr, *msg_asn);
```

```
}

// Drop the ICMP packet. Don't want it anymore.
return nfq_set_verdict(qh, id, NF_DROP, 0, NULL);
```

- ICMP Redirect Handling in the Kernel Module

```
if(icmp->type == ICMP_REDIRECT && (icmp->code == ICMP_REDIR_HIDRA_STALE_ENCAP ||
  icmp->code == ICMP_REDIR_HIDRA_L1_RECOVERY) &&
  ntohl(iph->daddr) == kmem->myIP) {
   // Mark the inbound packet so iptables can push it to userspace
   skb->mark = MARK_HIDRA_REDIR_L1;
   // Remove the attempted destination from the kernel's routing cache
   if(icmp->code == ICMP_REDIR_HIDRA_STALE_ENCAP) {
      deleteCacheEntry(msgHdr->daddr);
   } else if(icmp->code == ICMP_REDIR_HIDRA_L1_RECOVERY) {
      staleL1 = *((uint32_t *)(msgHdr));
      deleteCacheEntry(staleL1); // Payload holds the no-longer-staleL1 addr
   }
   // printk("Marked incoming ICMP HIDRA Redirect\n");
   return XT_CONTINUE;
}
```

D Analysis of Senior Project Design

This appendix addresses questions raised by the Computer Engineering department to analyze student senior projects.

D.1 Summary of Functional Requirements

This project adds reactive routing functionality, with complete support for route load-balancing as well as dynamic network failure and recovery to HIDRA. This allows DNS to be used in place of BGP to distribute mappings for route encapsulation in HIDRA networks.

D.2 Primary Constraints

The greatest challenges of this project stem from the fact that is extremely difficult to debug kernel modules. So, when HIDRA systems started to exhibit kernel panics,

tools such as `gdb` could not be used to diagnose the source of the panics. I also think that the sheer fact this project is networked does make this project more difficult, as perfectly reproducing test cases and other events is difficult when multiple devices are involved. This project did not have any strict constraints or parameters that impacted my approach to solving the problem.

D.3 Economic

No hardware or software was explicitly purchased to complete my senior project. The project does rely on several Cisco routers, Cisco switches, and Linux workstations though. When the equipment was new, it was probably worth tens of thousands of dollars in total. It is difficult to estimate the total development time for this project, but it involved approximately 6-12 hours a week spread over the duration of two quarters.

D.4 Environmental

There are no direct environmental impacts associated with this project. However, since emphasis in computer networks has always been on performance, network devices (switches, routers, etc.) typically are not designed to be environmentally-friendly. The software modifications in this senior project have no impact (positive or negative) on the environmental characteristics of HIDRA networks.

D.5 Manufacturability

The technology industry has experience manufacturing routers, switches, and desktop PCs. No issues exist in this area for this senior project.

D.6 Sustainability

When designing a new routing protocol for the Internet, it is difficult to support sustainability. This is the case because instituting a new global standard means that specifications, expectations, and behaviors of the design cannot change once the standard is in place. So, the project **cannot** be upgraded once it is in use. The technical community has already seen this in the slow move from IPv4 to IPv6. With this project, most changes are in software, so the impact on global resources is negligible.

D.7 Ethical

This senior project does not have any ethical implications. However, like the Internet routing technology used today, malicious users may attempt to use the routing protocol to redirect Internet traffic for illegal or ethically-inappropriate purposes. For example, a thief could attempt to redirect all network traffic destined for an international bank in hopes of acquiring bank account details and stealing customer funds. To help avoid malicious misuse of ENCAP records, DNSSEC technology can be used to protect these DNS records.

D.8 Health and Safety

The reactive routing additions to HIDRA do not impact the well-being of society. The software changes and new protocol design do not impact the way in which current routers and switches are used by humans.

D.9 Social and Political

This reactive routing design, and HIDRA as a whole, are proposals to change the way world-wide Internet routing works. Naturally, then, for global consensus to be reached, technical committees composed of experts from around the world must debate what is best for society as a whole. When smaller groups of people make decisions that change things on a global scale, there can always be societal and political impacts. However, the reactive routing design itself is not directly associated with any political concerns.

D.10 Development

During the course of this senior project, I learned how to better use several development tools and development techniques. First and foremost, I learned how to properly debug a Linux kernel module, as traditional tools like `gdb` cannot be used for this purpose. I also became extremely familiar with many of the Cisco IOS commands required to configure interfaces, routing, and BGP on Cisco routers. Along the same lines, I learned how design new rules and chains for `iptables` to redirect network packets on a Linux host. Finally, I also learned how to use `tcpdump` and how to better utilize Wireshark to analyze and debug network traffic.