HIDRA: HIERARCHICAL INTER-DOMAIN ROUTING ARCHITECTURE

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Bryan Clevenger

April 2010

© 2010

Bryan Clevenger

ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: HIDRA: Hierarchical Inter-Domain Routing Architecture

AUTHOR: Bryan Clevenger

DATE SUBMITTED: April 2010

COMMITTEE CHAIR: John Bellardo, Ph.D.

COMMITTEE MEMBER: Clark Turner, Ph.D.

COMMITTEE MEMBER: Hugh Smith, Ph.D.

Abstract

HIDRA: Hierarchical Inter-Domain Routing Architecture

Bryan Clevenger

As the Internet continues to expand, the global default-free zone (DFZ) forwarding table has begun to grow faster than hardware can economically keep pace with. Various policies are in place to mitigate this growth rate, but current projections indicate policy alone is inadequate. As such, a number of technical solutions have been proposed. This work builds on many of these proposed solutions, and furthers the debate surrounding the resolution to this problem. It discusses several design decisions necessary to any proposed solution, and based on these tradeoffs it proposes a Hierarchical Inter-Domain Routing Architecture - HIDRA, a comprehensive architecture with a plausible deployment scenario. The architecture uses a locator/identifier split encapsulation scheme to attenuate both the immediate size of the DFZ forwarding table, and the projected growth rate. This solution is based off the usage of an already existing number allocation policy - Autonomous System Numbers (ASNs). HIDRA has been deployed to a sandbox network in a proof-of-concept test, yielding promising results.

Contents

List of Tables	viii
List of Figures	ix
1 Introduction	1
2 History	3
2.1 Initial Development	3
2.2 CIDR	4
2.3 Current Situation	6
2.4 Standard Routing Metrics	6
3 Previous Work	8
3.1 IETF & RRG	8
3.1.1 Tunneling Route Reduction Protocol	9
3.1.2 Locator/ID Separation Protocol	10
3.1.3 A Practical Transit Mapping Service	11
3.1.4 Core Router-Integrated Overlay	13
3.1.5 IP With Virtual Link Extension	15
3.1.6 Internet Vastly Improved Plumbing Architecture	15
3.1.7 Six/One Router	16
3.1.8 IPv6 Dual Homing	18
3.2 Other Proposals	20
3.2.1 Hierarchical Architecture for Internet Routing	20
3.2.2 ViAggre	21
3.2.3 Shim6	22

3.2.4	NIRA	22
3.2.5	IPNL	23
3.2.6	HLP	24
3.3	Summary	25
4	HIDRA Framework	26
4.1	Overview	26
4.2	Forwarding Information Base vs. Routing Information Base . . .	28
4.3	L_0 Addresses	29
4.4	Encapsulation	30
4.4.1	Proactive Mapping	30
4.4.2	Reactive Mapping	33
4.5	Decapsulation	34
4.6	Routing	35
4.6.1	Effect of RIB vs. FIB	39
4.6.2	Multi-homed Networks	39
4.7	Projected Impact on DFZ Table Size	41
4.8	Load Balancing	43
4.9	Migration Strategy	44
5	Metrics	46
5.1	Applying Metrics	46
5.1.1	General Requirements	48
6	HIDRA Implementation	51
6.1	Implementation Specific Design Decisions	51
6.2	Testbed Configuration	53
6.3	Software Implementation	53
6.3.1	ICMP	56
6.3.2	MTU Errors	64
7	Experiments	66
7.1	Standard Connection Tests	69
7.2	In-Network Encapsulation Test	70

7.3	Failure Tests	72
7.4	Policy Tests	74
8	Future Work	84
9	Conclusion	85
	Bibliography	87

List of Tables

- 5.1 HIDRA affects many areas of the requirements stated in RFC 5773.
For more information about the individual requirements, see [10]. 47

List of Figures

4.1	Standard IPv4 packet header fields, and example encapsulation. The address 10.2.255.1 is mapped to AS 60553, yielding a mapped HIDRA L_0 address of 1.0.236.117.	28
4.2	A multi-site multi-homed network. From the standpoint of AS1, encapsulation can occur at point (1) if the AS itself provides encapsulation, point (2) if its upstream provider encapsulates, or at the originating end-host if a reactive routing scheme is used. Decapsulation can occur either at point (1) for end-site decapsulation, or point (3) for ISP decapsulation. (3) provides for slower DFZ FIB growth and better fail-over characteristics.	31
4.3	Example prefixes and ASNs ($\rightsquigarrow$) advertised in a small multi-homed network using legacy BGP, HIDRA with end site decapsulation, and HIDRA with ISP decapsulation. Routes in bold are the L_0 FIB. Duplicate L_0 routes are not in bold. All other routes are L_1 . $\dagger$ – Route originated with a private ASN.	38
4.4	Historical size of the DFZ table. The “Legacy DFZ Size” shows the historical and current size of the DFZ table. The “ASNs Announced” line shows the number of unique ASNs visible on the public Internet. “Transit ASNs” is the number of unique ASes that appear in the middle of at least one BGP AS path. The former AS count approximates customer decapsulation and the later ISP decapsulation. Graph (b) focuses on the ASN data. All data is from the public potaroo [27] web site from the RouteViews [38] vantage point. For readability only every hundredth data point is plotted in the graph.	42
6.1	The network testbed used to evaluate HIDRA.	53

6.2	Block diagram depicting the interaction between the HIDRA kernel module and the HIDRA daemon. Initial implementation was done for Linux.	55
6.3	A simple 2-AS HIDRA network, where each router and its connected hardware is in it's own AS.	58
6.4	The initial hops of a traceroute across the example network. These TTL Exceededs produce normal, unencapsulated ICMP errors responding to unencapsulated ICMP packets.	59
6.5	The packet has just been encapsulated and sent out to be routed across the network, but the TTL exceeds on its next hop.	60
6.6	The generated ICMP error message from point (3) in figure 6.5. Consists of 3 distinct components: the outside IP header, the ICMP header, and the ICMP payload. The “- - -” notation indicates the value of the field is not pertinent to the current example.	61
6.7	The encapsulated packet times out, outside of its own AS. At this point, the ICMP error response is generated, and sent to the source location in the original packet's L_0 header, which is the L_1 address of the original encapsulation device.	62
6.8	The packet is now decapsulated, and in standard “legacy” format. When generating an error message that needs to go back across the network, it needs to be encapsulated. The receiving end does not need to take any special action.	63
7.1	High Level ASN & Network Configurations	67
7.2	IPv4 addresses for endpoints in the testbed network	68
7.3	Ping latencies for HIDRA with end-site decapsulation vs. legacy routing.	69
7.4	Ping latencies for HIDRA between a host in AS4 and AS6 from figure 6.1. Compares legacy, end-host encapsulation, within AS encapsulation, and ISP encapsulation.	70
7.5	Ping latencies for HIDRA with manually induced failure after 30 packets.	72
7.6	Load balancing traffic paths. The packets originate at the same location, and end in the same AS, but based on the L_1 subnet the packet is destined for, it takes a different path.	75

7.7	On initial link failure, the originator still has the route entry in its BGP table, so the initial outgoing packet originated in AS 1 goes along the same path.	77
7.8	The packet is decapsulated, and sent out to be routed to its L_1 location. The desired L_1 network is no longer accessible though, and so it's sent back to the encapsulation device.	79
7.9	The packet is re-encapsulated, this time with the already updated BGP table. It is then routed around to AS 3, where it is decapsulated and delivered successfully to the customer network.	81
7.10	Packets are now routed directly through AS 3 to avoid the failed link.	83

Chapter 1

Introduction

The Internet originally began as a single connection between the TX-2 computer in Massachusetts and the Q-32 in California, talking over a low speed dial-up telephone in 1965 [32]. Since that modest beginning, the Internet has grown enormously, and along with it so have the paradigms and practices we associate with it.

One unfortunate result of this growth is that the way in which we route data from a source to a destination across the network has also grown at an alarming rate. In fact, today's forwarding table in core Internet routers has upwards of 300,000 entries [27], and is growing at a rate faster than the hardware can keep up with [4]. As such, many researchers and organizations have begun searching for a way to help alleviate the growing problem [4, 48].

This work is another step in further addressing the problem of a growing forwarding table. Recent work appears to be migrating towards a strategy known as a “locator/identifier split” [15, 17, 30, 48]. This work utilizes the concept

of a locator/identifier split, and proposes a Hierarchical Inter-Domain Routing Architecture (HIDRA). The main goal of HIDRA is to reduce both the immediate size of the Default Free Zone (DFZ) forwarding table, and to reduce its future growth rate. One of the major concerns driving the development of HIDRA is practical deployability; while many proposals suggest re-inventing the wheel and coming up with an all-around better system, the idea is simply not practical given the requirements of today’s Internet, as demonstrated by the fairly low market penetration of IPv6 ¹ [18, 37].

In addition to attempting to provide as smooth a transition process as possible, HIDRA also provides flexibility for future development and features that simply cannot be applied to the Internet as it currently stands [36]. Given the general agreement throughout the research community that a fairly fundamental change needs to be applied to our current routing system [4], it would be short-sighted not to attempt to build as robust and forward-looking solution as is practical.

¹A study by PC World shows IPv6 traffic at its peak in 2008 “representing less than one hundredth of 1% of Internet traffic” [37]

Chapter 2

History

This chapter will set the background for the current routing scheme. Specifically, it will develop how CIDR came to be, and what impact that has had on our current situation. Finally, some basic networking metrics and standards that will be used to judge the utility of various proposals will be outlined.

2.1 Initial Development

Originally, the Internet consisted of a single administrative network [9, 3]. As time went by, more people became aware of both the existence and the utility of such a network, and began building their own independent networks. By the time the concept was commercialized into private service providers in the early 1980's, several different networks, owned and controlled by different people, but still interconnected, already existed [3]. These networks had been built up in an ad-hoc manner, and as such, even today it can be stated that “Internet routing

often appears to be running close to the limits of stability” [9].

Eventually, separately owned individual routing domains were assigned specific, unique numbers and eventually became what are now designated as “Autonomous Systems” [9]. This domain concept has become what may be considered a fundamental axiom of today’s routing theory [9].

This organizational architecture has proved fairly robust especially because it allows the use of multiple (and different) protocols in various places, both inter-domain and intra-domain [9, 3].

2.2 CIDR

Initial addressing of the Internet split the standard Internet Protocol (IP) space into three categories [28]. Each of these categories was referred to as a “Class,” with half of the address space falling into Class A, or a /8 network which utilized 8 bits of the 32 bit address to denote the network, and the other 24 to specify a host. Class B and Class C categories each received a quarter of the total address space, and used /16 and /24 addressing techniques respectively. As such, Class C networks had the smallest amount of hosts.

Unfortunately, as what would soon become to be known as “the Internet” grew, so did the demand for Class C & B networks [28]. Even more unfortunately, the size disparity between a Class C and Class B network was huge (256 hosts vs. 65,535 hosts), and many of the desired networks required between one thousand and two thousand nodes, making a Class C too small and a Class B far too big for the purposes; as such, the typical solution was to use a number of Class

C networks for a single location. This standard practice led to the inflation of routing tables [28].

It soon became clear that the growth rate was easily outstripping the capability increases in the routing hardware, and so in the early 1990's several RFCs were submitted outlining Classless Inter-Domain Routing (CIDR) [28, 3, 25, 42, 21, 20, 8, 23, 7].

The adoption of CIDR in 1995 created the ability to allocate net blocks of any size [25, 42, 21, 28]. This gives a flexibility to the distribution of networks that was never before seen, and instead of having to register multiple smaller networks or use one large network address space only sparsely the address space could now be used much more effeciently. As a result, the number of routes in routing tables would be much more accurate without the redundancy of multiply registered C class domains.

This in turn led Internet Service Provider (ISP) route aggregation, where each ISP acts as a "Supernet," and is issued a large block of addresses that it then parcels out in smaller chunks to individual customers [20, 28]. Now, only the ISP specific advertisements needed to be global forwarding table, drastically reducing both its size and growth rate.

In the short term, CIDR was able to greatly reduce the number of routes in the DFZ forwarding table. Unfortunately, once again the route table growth is begining to stress the limits of the current hardware.

2.3 Current Situation

The application of CIDR given its original goals thus far has worked as expected. Unfortunately, the Internet is growing at such a rapid pace that the routing tables are expanding faster than the hardware technology tasked with storing the necessary routes can be improved. Currently, DFZ routing tables contain upwards of 300,000 entries [27]. While our hardware can handle current loads, the growth rate appears to be exponential, which is the true reason for the current concern [4].

2.4 Standard Routing Metrics

Routing has many metrics that can be used to help quantify a particular protocol. The most pertinent of these were recently revised by the Internet Routing Task Force (IRTF) in section 3 of RFC 5773 [10, 33, 12]. RFC 5773 and its predecessors outline and categorize the expectations and requirements for a routing protocol. This document contains a brief outline of the expectations and requirements.

The general requirements category refers to the correctness and scope of a protocol. Listed first in this category is that a protocol must provide “timely routing to all reachable destinations.” Additionally, a protocol must also be able to provide notification of failed services, be designed with growth in mind, and provide an environment that can be operated autonomously with minimal required resources [10].

The functional requirements category refers to somewhat more technical issues. Some examples of these types of requirements are that a protocol route

around failures dynamically, provide loop free paths, provide configurable paths based off outside preferences such as business agreements, and decouple inter- and intra-domain systems [10].

These requirements are mostly set forth in RFC 1126, and revised in RFC 5773 [33, 12, 10]. In addition to listing the requirement and expectations for a routing protocol, RFC 5773 also provides an assessment of how the current system accomplishes (or whether it accomplishes) these goals.

These metrics must be applied to any proposed solution to the up and coming routing problems in order to provide an accurate evaluation of the solution. The basics of these standards are applied to HIDRA in section 5.1.

Chapter 3

Previous Work

3.1 IETF & RRG

While many people have addressed routing issues for a long period of time, the looming problem concerning the ability to effectively upgrade router hardware to keep pace with the expansion of the forwarding table has caused a somewhat more focused effort than might typically be expected. In order to help catalyze and focus the effort, the Internet Engineering Task Force (IETF) chartered the Routing Research Group [48] (RRG) to “to explore routing and addressing problems that are important to the development of the Internet but are not yet mature enough for engineering work within the IETF.”

The IETF RRG is currently entertaining and discussing many different proposals, all targeted at creating or modifying the existing Internet architecture to

improve some aspect of the current system. Several of the proposals are discussed below.

3.1.1 Tunneling Route Reduction Protocol

Tunneling Route Reduction Protocol (TRRP) [53] is a similar proposal to HIDRA in terms of both goals and implementation. TRRP also proposes encapsulating traffic in an IPv4 packet that is compatible with the preexisting router base, and adding encapsulation/decapsulation devices to the network edges. However, TRRP requires a reactive routing protocol, even for initial deployment. TRRP uses DNS as that reactive scheme to create a mapping between hostname and destination address. While HIDRA can be, and is eventually expected to be expanded to incorporate a similar reactive scheme, it is not immediately required to realize benefits. This extension to HIDRA is demonstrated in [36].

TRRP suggests a design for the new DNS resource, stored in the TXT record. The record would consist of multiple entries, each entry containing a priority, a type, and a route. The suggested format is: pp,ii,route where pp represents a 2 digit hex value for priority, ii is a 2 character protocol identifier, and route is the appropriate addressing information, such as 192.168.100.1. Initial protocols would consist of dr, indicating flat routing, g4 for standard IPv4 GRE encapsulation, and g6 for standard IPv6 GRE encapsulation.

Because the specification is general, it leaves room for future protocols. In addition, because each record can contain more than one entry, it provides for multiple ways to reach a destination. In the event that the requesting encapsulation device does not understand a protocol, it simply ignores the entry and

moves on until it finds one it understands, with the highest priority.

Additionally, TRRP requires the use of an existing, preassigned IPv4 address as the endpoint of the tunnel. In contrast, HIDRA uses a new addressing scheme to quickly differentiate between different layers of the hierarchy, and derives endpoints directly from already existing number allocation policy.

3.1.2 Locator/ID Separation Protocol

Another very active proposal in the RRG is LISP: Locator/ID Separation Protocol [15]. This is another proposal that uses encapsulation to deliver packets across the Internet. LISP defines the specific packet formats, including the use of UDP encapsulation and reactive lookup ¹.

As initially envisioned, LISP injects itself into the current standard by taking packets from the IP layer up, and encapsulating them in another UDP packet with a special LISP prefix header. This process is expected to be done in a location past the end host, meaning the end host stack remains unaltered; encapsulation routers are expected to be responsible for all tunneling of the newly created packets over the Internet.

While LISP does not explicitly define a mapping protocol, it does provides behavioral constraints for an adequate mapping protocol. LISP-ALT [15] is an instance of LISP that creates an overlay network using GRE tunnels [16] with multi-protocol BGP peering sessions inside the tunnels. This overlay is used for the reactive lookup portion of LISP.

Unlike HIDRA, LISP's reactive routing uses new, untested protocols which may impeded any adoption process. Like TRRP, LISP does not support a purely

¹For more on reactive vs. proactive lookups, see section 4.4

proactive scheme.

3.1.3 A Practical Transit Mapping Service

APT: A Practical Tunneling Architecture [5] proposes addressing the scaling issue by eliminating edge networks from the core tables in an effort to reduce the stress being put on the core. The crux of this proposal is the distilling of used mappings versus total mappings in the routing tables. To do so, APT suggests a hybrid push-pull system to distribute mapping information, and data-driven notification for physical failures. Additionally, APT includes provisions for improved security in the form of a light-weight public-key distribution through its control protocol.

APT uses terminology for three new, or at least differently configured, pieces of hardware: an Ingress Tunnel Router (ITR), Egress Tunnel Router (ETR), and a Default Mapper (DM). An ITR and ETR can be combined into one physical device, denoted simply as a TR, and the three pieces of hardware are similar to that of the Encapsulation/Decapsulation devices utilized in HIDRA. In an APT architecture, a tunnel router is placed on the boundary between any edge network and its provider, on the provider side (since it is envisioned to give the providers the benefit). By doing so, it enables individual transit networks to deploy APT within themselves, making APT an incrementally deployable system. Additionally, each APT network requires at least one DM, but possibly more for a more distributed load.

This architecture works by tunneling traffic from an ITR to an ETR through a transit network. An ITR to ETR mapping is determined by the DM. DMs would be designed to contain all mappings, but each ITR would contain a cache for

individual mappings. In the event that an incoming packet incurs a cache miss, the ITR forwards the packet to the DM. The DM then selects the appropriate ETR, sends the packet on its way, and returns a MapRec that contains the ITR to ETR mapping back to the originating ITR. Each MapRec contains the ITR to ETR mapping, along with a cache idle timeout that denotes the period of time during which the record is valid.

Each DM may contain more than one possible ETR mapping that represents an ISP that serves the particular edge network. To support traffic engineering, each of these mappings also comes with a weight and a priority value. Because of this, when an ITR requests an ETR mapping, the DM will select one first according to priority, and in the event of a tie use each entry in proportion to its weight. This allows for the maintenance of any existing business policies and routing agreements. Each of these values, called a “MapSet” is advertised from the edge network owning the specific prefix.

An interesting attribute of this proposal is its unique handling for failure detection and recovery. Because an edge networks connectivity is reflected in a mapping table that does not adjust to physical failures, an ITR may attempt to tunnel packets to an ETR that has failed or lost connectivity to the edge network. In the event of a failure, packets heading to that ETR or failed edge network are redirected to a local DM via the (unspecified) intra-domain routing protocol. Once there, the DM will attempt to find an alternate ETR, as well as sending an appropriate failure message to the originating ITRs DM. Included in that failure message is a Time Before Retry value that is meant to keep the originating DM from using that ETR for a discrete period of time.

APT also specifies a specific protocol used to disseminate information between DMs, called Mesh Dissemination Protocol (MDP); this is the push part of the

architecture. All mapping information is pushed to the DMs throughout the transit network, creating a “DM Mesh.” This mesh will end up being congruent with AS topology, which once again facilitates incremental deployment. MDP functions very similarly to the way BGP currently functions, but contains the mapping information.

One additional consideration in MDP is the inclusion of a light-weight public-key mechanism. Each AS will include a public-key in its information dissemination. In order to prevent dissemination of faulty public-keys, each AS will additionally include its neighbors key in its dissemination. This way, an AS can determine the validity of an incoming key by the validity of its neighbors keys. The initial verification is done when two immediately neighboring ASes configure their DM connections through out of band communication, done offline. Keys are expected to have a finite time-to-live after which theyll expire. The protocol specifies ways for keys to be updated or revoked.

While APT does address the immediate issue of scalability, as the underlying number of networks grows, so must the number of tunneling routers at the edge, potentially recreating the problem currently being addressed.

3.1.4 Core Router-Integrated Overlay

Core Router-Integrated Overlay (CRIO) [60] is an architecture that makes a tradeoff between BGP table size, and route length. It touts ten to twenty times reduction in FIB size for a Tier 1 ISP, with marginal path length penalty. Through the use of tunneling, CRIO removes the requirement of topological contiguity, and instead introduces virtual prefixes which reduce the route table size. One large advantage to CRIO is it has no requirement for new hardware; all

changes can be implemented on currently existing hardware.

CRIO uses a basic encapsulation, like MPLS [57] or GRE [13], to create IP tunneling. Routers know which tunnel to select for a given IP prefix based on a distributed mapping. Each mapping entry contains an IP prefix, a tunnel endpoint, and possibly an optional policy. In order to support multi-homing, multiple entries with the same prefix may exist. This distributed mappings initial values are created through a dynamic routing protocol, like BGP, or setup statically. Once the initial values are configured for a local area, they are then distributed “through some external mechanism and installed” on other machines.

Once the mapping has been installed, packets are forwarded in a typical fashion, being encapsulated at an ingress router, and decapsulated at the egress router as determined by the mapping. An important note is that tunnels can be configured to be (and CRIO envisions they will be) one-ended tunnels. This means the tunnel endpoint accepts packets regardless of source, decapsulating them and forwarding them in a regular fashion. Because of this, initial tunnel configuration is greatly simplified.

Using the above mechanisms, a router can be configured to advertise every prefix within its virtual prefix; these routers are referred to as “VP-TSs”. A route is added to its virtual prefix through tunnels. As such, any ISP can independently determine which routers should be made a VP-TS. In general, its assumed that an ISP should not advertise virtual prefixes to any neighboring AS other than its customers to avoid carrying transit traffic for its peering, or possibly even upstream provider.

CRIO has been tested through static simulation on a Rocketfuel-measured Internet topology using traffic data from a Tier 1 ISP.

3.1.5 IP With Virtual Link Extension

IP with Virtual Link eXtension (IPvLX) [52] is a protocol that proposes using a combination of IPv4 and IPv6 to help alleviate scaling issues. This proposal suggests using IPv6 addresses mainly as identifiers, and IPv4 addresses mainly as locators, though neither of these classifications is strict. As a standard though, each IPv6 packet would be encapsulated in an IPv4 packet with IPvLX extensions.

Like several other proposals, IPvLX suggests using DNS to create a mapping between IPv6 identifiers and IPv4 locators, which would be an operation pushed onto the end host requiring full DNS support as a part of every end host routing stack. Using standard Locator Identifier split theory, only the locator addresses would need to be routed, allowing for a single IPv4 entry to represent many IPv6 end points.

Unfortunately, IPvLX does not present a transition plan to move from the current scheme to the proposed architecture, and initial assumptions seem to indicate that any such transition would not be smooth. Additionally, this proposal relies on heavy IPv6 adoption, which currently has not occurred [27].

3.1.6 Internet Vastly Improved Plumbing Architecture

Internet Vastly Improved Plumbing Architecture (IVIP) [29] is another core-edge separation based proposal. One of the key differences in this proposal is it's declared goal of eventually modifying the existing IP packet format to include extensions for a new routing system; effectively, IVIP proposes working changes into the existing architecture rather than (or perhaps in addition to) building on

top of it.

Though IVIP purports to provide benefits for all adopters, including even initial ones without full deployment of IVIP, the proposal is unclear about what exactly these benefits are or how the initial adoption would take place. Though IVIP has gone through a few revisions, the overall proposal is still relatively confusing, and not clearly layed out.

3.1.7 Six/One Router

The Six/One Router proposal [55] is based around the idea that current multi-homing and Provider Independent address space is what is causing the scalability issues of the Internet core routing system. It takes the tack that a way to resolve this issue is through the use of address indirection between provider-independent addresses at the edge of the network and aggregatable, provider-allocated addresses in the core.

Six/One Router attempts to avoid the standard pitfalls of address indirection: prolonged packet propagation latencies, extra bandwidth consumption, increased packet loss, etc, by a unique translation scheme of one-to-one transit addresses to edge addresses. While the theories and concepts presented in Six/One Router are applicable to IPv4, the author readily acknowledges that its reliance on a large address space forces this solution to be mainly applicable to IPv6, simply because there is not enough address space left in IPv4 to make it practical. This limitation in and of itself is fairly large, simply due to the relative lack of IPv6 adoption present in today's Internet [27].

This proposal envisions the addition of new routing hardware that does a direct translation of network source and destination address in IP packets. The

edge network on one side of the Six/One router would have its own specifically issued provider-independent address space, but the local network would be connected through a provider-dependent address. The Six/One router would do a direct translation on packets traveling across the network border, giving each individual provider-independent address, referred to as the “edge address,” a provider-dependent address issued by the ISP, referred to as the “transit address.”

Six/One Router depends on a DNS mapping of IP addresses to be available, but leaves this particular aspect as unspecified. Instead, it chooses to depend on other proposals to fill in that gap, such as APT Default Mappers [5] or DNS Map [54]. Given that a DNS mapping will be available, Six/One Router also intends to do a mapping of not only the source in the IP packets, but the destination as well. Given this combination, the idea is that multi-homing and general mobility are greatly increased. This increase is gained by the simple fact that multiple transit addresses can map to the same edge addresses, creating a network that is conducive to multi-homing.

In order to maintain backwards compatibility, Six/One Router also specifies the possibility for communication with non-upgraded edge networks. To enable this, Six/One Router makes transit addresses valid delivery points instead of limiting it to strictly edge addresses. Its able to accomplish this because of the unique one-to-one pairing of transit to edge addresses; this may seem unclear because the one-to-one mapping is not bi-directional. Specifically, one transit address always equates to one edge address, but one edge address could possibly be mapped to multiple transit addresses. Because each transit address can be mapped directly to a specific edge address, legacy networks would be supported by simply having them target the end transit address when doing DNS lookups.

In order to facility choosing the correct destination address, Six/One Router presents the idea of using an address prefix to denote the type of address, distinguishing between edge and transit addresses. Initially, the author considers a single leading bit in the address, but since there would be more transit addresses than edge addresses using this scheme, decides a single bit would be too inflexible. Instead, the author recommends creating a special edge address prefix, but gets no more specific than that.

3.1.8 IPv6 Dual Homing

IPv6 Dual Homing (V6DH) [11] seeks to reduce the pressure being put on the routing system, particularly in the Default Free Zone (DFZ). This solution is designed specifically for IPv6, in order to avoid making any tradeoffs for implementation on IPv4. The author expects IPv4 space to be treated as legacy.

V6DH proposes using a new addressing convention, rather than creating a new addressing space like many other proposed solutions. Specifically, V6DH assigns a special address within one Provider Aggregetable (PA) assignment which will correspond to a different PA assignment for a dual-homed site.

Given the two upstream providers PA-A and PA-B, the current addressing convention format would only include one of those assignments. Using V6DH, the second assignment would also be added, so that each assignment is visible. In order to take advantage of this modification, anything trying to find a route to the site would first look at PA-A, the first assignment, and try to find a path. If the destination is unreachable it instead looks for a path to PA-B, the second assignment.

The proposal specifically states that DNS servers should not be addressed

by the proposed convention, noting that It would make the server addresses no longer a Class 1 object. It also intends that hosts track connection reachability by using ICMP unreachable.

The V6DH proposal requires that instead of using a Mapping Table for routing announcements, which scales by the number of multi-homed sites, a table of “unreachable” end-site prefixes. It separates the information about how PA-A and PA-B are connected from the status of the links. This change makes it so the table scales by the number of links that are down, rather than those that are up. The supposition is that this number will be several orders of magnitude smaller. The proposal notes that the information given by BGP-4 “unreachable” is based off the “withdrawal” of a previously announced prefix, which is no longer the information needed. The proposal also points out that this information is not needed in the DFZ, but instead needed only on the edges. The details for this section are not in V6DH, though they are cited in IDR WG.

Given the new LinkDown behavior, V6DH proposes that rather than suppressing more-specific prefixes while performing BGP aggregation, the unavailability information gets pushed through the DFZ to the sending-side edge routers. Given this unavailability information the dual-homing mechanism can be utilized (swapping addresses when it is determined the first is unreachable) on the ingress side.

Once again, this proposal is useful only in IPv6, which severely limits its usefulness and applicability to todays Internet. Additionally, it requires modification of the existing BGP proposal, which would at the very least impede adoption as such changes are not to be taken lightly.

3.2 Other Proposals

Though the IETF RRG provides one focused outlet for proposals addressing current routing architecture, it's not the only venue in which proposals similar or relevant to HIDRA have been posted. In fact, many proposals with striking similarities have been displayed at various conferences or other publications, many of which are worth noting.

3.2.1 Hierarchical Architecture for Internet Routing

A proposal with very similar goals and implementation to HIDRA is HAIR: A Hierarchical Architecture for Internet Routing [17]. HAIR also uses an identifier/locator split based on current networking realities, and utilizes a reactive scheme to distribute mappings between locators and identifiers. While HAIR does not explicitly require a particular reactive mapping protocol, it envisions the use of DNS or something similar to fulfill this role.

HAIR specifically allows for n -levels in its hierarchy, with at least 3 different layers. Each layer would only be required to know how to route within its own local section of its layer, and how to go either up or down a layer as appropriate. A reactive mapping would be used to traverse different layers. HAIR also specifically mentions a few additional benefits such as IP mobility, a migration path, and traffic engineering possibilities. HAIR is also a fairly unique proposal in that it has a proof of concept testbed implemented.

Unfortunately, while the migration path for HAIR would be relatively straight forward, it depends on a reactive scheme utilizing a new mapping protocol (or at least, a use of an existing protocol in a new function). Because it lacks a detailed

incremental deployment plan and instead depends on a full-scale switchover, the adoption process would be greatly impeded.

The testbed for HAIR was implemented entirely using IPv6 as both locators and identifiers. Additionally, addresses were assigned using some from of DHCP, which is not explained further in the published paper. As such, the proposal offers interesting theories, but lacks greatly in its implementation.

3.2.2 ViAggre

The design of ViAggre [26] is motivated by the same observation that major architectural changes are unlikely without an incremental deployment strategy that doesn't require upgrading router hardware or software. ViAggre proposes reducing the DFZ FIB size by aggregating routes into virtual prefixes, and then deploying virtual networks inside a single ISP to utilize those virtual aggregations.

This type of abstraction is another form of locator-identifier split, in which the deploying ISP will have complete control over how the routes are divided up among its internal routers. Each set of ISP defined virtual prefixes is spread out along the network into "aggregation points," and are utilized as coarse routes for packets. Each of these points contains the more fine-grained information for routing a packet with in that prefix.

In order to tunnel to the correct "aggregation point" within the network, ViAggre utilizes MPLS encapsulation [47]. Through this clever use of tunneling and configuration of virtual networks, ViAggre is able to meet two important design goals: there are no required changes to router software or protocols, and the workings of the internal network remain completely transparent to the external network. HIDRA shares both of these design goals with ViAggre.

However, though the authors of ViAggre claim the negative impacts are negligible, ViAggre does impose some stretch on the path a packet will travel, as well as increased traffic load across the deploying ISP’s routers and links. ViAggre is deemed to be only a “short-term alternative” [26] as opposed to a long-reaching solution.

3.2.3 Shim6

Shim6 [49] is an end-host protocol stack modification that provides both load-sharing and failover capabilities to multihomed sites without the requirement for provider independent addresses. Because multihomed sites are a major contributor to the current inflation of the DFZ, Shim6 could have major reduction in the FIB size.

Unfortunately, Shim6 requires IPv6 deployment, which is not yet in place, and given the current pace may take many years before becoming a reality [18]. Another important shortcoming in Shim6 is its dependence on explicit support in all communication endpoints. While this may theoretically be nice, it’s not realistic to assume such a scenario can be achieved given the fundamental changes required. Finally, Shim6 *only* addresses multihomed sites; while this is a major factor in the DFZ size, it’s far from the only one. Utilizing the protocol may be a step in the right direction, but it’s far from a comprehensive solution.

3.2.4 NIRA

NIRA [59] describes a new comprehensive, policy based network architecture. The main focus of NIRA is to develop an architecture that encourages ISPs to provide Quality of Service (QoS) packages, and help to initiate innovation towards new features and services. This proposals attempts to do this by providing end users the ability to specify which general high-level path their packet will traverse, making ISPs compete for traffic, and thus business agreements.

In order to make this proposal scalable, it employs a hierarchical provider-rooted addressing scheme that can reduce the FIB size. Unfortunately, as scalability is not its main focus, it includes many changes that will impede adoption. In order to achieve the competitiveness among route selection, it uses new proactive and reactive protocols, a new representation for routes, and a different business model for provider compensation. While NIRA provides a unique, and interesting viewpoint, it is not realistic to implement in terms of adopting the solution to address scalability issues.

3.2.5 IPNL

Another pertinent proposal is IP Next Layer (IPNL) [19]. The authors of this proposal attempt to utilize and extend an existing tool to approach the address space problem: Network Address Translation (NAT) [14, 50]. They note that while typical use of NAT may break some existing applications such as peer-to-peer networks, a large majority of applications remain unaffected, so long as the servers themselves are not behind NAT. Additionally, NAT effectively creates a hierarchy; that is, NAT creates a sub-network where the end point address space is separate from the global address space.

Using this concept, a network could be setup such that individual customers

would no longer be dependent on their ISPs numbering. As such, it then becomes possible to setup a multihomed site without injecting site-dependent prefixes into the DFZ forwarding table. IPNL proposes the modification of NAT into end host network protocol stacks via a new IPNL protocol, to be utilized on top of existing IPv4. By doing so, they expect to be able to achieve benefits in addressing IPv4 address exhaustion, and an effective location/identifier split.

Unfortunately, IPNL *requires* the modification of all participating end hosts to be effective; this is not a realistic or easily achievable deployment scenario. HIDRA does not directly address exhaustion, but it does remove the major hurdle to issuing more sites *provider independent* addresses without increasing the size of the DFZ.

3.2.6 HLP

Another paper proposes the use of a new, hybrid link-state path-vector routing protocol called HLP to replace BGP in order to address scalability issues [51]. This proposal starts with the observation that BGP lies on one extreme of a spectrum: all routing policy information is kept private, done in local filters which are never transmitted. The authors of HLP assert that this gives BGP inherent problems such as poor scalability, minimal fault isolation, high rates of churn, and slow path convergence. The HLP protocol is designed specifically to address these shortcomings in BGP.

One of the key components of HLP is splitting the general topology into two levels in standard hierarchical fashion, based off the Autonomous System structure of the network. Root nodes are in the top level of the hierarchy, and are routed using a path-vector protocol, similar to the current BGP protocol. How-

ever, in order to help reduce the number of announcements that are propagated, and the granularity of those announcements, only the top-level hierarchy information is included. Within each of root node's local hierarchies, a link-state protocol is used to distribute routing information. The combination of these two protocols provides for a much more efficient and stable overall network.

While HLP provides the basis for a theoretically sound improvement to replace BGP, the business realities of the Internet are such that it would not be realistic to immediately deploy in a brand new protocol. Additionally, a change such as a switch to HLP provides no path for incremental deployment, again limiting the actual deployability of the proposal. The authors clearly acknowledge these shortcomings, and present their paper as a basis to stimulate informed debate on the topic of design and requirements of future inter-domain routing.

3.3 Summary

Many of the above proposals have ideas of merit. Unfortunately, they all contain some shortcoming, which is typically a failure to provide a realistic adoption strategy with incremental deployment possibilities and benefits, or the lack of support for standard business practices that any realistic proposal must address. Nearly all of these proposals depend on some form of a locator/identifier split, and the incorporation of some hierarchical structure and mapping to help ease the burden on the DFZ forwarding tables. HIDRA attempts to utilize these concepts and provide for a more realistically oriented deployment path.

Chapter 4

HIDRA Framework

The motivations behind the development of HIDRA have been two fold: a solution that will reduce the size of the DFZ forwarding table, and a solution that is realistically deployable. The following sections introduce the specifics of HIDRA, and how they attempt to meet those goals.

4.1 Overview

HIDRA is a hierarchical network architecture, using the basic map and encaps philosophy like many of the previous works discussed in Chapter 3. The top level of the hierarchy, level 0 or L_0 , always uses IPv4 as the network layer. Using location-identifier parlance, the destination location address is in the L_0 header. Level 0 consists of all transit networks. A transit network is generally responsible for carrying traffic between disparate networks under different administrative control. End sites, whether or not they are multihomed, are not part of L_0 , but

share responsibility for outbound routing.

The identifier address is found in L_1 . When an L_1 packet is traversing L_0 it gets encapsulated with the appropriate L_0 header. Before traversing the end site network, the packet will be decapsulated and all subsequent forwarding decisions will be based on the L_1 header. The protocol for L_1 is flexible, however the initial design and implementation assume it is IPv4. IP in IP encapsulation is used when IPv4 is the L_1 protocol. HIDRA has also been designed with the possibility of easily incorporating GRE encapsulation [24] to enable a wide range of L_1 protocols.

IPv4 is chosen as the L_0 protocol to maximize compatibility with existing hardware. By making calculated use of standard IPv4 forwarding logic, all present routers are able to forward HIDRA traffic and carry both L_0 and L_1 routes without hardware modifications or software upgrades. Additionally, most existing routers can be active participants in HIDRA L_0 routing with small configuration changes and the inclusion of an external encapsulation/decapsulation device. Though the map and encap scheme can support many L_0 protocols, usage of IPv4 is an extremely important part of facilitating HIDRA adoption.

An example mapping is shown in figure 4.1.

Standard v4 header fields

0	3	4	7	9	15	16	31
Version	Header length	Type of service	Total length				
Identification					Flags	Fragment offset	
Time to live		Protocol		Header checksum			
32-bit source address							
32-bit destination address							

Example encapsulated header

0	3	4	7	9	15	16	31
0x4	20	0x0	104				
0x05d3					0x4	0x0	
63		0x4			0x3e4c		
10.1.255.2							
1.0.236.117							
0x4	20	0x0	84				
0x0					0x4	0x0	
63		0x01			0x28a2		
10.1.255.2							
10.2.255.1							

Figure 4.1: Standard IPv4 packet header fields, and example encapsulation. The address 10.2.255.1 is mapped to AS 60553, yielding a mapped HIDRA L_0 address of 1.0.236.117.

4.2 Forwarding Information Base vs. Routing Information Base

An important subtlety of HIDRA is its reliance on the differentiation between the Forwarding Information Base (FIB), and the Routing Information Base (RIB) [58]. These are two areas of memory in which the route mappings are stored in a router. The FIB is size constrained and implemented in expensive hardware [56]. The RIB stores all the routes heard for every prefix, and is typically implemented in cheap RAM. Only the best route from the RIB get installed into the FIB. As explained in [26], most modern routers have the capability to selectively prevent a RIB route from being installed in the FIB.

HIDRA leverages this, and instead of attempting to reduce the total amount of routes announced immediately upon deployment which would affect only the RIB, it instead focuses reduces the number of routes that are actually installed

into the FIB. This is discussed further in section [4.6.1](#).

4.3 L_0 Addresses

The HIDRA prototype employs IPv4 as both the L_0 and L_1 protocol. This creates a challenge because both the location and identifier addresses are allocated from the same logical address space. Since most networks will mix both L_0 and L_1 traffic simultaneously, it is important to easily differentiate the packets. To perform this classification using existing equipment, HIDRA sets aside a well-known /8 prefix to contain all the L_0 addresses. Using this technique it is possible to install a set of routes that explicitly treat all L_0 and non- L_0 packets separately, and enables the usage of separate default behavior for each level of the hierarchy.

The L_0 addresses are computed as a direct function of existing *autonomous system numbers* (ASNs). This helps to leverage existing number allocation policy and network topologies. Employing ASNs in this way is a natural extension of their current use. Presently each transit, multihomed, or single homed site with unique routing policy is assigned a single ASN. This number already logically corresponds to the L_0 location. That is, the network the communication end point is attached to. Defining a mapping between ASNs and L_0 addresses also enables the reuse of two key pieces of the Internet infrastructure: number resource allocation mechanism and policy, and the BGP routing protocol. A secondary benefit is the ability to project the future size of the DFZ FIB based on historical number consumption, as seen in Section [4.7](#).

The actual mapping used to create the L_0 address in HIDRA is to use the /8 prefix for the high-order 8 bits of the address, and set the low-order 24 bits to the low-order 24 bits of the ASN. This technique only uses the lower-order 24 bits

of the 32-bit ASN. This is not a large limitation because current ASNs being issued have all 8 high-order bits set to 0 [6]. Additionally, as discussed in section 4.7, the projected consumption rate of ASNs in HIDRA is such that it will take far in excess of 10,000 *years* before it is necessary to use any of the high-order 8 bits.

4.4 Encapsulation

Encapsulation is the act of placing an L_0 header on an L_1 packet. This is an expensive operation in both time and space because it requires both performing a lookup on the L_1 destination to determine the corresponding L_0 destination, and then modifying an existing packet. In order to do this a mapping table must be available, which can become quite a complex, large structure when we consider the size of the Internet and all of the mappings that may end up being used. Mapping can be done in either a proactive or a reactive fashion, as discussed below.

4.4.1 Proactive Mapping

Currently, routes on the Internet are distributed through Border Gateway Protocol (BGP) [27, 43, 45, 44]. This is a purely “proactive” system meaning all routes are distributed, regardless of whether or not they are ever utilized. In contrast, another way in which routes can be distributed is through a “reactive” system, where the route is not distributed until needed. As a general rule, proactive systems typically have less first packet latency than a reactive system, but require more communication overhead [31, 34].

Figure 4.2: A multi-site multi-homed network. From the standpoint of AS1, encapsulation can occur at point (1) if the AS itself provides encapsulation, point (2) if its upstream provider encapsulates, or at the originating end-host if a reactive routing scheme is used. Decapsulation can occur either at point (1) for end-site decapsulation, or point (3) for ISP decapsulation. (3) provides for slower DFZ FIB growth and better fail-over characteristics.

In the current situation, the physical overhead of memory storage required by the existing system has become a concern, and must be addressed [4]. Most current proposals addressing this implement a completely reactive system, similar to or reliant on a DNS-like infrastructure [15, 17, 49, 53, 48]. While these systems may be ideal, they typically fail to address a critical issue: moving from a fully proactive system to a fully reactive system given the realities of today’s Internet administration is a difficult task, and doing so in any kind of flag-day scenario is unrealistic.

As such, HIDRA initially relies on a proactive distribution; more specifically, HIDRA utilizes the current BGP standard in an unmodified format to assist with integration. Eventually, it is expected that HIDRA will move to a mostly reactive route distribution protocol ¹ discussed further in Section 4.4.2.

A proactive mapping potentially requires a lookup table at least the size of

¹While HIDRA may be mostly reactive at some point, there will always be a few critical routes such as the root authority location for the reactive system that will heavily benefit from being distributed in a proactive fashion.

the current Default Free Zone (DFZ) FIB. Performing this expensive mapping close to the transmitting host is very important. First, it moves the incremental encapsulation burden of supporting additional devices from the core L_0 network equipment to edges of the network, where the absolute speed requirements of the network are much lower. This also places that burden where the growth is occurring, rather than forcing the relatively static core to handle more stress. It also provides for much better lookup cache locality, which in turn makes encapsulation more efficient. HIDRA uses at least one encapsulation device, and typically more to provide fail-over and load balancing, near the access links for each end site. Ideally the end hosts themselves will encapsulate the packets before transmission, removing the burden from the network entirely. HIDRA also works well *without* this optimization, as demonstrated in our testbed as shown in chapter 6.

It is unreasonable to expect all end hosts to participate in encapsulation. Upon initial deployment there will be almost no devices that have encapsulation software. Given the number of special purpose embedded hosts used today, it is also unreasonable to expect *every* “legacy” device in a network to become HIDRA aware. Therefore the network must support a transparent encapsulation service. This service is expected to be provided by specialized in-network encapsulation devices². These devices will be located either at the ISP’s end of a customer’s access link (for smaller customers) or within the customers network itself. In the later case the customer must ensure that all non-local L_1 addresses are routed to the encapsulation device(s), and all L_0 addresses are routed to the upstream ISP. This is doable because the L_0 addresses are all allocated from a previously discussed well-known prefix. If the ISP is providing encapsulation service, the routes in the end site network should be unaffected. In-network encapsulation

²Initially these encapsulation devices will be standalone network appliances. As routers get naturally refreshed they can take over this responsibility.

is provided as a mechanism for transitioning to end-host encapsulation, and to provide network service for legacy devices.

Figure 4.2 illustrates the possible encapsulation points within a network. Large stub and multihomed sites are expected to provide their own encapsulation service, with smaller sites leaving that responsibility to their upstream service provider. As HIDRA becomes more widely utilized, and vendors begin including support in their products for HIDRA, it is expected that the encapsulation will move down stream, from point (3) to (2) to (1), and eventually all the way to the end user.

4.4.2 Reactive Mapping

In order to further reduce DFZ forwarding table size and growth rate, it is envisioned that HIDRA will move to a mostly reactive system, and has been designed with this in mind. In the event that a reactive mapping becomes available, HIDRA will still query the available BGP information in order to maintain backwards compatibility. Additionally, critical infrastructure, such as the location of the reactive mapping authorities can still be distributed using the proactive protocol.

HIDRA is fairly agnostic as to what reactive mapping protocol is used. The easiest solution would be the use of DNS to distribute L_1 to L_0 mappings due to its proven scalability and current wide-spread usage [53, 36]. However, any system that can adequately meet the demands of a generalized reactive mapping protocol can be easily incorporated into the HIDRA framework.

A basic DNS mapping distribution system has been designed to help support HIDRA functionality in the proof-of-concept sandbox network. It incorporates

a new record type that contains the mapping information, as well as priority listings for multiple mappings to facilitate standard traffic engineering practices. This system is discussed in more detail in work by Mr. Nelson [36], and is left to be investigated further in future work.

4.5 Decapsulation

A packet is decapsulated as it traverses the $L_0 - L_1$ boundary. Decapsulation is a much faster operation than encapsulation because it only requires removing the outer-most header from the packet before forwarding it using standard techniques; there is no inherent requirement for a large, slow lookup operation. The only technical requirement placed on the decapsulation point is that it sits in both the L_0 and the L_1 networks. HIDRA takes advantage of this flexibility to minimize the number of routes in the L_0 DFZ.

Initially the decapsulation service will be provided by an external device, typically the same device that provides encapsulation. Later it will be transitioned to border routers as their software permits. Like encapsulation, when decapsulation is done using an external dedicated device, that device should be topologically close to the boarder routers to minimize stretch.

The obvious point of decapsulation is when a packet enters the destination site (either stub or multihomed). This is depicted as point (1) in figure 4.2. This is termed “end-site decapsulation.” End-site decapsulation requires every AS to originate an additional route to the DFZ FIB, the route for the site’s L_0 . This will be a /32 route.

Point (3) in figure 4.2 is the immediate upstream provider’s external gateway.

Performing decapsulation here is termed “ISP decapsulation.” ISP decapsulation has a number of advantages over end-site decapsulation. Most importantly, it enables traffic engineering for a multihomed site. The multihomed site can select the appropriate ingress link by advertising that *provider’s* L_0 address as the decapsulation point. It also enables efficient multi-site networks. Since the decapsulated L_1 packet traverses the provider’s network, it can take the most efficient path from the decapsulation point to any of the customer’s sites.

ISP decapsulation further reduces the size of the L_0 DFZ, because the only entries necessary are for transit provider’s decapsulation points. As shown in section 4.7, there are almost an order of magnitude fewer transit providers than single or multi-homed networks. Finally, the FIB burden of accepting new customers is placed on the upstream provider(s) that contract with the customer. Each customer will add one L_1 entry to the provider’s FIB for each route the provider accepts from the customer. In contrast, the provider will only occupy a single slot in the L_0 forwarding table, regardless of the number of customer routes or the length of customer prefixes it accepts.

4.6 Routing

To maximize compatibility and interoperability with existing network infrastructure, HIDRA uses BGP as its proactive routing protocol. Unmodified BGP already contains all the information necessary to map a L_1 address into the corresponding L_0 address. Each L_1 route has its own entry in the BGP RIB, so the normal longest prefix lookup can be used to extract that entry. Instead of using the next-hop value from that entry, which is standard on today’s Internet, the AS path attribute is used. The AS path attribute is an ordered list of all

the ASNs a packet traverses while it follows the path to the destination. HIDRA is only concerned with the final ASN in the path. This will be the ASN of the destination site. This ASN is extracted from the AS path and transformed into the corresponding L_0 address as previously described.

Selecting BGP as the proactive routing protocol for HIDRA enables the reuse of all *existing* route advertisements on today’s Internet. It further enables the reuse of network administrator’s knowledge of and device support for manipulating BGP advertisements for purposes such as traffic engineering and primary/backup link designation.

In addition to using the existing advertisements for proactive lookups, all HIDRA enabled ISPs must advertise their L_0 route via BGP. These routes are originated by every device within the site that can perform decapsulation. These devices may be dedicated decapsulation boxes or decapsulation routers themselves. Either way, this system uses any-cast to replicate the decapsulation service within a network. The encapsulation devices also need to originate a default route that is not propagated outside the local L_1 network. This route redirects unencapsulated L_1 traffic not destined for the current L_1 site to the encapsulation device, to then be encapsulated before the trip across L_0 .

Routers in HIDRA are configured in a similar fashion to current routers. That is, they exchange full L_0 and L_1 routes, both internally and externally, via BGP. All these routes are present in every router’s RIB. However all HIDRA aware routers are configured to prevent non-customer L_1 routes from entering the FIB. L_0 routes are easily identified by the well known prefix. As explained in the next section, all HIDRA routers have a complete RIB making it possible to have a long chain of alternating HIDRA aware and legacy sites each with the information they need to successfully forward packets across the entire network. This permits an

ISP to deploy HIDRA without requiring a customer to reconfigure its end of the peering sessions.

Legacy Internet

Peering Link	Routes Originated
AS1 $\Rightarrow$ AS2	192.168.1.0 $\rightsquigarrow$ AS1
AS1 $\Rightarrow$ AS3	192.168.1.0 $\rightsquigarrow$ AS1
AS2 $\Rightarrow$ AS1	192.168.2.0 $\rightsquigarrow$ AS2
AS2 $\Rightarrow$ AS3	192.168.2.0 $\rightsquigarrow$ AS2
AS3 $\Rightarrow$ AS1	192.168.3.0 $\rightsquigarrow$ AS3
AS3 $\Rightarrow$ AS2	192.168.3.0 $\rightsquigarrow$ AS3
AS3 $\Rightarrow$ AS4	192.168.3.0 $\rightsquigarrow$ AS3
AS4 $\Rightarrow$ AS3	192.168.4.0 $\rightsquigarrow$ AS4 192.168.44.0 $\rightsquigarrow$ AS4

HIDRA with ISP Decapsulation

Peering Link	Routes Originated
AS1 $\Rightarrow$ AS2	192.168.1.0 $\rightsquigarrow$ †
AS1 $\Rightarrow$ AS3	192.168.1.0 $\rightsquigarrow$ †
AS2 $\Rightarrow$ AS3	ASN.0.0.2 $\rightsquigarrow$ AS2 192.168.1.0 $\rightsquigarrow$ AS2
AS3 $\Rightarrow$ AS1	ASN.0.0.3 $\rightsquigarrow$ AS3 192.168.4.0 $\rightsquigarrow$ AS3 192.168.44.0 $\rightsquigarrow$ AS3
AS3 $\Rightarrow$ AS2	ASN.0.0.3 $\rightsquigarrow$ AS3 192.168.4.0 $\rightsquigarrow$ AS3 192.168.44.0 $\rightsquigarrow$ AS3
AS3 $\Rightarrow$ AS4	ASN.0.0.3 $\rightsquigarrow$ AS3 192.168.1.0 $\rightsquigarrow$ AS3
AS4 $\Rightarrow$ AS3	192.168.4.0 $\rightsquigarrow$ † 192.168.44.0 $\rightsquigarrow$ †

HIDRA with End Host Decapsulation

Peering Link	Routes Originated
AS1 $\Rightarrow$ AS2	ASN.0.0.1 $\rightsquigarrow$ AS1 192.168.1.0 $\rightsquigarrow$ AS1
AS1 $\Rightarrow$ AS3	ASN.0.0.1 $\rightsquigarrow$ AS1 192.168.1.0 $\rightsquigarrow$ AS1
AS2 $\Rightarrow$ AS3	ASN.0.0.2 $\rightsquigarrow$ AS2 192.168.2.0 $\rightsquigarrow$ AS2
AS3 $\Rightarrow$ AS1	ASN.0.0.3 $\rightsquigarrow$ AS3 192.168.3.0 $\rightsquigarrow$ AS3
AS3 $\Rightarrow$ AS2	ASN.0.0.3 $\rightsquigarrow$ AS3 192.168.3.0 $\rightsquigarrow$ AS3
AS3 $\Rightarrow$ AS4	ASN.0.0.3 $\rightsquigarrow$ AS3 192.168.3.0 $\rightsquigarrow$ AS3
AS4 $\Rightarrow$ AS3	ASN.0.0.4 $\rightsquigarrow$ AS4 192.160.4.0 $\rightsquigarrow$ AS4 192.160.44.0 $\rightsquigarrow$ AS4

Figure 4.3: Example prefixes and ASNs ($\rightsquigarrow$) advertised in a small multi-homed network using legacy BGP, HIDRA with end site decapsulation, and HIDRA with ISP decapsulation. Routes in bold are the L_0 FIB. Duplicate L_0 routes are not in bold. All other routes are L_1 . † – Route originated with a private ASN.

4.6.1 Effect of RIB vs. FIB

To help depict the effect HIDRA has on routes, figure 4.3 shows a small, four site network, and illustrates how routes are announced and propagated under the current Internet architecture, HIDRA with end-site decapsulation, and HIDRA with ISP decapsulation. In the legacy example, all routes are originated by their own AS and installed in the DFZ FIB, for a total of 5 entries. When HIDRA and end-site encapsulation is employed, each AS originates a single L_0 route and enough proactive routes to advertise their entire address space. This results in 4 entries in the DFZ FIB.

The second example in figure 4.3 shows HIDRA with ISP decapsulation. Each ISP originates a single L_0 route, for a total of 2 entries in the DFZ FIB. In addition, the stub and multihomed sites advertise their prefixes with a *private* ASN to their immediate upstream providers. The providers routers automatically remove the private ASN from the AS path before propagating the route, so it appears as though the route originates from the provider's network. This ensures the packet is addressed to the provider's L_0 decapsulation address and *not* an L_0 address based on the customer's private ASN.

4.6.2 Multi-homed Networks

Another benefit of using BGP to distribute proactive routes is robust support for multihomed sites. AS1 in figure 4.3 is a multihomed AS. When either one of its access links fail, the traffic will automatically be routed through the other link regardless of the decapsulation point. Both of these scenarios are illustrated in the next paragraphs.

Assume HIDRA is using end-site decapsulation. The multihomed site will be advertising its L_0 route to both its upstream providers, AS2 and AS3. AS3 will also hear the multihomed site's L_0 route from its peering session with AS2, however it will prefer the direct link between AS3 and AS1 due to the shorter AS path. AS4 will hear AS3's best route, which will be AS3–AS1. Further assume that a host in AS4 is communicating with a host in AS1. The packets get encapsulated at AS4's border gateway and then sent along the L_0 route to AS1. When the AS1–AS3 link fails it will be detected by BGP. AS3 will fall back on the next best L_0 route, AS1–AS2–AS3. This modified route will be propagated to AS4, and all L_0 packets will continue to reach AS1 using the updated route.

In the second scenario HIDRA is using ISP decapsulation. In this case AS2 is originating a proactive route to AS1's prefixes, as is AS3. These routes have AS2 and AS3, respectively, as the last AS in the AS path. AS4 will only hear AS3's advertisement since that route has a shorter AS path. In addition both AS2 and AS3 are advertising their own L_0 routes. Packets sent from a host in AS4 to a host in the multihomed site gets encapsulated with AS3's L_0 address, because that is the best proactive route. When the link between AS1 and AS3 fails, BGP will withdraw the *proactive* route originated by AS3. This causes AS3 to advertise its next best proactive route, the route that AS2 is originating. BGP then updates AS4 with the new route, and all future encapsulations will use AS2's L_0 decapsulation address.

In both scenarios recovery automatically takes place on the same time scale and using the same mechanisms as recovery in today's Internet. This is true when using both end-site decapsulation and ISP decapsulation.

4.7 Projected Impact on DFZ Table Size

One of the main motivators behind HIDRA is to reduce the number of routes that get stored in the DFZ FIB table. HIDRA does so by aggregating the current routes into more coarse routes, based on the ASN, as discussed in the previous sections. One advantage of utilizing ASNs as the basis for aggregation is the large body of historical ASN utilization data that is available. Since 1998, [27] has been archiving enough data to project both the absolute size and growth of the L_0 table for legacy routing, HIDRA routing with end-site decapsulation, and HIDRA with ISP decapsulation. The two graphs in figure 4.4 show this data.

The data supports an immediate projected reduction from approximately 315,000 entries to 34,000 entries, equivalent to one order of magnitude. This assumes every existing stub and multihomed site performs its own decapsulation, which is a reasonable initial deployment scenario. Long term, stub and multihomed sites should migrate to ISP decapsulation, reducing the table size from 34,000 to approximately 5,000, another order of magnitude.

In addition to immediately reducing the DFZ FIB size, HIDRA also substantially changes the table’s growth trend. The current table is growing either exponentially or super-linearly with a steep slope. Either way, the growth trend in ASN usage is linear with a much shallower slope. Additionally, the growth in transit ASNs is even flatter than total ASN growth. Under the assumption that new non-transit sites would *not* be issued ASNs after HIDRA implementation, the transit ASN trend is the one that will most closely predict future L_0 growth. Therefore widespread adoption of HIDRA can both immediately reduce DFZ size by at least one order of magnitude, and flatten out the expansion trend so much

Figure 4.4: Historical size of the DFZ table. The “Legacy DFZ Size” shows the historical and current size of the DFZ table. The “ASNs Announced” line shows the number of unique ASNs visible on the public Internet. “Transit ASNs” is the number of unique ASes that appear in the middle of at least one BGP AS path. The former AS count approximates customer decapsulation and the later ISP decapsulation. Graph (b) focuses on the ASN data. All data is from the public potaroo [27] web site from the RouteViews [38] vantage point. For readability only every hundredth data point is plotted in the graph.

that the L_0 table will not reach the size of today’s table again for at least 50 years.

4.8 Load Balancing

An accomplishment of HIDRA is retaining as many existing network management practices as possible. Load balancing is an important example. This technique is still available within HIDRA, but requires ISP Decapsulation as discussed in the second scenario in section 4.6.2. Given a HIDRA network with ISP decapsulation, the end-site AS can balance incoming traffic by advertising different portions of its netblock to different upstream ISPs. The netblock split is under control of the end site, as are the MEDs, communities, AS prepending, and other BGP traffic engineering techniques ³.

For a more in-depth look, this section examines the common technique of AS path pending [41, 22]. BGP utilizes the AS path attribute associated with a route primarily to ensure no loops are created when routing. However, it also uses it as a criteria for determining which route is best by selecting the path with the shortest number of AS hops [43, 44]. Modern routers take advantage of this by allowing the creation of route-maps ⁴ to add the local AS multiple times into the beginning of the AS path attribute. Using these user controlled route-maps, the AS path attribute length on advertised routes can be artificially increased, lowering the preference for the route.

HIDRA can effectively leverage this concept in standard traffic engineering practice by selectively lengthening the advertised path on one link between the

³For more on BGP traffic engineering techniques, see [2].

⁴“Route-map” is a Cisco specific term, though the functionality exists in most modern routers [26]

customer and one of its ISPs. The result is that BGP will prefer a different link between the same customer and ISP. Because the other route still exists (just with a longer AS path attribute) BGP will still fall back on the other link in the case of a failure.

This is demonstrated on our testbed in section [7.4](#).

4.9 Migration Strategy

One of the unique aspects of HIDRA is the ease of migration from a currently existing network to a HIDRA network. This is made possible by the fact that HIDRA takes advantages of already in place constructs such as BGP and ASN assignment to work. In order to move from a flat network to a HIDRA network, a default route must be added to send all traffic not destined for the local L_1 network to an encapsulation point. Conversely, a route for all encapsulated traffic, which is easily identifiable by the $/8$ well-known prefix, to a decapsulation point must also be added. Given these two routes, all intra-as configurations remain the same.

Perhaps the most complicated part of setting up a HIDRA network is the modification to the BGP configurations of border routers. These must be configured for two different purposes: to announce the appropriate L_0 addresses, and to actually realize a reduction by removing all non-local, non- L_0 routes from it's forwarding table and leaving them only in the route table. Announcing the appropriate L_0 address requires nothing more than a standard announcement and inclusion of the appropriate neighbor in the BGP configuration. Removing all non-local, non- L_0 routes from the forwarding table is a less well known technique, but is accomplished through the use of IP prefix-lists and an appropriate route

map.

An important point here is that a HIDRA network will save memory space and still provide correct functionality, *regardless* of whether or not its neighboring ASes are legacy or HIDRA.⁵ This means that there's an incentive for network administrators to deploy HIDRA, regardless of their neighbor's status. Because of this, HIDRA can be deployed incrementally without reliance on its neighbors also deploying HIDRA.

⁵Though a HIDRA network can provide correct functionality regardless of whether the neighbor is legacy or HIDRA, it is necessary for the administrator to configure the network differently based on whether the neighbor is utilizing HIDRA or legacy.

Chapter 5

Metrics

5.1 Applying Metrics

The standards and requirements set out for routing protocols detailed in RFC 5773 [\[10\]](#) are quite extensive. They are applied in this section to HIDRA, to verify how well HIDRA fits into the expected standard. The table below lists all requirements set forth by the RFC, and whether or not HIDRA affects these particular areas. Some of the results are relatively straight forward; those that are not as obvious are discussed further in the next section.

RFC Requirement	HIDRA's Affect
Route to Destination	Maintained
Routing is Assured	Maintained
Large System	Improved
Autonomous Operation	Maintained
Distributed System	Maintained
Provide A Credible Environment	Maintained
Be A Managed Entity	Maintained
Minimize required Resources	Improved
Route around failures dynamically	Maintained
Provide loop free paths	Maintained
Know when a path or destination is unavailable	Maintained
Provide paths sensitive to administrative policies	Modified
Provide paths sensitive to user policies	Modified
Provide paths with appropriate QoS	Maintained
Provide autonomy between inter/intra-AS	Maintained
Decouple inter/intra-AS forwarding	Improved
Do not forward administratively inappropriate packets	Modified
Do not forward datagrams to failed resources	Maintained
Forward datagram according to its characteristics	Maintained
Provide a distributed/descriptive information base	Improved
Determine resource availability	Maintained
Restrain transmission utilization	Improved
Allow limited information exchange	Maintained
Support a packet-switching environment	Maintained
Accommodate a connection-less oriented user transport service	Maintained
Accommodate 10K autonomous systems and 100k networks	Improved
Allow for arbitrary interconnection of autonomous systems	Maintained
Provide routing services in a timely manner	Maintained
Minimize constraints on systems with limited resources	Improved
Minimize impact of dissimilarities between ASes	Maintained
Accommodate the mechanisms of the AS	Improved
Implementable by network vendors	Maintained

Table 5.1: HIDRA affects many areas of the requirements stated in RFC 5773. For more information about the individual requirements, see [10].

5.1.1 General Requirements

Route to Destination

This requirement states that all locations must be reachable in a timely fashion. In a HIDRA network, the same locations are routable as in the current flat network, with only a few extra hops in any given path, even across the largest networks. The system performs on a very similar timescale with current routing systems.

Routing is Assured

This requirement specifically states that a user be notified within a “reasonable time period” of failure to provide a service. HIDRA utilizes BGP and standard IP practices such as ICMP seamlessly, maintaining all the failover and information exchange techniques involved in these protocols. Because of this, the status quo is maintained.

Large System

This standard requires that any architecture be designed to accommodate the growth of the Internet. This is the point that most proposals in the area are attempting to address. HIDRA is designed specifically with the growth trend in mind, not just an immediate reduction. Using HIDRA will link the forwarding table growth rate with the ASN growth rate, which has been roughly linear [27]; this provides much better growth rate than the current super-linear growth of the forwarding table [4].

Autonomous Operation

Administrative domains, currently signified by the distribution into Autonomous Systems, must be able to set their out intra-system routing policies. HIDRA's use of BGP once again makes maintaining this standard easy, as all intra-AS routing decisions are the same. Additionally, all configuration for HIDRA can be done on a per-AS basis, maintaining the standard.

Distributed System

In order to support redundancy and diversity of nodes and links, any routing system being used on a large scale should be distributed. HIDRA measures up extremely well in this category, as not only does it maintain the current level of distribution through its use of BGP, but it also paves the way for the introduction of a reactive mapping system for fully (or at least, nearly fully) reactive routing, which can be accomplished through another proved distributed system such as DNS.

Route Around Failures Dynamically

Once again, HIDRA's use of BGP allows it to maintain the current level of failure detection and recovery.

Provide Paths Sensitive to Administrative Policies

Through the use of AS pending, the administrators of an AS can control the way in which routes are mapped, giving HIDRA load balancing capabilities. On the down side, in the current incarnation of HIDRA, encapsulated traffic

becomes much harder to balance or control, since routes can't be deaggregated unless multiple numbers are used to represent the same AS. It is expected that this issue can be addressed in future iterations of HIDRA.

Provide Autonomy Between Inter- & Intra-Autonomous System Route Synthesis

HIDRA maintains the status-quo in this regard. While HIDRA requires the use of BGP to distribute at least critical locations (eventually, all other routing information could conceivably be pushed to a reactive system), the intra-AS protocol remains unaffected.

Chapter 6

HIDRA Implementation

In order to demonstrate the feasibility and effects of HIDRA, a sandbox network has been created in the Cal Poly networks lab using a prototype implementation. This implementation is built in Linux, and contains both a userspace application linked into the network stack through netfilter [1], as well as a custom kernel module to provide the encapsulation and decapsulation services.

Early experimentation shows promising results, with the forwarding table showing the expected significant reductions in space while the network itself maintains the same level of approximate failure tolerance and flexibility.

6.1 Implementation Specific Design Decisions

As discussed in chapter 4, the general concept is to utilize the ASN as the locator in a locator identifier split, and to then encapsulate packets in such a way as to do routing based on the locator, and then the identifier. BGP is used as

the initial mapping utility, and as time passes a reactive mapping system should be introduced. While these are the high level concepts, our implementation has made some very specific design decisions to make HIDRA work in actuality, not just theoretically.

In order to maximize comparability and flexibility, we chose to utilize the IP in IP standard [39] and encapsulate existing packets with another IP layer, as discussed in section 4.3. An important aspect of encapsulation is how the L_0 header specifics outside of the mapping are determined, and how it corresponds to the incoming L_1 address. The three most important fields in consideration are the source address, the TTL, and the ID field. The design decisions concerning the source and TTL are below, and the usage of the ID field is discussed in section 6.3.

In the current design, the L_0 source address is set as the L_1 address *of the device responsible for doing the encapsulation*. Thus, every packet being encapsulated by a specific physical device must result in the same, *unique* source address. This may seem odd at first, but on further consideration becomes quite obvious: this location has modified the packet, and any error responses to the *encapsulated* version of the packet must come back to the device that did that encapsulation, so appropriate action may be taken. An example scenario is given in 6.3.1.

The TTL field is fairly straightforward after careful consideration: in general we want an encapsulated packet to time out at the same place in the hierarchical network that an unencapsulated packet would fail in a flat, regular network. As such, we simply copy the TTL up on encapsulation, and back down on decapsulation. This keeps the TTL consistent across the L_0 and L_1 layers. Because of this, the prototype network is able to support protocols or applications relying on the TTL, such as traceroute.

6.2 Testbed Configuration

A testbed network has been constructed to more accurately evaluate HIDRA. The AS-level network topology for this prototype network is depicted in figure 6.1. This network was physically created in one of the laboratories at Cal Poly, San Luis Obispo. Each AS consists of one Cisco 3640 or 3820 router with multiple 100Base-T or 1000Base-T network interfaces. Each AS also has either an older 500Mhz Pentium III PC with 256MB of RAM, or a 3.2 Ghz Pentium 4 CPU with 512MB of RAM running the HIDRA software stack. BGP is used to exchange both L_0 routes and proactive L_1 routes. The network uses private addresses and is otherwise not related to the commodity Internet.

In addition to HIDRA-based IPv4, the testbed routes IPv6 traffic between all ASes. This traffic does *not* use HIDRA. IPv6 is used as a control plane to coordinate experiments. All other traffic uses IPv4 and HIDRA.

Figure 6.1: The network testbed used to evaluate HIDRA.

6.3 Software Implementation

HIDRA has been implemented and tested using generic Linux computers to provide encapsulation and decapsulation. There are two primary parts to the

implementation, a user-level daemon and a kernel-level encapsulation / decapsulation module. The interactions between the components and the general flow of packets through the software is summarized in figure 6.2.

The implementation uses Linux’s firewall hooks to identify packets in need of encapsulation or decapsulation. Packets are first sent to the kernel module. If the packet needs decapsulation, the module strips off the L_0 header¹ and re-injects the decapsulated packet into the Linux network stack. The network stack makes a forwarding decision to either deliver the packet to the localhost (implementing host-based decapsulation) or retransmit it back into the network (implementing network-based decapsulation).

The kernel module does not contain the logic to resolve an L_1 address into the corresponding L_0 address. This design decision was made because the mapping logic is a management-plane operation, and would needlessly complicate and slow the forward-plane encapsulation code. Instead, the kernel module has an internal mapping cache. When a packet needs encapsulation, the cache is consulted. If the mapping is present in the cache, the module encapsulates the packet and re-injects the new L_0 packet into the network. The kernel module also determines if the encapsulated packet will exceed the MTU and generates the appropriate ICMP error as necessary. If there isn’t an entry in the kernel cache, the packet gets sent to the user-level HIDRA daemon via Linux’s NFQueue subsystem.

In order to handle ICMP error messages, a copy of any newly-encapsulated packet is stored, indexed by a unique ID field in the new L_0 IP header. Because of this, when an ICMP error message (such as a *Destination Unreachable* or a *ttl exceeded*) is returned on the encapsulated packet, the original packet is available

¹This process is slightly more complicated than initially expected, and is examined more closely in section 6.3.1.

Figure 6.2: Block diagram depicting the interaction between the HIDRA kernel module and the HIDRA daemon. Initial implementation was done for Linux.

and the correct error message can be generated and returned to the original sender. For further details, see section 6.3.1.

The HIDRA daemon resolves the L_1 destination into the appropriate L_0 destination. Once the mapping is resolved, the daemon creates a new entry in the kernel’s mapping cache. This ensures that all subsequent packets sent to the same destination will be encapsulated by the low overhead kernel module, instead of the high overhead daemon. In addition to resolving the mapping, the daemon is responsible for encapsulating and re-injecting all packets redirected to it from the kernel module.

The user-level daemon has two more important responsibilities. First, it is responsible for installing the correct firewall rules when it starts and cleaning them up when it exits. This maintains the firewall in a working state, regardless of if the daemon is currently running. The second major responsibility is exchanging

iBGP information with the other routers in the AS. To facilitate this, the routers are configured to peer with the encapsulation daemon. Receiving the full BGP table enables the proactive mapping lookup. It also allows the daemon to update the kernel’s mapping cache and firewall rules automatically as new L_1 routes are seen and old ones change. This is a necessary part of handling network failures.

The daemon uses the iBGP peering sessions to originate the L_0 decapsulation route for its AS, and originate a default route that directs all L_1 traffic to itself for encapsulation before the packets leave the AS. The routers are configured to tag all routes in their L_1 with a well-known BGP community. The encapsulation boxes use this community to determine which L_1 destinations are directly reachable and which ones require encapsulation.

6.3.1 ICMP

One of the trickiest parts of the proof-of-concept implementation of HIDRA was enabling full use of the ICMP protocol [40]. While standard ICMP messages are relatively straightforward, normal packets, the responses generated to them due to errors cause encapsulation schemes some grief because they include the original packet that generated the error. Since encapsulation modifies the original packet, it’s frequently the case that the error message responds to that modified version. In order to handle the error correctly, it is essential that the included information in the error message be correct for the originator to identify which packet triggered the error. Because the packet was modified for encapsulation (the L_0 header was added), the originator may no longer be able to recognize which packet triggered the error.

ICMP errors can come in four different ways in a HIDRA network:

- Unencapsulated error message, generated in response to an unencapsulated packet
- Unencapsulated error message, generated in response to an encapsulated packet
- Encapsulated error message, generated in response to an unencapsulated packet
- Encapsulated error message, generated in response to an encapsulated packet

This aspect of ICMP error generation and handling is the main motivator behind the choice of L_0 source address. If the error is generated in response to an encapsulated packet, the ICMP response is routed back to the encapsulation device so that it can undergo the reverse modification to prepare it for the original sender.

To handle this on the software side, each time a packet is encapsulated, the first 48 bytes (L_0 , L_1 , and 64 bits of data as specified by the original ICMP RFC [40]) are stored in a local cache. The cache index is then written into the L_0 header IP ID field. In order to keep unique indices, the local cache will store up to 65,536 packets so the index will coincide with the 16 bit ID field in a standard IPv4 header, after which it will rollover, overwriting old packets. Because of the utilization of the ID field, any error response to an encapsulated packet contains the index into the cache, since it contains the original message that generated the error. This index is verified to make sure the device is indeed the one that did the original encapsulation by checking against the contained packet's L_0 and L_1 headers. If the fields in the ICMP payload match those in the stored packet at the location in the cache index by the IP field, it is assumed that the local device

Figure 6.3: A simple 2-AS HIDRA network, where each router and its connected hardware is in it's own AS.

did the encapsulation, and the appropriate error message is then generated from the contents of the cache and routed back to the original source location.

As is typical in any networks application, this processes contains many edge cases. The edge cases surround the packet that generates the error message, or more specifically where in the network an ICMP error is generated. Each of the four possibilities occurs in a fairly typical trace route across a HIDRA network with multiple ASes. The network depicted in figure 6.3 is used to demonstrate each of the four possibilities, which are examined below.

Unencapsulated Header with Unencapsulated ICMP Payload - In this case, the packet is a “normal” packet, and no special considerations need to be made. Due to implementation constraints, all ICMP error messages are routed through the kernel module, so this case must be detected, and returned to the standard packet stream.

Figure 6.4: The initial hops of a traceroute across the example network. These TTL Exceededs produce normal, unencapsulated ICMP errors responding to unencapsulated ICMP packets.

In a traceroute across the example network, this occurs in the first two hops shown in figure 6.4, where a standard, legacy packet generates an ICMP TTL Exceeded error message.

Unencapsulated Header with Encapsulated ICMP Payload - This edge case is uncommon in most communications, but becomes readily apparent when doing a simple traceroute. This occurs when a packet is encapsulated, and then generates an error message before leaving its L_1 network. In the case of a traceroute, when the packet about to be encapsulated has a TTL of 2. The packet will be encapsulated (and thus have its TTL decremented), and then be sent to the encapsulation device's upstream router, at which point the TTL will exceed and an error will be generated. The error response uses the L_0 source address, which is the L_1 address of the encapsulation device. In this case, the L_1

Figure 6.5: The packet has just been encapsulated and sent out to be routed across the network, but the TTL exceeds on its next hop.

network is still directly routable, and is in fact directly connected to the device generating the error.

Because of this the ICMP packet is sent back unencapsulated to the encapsulation device. In order to handle this correctly, the software must restore the correct inner packet and L_1 destination address ² as well as the ICMP error payload.

As a more concrete example, we examine hop 3 in the traceroute of the example network, shown in figure 6.5. At this point, the packet has just been encapsulated by the lower AS encap device, and sent out to attempt to be routed to its final destination. When it arrives at the router, the TTL exceeds and an error is generated in response to the current packet. The packet generating the

²The notation L_1 is used to describe the protocol, and layer in the network that the packet is operating; it is important to realize that at this point, there is only one IP header on the packet, and L_1 is referring to that.

Figure 6.6: The generated ICMP error message from point (3) in figure 6.5. Consists of 3 distinct components: the outside IP header, the ICMP header, and the ICMP payload. The “- - -” notation indicates the value of the field is not pertinent to the current example.

error has an L_0 source of the device that encapsulated it, in this case 10.2.255.2, which now becomes the destination address of the ICMP error.

Because the device is directly connected, the ICMP error packet is able to be routed directly back to the encapsulation device. The actual values in the error message are shown in figure 6.6. At this point, the encapsulation device utilizes the ID field from the ICMP *payload* packet to do a lookup in its local cache. Given the cache information, the correct error message is generated to be sent back to the originator. Specifically, a new ICMP error packet is generated, using the original sender’s source address as the new destination address, the generator of the original ICMP error packet’s address as the source, and copying the original packet in as the ICMP payload.

Encapsulated Header with Encapsulated ICMP Payload - This occurs when an error is generated in response to a packet outside of the original

Figure 6.7: The encapsulated packet times out, outside of its own AS. At this point, the ICMP error response is generated, and sent to the source location in the original packet's L_0 header, which is the L_1 address of the original encapsulation device.

encapsulator's AS, because the L_0 source address of any packet is the L_1 address of the device that did the encapsulation. If the L_0 source is not the local AS, the error response generated to that packet will be handled in the standard fashion, meaning the error message now needs to be encapsulated for its return path. This situation occurs at hop 4 on the example traceroute, shown in figure 6.7.

When the packet reaches the L_0 destination of the original encapsulator, the packet is decapsulated. However, the additional step of reconstructing the original packet as discussed in the previous example must also take place. Again, the ID field from the ICMP payload is used as an index into the local cache, and the appropriate error message is generated in the same fashion. ³

³This case was frustrating, as in the original testbed two of the devices were running on CentOS 5.2 which contains a bug in its routing stack that corrupts L_1 destination addresses when using the IP in IP protocol. The problem was rectified by moving to Ubuntu 9.08.

Figure 6.8: The packet is now decapsulated, and in standard “legacy” format. When generating an error message that needs to go back across the network, it needs to be encapsulated. The receiving end does not need to take any special action.

Encapsulated Header with Unencapsulated ICMP Payload - When the error occurred, it was in response to an unencapsulated packet. The generating side was in a different AS than the original source, and thus the ICMP packet was encapsulated to route back across the HIDRA network. No special action is needed; if the L_0 destination is local, then decapsulate the packet, otherwise let it continue on its way. This situation occurs when a packet times out at hop (6) of figure 6.8.

Given ICMP handling for all four error cases, HIDRA has the fairly unique attribute that a traceroute will show all L_1 hops, even inside of a tunnel. In many encapsulation or tunneling schemes the tunnel itself gets hidden, but because of the design decisions made concerning how the L_0 header is constructed, HIDRA becomes transparent and shows the full physical route.

6.3.2 MTU Errors

Maximum Transmission Unit (MTU) path discovery is described in RFC 1191 [35]. Essentially, the process attempts to find the maximum size a single packet can be across a certain path; when a packet is too large, an ICMP *Destination Unreachable* error message is generated, which contains the maximum packet size for that particular path. This ICMP error message is subject to all the considerations previously discussed, but also poses a new quirk – this MTU must also take into account encapsulation, and the end host may or may not know about encapsulation.

Handling this issue is fairly straightforward. The ICMP Destination Unreachable is like all ICMP error messages, and contains the packet which generated the error. The MTU error must first be handled as a normal ICMP error, as discussed in the previous section. Once the correct message has been generated through the previously outlined process an additional step required to process MTU specific errors, again based on where in the network the error occurred. In this case however, there are only two possibilities: the error was generated in response to an encapsulated packet, or the error was generated in response to an unencapsulated packet. This state is determined again by examining the incoming ICMP error's payload packet, and checking the destination field to see if it is in the well-known HIDRA /8 prefix or not.

In the event that the error message was generated in response to an encapsulated packet, in addition to doing the normal ICMP error message restoration the field containing the MTU size must be decremented by 20 bytes. This is necessary, because the end host that generated the packet does not account for HIDRA's encapsulation. If the packet must be encapsulated to reach its location

(which, given that the error message contained an encapsulated response must be the case), HIDRA will always add 20 bytes in the form of the L_0 header to the original packet. As such, the extra headroom must be left on the original packet, meaning the effective MTU at the end host is 20 bytes smaller than the true network MTU.

Conversely, if the error message was generated in response to a packet that was not encapsulated, only the normal ICMP error message handling must occur. This is because at the point in the network where the packet exceeded the MTU, the L_0 header had already been decapsulated. As such, it's not necessary to account for that 20 bytes.

Chapter 7

Experiments

All experiments were run on the network topology depicted in figure 6.1, and the specific numbering depicted in figures 7.1 and 7.2. To ensure complete connectivity, an all-pairs ping was conducted before performing any experiment.

Figure 7.1: High Level ASN & Network Configurations

Figure 7.2: IPv4 addresses for endpoints in the testbed network

Figure 7.3: Ping latencies for HIDRA with end-site decapsulation vs. legacy routing.

7.1 Standard Connection Tests

The initial experiment was a simple base-line ping test between machines located in AS 6 and AS 4. This test was first run with the network in a legacy configuration. The legacy configuration is similar to how the Internet is operated now. The testbed was reconfigured for HIDRA using end-host encapsulation and decapsulation, and the ping was re-run. With end-host encapsulation each packet is encapsulated by the actual host it is being sent from before it is transmitted across the network. Decapsulation also takes place on the end-host. The results in both cases were very similar; both tests averaged pings of slightly under 1 millisecond, with the HIDRA network performing slightly slower as shown in figure 7.3. The performance difference is due to the extra CPU overhead of encapsulation and decapsulation.

Figure 7.4: Ping latencies for HIDRA between a host in AS4 and AS6 from figure 6.1. Compares legacy, end-host encapsulation, within AS encapsulation, and ISP encapsulation.

7.2 In-Network Encapsulation Test

In-network encapsulation is tested by adding an additional machine to both AS 6 and AS 4. One machine was running a recent unmodified Ubuntu release and the other Windows XP. Neither machine has the HIDRA software installed. These two new machines were used to measure ping latency. For the packets to get encapsulated correctly they get routed to an encapsulation device in the same site as the host. The L_0 header is placed on the packet by this device before the packet leaves the site. The same is true for decapsulation. There is a noticeable increase in latency because the packet is traversing an additional 4 links (to and from both the encapsulation and decapsulation device).

Next we look at ISP decapsulation. In this scenario, AS 4, 5, and 6 are all

configured as legacy networks and there is no HIDRA specific software running in any of those sites. AS 1, 2, and 3 are configured as HIDRA networks and tag routes originated by their customers as being part of L_1 . Packets may traverse unencapsulated from AS 6 to AS 4 because AS 3 contains the necessary L_1 routing information for both AS 6 and 4; the same relationships are true of ASes 3, 5, and 6. If the packet is destined for another AS the encapsulation device in AS 2 will encapsulate it. The specific path used in this experiment, from AS 5 to AS 6, requires encapsulation. The L_0 packet will be decapsulated as it enters either AS 2 or AS 3, depending on the direction of communication. Again, because the packet is traveling along four extra hops we see an increase in round trip latency, shown in figure 7.4. Regardless of latency, the success of the pings demonstrates that all the configurations can correctly forward traffic.

Figure 7.5: Ping latencies for HIDRA with manually induced failure after 30 packets.

7.3 Failure Tests

The next test utilized the same two network configurations as in the initial connection tests and demonstrates the failover capabilities of both networks. After the 30th packet was sent and the response received, the connection between AS 4 and AS 2 was manually broken. In both cases, this causes the active path between AS 4 and AS 5 to fail. BGP takes roughly 10 packets to route around the failure and use the longer path, AS 5 – AS 2 – AS 1 – AS 3 – AS 4. Because the packets traverse two extra hops, the round trip latency increases noticeably, shown in figure 7.5. After the 90th ping response was received, the link was restored. In both instances BGP detected and utilized the recovered link after

a short period of time. This is visible in the graph when the round trip latency dropped back down to the original time.

Failover in the ISP encapsulation configuration was the final experiment. Pings were sent from AS 6 to AS 4. Since the best route between AS 6 and AS 4 remains in the same L_1 , and because ISP encapsulation is used, the ping probes and responses are *not* encapsulated. After the 30th packet, the link between AS 2 and AS 4 is manually failed. Roughly 10 seconds after failure the pings are successful with a higher latency, due to the longer route. However this is more interesting because of the encapsulation involved. To utilize the AS 3 – AS 1 – AS 2 – AS 4 path, the packet must traverse L_0 and must be encapsulated. So, HIDRA automatically routed around the link failure *and* began encapsulating previously unencapsulated packets to do so. The link was restored after the 90th packet, and the network was able to adjust back to the original, unencapsulated path.

7.4 Policy Tests

Next we examine utilizing business peering agreements and failover policies. The network is configured for in-network encapsulation. Two machines are attached to AS 5, and AS 5 is configured to load balance it's traffic across it's two ISPs, AS 2 and AS 3. This is done by splitting AS 5's network into two smaller subnets, and configuring the upstream announcements to append an extra AS onto the BGP AS path attribute, a common traffic engineering practice.

Two ping packet streams are started from a location in AS 1, each destined for one of the different subnets in AS 5 as shown in figure [7.6](#). One stream travels via AS 2, and the other via AS 3. The actual paths are verified by use of a traceroute to the specific location the data stream is destined for.

Figure 7.6: Load balancing traffic paths. The packets originate at the same location, and end in the same AS, but based on the L_1 subnet the packet is destined for, it takes a different path.

After 30 packets, the AS 2 to AS 5 link is manually broken. The stream of packets continues, uninterrupted, now traversing through AS 3 instead of AS 2. Here we can see that the traffic which was previously being load balanced to take separate paths is now being sent along the same path, correctly accounting for the failed link. All traffic now follows the orange path in figure [7.6](#).

As an interesting side note, this test inherently reveals some complicated HIDRA-Legacy behavior. When the link between AS 2 and AS 5 is broken, no packets end up being dropped. During the BGP reconfiguration, packets from the originator are still routed as if there was no failure, displayed in figure [7.7](#).

Figure 7.7: On initial link failure, the originator still has the route entry in its BGP table, so the initial outgoing packet originated in AS 1 goes along the same path.

At this point, the packet is decapsulated because the destination AS is AS 2, the local AS. The packet is then sent back out to be routed to its correct L_1 address. Unfortunately, the link has broken, and the router has no direct connection to the specified L_1 address. Instead, the packet is routed back to the default entry announced by the encapsulation device for the AS; this routes the packet directly back to where it has just come from, as shown by figure [7.8](#)

Figure 7.8: The packet is decapsulated, and sent out to be routed to its L_1 location. The desired L_1 network is no longer accessible though, and so it's sent back to the encapsulation device.

At this point, the mapping lookup is done. Since the link failure was a local failure ¹, the BGP table on the local device has already been updated, and the second entry is utilized. The packet is encapsulated using AS 3's L_0 address, and sent back out for routing once again. The packet travels through AS 1, to AS 3, where it is then decapsulated, and sent on to the customer network, as depicted in figure 7.9.

¹The failure was not quite local, but occurred on a neighbor which is peered through iBGP rather than eBGP, and as such the updates are nearly instantaneous [46].

Figure 7.9: The packet is re-encapsulated, this time with the already updated BGP table. It is then routed around to AS 3, where it is decapsulated and delivered successfully to the customer network.

While this is not the most efficient route, as it bounces between AS 1 and AS 2 in a circular fashion, even in the event of immediate failure, the packet is still eventually routed to the correct location. This state lasts only until BGP fully reconfigures, at which time both streams travel the path through AS 3, and are delivered successfully to their respective end points, as depicted by figure [7.10](#).

Chapter 8

Future Work

The HIDRA design provides a strong fundamental basis to help reduce strain put on the DFZ forwarding table. However, almost equally importantly, HIDRA also provides a strong jumping off point from which many new innovations can take place. One of the most important pieces of possible future work is the incorporation of a reactive rather than proactive mapping; this work is already underway [36]. Additionally, new services such as IP Mobility become realistic possibilities given the hierarchical nature of HIDRA.

Another important aspect that should be taken into consideration is how IPv6 and HIDRA can interact. This is slightly more complicated in the proactive architecture because the existing Internet routers do not universally exchange IPv6 routes with BGP. Integrating IPv6 support with reactive routing is the path of least resistance for IPv6. Future work entails adding both proactive and reactive IPv6 support.

Chapter 9

Conclusion

As the Internet and the number of routes required to make the Internet run continues to expand at such an alarming rate, there is a pressing need for change to enable our hardware to keep pace. The concept of a hierarchical system has presented itself in many of the recent proposals, each with a different way to limit the required size of the DFZ forwarding table. Unlike other proposals, HIDRA offers a path to help effectively reduce both the immediate size as well as the rate of growth of the global DFZ forwarding table in an incremental fashion that attempts to remain fully backwards compatible. It utilizes many preexisting structures and protocols such as existing number allocation policy, BGP, and current router firmware. These pragmatic concerns separate HIDRA from many other proposals.

Additionally, HIDRA enables future improvements, such as adding a reactive routing protocol which will further reducing the strain put on core routers. In general, HIDRA provides a much more flexible base routing system that could

eventually be taken advantage of to provide new and unique services. As such, we feel it surpasses many other proposals in that it can be realistically integrated to the existing Internet architecture.

Bibliography

- [1] Netfilter - firewalling, nat, and packet mangling for linux.
- [2] *BGP: Building Reliable Networks with the Border Gateway Protocol*. 2002.
- [3] Internet history. http://www.computerhistory.org/internet_history, 2006.
- [4] A proposal for scalable internet routing & addressing.
<http://tools.ietf.org/html/draft-wang-ietf-efit-00>.
- [5] APT: a practical transit mapping service. <http://tools.ietf.org/html/draft-jen-apt-01>.
- [6] Autonomous system (as) numbers allocation table.
<http://www.iana.org/assignments/as-numbers>.
- [7] V. Cerf. IAB recommended policy on distributing internet identifier assignment and IAB recommended policy change to internet “connected” status. RFC 1174 (Informational), Aug. 1990.
- [8] R. Colella, E. Gardner, and R. Callon. Guidelines for OSI NSAP Allocation in the Internet. RFC 1237 (Proposed Standard), July 1991. Obsoleted by RFC 1629.

- [9] E. Davies and A. Doria. Analysis of Inter-Domain routing requirements and history. <http://smakd.potaroo.net/ietf/idref/draft-irtf-routing-history/index.html>.
- [10] E. Davies and A. Doria. Analysis of Inter-Domain Routing Requirements and History. RFC 5773 (Historic), Feb. 2010.
- [11] B. Dickson.
- [12] A. Doria, E. Davies, and F. Kastenholz. A Set of Possible Requirements for a Future Routing Architecture. RFC 5772 (Historic), Feb. 2010.
- [13] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. The COPS (Common Open Policy Service) Protocol. RFC 2748 (Proposed Standard), Jan. 2000. Updated by RFC 4261.
- [14] K. Egevang and P. Francis. The IP Network Address Translator (NAT). RFC 1631 (Informational), May 1994. Obsoleted by RFC 3022.
- [15] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/ID separation protocol (LISP). <http://tools.ietf.org/html/draft-farinacci-lisp-12>.
- [16] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). RFC 2784 (Proposed Standard), Mar. 2000. Updated by RFC 2890.
- [17] A. Feldmann, L. Cittadini, W. Mühlbauer, R. Bush, and O. Maennel. Hair: hierarchical architecture for internet routing. In *ReArch '09: Proceedings of the 2009 workshop on Re-architecting the internet*, pages 43–48, New York, NY, USA, 2009. ACM.

- [18] Final report on ipv6 deployment issues.
<http://www.6net.org/publications/deliverables/D2.5.3.pdf>.
- [19] P. Francis and R. Gummadi. Ipn1: A nat-extended internet architecture. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 69–80, New York, NY, USA, 2001. ACM.
- [20] V. Fuller, T. Li, J. Yu, and K. Varadhan. Supernetting: an Address Assignment and Aggregation Strategy. RFC 1338 (Informational), June 1992. Obsoleted by RFC 1519.
- [21] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. RFC 1519 (Proposed Standard), Sept. 1993. Obsoleted by RFC 4632.
- [22] R. Gao, C. Dovrolis, and E. W. Zegura. Interdomain ingress traffic engineering through optimized as-path prepending. In *In Proceedings of IFIP Networking*, pages 647–658, 2005.
- [23] E. Gerich. Guidelines for Management of IP Address Space. RFC 1466 (Informational), May 1993. Obsoleted by RFC 2050.
- [24] Generic routing encapsulation (gre). rfc2784.
- [25] I. E. S. Group and R. Hinden. Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR). RFC 1517 (Historic), Sept. 1993.
- [26] Hitesh Ballani and Paul Francis and Tuan Cao and Jia Wang. Making Routers Last Longer with ViAggre. In *Proc. of USENIX Symposium on Networked Systems Design and Implementation*, Apr 2009.

- [27] G. Huston. BGP Routing Table Analysis Report.
<http://bgp.potaroo.net/as6447/>.
- [28] G. Huston. Scaling the internet's routing system - ISOC member briefing #3. <http://www.isoc.org/briefings/003/>, 2001.
- [29] Ivip - scalable routing and addressing architecture for the internet.
<http://www.firstpr.com.au/ip/ivip/>.
- [30] D. Jen, M. Meisel, H. Yan, D. Massey, L. Wang, and B. Zhang. Towards a new internet routing architecture: Arguments for separating edges from transit core. New York, NY, USA, 2008. ACM.
- [31] S. Lee, Y. Yu, S. nelakuditi, and Z. li Zhang. Proactive vs. reactive approaches to failure resilient routing. Technical Report 03-031, University of Minnesota - Computer Science and Engineering, August 2003.
- [32] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, and S. Wolff. Internet society (ISOC) all about the internet: History of the internet. <http://www.isoc.org/internet/history/brief.shtml>, 2003.
- [33] M. Little. Goals and functional requirements for inter-autonomous system routing. RFC 1126, Oct. 1989.
- [34] C. Mbarushimana and A. Shahrabi. Comparative study of reactive and proactive routing protocols performance in mobile ad hoc networks. In *AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*, pages 679–684, Washington, DC, USA, 2007. IEEE Computer Society.

- [35] J. Mogul and S. Deering. Path MTU discovery. RFC 1191 (Draft Standard), Nov. 1990.
- [36] D. Nelson. Internet-scale reactive routing and mobility. Master's thesis, California Polytechnic State University, San Luis Obispo, June 2009.
- [37] J. Niccolai. Study shows glacial pace of ipv6 adoption. http://www.pcworld.com/businesscenter/article/150013/study_shows_glacial%_pace_of_ipv6_adoption.html, August 2008.
- [38] U. of Oregon. Route views project. <http://www.routeviews.org/>.
- [39] C. Perkins. IP Encapsulation within IP. RFC 2003 (Proposed Standard), Oct. 1996.
- [40] J. Postel. Internet Control Message Protocol. RFC 792 (Standard), Sept. 1981. Updated by RFCs 950, 4884.
- [41] J. Qiu and L. Gao. Robust egress interdomain traffic engineering.
- [42] Y. Rekhter and T. Li. An Architecture for IP Address Allocation with CIDR. RFC 1518 (Historic), Sept. 1993.
- [43] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771 (Draft Standard), Mar. 1995. Obsoleted by RFC 4271.
- [44] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), Jan. 2006.
- [45] Y. Rekhter and E. Rosen. Carrying Label Information in BGP-4. RFC 3107 (Proposed Standard), May 2001.

- [46] E. Rodriguez. An introduction to the bgp protocol. <http://www.skullbox.net/bgp.php>, June 2004.
- [47] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), Jan. 2001.
- [48] Routing research group - IRTF. <http://trac.tools.ietf.org/group/irtf/trac/wiki/RoutingResearchGroup>.
- [49] Shim6: Level 3 multihoming shim protocol for ipv6. rfc5533 ; <http://www.shim6.org/>.
- [50] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663 (Informational), Aug. 1999.
- [51] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica. Hlp: a next generation inter-domain routing protocol. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 13–24, New York, NY, USA, 2005. ACM.
- [52] F. Templin. The IPvLX architecture. <http://tools.ietf.org/html/draft-templin-ipvlx-08>.
- [53] TRRP. <http://bill.herrin.us/network/trrp.html>.
- [54] C. Vogt. Dns map - a dns-based resolution system for ip address mappings.
- [55] C. Vogt. Six/one router: a scalable and backwards compatible solution for provider-independent addressing. In *MobiArch '08: Proceedings of the 3rd international workshop on Mobility in the evolving internet architecture*, pages 13–18, New York, NY, USA, 2008. ACM.

- [56] R. Whittle. How do modern high-end routers implement the forwarding information base function? <http://www.firstpr.com.au/ip/sram-ip-forwarding/router-fib>.
- [57] T. Worster, Y. Rekhter, and E. Rosen. Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE). RFC 4023 (Proposed Standard), Mar. 2005. Updated by RFC 5332.
- [58] L. Yang, R. Dantu, T. Anderson, and R. Gopal. Forwarding and Control Element Separation (ForCES) Framework. RFC 3746 (Informational), Apr. 2004.
- [59] X. Yang. Nira: a new internet routing architecture. In *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pages 301–312, New York, NY, USA, 2003. ACM.
- [60] X. Zhang, P. Francis, J. Wang, and K. Yoshida. Scaling ip routing with the core router-integrated overlay. In *ICNP '06: Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 147–156, Washington, DC, USA, 2006. IEEE Computer Society.