REACTIVE ENCAPSULATION MAPPINGS IN HIDRA

Scott M. Marshall, John M. Bellardo
Computer Science Department
California Polytechnic State Univ.
San Luis Obispo, CA, USA
email: {scmarsha,bellardo}@calpoly.edu

Daniel Nelson, Bryan Clevenger
Research done while students at:
California Polytechnic State Univ.
San Luis Obispo, CA, USA
email: daninels@cisco.com, bcleveng@gmail.com

ABSTRACT
Scalability analysis of the Internet has resulted in two main concerns: rapid growth of the forwarding table and BGP's poor convergence properties when distributing hundreds of thousands of routes. HIDRA [5], a backward-compatible architecture designed with feasibility-of-implementation in mind, has been proposed as one solution to reduce the size of the default-free zone (DFZ) forwarding table.

This work extends HIDRA, greatly reducing the number of routes maintained by BGP, yet preserves a practical, incremental deployment strategy. The proposed protocol also provides end networks direct, finer-grained control over the distribution of packets flowing into their network and provides for efficient mobility support.

The new mapping protocol is prototyped on a small network testbed and shown to work in all tested circumstances, including normal network operation, link failures, and transitional routing environments. Additionally, IP Mobility is discussed and shown to work in this environment without triangle routing and only minimal additional overhead.

KEY WORDS
Routing Protocols, IP based Networks, Internet Architectures, HIDRA

1 Introduction

Concerns about the scalability of today's Internet structure are well-known and generally fall into one of two categories. There is substantial concern about the continued growth of the DFZ forwarding table and its resulting impact on the cost of routers and routing policy [3]. The second area of concern is the performance, stability, and convergence properties of BGP as it exchanges an ever increasing number of less-stable routes [16].

HIDRA [5], a Hierarchical Inter-Domain Routing Architecture, is an architectural approach to reducing the size of the DFZ forwarding table. HIDRA divides the Internet into a multi-level hierarchy, using IPv4 encapsulation to forward packets between different levels of the hierarchy. HIDRA's overriding design constraint is deployability. It goes to great lengths to maximize compatibility with existing protocols such as IPv4 and BGP, current network hardware, number resource policy, and existing business constraints. HIDRA substantially reduces both the number of routes in the DFZ forwarding table and the table's projected growth rate.

This work extends HIDRA by adding a reactive mapping system for all levels of the network hierarchy below the root. This mapping protocol addresses the scalability concerns of BGP, reducing the number of mappings BGP announces by an order of magnitude. The mapping protocol also enables traffic engineering and other coarse routing policies to be applied at the individual host level without exponential growth of any single mapping table.

This work preserves HIDRA's original focus on practical, deployable solutions. In particular, it doesn't change the overall architecture or depend on any router hardware or firmware upgrades. The mapping protocol leverages most of the existing DNS software and management infrastructure. The only additional changes are updates to the DNS servers of organizations hosting the mapping records. HIDRA can seamlessly use either its original, BGP-based mapper or the reactive, DNS mapper.

The reactive mapper has been implemented and tested within the original HIDRA testbed. This work presents results demonstrating that normal operations and link failure recovery function as expected. In addition, it provides some initial insight into the performance and latency of the mapper.

This work also discusses the positive impact that reactive routing has on IP mobility, namely that zero-stretch IP mobility is feasible. Initial mobility results are shown that demonstrate both the operational correctness of mobility and its minimal additional overhead.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 provides the minimally necessary background context on HIDRA and proactive mapping to understand this work. Section 4 details the reactive mapping protocol that was added to HIDRA. Section 5 addresses reactive mapping's impact on mobility. Section 6 presents the prototype implementation and experimental results. Section 7 explains some future directions and Section 8 concludes.

2 Related Work

There are several existing papers which contain various aspects of reactive routing similar to those added to HIDRA.

However, HIDRA is unique in that it strives to maintain a viable migration strategy from the current Internet architecture to the proposed alternative. This overarching objective results in a combination of design decisions not found in other work.

Feamster *et. al.* [7] identify many of the benefits and motivation for using reactive routing. Most of the research focuses on peer-to-peer overlay networks, so all of their mechanisms aren't directly applicable.

In [21], Yang *et al*. describe NIRA, A New Inter-Domain Routing Architecture. The proposed reactive routing scheme uses rate-limited ICMP messages as a means to notify network nodes of a change in route availability. This approach is similar to that used in HIDRA. Adoption of NIRA may be impeded due to the new proactive and reactive routing protocols proposed.

Carpenter *et.al.* presents a series of architectural principles for the Internet [2]. One of the principles, "Circular dependencies must be avoided", motivates the use of HIDRA's mixed proactive and reactive routing to avoid circular dependencies.

The Tunneling Route Reduction Protocol (TRRP) [18] proposes using DNS [11] to distribute gateway addresses globally and tunnels to reduce total route count. This would provide reactive routing capabilities to end users via the use of reverse lookups on IP addresses to determine encapsulation information for a destination. The data contained in HIDRA's ENCAP records is based on this work, however TRRP's use of DNS TXT records rather than a more specific resource record type may impede implementation. Other routing proposals, such as [1], also use DNS TXT records as the basis for exchanging mapping information.

Feldmann *et.al.* [8] propose hierarchical architecture for Internet routing (HAIR). HAIR includes many of the same concepts as HIDRA, such as multiple level routing hierarchy and support for efficient mobility. It uses an alternative reactive mapping service. It also includes initial performance results showing the benefits a hierarchal network architecture can have for mobility. HAIR's use of IPv6 and a new mapping service separate it from HIDRA.

3 HIDRA Background

The reactive mapping system described in this paper was developed within the context of HIDRA, however most of the mechanisms aren't HIDRA specific. This section provides an overview of general HIDRA operation, including encapsulation and proactive mapping. Readers are referred to the original HIDRA [5] work for more complete details.

HIDRA is a hierarchal network architecture, with level 0, abbreviated L_0, at the top of the hierarchy. There are at least two levels in the network. ISPs are generally members of both L_0 and L_1, enabling them to forward traffic between levels of the hierarchy and transmit packets across L_0. Stub and multi-homed end-site networks are generally members of L_1 only. Those networks reach L_0 through their upstream provider(s).

Packets are encapsulated as they traverse up the hierarchy, such as from L_1 to L_0. Each level adds an additional IPv4 encapsulation header. In the common case of a two-level hierarchy, packets being forwarded through L_0 have two IPv4 headers. The packets are forwarded based on their L_0 destination address, arriving at the immediate upstream provider for the destination network. After arriving at the last-hop ISP, the L_0 header is removed via decapsulation and the original packet is forwarded to its L_1 destination. The process is similar when there are more than two levels in the hierarchy, except there are more encapsulation and decapsulation operations. As a rule of thumb, a hierarchy with n levels will result in $n - 1$ encapsulation and decapsulation operations. This general network architecture, map-encap [12], is not novel and has been both used and studied in other literature, *e.g.* [19, 6].

3.1 L_0 Addressing

HIDRA uses IPv4 as the L_0 protocol. This places an additional burden on the encapsulation operation, as it must map a destination L_1 address into an IPv4 L_0 address. The L_0 address must be that of an immediate upstream provider for the L_1 destination. This address can be thought of as a node's *location* in the network. HIDRA uses a network's *autonomous system number* (ASN) to determine its L_0 address. ASNs are already managed by the existing Internet Assigned Numbers Authority (IANA) infrastructure.

L_0 addresses are formed by combining a well-known /8 prefix with the 24 least significant bits of the ASN. This mechanism has a number of benefits, including the ability to filter and route all L_0 traffic with a single rule that matches the well-known /8. It also partially solves the mapping problem. It is no longer necessary to map an arbitrary L_1 address to an arbitrary L_0 IPv4 address. All that is needed is to map an L_1 address to an ASN.

3.2 Proactive Mapping

Proactive mapping distributes the entire L_1-to-ASN mapping table *before* any mapping is performed, and continuously updates it in response to topology changes. The primary drawbacks of this technique are higher memory requirements for storing the entire table and more data exchanged by the mapping protocol. The biggest benefit is an extremely low first packet latency. Since the entire table is already stored in DRAM, the actual mapping operation is simply an efficient table lookup.

HIDRA uses unmodified *Border Gateway Protocol* [14] (BGP) to proactively maintain the L_1-to-ASN mapping table. This was chosen to maximize backward compatibility with existing Internet routes, route advertisements, and routing policy. Unmodified BGP works well because each route advertisement already contains all the necessary information – the destination network prefix and

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

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

ASN which originated the route. HIDRA keeps all proactive routing information in the BGP routing table (RIB), but prevents it from being added to the expensive forwarding table (FIB), thereby reducing the number of routes in the DFZ.

3.3 Encapsulation

HIDRA requires at least one device within each network be able to encapsulate traffic before it traverses L_0. This device is responsible for resolving L_1 addresses to ASNs. The first encapsulation device must be reachable by the packet's source without needing additional encapsulation. Encapsulation is a relatively slow operation. To minimize negative impacts on scalability, the encapsulation device should be as close to the source as possible. Ideally the originating host will encapsulate its own packets (termed *host-based encapsulation*).

In proactive mapping, each encapsulation device needs to actively receive BGP advertisements. This effectively precludes host-based encapsulation, necessitating a network-based encapsulation device. In contrast, the reactive mapping scheme proposed here removes the need to run BGP on encapsulation devices, enabling host-based encapsulation based on DNS records.

3.4 Decapsulation

Decapsulation removes the L_0 header and forwards the packet according to the remaining L_1 header. This should take place as close to the $L_0 \Leftrightarrow L_1$ boundary as possible. In HIDRA, this point is immediately after the packet enters the nearest upstream network provider of the final destination. This point is *not* within the destination network, even if that network has been issued an ASN. This placement enables the decapsulated packet to traverse the most efficient path through L_1 to its destination. Decapsulation is a relatively fast operation that doesn't require table lookups.

Decapsulation within the final transit AS creates an asymmetry between the encapsulation and decapsulation location. Packets are typically encapsulated within the customer's network, or, in the worst case, at the provider's end of the customer's access link. They are decapsulated as soon as they enter the destination provider's network.

4 Reactive Mapping

The key difference between the original proactive mapping system presented in [5] and reactive mapping is *when* the mapping data is distributed over the network. Reactive protocols don't distribute the state information *a priori*. Instead, they resolve the mapping information on-demand as part of the encapsulation process. This results in more mapping latency when compared to proactive systems. Mapping entries are cached to minimize the lookup latency of future packets sent to the same destination. As a result, the latency on subsequent packets is the same in both proactive and reactive schemes.

The reactive mapping software must be installed on all encapsulation devices. During initial deployment, there will only be a small number of these per HIDRA-aware network. This minimizes the impact of adopting reactive routing. Some features enabled by reactive routing add additional encapsulation requirements. For instance, when using mobility, it is necessary for the mobile host to perform all of its own encapsulation and mapping.

DNS [11] is used to store and distribute the destination-specific mapping data. This choice was made for several reasons. First, control of DNS is distributed to organizations by a recognized central authority. Organizations are free to modify their own entries as they deem necessary, allowing distributed, autonomous control. Any new lookup protocol would need to have a similar administrative structure.

DNS is already widely deployed. Its strengths, weaknesses, and operational characteristics are well understood. It is critical for many common Internet activities, such as web browsing and email. DNS servers cache entries closer to the encapsulation device, reducing lookup latency. This is true even if the server doesn't understand the format of the resource record it is caching.

DNS is already used during the initial stages of establishing most Internet connections. These connections begin by resolving a DNS name into an IP address. Piggybacking the reactive mapping record in the response to the forward name lookup virtually eliminates any first-packet latency for these connections.

Finally, DNS can be made secure through technologies such as DNSSEC [20], which allows for signing of records, preventing spoofing attacks. It would be possible

to use an entirely new protocol for reactive routing lookups rather than DNS, but the overhead in designing, testing, deploying, and building a general operational consensus around such a new system will most likely be too high a hurdle to overcome.

In addition to including the reactive mapping information in a forward DNS lookup, an end host can explicitly lookup an appropriate mapping by performing a standard reverse DNS lookup for the L_1 destination addresses. The reverse lookup query requests the ENCAP record instead of the standard PTR record. This technique doesn't mask first packet latency, but is a necessary fallback for packets not preceded by a forward lookup.

4.1 ENCAP Records

To distribute mapping records with DNS, a new resource record type, called the ENCAP record, is defined. ENCAP records are returned from both a forward or reverse DNS lookup. They contain the six fields summarized in Table 1 and described here. The *Data* field specifies the encapsulation address. Its exact format and length is variable. For example, an ASN or IPv4 encapsulation address requires 4 bytes while an IPv6 encapsulation address would require 16 bytes.

The two-byte *Type* field defines the format of the data. Currently, two types are implemented: ASN, and A. ASN signifies that a four byte ASN is contained in the data field. The A type indicates that the data field contains a standard DNS type A resource record – a four byte IPv4 address.

The one-byte *Flags* field contains various informational flags from the server. All bits in this field are reserved for future use. The one-byte *Level* field specifies the hierarchal level of the ENCAP resource record. This is a performance optimization for networks with more than 2 levels. A query response for a network of n levels must contain all records for levels $0 \ldots n-1$.

The one-byte *Priority* and *Preference* fields allow network administrators to provide failover and load balancing capabilities for encapsulation addresses. When several records with the same level are returned, the encapsulation device must choose the record with the highest priority, and fail over to entries in decreasing priority order. If there are multiple records with the same level and priority, the *Preference* field provides load-balancing. Subsequent packets to the same destination should be encapsulated using each different record with the same priority. The preference field defines a weighted distribution used to determine how frequently a particular mapping should be used.

4.2 Circular Dependency

A circular dependency is created by the reactive system. A reactive lookup requires a DNS Query packet be sent to a remote DNS server, which is typically located in a different L_1 network. For instance, most ASes don't operate their own *root* DNS server. This query requires encapsulation to reach the server. Encapsulating the DNS Query requires resolving the DNS server's L_1 address into its L_0 address, which requires another reactive lookup completing the cycle.

DNS already addresses a similar dependency. A resolver needs to know where the root nameserver is before any lookups can be performed. The resolver uses a "hints" file to seed this information. As the lookup progresses down the name hierarchy, each nameserver provides the address "glue" records for the next server down the chain.

There are two ways to break this dependency in HIDRA. The first option is to include ENCAP records in the standard DNS "hints" file alongside the A records for the root servers. Since this may prove problematic to deploy, a second option is proposed. Proactive mapping can be used to distribute the encapsulation state for, and forward packets to, the root nameservers. The root nameservers contain the "glue" encapsulation records for the next nameservers down the chain. This is termed a *mixed-mode* network, where the most critical infrastructure routes are handled proactively and everything else reactively. Mixed-mode mapping breaks the aforementioned dependency cycle because the mapping for the root DNS server's L_0 address no longer depends on a DNS lookup.

4.3 Failure Detection

Reactive mapping complicates link failure detection and rerouting. In a BGP-based proactive system, link failures are detected and routed around quickly, on the order of tens of seconds [1]. Reactive schemes don't have this luxury. Instead, the primary mechanism for refreshing bad mappings is timeouts. When a cached mapping times out, it is discarded. If another packet for the same destination needs to be encapsulated, a new reactive lookup is performed. Achieving the same fast detection properties of proactive mapping requires a very low timeout value. This has the undesired impact of dramatically increasing the overhead and network resource requirements of the reactive mapping service, negating the performance benefits achieved with caching.

HIDRA's reactive mapping mitigates the slow detection speeds by requiring all encapsulation devices to notify the L_1 packet source when the L_1 destination is no longer reachable with a particular ENCAP record. This is determined by maintaining a small amount of additional state information in the encapsulation device. The device already tracks the set of *present* L_1 routes. It also tracks the set of *past* L_1 routes – those routes that were previously in its L_1 but are currently not. This information is maintained via the iBGP peering session(s) the encapsulation device establishes with its L_1 routers. Before a packet is encapsulated, its L_1 source and destination address are verified. If both addresses are *not* part of the encapsulation device's

[1] Exact timings will vary with the BGP settings of the routers involved in detecting the failure and propagating the associated update(s).

L_1 network, an ICMP Redirect [15] packet is returned to the L_1 source of the packet.

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

After a node receives an ICMP Redirect, it marks the encapsulation mapping as unavailable. The unavailable entry is determined by examining the ASN within the message payload and the destination address found in the IP header within the payload of the ICMP packet. If the newly unavailable entry is the only remaining mapping with the highest priority, the next highest priority set of mappings is used. If, after marking the current entry as unavailable, there are more mappings with the same priority, only the remaining valid mappings are considered.

4.4 Failure Recovery

Detecting and responding to a link recovery in a timely manner has the same challenges as detecting the failure. Namely, using a strictly timeout-based technique creates an undesirable tradeoff between recovery performance and network overhead. HIDRA's solution to this problem is similar to the technique it uses to detect failures, but it requires that even more state be tracked by the encapsulation devices. Every time an encapsulation device sends an ICMP Redirect (Section 4.3) to a new L_1 destination, this destination is added to a *recovery notification table*. Once the link has been restored, the encapsulation device sends an additional ICMP packet to every L_1 destination in this list.

Link recovery is inferred based on the contents of the *past* and *present* L_1 route sets. If a new L_1 route is learned that was not known in the past, no link recovery has occurred. If a L_1 route was known in the past, had its route advertisement withdrawn and then re-announced, a link recovery took place and the notifications are sent. Since the past and present L_1 routes are maintained via iBGP, this detection occurs shortly after a link is brought up.

The amount of memory required to store this state is much greater than that needed for the failure detection. Sending the ICMP Redirects on link recovery requires one table entry per actively transmitting $< source, L_1 >$ route pair. Failure detection requires just one table entry per L_1 route that is withdrawn as a result of the failure.

5 Mobility

Widespread use of IP mobility has two main requirements. First, a mobile host must be addressable by the same identifier, no matter its location in the network. Second, communicating with the host while it is mobile must not incur a substantial performance penalty [9, 10]. Existing Mobile IP standards [13] meet the first requirement via tunnels and triangle routing, but fall short of the second requirement [17]. HIDRA easily meets both these requirements with only slight modifications.

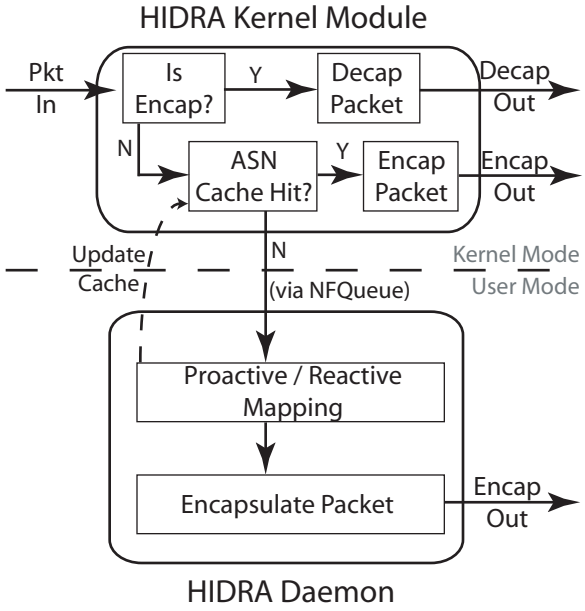

Figure 1. HIDRA software design overview and packet processing flowchart.

Zero-stretch mobility requires the addition of a third level to the routing hierarchy and a corresponding encapsulation header. The innermost layer, L_2, is the mobile node's constant identifier. The L_1 address is assigned by the current host network and is non-mobile. It is part of the mobile node's location identifier. L_0 is the network address for L_1, completing the node's location identifier. This system enables efficient routing of the packet directly to L_1. The only performance penalty is due to the extra encapsulation header, which causes a slight reduction in the amount of useful data that can be transmitted in a single packet. There is a very slight increase in per-packet latency due to the additional decapsulation operation and more complex encapsulation operation. Mobility also places additional requirements on the mobile node.

Mobile nodes must decapsulate a portion of each packet. They must be able to remove the L_1 header, making it appear as if the packet only has the L_2 header. The mobile node must also be able to update its current location in the mapping system. In HIDRA, this entails modifying one or more of the ENCAP records (Section 4.1) for the corresponding mobile identifier address. Existing dynamic DNS update protocols already provide the necessary secure mechanism for this.

Mobility also creates new constraints on address allocation. The mobile node must be able to retain the identifier address from its home network while acquiring a new L_1 address from its new host network. The home network must not re-assign the mobile node's identifier address. The

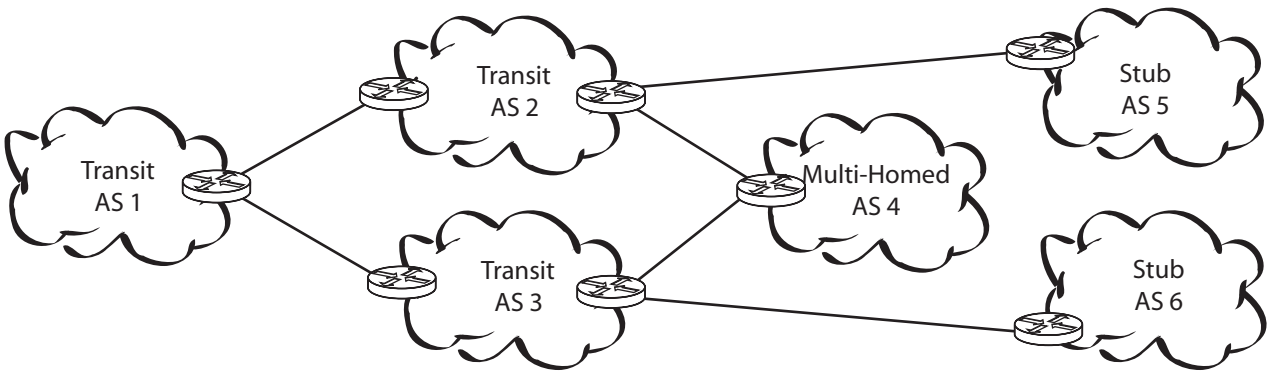

Figure 2. Testbed network topology diagram

uniqueness of these addresses must be preserved.

Detecting when a host moves is challenging because the only directly-observable event is a communication failure. The same event can be caused by many conditions, including a power failure, a transient network fault, or migration of the mobile node. In any of these cases, the stationary node's encapsulation device immediately performs another reactive lookup for the mobile host.

Timeouts are typically used to infer communication failure. These can take anywhere from tens of seconds to minutes. Depending on the mobile node's rate of movement, this delay may introduce unnecessary overhead into the communication. A more graceful alternative allows the mobile node to provide a "mobility hint" to the other node. For security reasons, this can only be a hint and can't actually contain the updated mapping information. The mechanism for delivering this hint is an IP Options flag. Responding to the hint is the same as responding to a communication failure.

6 Experimental Validation

The reactive mapping protocol presented in this work has been added to the existing HIDRA prototype. This implementation demonstrates basic proof-of-correctness and enables an early understanding of the additional overhead involved with the network architecture.

6.1 Software

The original Linux-based proactive HIDRA implementation [5] was extended to include support for reactive mapping. An overview of the design is shown in Figure 1. It includes a kernel module for performing fast encapsulation and decapsulation operations, and a user-level daemon for resolving all mappings. Once a mapping is resolved by the daemon, an entry is added to the kernel's cache to prevent future packets sent to the same destination from making the slow kernel→user trip.

Adding reactive routing support involved three substantial changes to the existing implementation. HIDRA's user-space daemon was modified to perform the reverse lookups and parse the resulting DNS ENCAP records as described in Section 4. The kernel module and HIDRA daemon were modified to generate the new ICMP Redirect errors after a link failure that results in an encapsulation change. Finally, both pieces were modified to correctly respond to the new ICMP Redirect errors by falling back on ENCAP records with a lower priority. Recovery after a link is restored remains future work.

It was also necessary to modify a DNS server to add support for the new ENCAP resource records, including returning the appropriate glue records in response to queries for NS records. Due to its popularity and relatively clean design, ISC BIND [4] was modified to include these features.

6.2 Experimental Setup

A six-AS network testbed was created using the prototype HIDRA implementation, Cisco routers, and Cisco switches. The AS-level topology of the testbed is illustrated in Figure 2.

During reactive routing tests, the root DNS server was located in AS1. Two top level domain (TLD) servers were located in AS3 and AS2. These were authoritative for different TLDs. There are no redundant DNS servers in this configuration.

The encapsulation mapping for the root DNS server, a /32 IPv4 route, is distributed with HIDRA's proactive mapping protocol. This mapping for the root DNS server is the *only* mapping distributed via the proactive protocol. All mappings for other DNS servers, including TLDs and reverse name lookups, are reactive. Using this setup for all the experiments demonstrates that the "mixed-mode" proactive-reactive mapping environment described in Section 4.2 works as expected.

6.3 Reactive Protocol Results

Two sets of experiments were done with the testbed to verify operational characteristics of reactive mapping in HIDRA. The first focuses on quantifying the amount of

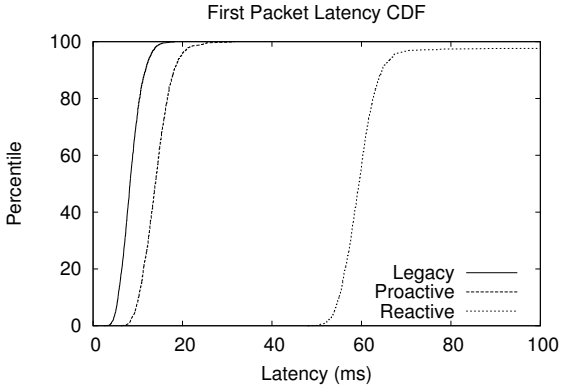

Figure 3. CDF of observed first packet *roundtrip* latencies for legacy (*e.g.*non-HIDRA), proactive mapping, and reactive mapping within the testbed. All caches, including host ARP caches, were cleared. A few reactive data points at 850 ms (starting around the 97^{th} percentile) are truncated to make the graph more readable.

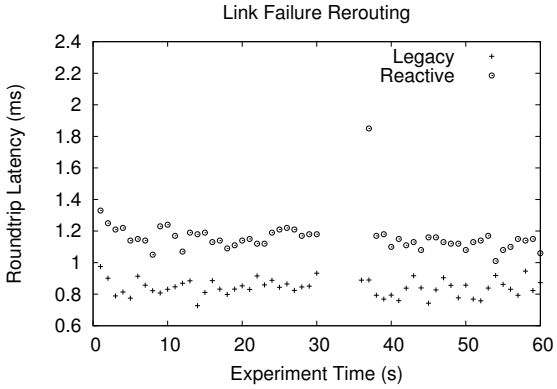

Figure 4. Ping latencies between a host in AS1 and AS4 before and after link failure. The link between AS2 and AS4 was unplugged at time 30. Connectivity is restored, using the path through AS3, at time 36 in the legacy network and time 37 with reactive mapping. Missing data points indicate packets never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

first packet latency that reactive mapping introduces, and the second demonstrates the network's responsiveness to link failures.

6.3.1 First Packet Latency

In this experiment, 2000 ICMP Echo (ping) packets were sent from a host in AS6 to a host in AS5. All encapsulation and decapsulation was done in AS3 and AS2. Before each ping was sent, all caches (ARP, DNS, HIDRA kernel module, etc) were flushed. Figure 3 is a CDF comparing all 2000 *roundtrip* ping latencies for three different network configurations: "legacy" networks (*e.g.* the current Internet architecture), HIDRA with proactive mapping only, and HIDRA with reactive mapping. The difference between the legacy and proactive HIDRA curves, roughly $5.5ms$, is attributable to the extra CPU overhead and slightly longer paths taken by the packets for encapsulation and decapsulation. The difference between the proactive and reactive curves, $45.5ms$, is the additional latency incurred with reactive mapping. This additional time is attributable to DNS operations including recursively resolving the reverse DNS lookup, encapsulating the DNS request packets, resolving the L_0 destination for the requests, and encapsulation / mapping for the responses. One-way latencies are slightly more than half the roundtrip latencies due to caching while the ICMP Echo Request traverses the network. Note that only the relative timing values are meaningful. The absolute time values can vary greatly depending on node CPU power, network equipment capability, and size of the network.

6.3.2 Link Failure Test

This test demonstrates how quickly HIDRA with reactive mapping can detect and route around a link failure that results in a mapping change. Correctly and quickly fixing these failures is one of the primary roles of a routing protocol. To test network operation in spite of link failures, a steady stream of pings, one per second, was sent from AS1 to AS4. Due to the *priority* setting in the reactive mapping, these packets traverse the path AS1→AS2→AS4. After the 30th packet was sent and the response received, the connection between AS2 and AS4 was unplugged. This is seen as packet loss (*e.g.* no data points) in Figure 4 starting at time 31. Packets continue to get lost until the network repairs itself, at time 36 for the legacy network and time 37 for the reactive network. This extra delay is inherent in reactive mapping, as the ICMP Redirect packet isn't sent until the underlying BGP routes in L_1 have stabilized. This causes reactive recovery to be delayed by 1 packet, which equates to 1 second in this experiment.

Also note the higher latency on the 37^{th} reactive packet. This is due to the packet being processed by the HIDRA daemon. When the ICMP Redirect is received, the corresponding entry in the kernel encapsulation cache is flushed, causing the next packet to be processed by the user-space daemon. The daemon determines the current set of valid mappings and installs them into the kernel's cache. Since there was a lower priority, but still valid, mapping returned in the original DNS response, no DNS query took place. Had a DNS query been necessary this latency would have been higher.

6.4 Mobility Results

Mobility, as described in Section 5, was implemented and tested within HIDRA's reactive network architecture. Since mobility was tested using an earlier version of the testbed and software, there are a few important differences from the previous experimental setup. The testbed consisted of two ASes, the minimum to test inter-AS mobility. Due to lim-

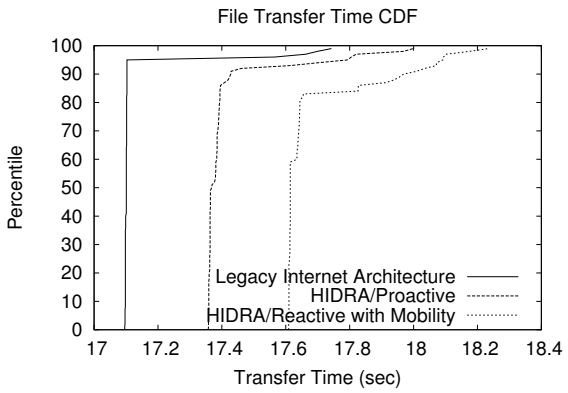

Figure 5. CDF of seconds required to transfer a 20MB file in the legacy, HIDRA proactive, and HIDRA reactive with mobility enabled network architectures. All tests performed 100 times.

itations in available network equipment, the link between the two AS's routers was 10Mbit Ethernet. The end hosts and nameservers in the network were also older, slower computers.

Figure 5 compares the length of time necessary to transfer a 20 Megabyte file using three different architectures: current Internet architecture (legacy), HIDRA with proactive mapping, and HIDRA with reactive mapping and a mobile host. During the mobility experiments, the host was actually mobile – it was not connected to its home AS. However, the mobile host wasn't moving, and it didn't change attachment points during the file transfer. The slight slowdown in the mobile case is due to the three-level network hierarchy. This extra level adds an additional 20 bytes of header to every packet. This results in a roughly 1.36% reduction of the amount of file data in a single packet. Data collected, but not graphed here, shows the packet count increases roughly 1.45% over the proactive architecture. As seen in Figure 5, absolute transfer time increases roughly 1.55%. These experiments validate both the feasibility of mobility within HIDRA and its low cost.

7 Future Work

This paper presents a design for quicker detection of link recoveries when using reactive mapping. This recovery mechanism hasn't been prototyped yet. An important future direction is completing this implementation, testing, and working out any design quirks.

Support for using IPv6 as the L_1 protocol is another important feature that needs to be added to HIDRA. The IPv6 implementation should focus on using reactive mapping since IPv6 doesn't currently enjoy widespread deployment. Using a purely reactive environment enables deployment over today's IPv4 backbone with unmodified routers. Adding IPv6 proactive mapping and the ability to use IPv6 as the L_0 protocol is also future work.

8 Conclusion

This work presents a new reactive mapping scheme for HIDRA. The scheme leverages existing DNS servers by distributing a reactive mapping resource record through the existing DNS infrastructure. The reactive mapping scheme substantially reduces the number of routes exchanged with BGP at the expense of additional first packet latency.

Reactive mapping's impact on IP mobility is also examined. The map-encap scheme of HIDRA enables mobility without triangle routing, at the cost of slightly more per-packet header overhead. Mobility is implemented and tested. It is shown to both work correctly and experience only a slight decrease in performance due to the reduction in per-packet goodput.

References

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

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

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

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

[5] B. Clevenger, D. Nelson, and J. M. Bellardo. Hidra: Hierarchical inter-domain routing architecture. In *ACIT-ICT*, 2010.

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

[7] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek. Measuring the effects of internet path faults on reactive routing. In *Proc. of SIGMETRICS 2003*.

[8] A. Feldmann, L. Cittadini, W. Mühlbauer, R. Bush, and O. Maennel. Hair: hierarchical architecture for internet routing. In *Proc. of ReArch 2009*.

[9] D. Krioukov, K. Fall, and A. Brady. On compact routing for the internet. 2007.

[10] D. Krioukov, K. Fall, and X. Yang. Compact routing on internet-like graphs. In *INFOCOM 2004*, volume 1, 2004.

[11] P. Mockapetris and K. J. Dunlap. Development of the domain name system. In *Proc. of SIGCOMM 1988*.

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

[13] RFC 3344 (rfc3344) - IP mobility support for IPv4. http://www.faqs.org/rfcs/rfc3344.html.

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

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

[16] 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 *Proc. of SIGCOMM 2005*.

[17] The TCP/IP guide - mobile IP efficiency issues. http://www.tcpipguide.com/free/t_MobileIPEfficiencyIssues.htm.

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

[19] P. F. Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. *Proc. of SIGCOMM 1988*.

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

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