NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

APPLIED CYBER OPERATIONS CAPSTONE PROJECT REPORT

HIGH-FREQUENCY MAPPING OF THE IPV6 INTERNET USING YARRP

by

Eric W. Gaston

March 2017

Capstone Co-Advisors: Robert Beverly
David Plonka

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2017	Master's Thesis 03-30-2016 to 03-30-2017

4. TITLE AND SUBTITLE

HIGH-FREQUENCY MAPPING OF THE IPV6 INTERNET USING YARRP

5. FUNDING NUMBERS

RCKHX

6. AUTHOR(S)

Eric W. Gaston

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

LTS
8080 Greenmead Dr, College Park, MD, 20740

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Both the number of hosts using Internet Protocol version 6 (IPv6), and the volume of IPv6 traffic, has increased exponentially since 2012. With this adoption, the IPv6 routed infrastructure becomes an increasingly important component of global critical infrastructure and network policy. Unfortunately, the tools and techniques used to perform active network topology discovery were designed for Internet Protocol version 4 (IPv4), leading to a potentially opaque view of the IPv6 Internet. In this thesis, we extend nascent work on stateless high-speed IPv4 active topology probing to develop a new IPv6 traceroute method "Yelling At Random Routers Progressively version 6" (Yarrp6). Yarrp6 randomly permutes the set of IPv6 targets and hop counts to distribute load, thereby helping to avoid IPv6 response rate limiting. Further, we encode state in the IPv6 payload to permit Yarrp6 to both match responses with probes and use different probe transport protocols. Via active experimentation on the public IPv6 Internet, we compare the results obtained from Yarrp6 against the current state-of-the-art IPv6 topology mapping tool. We show that Yarrp6 can discover topology at more than an order of magnitude faster than previously possible. Finally, we conduct a study of the effect of transport layer protocol on forward Internet Protocol (IP) path inference to determine what protocol is best used for active IPv6 topology discovery.

14. SUBJECT TERMS

IPv6, active topology mapping

15. NUMBER OF PAGES 71

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

HIGH-FREQUENCY MAPPING OF THE IPV6 INTERNET USING YARRP

Eric W. Gaston
Information Systems Technician First Class, United States Navy
B.S., Hawaii Pacific University, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED CYBER OPERATIONS

from the

NAVAL POSTGRADUATE SCHOOL
March 2017

Approved by: Robert Beverly
Capstone Project Co-Advisor

David Plonka
Project Co-Advisor, Akamai Technologies

Cynthia Irvine
Chair, Cyber Academic Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Both the number of hosts using Internet Protocol version 6 (IPv6), and the volume of IPv6 traffic, has increased exponentially since 2012. With this adoption, the IPv6 routed infrastructure becomes an increasingly important component of global critical infrastructure and network policy. Unfortunately, the tools and techniques used to perform active network topology discovery were designed for Internet Protocol version 4 (IPv4), leading to a potentially opaque view of the IPv6 Internet. In this thesis, we extend nascent work on stateless high-speed IPv4 active topology probing to develop a new IPv6 traceroute method "Yelling At Random Routers Progressively version 6" (Yarrp6). Yarrp6 randomly permutes the set of IPv6 targets and hop counts to distribute load, thereby helping to avoid IPv6 response rate limiting. Further, we encode state in the IPv6 payload to permit Yarrp6 to both match responses with probes and use different probe transport protocols. Via active experimentation on the public IPv6 Internet, we compare the results obtained from Yarrp6 against the current state-of-the-art IPv6 topology mapping tool. We show that Yarrp6 can discover topology at more than an order of magnitude faster than previously possible. Finally, we conduct a study of the effect of transport layer protocol on forward Internet Protocol (IP) path inference to determine what protocol is best used for active IPv6 topology discovery.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ACL Access Control List

ARIN American Registry for Internet Numbers

Ark Archipelago

AS Autonomous System

BGP Border Gateway Protocol

CAIDA Center for Applied Internet Data Analysis

CalREN California Research and Education Network

CDF Cumulative Distribution Function

CDN Content Delivery Network

CIDR Classless Inter-Domain Routing

DNS Domain Name System

HMAC Hashed Message Authentication Code

HTTP Hypertext Transfer Protocol

IANA Internet Assigned Numbers Authority

ICMP Internet Control Message Protocol

ICMPv6 Internet Control Message Protocol Version 6

IID Interface Identifier

IP Internet Protocol

IPv4 Internet Protocol Version 4

IPv6 Internet Protocol Version 6

ISP Internet Service Provider

MAC Media Access Control

MRA Multi-Resolution Aggregate

MTU Maximum Transmission Unit

NAT Network Address Translation

NPS The Naval Postgraduate School

NSA National Security Agency

PPS Packets-Per-Second

RFC Request for Comments

RIR Regional Internet Registry

RSI Recursive Subnet Inference

RSI6 IPv6 RSI

RTT Round-Trip Time

TCP Transmission Control Protocol

TOS Type of Service

TTL Time To Live

UDP User Datagram Protocol

VP Vantage Point

Yarrp Yelling at Random Routers Progressively

Yarrp6 Yelling at Random Routers Progressively Version 6

Acknowledgments

First, I would like to thank Robert Beverly for his guidance and encouragement throughout this process. Without his assistance, I would still be trying to figure out where to start on writing my thesis. I thank him for providing me the chance to speak at two Internet measurement conferences, the introduction to the big names in the Internet measurement community, and for allowing me the opportunity to work with him on his amazing research. I have learned so much from him in my short time here at NPS.

Second, I would like to thank David Plonka for providing his knowledge and expertise to make this thesis possible. Without his insight and critique, this project would not have been as successful as it was.

Third, I would like to thank the Center for Applied Internet Data Analysis (CAIDA) and Young Hyun for his assistance in gaining access to and allowing the use of the infrastructure to conduct our scans and for the insight he provided into the current active topology mapping being conducted by CAIDA.

Finally, I would like to thank my wife, Katelyne, and my three boys, William, Christopher, and Matthew, for their love and support while I worked and studied for my degree after coming off an arduous sea tour in Japan. Without them none of this would have been possible.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

There are currently two versions of the Internet Protocol (IP) used by hosts to communicate over a network: the original Internet Protocol Version 4 (IPv4) and its successor, Internet Protocol Version 6 (IPv6). While IPv6 is not a new protocol (the specification for IPv6 was published in 1998 with Request for Comments (RFC) 2460), it has seen relatively slow adoption until around 2012. IPv4 uses 32-bit addresses, frequently represented in human-readable form as "dotted decimal" notation (e.g., 192.168.1.1). The 32-bit addresses used by IPv4 allow for a total of 2^{32} ($\sim 4.2 \times 10^9$) possible unique addresses. Despite this seemingly large number of addresses, the growth of the Internet has lead to IPv4 address exhaustion and the need for the larger address space afforded by IPv6.

Allocation of IP addresses is handled by the Internet Assigned Numbers Authority (IANA), which allocates contiguous blocks of IP addresses to each of the five Regional Internet Registries (RIRs). The RIRs then use these allocated address blocks to sub-allocate IP addresses to network service providers, and ultimately, for use by end users. The American Registry for Internet Numbers (ARIN), the RIR for the U.S. and Canada, depleted its allocated IPv4 address space on September 24, 2015 [1]. Similarly, the four remaining RIRs have all started sub-allocating IPv4 addresses from their last allocated block of addresses [2]–[5].

A long-term solution to IPv4 exhaustion is to transition to native IPv6. One of the primary design goals of IPv6 was expanding the address space to a point where address exhaustion would no longer be an issue. IPv6 thus uses 128-bit addresses, typically written in human-readable form as eight groups of 16 bits in hexadecimal (e.g., 2607:f8b0:400f:803::200e) [6]. The 128-bit addresses used by IPv6 allow for a total of 2^{128} ($\sim 3.4 \times 10^{38}$) possible unique addresses.

Until recently, IPv6 has seen relatively slow adoption. The reason for this slow adoption may be due to the difficulties that come with the transition from IPv4 to IPv6. Czyz et al. look at adoption of IPv6 using six metrics divided into two classes [7]. The first four of these metrics are classified as "prerequisite functions" and are functions that must be met for communication to take place over the IPv6 Internet. The final two metrics are classified as

"operational characteristics" and are only evident once nodes start exchanging packets. The four prerequisite functions are Addressing, Naming, Routing, and End-to-End Reachability. While all four prerequisites must be functioning to allow communication over the IPv6 Internet, the two which present the greatest challenge to Internet Service Providers (ISPs) and web content providers are routing and end-to-end reachability. This is due to the fact that IPv6 is not backwards compatible with IPv4. While transitioning to IPv6, providers must maintain a complex network configuration to allow access to content over both IPv6 and IPv4.

The exhaustion of IPv4 addresses with the continued growth in the number of hosts has lead to a resurgence in the transition from IPv4 to IPv6. On June 6, 2012, major ISPs and web content providers such as AT&T, Comcast, Google, and Facebook permanently enabled IPv6 in what was called the "World IPv6 Launch." Since this event, there has been a 500% increase in the volume of Global IPv6 traffic [8]. The relatively recent widespread use of IPv6 has generated considerable recent research into its operation and usage [7], [9]–[14]. However, the active measurement tools and techniques used to characterize IPv4 traffic and networks are not always immediately applicable to IPv6. In this thesis, we consider the specific problem of active IPv6 network topology mapping.

Network topology mapping is the mapping of the way a network is connected (i.e., the interconnection of the routers that make up the network). Topology mapping can be conducted through either passive or active means. In passive topology mapping, inferences are made about network connections based on data-plane traffic observed at specific points such as web servers or routers. In passive topology mapping there is no attempt to send probes to elicit this information. In contrast, active topology mapping is the inference of network connections based on active probes sent with the sole purpose of eliciting a response from a network device or host. Active topology mapping gives greater control to the researcher to obtain the data which they feel is most important to their study. While active topology mapping gives a researcher this control, the extra network traffic it generates can cause overloading of the bandwidth or processing power for routers in the path of the probes and is also susceptible to filtering of packets which can effect the overall results. Active topology mapping can be used for mapping hosts, Autonomous Systems (ASs), routers, interfaces, and links. In this study we are primarily interested in mapping router interfaces and determining links between these interfaces.

Active topology mapping of the IPv6 Internet presents challenges that have not been seen with active topology mapping of the IPv4 Internet. The vast address space, the requirement to implement rate-limiting of Internet Control Message Protocol Version 6 (ICMPv6) Error Messages [15] more aggressively than with IPv4, and different network configurations present new challenges that the Internet measurement community must determine how to best handle. While there has been a great deal of research on mapping the IPv4 Internet [16]–[25], there has been relatively little work on IPv6 mapping. Many of the current techniques used for IPv4 do not scale well to the vast address space of IPv6. Most tools developed for active topology mapping of the IPv4 Internet ascribe to the slow and steady method, where probes are sent relatively slowly with the tool waiting for a response for the probe or a timeout before sending another probe. Given the much larger address space that IPv6 represents, this method does not scale well to the IPv6 Internet and makes large scale mapping infeasible. New techniques must be developed to deal with this much larger address space in order to make large scale scanning possible. Prior IPv6 specific research has concentrated on methods of reducing the search space by making inferences into what portions of this address space contain active hosts [11]–[13]. In this study, we concentrate on the speed at which probing is conducted. We introduce a new method to perform active topology mapping faster than the current state-of-the-art tools. Our method is complementary to previous studies into discovering the active portions of the IPv6 Internet.

1.1 Motivation

Since the World IPv6 Launch in 2012, IPv6 traffic has doubled every year. Figure 1.1 shows the percentage of users that access Google over IPv6 . This trend shows that IPv6 adoption is increasing and suggests it will continue to increase as more and more ISPs and Web content providers support IPv6.

With this increased adoption and use comes the need and desire to study and understand the IPv6 Internet. Network and security researchers rely on topological maps of the logical Internet to address problems ranging from critical infrastructure protection to policy development. These maps of the logical Internet are important for making informed decisions in the public and private sectors in, at least, the following ways:

- Researchers use them in determining peering on the Internet [20].

- Large Content Delivery Networks (CDNs) use them to determine how they can better deliver data to customers [23].
- Governments use them to determine the effects of outages to critical infrastructure [26].
- Researchers also use them to understand logical network connections to better determine how traffic flows through a network [21].

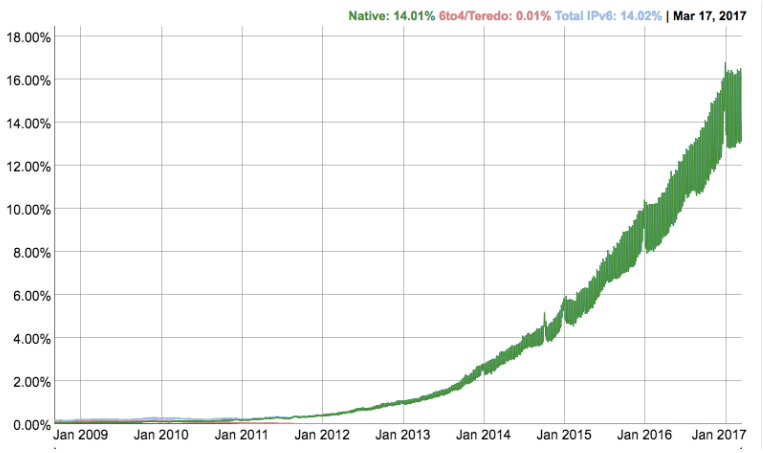

Figure 1.1. Percentage of Users Accessing Google over IPv6 (March 2017). Source: [27].

Active topology mapping is critical to producing accurate topological maps of the logical Internet, but state-of-the-art tools have not been able scale with the size of the IPv6 Internet. To address the shortcomings of these state-of-the-art tools, we develop a new tool called Yelling at Random Routers Progressively Version 6 (Yarrp6), which is a version of Yelling at Random Routers Progressively (Yarrp) [16] which supports IPv6. Yarrp implements a new stateless traceroute technique to enable high-speed active topology mapping which has been demonstrated to do 100K Packets-Per-Second (PPS) [16]. We hope to leverage this capability to enable scanning of a large protion of the IPv6 Internet. While scanning of the entire IPv4 address space has become common place, the sheer size of the IPv6 Internet makes the possibility of scanning the entire address space infeasible. Despite the fact that scanning the entire IPv6 address space may be infeasible, using previous research techniques that attempt to discover the active portion of the IPv6 Internet can facilitate active scanning of the IPv6 Internet by providing candidate addresses to probe [11], [12].

Additionally, while there has been research conducted on what is the best transport protocol to use for active probing of the IPv4 Internet [18], we have been unable to find equivalent research for the IPv6 Internet. Knowing which transport protocol gives the best results is essential to ensuring accurate and valid results for topology mapping. The speed at which Yarrp6 is able to complete scans enables us to conduct a study of which transport protocol returns the best results by allowing us to run scans in rapid succession using different transport protocols.

1.2 Research Questions

The focus of this thesis is the extension of Yarrp to support IPv6, allowing for high-speed and rapid scanning of the IPv6 Internet. In extending Yarrp to cover IPv6, we hope to leverage high speed probing to enable scanning of a large portion of the IPv6 Internet. We also wish to determine which transport protocol (Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or ICMPv6) returns the best results in terms of infrastructure discovered. Determining which transport protocol is best to use for active scans is imperative for future research in active topology mapping of the IPv6 Internet.

We begin by using IPv6 scan data from the Center for Applied Internet Data Analysis (CAIDA) Archipelago (Ark) infrastructure [28] as a basis for comparison against results from Yarrp6. Using the same target list used for the March 4, 2017, CAIDA scans as an input for Yarrp6, we compare topological recall. Topological recall is the fraction of routers and links discovered versus the actual number of routers and links that exist. We also do a comparison of the two scans' recall as a function of time to determine how much faster Yarrp6 scans. We then use the same target list to conduct scans using ICMPv6, UDP, and TCP based probes and do a comparison of the recall as a function of the transport protocol.

In our research we hope to answer the following questions:

- How should Yarrp be modified to support IPv6 while maintaining high probe rates?
- How do the results obtained using Yarrp for IPv6 compare with prior topology studies [10], [13], [28] on IPv6?
- Which IPv6 probe transport protocol returns the highest infrastructure recall?

1.3 Summary of Contributions

In this thesis we develop new methods of active topology probing on the IPv6 Internet which improve on the speed of state-of-the-art methods. This research makes the following contributions:

1. Development of Yarrp6 (i.e., Yarrp with IPv6 support) to allow for rapid mapping of the IPv6 Internet.
2. Analysis of Yarrp6 recall compared to CAIDA as a function of time.
3. Analysis of Yarrp6 recall as a function of transport protocol (e.g., TCP, UDP, ICMPv6).

1.4 Thesis Structure

The remaining chapters of this thesis are organized as follows:

- Chapter 2 introduces IPv6 and associated protocols, tools and techniques for IPv6 mapping, and prior research on IPv6 topology measurement.
- Chapter 3 discusses the data set used by this study, our methodology in comparing Yarrp6's results, and our methodology for evaluating the performance of probe transport protocols.
- Chapter 4 provides our results and analysis of Yarrp6's recall as a function of time as well as a function of transport protocol.
- Chapter 5 details our research conclusions as well as our recommendations for future work.

CHAPTER 2:
Background

As discussed in Chapter 1, we have reached a point of address exhaustion for IPv4. One long-term solution for this is the adoption of IPv6. Czyz et al. examine IPv6 adoption using 12 metrics applied to 10 global datasets to examine the state of IPv6 [7]. They show that the number of IPv6 DNS records, queries for IPv6 DNS records, amount of traffic being carried over IPv6, the number of ASs that support IPv6, the number of IPv6 prefixes being globally advertised, and the number of IPv6 enabled clients and servers, while still relatively small compared to IPv4, have all grown at an exponential rate since 2012 and will only continue to grow. They conclude that IPv6 is now being used as a real production protocol.

2.1 IPv6

IPv6 was developed in response to the rapid expansion of the Internet and the exhaustion of IPv4 addresses. The address space of the IPv6 Internet is vast compared to that of the IPv4 Internet, using 128 bits to represent an address of an interface on the Internet compared to the 32 bits used by IPv4. This allows for the use of 2^{128}, or $\sim 3.4 \times 10^{38}$, unique addresses. This increased address space gives IPv6 the ability to easily expand with the Internet well into the future.

For human-readable presentation, IPv6 addresses are broken into eight 16-bit groups written in the form x:x:x:x:x:x:x:x. Each "x" is a four hexadecimal digit representation of the corresponding 16-bit portion of an address with each hexidecimal digit representing one nyble (4 bits) of the address(e.g., 2607:f8b0:400f:0803:0000:0000:0000:200e.) Additionally, two shorthand notations are used to reduce the amount of space required to display an address: (i) removal of all leading zeros within a group; and (ii) a variable length run of zeros can be replaced by the double colon '::' once in an address [29] (e.g., the above address can be shortened to 2607:f8b0:400f:803::200e.)

One of the main considerations given in the development of IPv6 was the reduction in processing and bandwidth cost of the IPv6 header [30]. Many of the fields in an IPv4 header were dropped in an effort to achieve this. Figure 2.1 shows the fields of the IPv6 header and a brief description of each field.

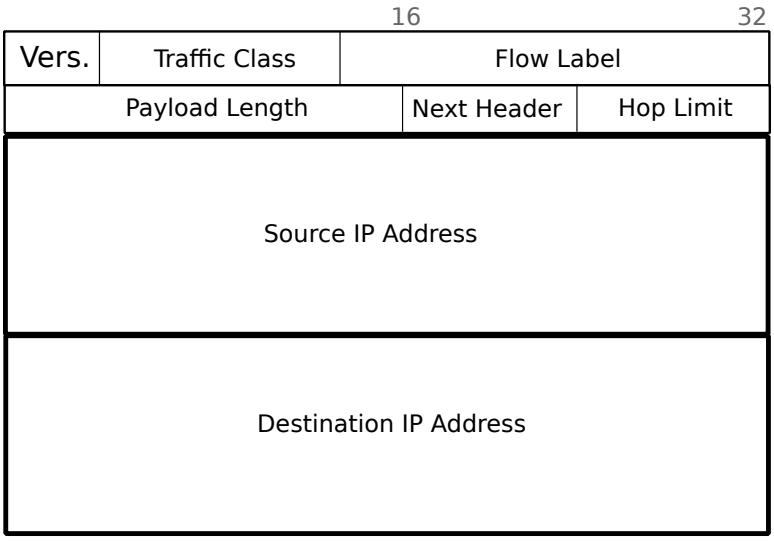

Figure 2.1. IPv6 Header.

- Version (4 bits) – The version of IP (always 6 in IPv6 packets)
- Traffic Class (8 bits) – Can be used to distinguish different classes or priorities of packets
- Flow Label (20 bits) – Can be used to label a sequence of packets for which special handling is desired
- Payload Length (16 bits) – The length of the packet after the IPv6 header
- Next Header (8 bits) – Identifies the type of the next header immediately following (e.g., ICMPv6)
- Hop Limit (8 bits) – The maximum number of hops the packet can be forwarded
- Source IP Address (128 bits) – The IPv6 address of the packet's sender
- Destination IP Address (128 bits) – The IPv6 address of the packet's destination

2.1.1 ICMPv6

RFC 4443 states that, "ICMPv6 is used by IPv6 nodes to report errors encountered in processing packets, and to perform other internet-layer functions such as diagnostics" [15]. ICMPv6 has two key features that are important for the purpose of this study. The first feature relates to the Time Exceeded error message and how much of the packet invoking the Time Exceeded error message is returned when an error occurs. A Time Exceeded error message is sent to a given packet's source address by a router if it receives a packet where

the hop limit is zero, or if the router decrements the hop limit to zero (each router in the path decrements the hop limit by one before forwarding the packet.) In ICMPv6, a Time Exceeded error message returns "as much of the invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 Maximum Transmission Unit (MTU)" [15]. The second feature which is important to this research is rate-limiting of ICMPv6 error messages. RFC 4443 states that, "in order to limit the bandwidth and forwarding cost incurred by originating ICMPv6 error messages, an IPv6 node MUST limit the rate of ICMPv6 error messages it originates." It later states that, "rate-limiting mechanisms that cannot cope with bursty traffic (e.g., traceroute) are not recommended" [15]. To the best of our knowledge, the exact implementation of IPv6 rate-limiting varies among vendors and is not public knowledge. As we will show in Section 4.1.1, rate-limiting in practice is a significant contributor to missing traceroute hops, especially when performed at the high rates we envision.

2.2 Traceroute

Traceroute is a tool used to trace the forward data-plane path an IP packet takes from the current host to a selected destination [31]. It is used by system administrators for diagnosing network issues as well as researchers who wish to actively scan the Internet to infer properties such as topology. To determine the path taken, traditional traceroute sends a series of IP packets to a destination, incrementing the Time To Live (TTL), in the case of IPv4, or the Hop Limit, in the case of IPv6, by one in each subsequent IP packet. This continues until the destination is reached or a predetermined number of packets are sent with no response. A response may not be received for several reasons: (i) a node being configured to not send ICMPv6 Time Exceeded in transit messages; (ii) the probe or response packet being dropped by another node in the path; (iii) the probe or response packet being lost due to network errors; or (iv) a node dropping the probe packet due to rate-limiting. As discussed in section 2.1.1, when a router receives an IP packet with a TTL or Hop Limit of zero, or the router decrements the TTL or Hop Limit to zero, instead of forwarding the packet, it generates a Internet Control Message Protocol (ICMP) Time Exceeded error message and sends it to the packet's originating source address. One limitation of forward IP path inference using traceroute is how nodes determine what source address to use when originating ICMP error messages. RFC 4443 states that, when a router originates an ICMP

message, it MUST use an address belonging to the node, but leaves the choice of which address to use up to the node [15]. In a node that has multiple IP addresses, this means we could get an ICMP error message from a source address that is not in the direct path to the destination, resulting in incorrect path inferences. Traceroute reconstructs the path to the destination by recording the source addresses of these ICMP Time Exceeded error messages. The ICMP Time Exceeded error messages are matched to the probe that induced the message by comparing the destination port header field in UDP probes, the ID/Sequence number header fields in ICMP probes, and the IPID header field in TCP probes contained in the quote of the ICMP time exceeded message. The quote is the portion of the ICMP Time Exceeded error message that contains the headers of the packet that induced the message. By matching these quotes with the corresponding probe, traceroute is able to determine the addresses for each hop the IP packet takes to the destination.

Traceroute probes can use any transport layer protocol that IP supports. It can use TCP, UDP, or ICMP packets and each has distinct advantages and disadvantages. In [18], Luckie et al. look at the result of using different transport layer protocols on forward IPv4 path inference and lay much of the groundwork for the selection of different probing methods to benefit different research goals. They find significant differences in observed topology based on the transport layer protocol used for the probes. They concluded that ICMP probes tended to reach more destinations, reaching their destination 72.2% of the time, and yield better results for AS path inferences, discovering 92.3% of all AS links. In comparison UDP reaches fewer destinations, reaching only 42.7% of their destinations, but performs better for IP link inferences, discovering 89.1% of all IP links. To our knowledge no equivalent study has been published for IPv6.

While traceroute is essential to active topology mapping, it is susceptible to error "in the presence of routers that employ per-flow load balancing on packet header fields" [19]. Per-flow load-balancing is the selection of a path by a router based on a packet's flow identifier where a flow identifier is defined as the five-tuple consisting of the source IP address, destination IP address, source port number, destination port number, and the protocol (i.e., TCP or UDP). When using traditional traceroute, errors induced by per-flow load balancing routers can cause incorrect route inference which may result in erroneous results such as missing links, false links, routing loops, and routing diamonds.

To resolve these issues, a new method of traceroute was proposed by Agustin et al. known as Paris traceroute [19]. In Paris traceroute, certain probe packet header fields are maintained throughout the entire probe to a single destination to ensure that per-flow load balancing routers within the path allow the packets to follow the same path on each successive hop. The header fields which must be maintained for the duration of a single trace to a destination are the five-tuple discussed above as well as the Type of Service (TOS) field in the IPv4 header, the Traffic Class and Flow Label in the IPv6 header, and the Code and Checksum in the ICMP and ICMPv6 headers.

The current de-facto standard tool used for active Internet topology mapping of both the IPv4 and IPv6 Internet is *scamper* which is developed by CAIDA [32], [33] and is used by many of the major active Internet topology studies [10], [17], [28]. Scamper uses the Paris traceroute method and has the capability to conduct scans using ICMP, TCP, and UDP. Scamper, like traditional traceroute, maintains state on each probe sent to match a response with its corresponding probe. Scamper outputs results in a binary format known as *warts*. Warts output contains a substantial amount of metadata about the individual scan as well as a substantial amount of detail about responses received.

2.3 Yarrp

Yarrp is a new traceroute technique designed for high-rate, Internet-scale probing [16], [34]. As discussed in Section 2.2, In classic traceroute, we send each probe *toward a single IP destination* in sequence (i.e., TTLs 1, 2, 3, 4, and so on.) When initially developed, traceroute sent only one probe at a time, waiting for a response to this probe or a timeout before sending the next probe. In order to reduce the amount of time required to complete a trace, modern implementations now send multiple probes at one time. The requirement to maintain state on these probes limits the number of simultaneous probes that can be sent at one time. Upon receiving a response to a probe, the response is matched to the corresponding probe packet and the TTL of the originating probe packet and the source address of the response packet is recorded. This requirement to maintain the state on each probe can significantly slow down scans due to unresponsive or non-existent hosts and routers that do not return ICMP Time Exceeded error messages. Yarrp achieves 100K PPS by separating the scanning and path reconstruction tasks. This separation allows Yarrp to remain stateless, and only concern itself with sending packets. To achieve this statelessness

but still allow for path reconstruction, Yarrp encodes data about each probe within the header fields of the probe packet. Figure 2.2 displays the IPv4 and TCP headers with the Yarrp encoded fields highlighted. Upon receiving the response to the probe, Yarrp records the encoded information from the ICMP message quotation, which contains the first 28 bytes of the packet that induced the expiration, into a plain text file in the order they are received. The file is then processed offline at a later time and paths are reconstructed and written to a file in the standard warts format.

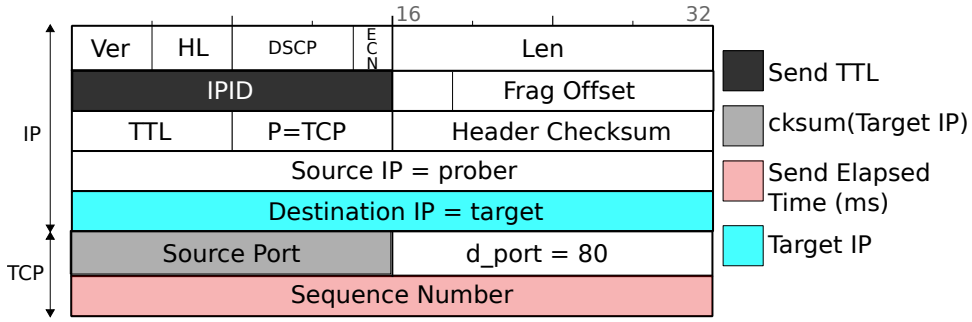

Figure 2.2. Yarrp Encoding of IPv4 and TCP Header. Source: [16].

In order to prevent the overloading of any one path, Yarrp randomizes the probing order over the domain of IPs and TTLs (D = IPs × TTLs). For example, Yarrp will send a probe to IP address A with a TTL = 10, then to address B with a TTL = 3, then address C with a TTL = 6, and so on until the entire space of TTLs for each target IP is covered. In order to achieve this randomization, Yarrp uses RC5 [35], a 32-bit symmetric block cipher, to permute over the domain of IPs and TTLs. Yarrp encrypts the sequence $i = 0, 1, ..., |D| - 1$ with key k using the results of $C_i = RC5_k(i)$ to determine what target IP address ($C_i[0:23] * 2^8 + (C_i[0] + C_i[1] + C_i[2])\%256$) and TTL ($C_i[24:31]$) to use for the probe. By doing this, Yarrp ensures that same random address for each /24 is probed for each TTL.

2.4 Actively Mapping the IPv6 Internet

We have found two examples of previous or ongoing attempts to actively map the IPv6 Internet. The first example is CAIDA's ongoing effort since December 12, 2008, in which they probe one random address and the ::1 of all /48 or shorter prefixes announced in

the global IPv6 Border Gateway Protocol (BGP) routing tables once every 48 hours from multiple globally distributed monitors within the Ark Infrastructure [28]. The overall goal of this study is the discovery of IPv6 topology and is not concerned with discovering responsive IPv6 destinations. The scans utilize scamper to perform ICMPv6 Paris traceroute probes of the ::1 and one random address from each announced /48 or shorter. As of March 4, 2017, this amounts to a total of 39,196 prefixes being scanned by 46 monitors from around the globe. These scans are designed in an attempt to balance the time required for scans with the expected coverage.

The second example is Rohrer's et al. scan conducted over the period of November 13, 2014, to March 1, 2015. In this study, probing was performed to one address in each /48 in all /32s advertised in the global IPv6 BGP tables [10]. The design of their experiment is based on recommendations for allocations provided in RFC 3177 [36]. While RFC 3177 was obsoleted by RFC 6177 [37], they conclude that because the recommendations from RFC 3177 were used by providers for a decade, scanning all /48$'s$ gives a reasonable compromise between completeness and probing time. For their study they scanned 406,388,736 /48 prefixes over this 4 month period.

These studies have fallen short in either the timeliness of complete scans or the number of prefixes scanned. In this study, we extend Yarrp, which previously only supported IPv4, to map IPv6 routers. The speed at which Yarrp is able to scan will reduce the amount of time required to complete scans from months to days, while still allowing for the scanning of a significant portion of the IPv6 address space.

2.5 Reducing the Target Space

The sheer size of the IPv6 Internet makes scanning the entire address space impossible. Even at the 100K PPS rates Yarrp can achieve, it would take $\sim 3.45 \times 10^{27}$ years to probe each address. This does not mean that active topology mapping is not possible or should not be done, but we must develop methods to reduce the target space to a more realistic size for active scanning. We could use simplistic methods to reduce the size (i.e., only scanning blocks of the IPv6 Internet that have been allocated by IANA or sub-allocated by the RIRs), but this still leaves a very large target space which is not much more feasible to scan then the entire IPv6 Internet. This forces us to take new approaches to reducing the target space

for our probes. In the following paragraphs, we look at four studies that attempt to reduce this target space.

In the first study, LaFever develops Recursive Subnet Inference (RSI) for IPv6 (IPv6 RSI (RSI6)) which is a method to determine if subnetting is present within an advertised prefix [14]. To determine if subnetting is present within the prefix, RSI6 sends a traceroute probe to two addresses within the prefix. Generally, addresses located at $\frac{1}{4}$ and $\frac{3}{4}$ portion of the prefix are probed and if the results of the two probes match, he infers that subnetting is not present. However, if the two probes' routes differ, he infers that subnetting is present and the prefix is further divided until all subnets are found. He also proposes that, due to common subnetting practices on IPv6, probing the ::1 and $\frac{1}{2}$ portion provide additional topology with little to no extra cost. This method allows us to create a list of candidate addresses for our active probes by determining the existence of subnetting within a prefix. The existence of subnetting would imply that this prefix is actively being used and would provide us with candidate subnets to probe using Yarrp6.

In the second study, Gray develops heuristics for generating candidate addresses for active probing [13]. To create his method for candidate generation, he uses historical IPv6 topology probing results to determine the most common lower 64-bit values present. He then obtains a list of all globally advertised BGP prefixes and generate a candidate list by combining each advertised prefix, regardless of size, with one of the most common lower 64-bit values. The list generated by this method can then be used for active probing with Yarrp6.

In the third study, Plonka et al. develop methods to classify IPv6 addresses spatially and temporally using log data from a large CDN [12]. Their temporal classification method is designed to determine the lifetime or stability of an IPv6 address. By examining active client addresses within the log data over a period of one year, they determine the stability of a client IPv6 address by looking for common IPv6 addresses within each log. For example, if an address is observed as active in a log over a 1 year time period, then they classify this as a stable address and assign it to a stability class. Stability classes are named according to length of time (i.e., the previous example would be classified as 1 year stable.) Their spatial classification method is designed to determine the proximity of addresses and prefixes as well as to give a visualization of the containing address block. They introduce two metrics for spatial classification, Multi-Resolution Aggregate (MRA) Count Ratios and

Prefix Density. Both MRA plots and Prefix Density are used to expose structure within prefixes and can be used to determine dense, or highly active portions of a prefix. By determining portions of the IPv6 address space that have a high density of active addresses, we can generate list of candidates for active probing with Yarrp6.

In the fourth study, Foremski et al. develop a method to discover IPv6 address structure based on analysis of a subset of known active IPv6 addresses obtained through active and passive means [11]. To infer structure in IPv6 addresses, they first plot the entropy of each address nyble across the entire dataset and then each nyble is segmented into groups with similar entropy. Within each segment the values are then sorted by probability and a heatmap is generated. This method can be used to probabilistically model IPv6 addresses based on information theory and machine learning for generation of candidate address lists for active probing with Yarrp6.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology

As discussed in Section 2.4, we have found two studies that conducted active topology mapping of the IPv6 Internet. These studies have fallen short in either the amount of time required to complete scans or in the number of prefixes that they scanned. The main thrust of this study is the extension of Yarrp to support IPv6 topology discovery. We call this new tool "Yarrp6." Our hope is that Yarrp6 will serve as an initial step toward the ability for active topology measurement systems to cover a significant number of IPv6 prefixes within a reasonable amount of time.

This chapter discusses the key design considerations, unique challenges, and development of Yarrp6. After initial development of Yarrp6 we validate our tool by comparing its results with the results of CAIDA's IPv6 dataset [28] and quantify Yarrp6's speed improvement. We then use Yarrp6 for a study of the effect of transport protocol on forward IP path inference, partially recreating the experiments of Luckie et al. [18] for IPv6.

3.1 Yarrp6

Yarrp is a tool that implements a stateless traceroute technique designed for high-rate, Internet-scale topology mapping. Yarrp6 is an extension of Yarrp's techniques to IPv6 and further allows for the capability to conduct scans using ICMPv6, UDP, and TCP. In extending Yarrp to IPv6 we must address three primary challenges:

- Accommodating per-flow balanced paths in IPv6
- Encoding state in IPv6, where the packet header is different than in IPv4
- Permuting the probing order of IPv6 prefixes and TTLs.

For each packet we ensure that we are following the Paris traceroute methods discussed in Section 2.2. To do this, we ensure that all packets sent by Yarrp6 have the traffic class and flow label set to 0 to ensure that any per-flow load balancing routers along the path do not cause our scans to infer false or missing links. Additionally, there are specific considerations given for header fields of each transport protocol we use, which we discuss in the following sections. In order to control the entire packet header, we must craft raw packets which we will discuss in greater detail in section 3.1.4.

Recall from Section 2.3 that Yarrp achieves its stateless operation by encoding the state within the IP and TCP headers and reconstructing this state offline at a later time based on the recorded ICMP response's quotation. Instead of encoding the information needed to reconstruct the path in the packet headers, as Yarrp does, Yarrp6 includes the necessary state within the packet's data payload. This simplification is a result of the difference in behavior between ICMP and ICMPv6. RFC 4443 states that, when a node sends an ICMPv6 Time Exceeded error message, it should include, "as much of the invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU" [15]. This is a change from ICMP where RFC 792 states that, the ICMP Time Exceeded error message should only include, "the Internet header + 64 bits of the Original Data Datagram" [38]. Figure 3.1 shows the structure used for Yarrp6's payload and below this we provide a description of each field within the payload.

```
struct payload {
    char id[7];
    uint8_t hlim;
    uint32_t diff;
};
```

Figure 3.1. Yarrp6 Payload Structure.

- ID (7 bytes) – Always set to "Yarrp6" (0x59417272703600). Used as a unique identifier that ensures we only examine ICMPv6 packets received as a result of a Yarrp6 probe.[1]
- hlim (1 byte) – Holds the hop limit with which we sent our packet and is used later to match our packet with its corresponding hop.
- diff (4 bytes) – holds the time at which we sent our packet as the time elapsed since the start of our scan in milliseconds. Used to calculate the Round-Trip Time (RTT) of each hop.

[1]The ID field was added after running Yarrp6 on CAIDA's Ark infrastructure for the first time. During these first scans, we were unable to differentiate Yarrp6 probes from scamper probes being run on the same Vantage Point (VP).

Using a structure to define our payload makes referencing our fields easier when decoding the responses to our probes. When we receive responses to our probes we know that our data structure will be included in the response and this allows us to reference these fields by name when decoding. Using a structure to define our payload also makes Yarrp6 easily extensible in the future if the need arises to include more information within the payload. We will discuss two such additions in Section 5.1. These additions are, (i) a version field to differentiate probes from different versions of Yarrp6; and (ii) a checksum of the payload to determine any modifications to our encoded information.

3.1.1 TCP Probes

For TCP-based probes, we implemented the capability to send both TCP SYN as well as TCP ACK packets. TCP SYN packets are the first packets sent as part of the three-way handshake in the establishment of a TCP connection. We send the TCP SYN packets with the destination port set to 80, which is the well known port for Hypertext Transfer Protocol (HTTP) traffic. We choose these parameters due to the fact that TCP SYN packets to port 80 are very common and are less likely to be dropped by firewalls within the path.

TCP ACK packets are sent between two nodes communicating over TCP to acknowledge receipt of packets. For every TCP packet sent, there is a corresponding TCP ACK sent by the receiving side acknowledging receipt to the sender. For TCP ACK packets, we use the well known port 80 as our destination port. These parameters are chosen again, due to the prevalence of TCP ACK packets for communication with web servers. However, a potential problem with TCP ACK packets is that many stateful firewalls will not allow TCP ACK packets that do not belong to an established connection. Therefore, the potential for dropped packets may increase with this method.

We ensure that we follow Paris traceroute methods with our TCP probes in order to reduce anomalies and erroneous results. This dictates that we must use the same source and destination port for each packet within a traceroute to a single host. As discussed above, we use port 80 as the destination port in all packets. For our source port we use port 51234, which is an ephemeral, unreserved port and fits with the expected behavior of a typical TCP connection. This helps ensure that our packets are not dropped by firewalls that filter out TCP connections from well known source ports (i.e., ports less then 1024.) We set our

initial sequence number to 1 and we do not include any TCP options. We set only the SYN flag in TCP SYN probes and only the ACK flag in TCP ACK probes.

3.1.2 UDP Probes

We send all UDP packets with the destination port of 53, which is the well known port for Domain Name System (DNS) traffic. DNS is the protocol which converts a domain name (e.g., google.com) into an IP address. The queries that are used to determine these hostname-to-IP mappings are sent as UDP packets to a resolving DNS server on port 53. Much as we used port 80 in TCP to mimic common traffic, we use port 53 in UDP in order to increase the number of firewalls our packet can traverse and increase our forward IP path discovery.

We ensure that we follow Paris traceroute methods with our UDP probes in order to reduce anomalies and erroneous results. This dictates that we use the same source and destination port for each packet within a traceroute to a single host. As discussed above, we use port 53 as the destination port in all packets. For our source port we again use port 51234, which is an unreserved port and fits with the expected behavior of typical UDP traffic. This ensures that the resulting packets are not dropped by firewalls that filter out UDP traffic from well known source ports (i.e., ports less then 1024.)

3.1.3 ICMPv6 Probes

For ICMPv6 probes, we use ICMPv6 Echo Request packets. One issue that may impact our ICMPv6 Echo Request probes is the fact that many sites follow the security best practice of not allowing hosts to respond to ICMPv6 Echo Request message, blocking outgoing ICMPv6 Time Exceeded error messages, and blocking Echo Reply messages at the firewall [24], [39]. Despite this, ICMPv6 Echo Request messages are still very common and both active topology mapping studies discussed in Section 2.4 used ICMPv6 probes in their study.

We ensure that we follow Paris traceroute methods with our ICMPv6 probes in order to reduce anomalies and erroneous results. This dictates that we use the same code and checksum for each packet within a traceroute to a single host. The code for an ICMPv6 echo request message will always be 0, but keeping the checksum the same presents a

problem in ICMPv6. In calculating the checksum for ICMPv6 we are required to include a pseudo IPv6 Header. In every packet Yarrp6 sends, we change the hop limit within our IPv6 header which will cause our checksum for each packet to differ. While this means that we cannot do Paris traceroute as specified in [19], per-flow load balancing is less prevalent on ICMP packets [25]. We will discuss possible solutions to this problem as future work in Section 5.1

3.1.4 Challenges

Recall from Section 2.3 that in order to prevent the overloading any one path, Yarrp randomizes its probing order by using RC5 to permute over the target domain of IPs and TTLs ($D = IPs \times TTLs$). The permutation library encrypts the sequence $i = 0, 1, ..., |D| - 1$ where $C_i = RC5_k(i)$. The ciphertext returned by the RC5 algorithm is 32 bits, which Yarrp then takes and splits into the most significant 24 bits and the least significant 8 bits. The 24 bits is used as the first three octets of the IPv4 address and the 8 bits is used as the TTL.

Our first challenge in developing Yarrp6 is the fact that the permutation library used for probe randomization in Yarrp makes use of the RC5 cipher which only supports 32-bit block size in the implementation used. While RC5 does support 64-bit and 128-bit block sizes as well as the 32-bit block size, neither of these 3 present a ideal block size for use in IPv6. To understand this, we must look at the target space over which we permute. For example, in both [10] and [28], the authors choose to divide the prefixes they are probing into /48s. Looking at the RC5 block sizes, we see that 32 bits is too small to permute over this space, but 64 bits is to large, causing us to potentially waste 2^{32} (4,294,967,296) calculations as we permute over a target space that is 32 bits larger then we need. The wasted calculations are due to the cycle-walking algorithm that is used to permute over the target space [40]. In order to address the block size issue we change the block cipher used by the permutation library from RC5 to a new lightweight block cipher developed at the National Security Agency (NSA) known as Speck [41]. Speck is designed to be lightweight, fast, and most importantly, supports block sizes of 32, 48, 64, 96, and 128-bits. The number of different block sizes gives a great deal of flexibility. The block size can be changed to the smallest block size needed to achieve permutation of the target space (i.e., if you only need 48 bits for the permutation of your target space you can use the 48-bit block size instead of wasting the extra calculations that would be required if you were using the 64-bit block size.) In the

development of Yarrp6 we use a block size of 32-bits, which was sufficient for permutation of the target space we scanned in this research. Changing the block size of the cipher to support the scanning of larger target spaces only requires the modification of one parameter within the compilation of the library and makes switching between block sizes very simple.

Our second challenge in developing Yarrp6 is that IPv6 does not give the ability to control all header fields in the packet when using raw sockets [42]. Yarrp uses raw sockets in order to control each field of the IPv4 header. IPv6 raw sockets only allows changes to the hop limit within the header which is insufficient for implementing the Paris traceroute method. In order to allow us to control all fields of the headers in our packets, we must use a packet socket instead of a raw socket [43]. While this gives us the ability to control our packet's header fields, it also places the burden of creating the layer 2 header on the developer instead of the operating system. To create the layer 2 header we must know the Media Access Control (MAC) address of our default gateway and the MAC address of the host we are conducting the scans from. We leave the responsibility of supplying the two required MAC addresses to the user. In order to conduct scans the user must supply these MAC addresses as command line options when running Yarrp6. On Linux, the source MAC address can be obtained by using the "ifconfig" command and recording the "HWaddr" for the interface they will be conducting scans from. On Linux, the destination MAC address can be obtained using the "ip neighbor show" command and recording the "lladdr" for the default gateway.

Our third and final challenge in developing Yarrp6 is that rate-limiting of ICMPv6 error messages can impact scan results at the rates at which Yarrp6 scans. As discussed in Section 2.1.1, an IPv6 node, "MUST limit the rate of ICMPv6 error messages it originates" [15]. We postulate that rate limiting will have a significant impact on the responses we receive from routers within 1 or 2 hops of the host we are running scans from. This is due to the tree like structure of the network which implies that the set of router interfaces close to our host (i.e., 1–2 hops) is small compared to the set of router interfaces at hops further along the path. Yarrp has the capability to maintain a list of "neighborhood" interfaces from 1 to a runtime configurable number. We currently have not yet implemented this capability in Yarrp6, but, if analysis shows significant rate limiting within these lower hops, this feature could alleviate the problem. To detect rate limiting we will look at the distribution of missing hops within our scan results.

3.2 Validating Results of Yarrp6

To evaluate Yarrp6, we wish to examine its topological recall as a function of probing speed and probe protocol. Instead of running Yarrp6 in a closed test environment, we chose to evaluate it on the public world-wide Internet in order to obtain real-world performance metrics. However, there is no ground truth of the IPv6 Internet's topology, and it is not feasible to probe every IPv6 address. Therefore, we must (i) choose prefixes (and addresses within those prefixes) to probe; and (ii) perform relative comparisons against results obtained via other techniques.

We use the March 4, 2017, scans which were started at 7:08 PM from the san-us (San Diego, CA) Ark VP [28] for the validation of our results. These scans were conducted using scamper [32] with ICMPv6 Paris traceroute probes to 75,594 IPv6 targets. In order to validate the results of our scans, CAIDA granted us access to probe from the san-us VP. We start by extracting the target list and then probe from the same VP with Yarrp6 to the same target list. By ensuring the same VP and target list is used, we would expect the paths to be largely congruent, allowing us to do a valid comparison of topological recall between the two methods. We conducted our scans on March 6, 2017, starting at 0845 GMT. We accept that this difference of 37 hours may cause some difference in Yarrp6 scan results when compared to CAIDA's.

In order to preform the validation, we reconstruct the traces offline by converting the yrp file produced by Yarrp6 into a warts binary trace. We must do the reconstruction offline because Yarrp6 maintains no state while conducting scans, it only receives responses from probes it sends and writes them to file instead of pairing them to the corresponding probe that produced the response. We compare our results to CAIDA by using two metrics. The first metric we compare is the degree distribution of Yarrp6's data compared to CAIDA's. Showing that we have a similar degree distribution when scanning from the same vantage point using the same target list suggests that we discover similar infrastructure. Additionally, we preform a comparison of the edit distance between Yarrp6's results and CAIDA's. The edit distance is the number of differences there are between two traces. It can be expressed as the number of insertions, deletions, or substitutions there are in a Yarrp6 trace when compared to CAIDA's trace to the same destination. For the purpose of this study, an insertion is a hop that is present in the Yarrp6 trace and is not present in the corresponding CAIDA trace, a deletion is a hop that is present in the CAIDA trace and is not present in

the corresponding Yarrp6 trace, and a substitution is a hop that is present in both traces but which contains a different address for each.

We are also interested in the rate at which Yarrp6 discovers unique interfaces compared to CAIDA. After initial validation of topological recall with CAIDA's data, we will compare the speed at which we discover unique router interfaces compared to CAIDA as a function of time. To determine this we will plot the number of unique interfaces found as a function of time and calculate number of unique interfaces found per second.

3.3 Determining the Effect of Transport Protocol

In the second part of this study, we are interested in determining the best transport layer protocol to use for active IPv6 topology mapping. As discussed in section 2.2, Luckie et al. studied the effect of transport layer protocols on forward IP path inference for IPv4 [18]. Our hope is to recreate parts of this experiment to obtain a better understanding of transport layer protocol impact on forward IPv6 path inference.

In their study, Luckie et al. define four metrics to evaluate the effect of different transport layer protocols on forward IP path inference [18]:

1. Destinations Reached
2. Complete IP Paths
3. Unique IP Links
4. Unique AS Links

In this study we will only use three of these four metrics in determining the best transport layer protocol to use, leaving the study of unique AS Links to future work.

Destination Reached is a metric of how many of our probes reached their intended target. Luckie et al. consider a destination to be reached if any of the following conditions occur:

- an ICMP port unreachable message is received in response to a UDP or TCP probe.
- a TCP packet is received from the destination in response to a TCP probe.
- an ICMP echo reply packet is received from the destination in response to an ICMP echo request packet.
- an ICMP destination unreachable message is received with a source address matching the destination address probed. [18]

Due to the stateless nature of Yarrp6, we are currently unable to determine if a destination was reached with a TCP probe where the target sends a TCP SYN or TCP RST. While these are valid responses to a TCP probe, they do not contain the quotation with our encoded information like ICMPv6 error messages. This means that, because we cannot determine if a TCP SYN or TCP RST is in response to one of our probes, we cannot determine if the destination was reached in these cases. Compounding this problem is the fact that TCP SYN and TCP RST packets are very common among network traffic and are not unique to traceroute traffic meaning we would have to try and find the packet which corresponds to our probe within a myriad of traffic. Reaching a destination gives us a better understanding of forward IP path length and RTT when reconstructing the paths offline. To determine if a destination was reached, we iterate through each hop of all probes and if the target address is seen as a hop in the probe we count the probe as reaching its destination.

Luckie et al. define a complete IP path as any path where, "the destination is reached and there is a response from all intermediate hops" [18]. Complete IP Paths allow for more accurate inferences because the entire path the probes took enroute to the target is recorded. To determine complete paths, we iterate through each hop in a probe checking if we have any missing hops until we find our target's address. If we are missing no hops, and have reached the destination, we count the probe as a complete path.

Unique IP Links is the measure of how many interface level connections we find within a trace. An IP link is any pair of adjacent IP hops seen within a probe. IP links are important to topology measurement because they suggest router adjacencies and allows us to make much stronger inferences. To determine the number of unique IP links, we create a graph from the probes where unique hops are added as nodes and any adjacent hops in the probe are added as edges connecting these nodes. We then count the number of edges as the number of unique IP links within the probe.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Results

In chapter 3 we introduced our methods for evaluation of Yarrp6 using current CAIDA data which represents the state-of-the-art. After initial validation, we compare the rate at which Yarrp6 discovers unique interfaces, performing a comparison of our results to CAIDA's results of unique interfaces discovered per second and unique interfaces per probe sent. Finally, we investigate the effect of the transport layer protocol on forward IP path inference, comparing the number of destinations reached, complete IP paths, and unique IP links for probes sent using ICMPv6, TCP SYN, TCP ACK, and UDP.

4.1 Validation of Yarrp6

Validation of active topology requires the scanning of a network where ground truth can be obtained. While comparing our results to a ground truth network would be the ideal form of validation, networks that allow for running ground truth are very hard to find. We leave ground truth validation to future work and instead define validity in this study *relative* to CAIDA's probing.

In order to determine the validity of the results produced by our tool, we must do a comparison of the data our tool gathers against data gathered by a state-of-the-art tool. As discussed in Section 3.2, we choose to validate our tool against the IPv6 scans conducted by CAIDA. We validate our results against CAIDA's results for scans conducted on March 4, 2017, by running our scans using the same target list, from the same vantage point, and nearly on the same day. The day we chose to scan had no significance and was done for experimental convenience. While we do not expect results to differ when comparing different days, we will leave this comparison for future work and concentrate on validating our tool against a single scan from CAIDA. CAIDA's IPv6 scans make use of the scamper tool to conduct ICMPv6 Paris traceroutes of the target destinations [28], [32], [33].

For the remainder of this study we label all graphs key legends for CAIDA scans simply as CAIDA. This was done for efficiency of explanation due to the fact that the results from only one scan were used for the analysis. This label refers to the production IPv6 active topology mapping scans conducted on March 04, 2017, by CAIDA using scamper on the san-us VP.

We initially validate the correctness of Yarrp6 by probing the same destination used in CAIDA's scans. By probing the same destinations used in the CAIDA scans from the same VP, we expect that the paths will be largely congruent allowing for a fair determination of the recall of our method. We first pursue this by creating the resulting interface-level graph from the scans. The interface-level graph has unique interfaces represented as nodes and connections between nodes in consecutive hops represented as edges. we do not perform alias resolution to create the router-level topology as this is not one of the goals of this study. Instead, we are interested in comparing results at the IP address level from the two methods. Figure 4.1 displays the degree distribution of this graph on a log-log scale. While degree distributions are known to miss structural differences when comparing graphs [44], the two graphs represent the same underlying probed network. In particular, since the two graphs are produced by running the tools from the same source to the same destinations, we use the degree distribution plot to provide a first-order comparison.

We can see from Figure 4.1 that the distribution matches closely, providing a level of confidence in the ability of Yarrp6 to accurately recover similar topologies. We do see some areas of difference which we attribute to the more aggressive rate-limiting of ICMPv6. Yarrp6 discovers $\sim$ 0.035% fewer interfaces (51,909 versus 51,927) and discovers $\sim$ 3% more edges (63,422 versus 61,049) than CAIDA's measurements.

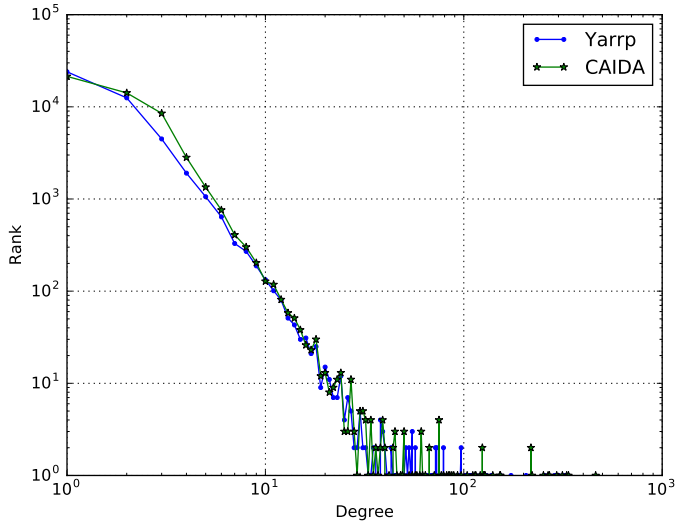

Figure 4.1. Degree Distribution of Yarrp6 versus CAIDA.

In addition to examining the degree distribution, we compare methods by evaluating the per-path edit distance for the trace to each target. By applying the Levenshtein edit distance method to each path, we can evaluate how many deletions, insertions, and substitutions there are in each of the 78,435 paths in the Yarrp6 results when compared to same path in CAIDA's results. In the context of this study, a deletion is a hop that is present within the CAIDA trace which is not present in the Yarrp6 trace, an insertion is a hop that is present within the Yarrp6 trace which is not present in the CAIDA trace, and a substitution is a hop that is present in both traces but contains a different address for each. An edit distance of zero indicates that the discovered paths to a particular target are identical between the two methods – the ideal outcome for our validation. Figure 4.2 provides the Cumulative Distribution Function (CDF) of the overall edit distance, missing hops, substitutions, and deletions in comparing Yarrp6 and CAIDA scans. We see that ~ 52% of the paths have an edit distance of 0, indicating that the paths are the same, while ~ 25% of the paths differ by one hop and ~ 13% differ by two hops.

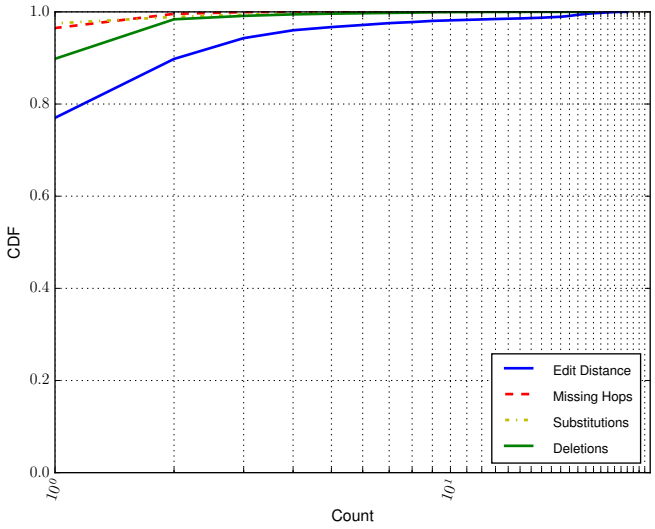

Figure 4.2. Edit Distance of Yarrp6 versus CAIDA.

Because we expect a larger percentage of path comparisons to yield an edit distance of zero, we conduct further analysis by comparing the edit distance between two CAIDA scans, each conducted with scamper. In this fashion, we seek to better isolate the influence of different methods on the edit distance from differences attributable to network. Using the

same Levenshtein edit distance method, we compare CAIDA scans conducted on March 3, 2017, at 7:56 PM and March 4, 2017, at 11:08 AM. When comparing these scans, it is important to note that we can only compare the results of traces to the ::1 of each prefix. This is due to the fact that CAIDA performs a trace to the ::1 and a random address in each prefix. The random addresses probed in each cycle differ and thus we can only compare the results of the traces to the ::1. Figure 4.3 shows the results of this comparison. When comparing results of scans which used the same tool conducted only 15 hours apart, we see that only $\sim$ 46% of the paths have an edit distance of zero, while $\sim$ 43% of the paths differ by one hop and $\sim$ 6% differ by two hops. These results show that only a small portion of the paths scanned are the same.

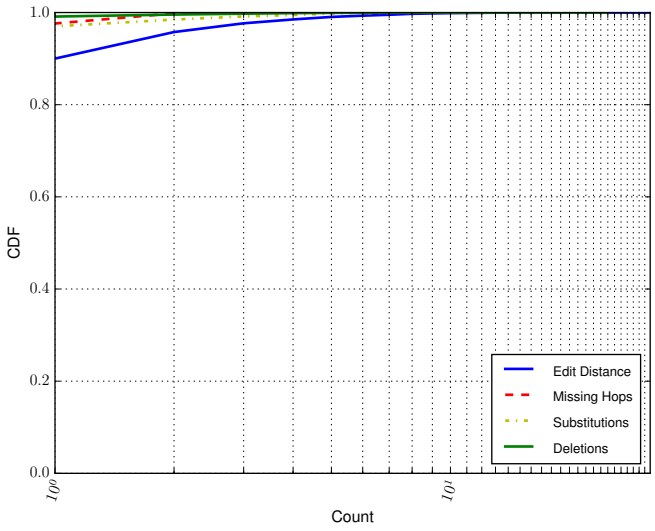

Figure 4.3. Edit Distance of Two Different CAIDA Scans Separated by 15 Hours.

These results require a more in depth analysis of the traces in an attempt to discover the reason for these anomalies. We look at the resulting warts binary trace files by hand to see if we can determine any patterns. We find two interesting cases that show up repeatedly and help explain the high number of differences in our edit distance analysis.

The first case is what appears to be traces taking different load balanced paths. This is possible because, while both methods use Paris traceroute, a per-flow load balancing router could route the probes from CAIDA's scans through one path and the probes from Yarrp6

through a different path. We see that most of these cases result in the edit distance differing by N number of substitutions where N appears to be the number of hops beyond the per-flow load balancing router. Table 4.1 shows one example of this. We see that the CAIDA trace and Yarrp6 trace take the same hop thru hop 5, at which point they diverge.

Table 4.1. Yarrp6 and CAIDA ICMPv6 Traces to 2607:b400::1.

Hop	Yarrp6	CAIDA
1	2001:48d0:101:501::18	2001:48d0:101:501::18
2	2001:468:e00:c48::1	2001:468:e00:c48::1
3	*	2607:f380::108:9a41:af60
4	2001:468:e00:801::2	2001:468:e00:801::2
5	2001:470:0:72::2	2001:470:0:72::2
6	2001:470:0:362::2	2001:470:0:15d::1
7	2001:470:0:360::1	2001:470:0:204::2
8	2001:470:0:35f::2	2001:470:0:2c9::1
9	2001:478:132::182	2001:504:0:2:0:4:220:1
10	2607:b400:f0:8002::f0	2607:b400:f0:2002::f0

The second interesting case is created as a result of the fact that we run our reconstruction offline. An error in our *yrp62warts.py* script, which converts the file from the yrp file output by Yarrp6 into a warts binary trace file, handles the reconstruction of loops incorrectly. Table 4.2 shows an example of this case. Looking at the edit distance we see that this trace has 2 deletions where the Yarrp6 trace is missing hops 12 and 13. We further analyze this by pulling out all responses received from probes to 2801:80:330::1 from our yrp file. In looking at these responses, we discover that we receive a response for all hops sent by Yarrp6 to this destination with the exception of hop 2. Our results also show a loop with the same two addresses starting at hop 11. We leave the improvement of the *yrp62warts.py* to future work.

In using the above methods of comparison we take a first step toward demonstrating that the results of Yarrp6 sometimes match with the results obtained using state-of-the-art tools. We have shown that Yarrp6 discovers similar number of interfaces and produces more links between these interfaces.

Table 4.2. Yarrp6 and CAIDA ICMPv6 Traces to 2801:80:330::1.

Hop	Yarrp6	CAIDA
1	2001:48d0:101:501::18	2001:48d0:101:501::18
2	*	2001:468:e00:c48::1
3	2607:f380:1::108:9a41:ac91	2607:f380:1::108:9a41:ac91
4	2001:468:ff:302::1	2001:468:ff:302::1
5	2001:468:ff:100::1	2001:468:ff:100::1
6	2001:468:ff:110::2	2001:468:ff:110::2
7	2001:468:ff:9c5::2	2001:468:ff:9c5::2
8	2001:12f0:0:fd::26	2001:12f0:0:fd::26
9	2001:12f0:0:fc::3e	2001:12f0:0:fc::3e
10	2001:12f0:0:fe::82	2001:12f0:0:fe::82
11	2801:82:0:f8::8	2801:82:0:f8::8
12		2801:82:0:f8::1
13		2801:82:0:f8::8

4.1.1 Rate-limiting

As discussed in Section 2.1.1, RFC 4443 states that, "an IPv6 node MUST limit the rate of ICMPv6 error messages it originates" [15]. The effect of this rate-limiting at the speeds at which Yarrp6 scans is unclear. In this section we present some initial analysis that appears to support that rate-limiting does affect the scan results of Yarrp6. In Figure 4.4 we plot a CDF of the hop depth of missing hops in our scan results. We see that in our scans, hops 1–3 account for ~ 64% of all of the missing hops. Additionally, hop 3 accounts for ~ 47% of all missing hops.

We do a further analysis of missing hops by comparing our scans with the scans conducted by CAIDA. We look at each probe at a given hop (e.g., hop 3) and if we are missing this hop in a trace we check the corresponding trace to the same destination in CAIDA's scans. If CAIDA's scans are not missing the corresponding hop, we record the address for that hop. We find that in all traces, 36,864 of 78,435 (~ 47%) are missing hop 3. Of these 36,864 missing hops, CAIDA's scans have an address for 31,312 (~ 85%) of these hops. The list of addresses for missing hops in Yarrp6's scan for which we found an address in the corresponding CAIDA trace contains only 6 unique addresses. Additionally, 4 of the 6 addresses found had IPv6 addresses assigned in an address block allocated to the California Research and Education Network (CalREN). This could be due to more aggressive rate-limiting put in place by CalREN, or some other unrelated circumstance.

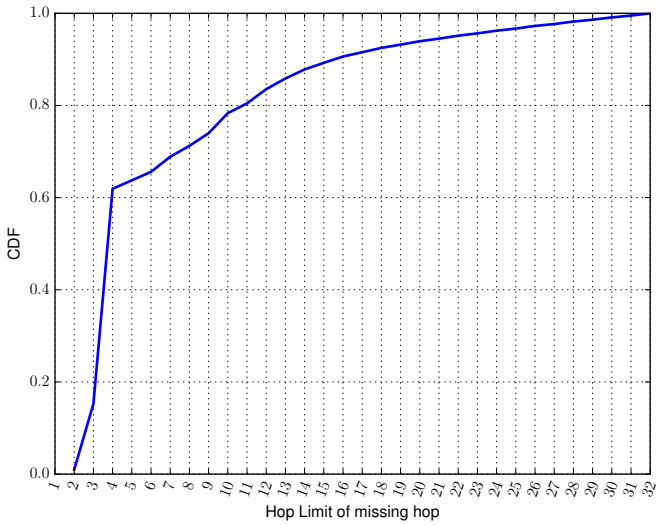

Figure 4.4. CDF of Missing Hops by TTL in Yarrp6 ICMPv6 Scan Results.

While these results do not definitively show the presence of rate-limiting affecting our traceroute results, they suggest possible rate-limiting. We are unaware of any current or past research into rate-limiting of ICMPv6 Error messages and we believe that there is a strong need for continued research into rate-limiting of ICMPv6 error messages. In Section 5.1 we discuss some ideas for possible future research that might be done into ICMPv6 rate-limiting [22].

4.1.2 Comparison of discovery rate of Yarrp6 vs. CAIDA

A primary goal of this research is to develop a tool that has the ability to scan the vast IPv6 address space more rapidly then current tools. In this section we evaluate how much quicker Yarrp6 scans than the state-of-the-art tool. We evaluate the scan speeds of Yarrp6 by comparing our scan rates to the scan rates obtained by CAIDA's IPv6 active topology measurements that use scamper.

We start by looking at the number of unique interfaces discovered by Yarrp6 and CAIDA's scans as a function of time. We plot the number of unique interfaces discovered by each tool, sampling every 10 seconds for the duration of the time it takes Yarrp6 to complete the scan (601 seconds). Figure 4.5 displays this plot and shows that Yarrp6 discovers many more unique interfaces per second then CAIDA. Yarrp6 discovers 51,898 unique interfaces

in 601 seconds averaging ~ 86 unique interfaces per second, while CAIDA discovers 1,623 unique interfaces within the same time period averaging ~ 3 unique interfaces per second.

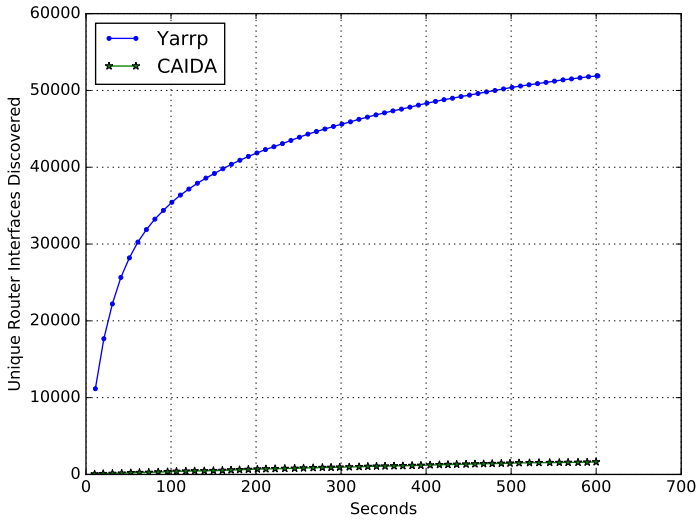

Figure 4.5. Discovery Rate of Yarrp6 versus CAIDA as a Function of Time.

While this is a substantial increase in speed, we also wish to compare the number of unique interfaces discovered as a function of probes sent by each tool. We use this metric to determine if Yarrp6 is more efficient by examining the probing cost relative to the infrastructure discovery benefit. We plot the discovery rate of Yarrp6 and CAIDA as unique interfaces discovered per probe as a function of the number of probes in Figure 4.6. We can see from the shape of the graph that Yarrp6 initially begins with a much higher rate of interfaces discovered per probe sent, staying above the CAIDA scans until around 10^6 probes. Overall, Yarrp6 discovers 51,909 unique interfaces using 2,508,576 probes averaging ~ 0.02 unique interfaces per probe, while CAIDA discovers 51,940 unique interfaces using 1,673,211 probes averaging ~ 0.03 unique interfaces per probe. This increase in the number of probes sent can be attributed to the fact that Yarrp6 is stateless. Scamper is designed to stop probing a destination if it has been reached or if there are five consecutive non-responsive hops, but Yarrp6 will continue to probe in these cases due to its stateless nature.

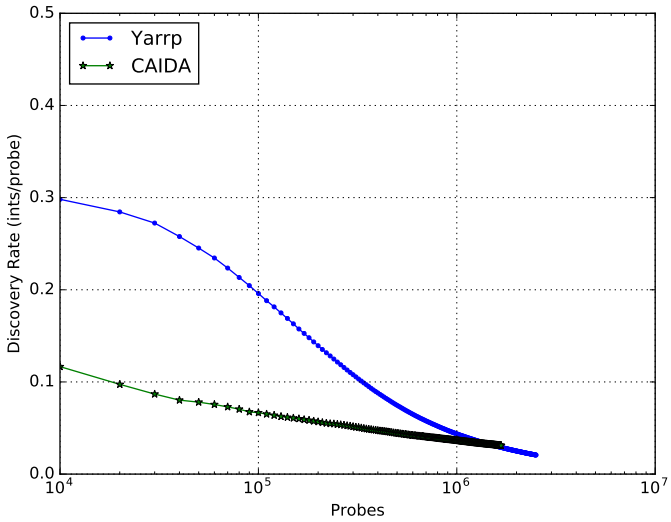

Figure 4.6. Discovery Rate of Yarrp6 versus CAIDA as a Function of Probes Sent.

The max hop limit used by Yarrp6 is runtime configurable via a command line option and has a direct impact on the number of packets Yarrp6 will send. The total number of packets Yarrp6 sends in a trace can be calculated using IP$s \times HopLimit$. For example, the scans we conducted in this study probed 78,393 destinations using the hop limit of 32. This means that we send 2, 508, 576 (78, 393 $\times$ 32) packets. The default hop limit used by Yarrp6 is 32, and is based on our analysis of the number of unique addresses observed at each hop limit in historical scan data.

In Figure 4.7, we plot unique addresses discovered at the corresponding hop limit. We use data from all scans conducted from the 46 VPs used in CAIDA's February 12, 2017, IPv6 cycle. The red line indicates the median number of interfaces per hop limit among each VP. We can see from this graph that the most interfaces are discovered in hops 9–14 with the inflection point around hop 10. We also see that the number of interfaces discovered beyond hop 27 is negligible. We also compare this to the same analysis done for IPv4 in [16]. We see that, compared to IPv4, the graph of CAIDA's IPv6 data shows that the distribution of unique router interfaces discovered is skewed toward lower hop limits. While reducing the number of hops that we probe will reduce the number of packets that we are required to send, it could come at the cost of possibly discovering less infrastructure. We leave a thorough investigation of the optimum max hop limit to future work.

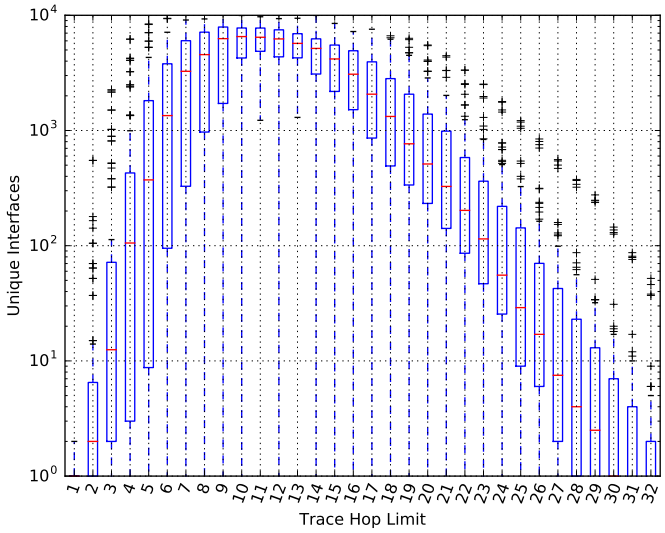

Figure 4.7. Distribution of Interfaces Discovered as a Function of Hop Limit.

While Yarrp6 scans at a higher rate than other tools, it requires more packets to do so. The stateless nature of Yarrp6 allow it to scan 28 times faster than scamper, as configured in the CAIDA results, but at the cost of sending 1.5 times the packets. The amount of packets sent can be reduced by reducing the amount of hops Yarrp6 probes, but this has the potential to reduce the amount of infrastructure Yarrp6 discovers. In our opinion, the reduced completion time that we achieve using Yarrp6 outweighs the relatively small cost of sending more packets.

4.2 Effect of Transport Layer Protocol

The second component of the results is aimed at determining which transport layer protocol performs best for active topology mapping of the IPv6 Internet. As discussed in Section 3.3, we define best by using three metrics: (i)destinations reached; (ii) complete IP paths; and (iii) unique IP links. The three metrics used in this study mirror three of the four metrics used in the same study of IPv4 [18]. We excluded unique AS links in this study, and will leave this for future work. We also include the number of unique addresses discovered for a direct comparison of actual infrastructure discovered.

We start by using Yarrp6 to probe the same destinations from the same san-us VP used for

the validation portion of this study. We conduct scans using UDP, TCP SYN, and TCP ACK probes. We use the same scan conducted for the validation portion as our results for ICMPv6. Table 4.3 list the results of these scans for each metric.

Table 4.3. Result of Probes for Different Transport Layer Protocols.

Probe Method	Unique Interfaces	Destinations Reached	Complete IP Paths	Unique IP Links
ICMPv6	51,909	10,320	4,339	73,590
UDP	44,353	4,546	1,126	59,884
TCP SYN	40,592	N/A	N/A	53,013
TCP ACK	40,602	N/A	N/A	52,664

4.2.1 Destinations Reached

We first examine the results of the Destinations Reached metric by looking at the total number of destinations reached for each method. ICMPv6 elicits a response from far more destinations than UDP: 13% of the time compared to 6% of UDP. As discussed in Section 3.3, Yarrp6 does not attempt to determine when a destination is reached while probing with TCP SYN or TCP ACK packets. The expected response from a node when a TCP SYN or TCP ACK packet is received is a TCP SYN ACK or a TCP RST. These responses will not contain the quote of the invoking packet and therefore none of the encoded information Yarrp6 uses for path reconstruction will be received. For this reason we do not include any results for destinations reached in the case of TCP SYN or TCP ACK.

As part of our analysis of the destinations reached metric, we also compare the halting reasons of ICMPv6, UDP, TCP SYN, and TCP ACK probes. The halting reason of a probe is the reason for which the probing of a destination is stopped. Scamper has four halting reasons that will cause it to stop probing a destination:

1. A probe can halt due to the destination being reached.
2. A probe can halt due to receiving an ICMPv6 Destination Unreachable error message with a source address other then the probed destinations address.
3. A probe can halt due to the detection of a routing loop, where an address appears for a second time in the same trace.

4. A probe can halt when a gap limit of unresponsive hops is reached. In scamper the gap limit is reached when there are five consecutive unresponsive hops.

Due to Yarrp6's stateless operation, it will continue probing a destination until the configured number of max hops for a destination are probed (i.e., 32 hops.) While Yarrp6 does not halt for any of the reasons listed above, we can determine the halting reason that would occur in our offline reconstruction of the paths and thus we are able to conduct an analysis of the halting reasons.

Table 4.4 shows the results of the analysis of the halting reason. Gap Limit is the largest reason for halting with ~ 52% of all ICMPv6 and ~ 57% of all UDP probes halting for this reason. The second largest reason for halting was due to destination unreachable messages that contained a different source address than that of the target we were probing.

Table 4.4. Halting Reasons for Probes.

Probe Method	Total Probes	Destination Unreachable	Loop	Gap Limit
ICMPv6	78,435	19,301	8,112	40,780
UDP	78,343	21,911	7,260	44,654
TCP SYN	78,412	19,703	7,095	N/A
TCP ACK	78,609	18,712	7,262	N/A

According to RFC 4443, an ICMPv6 Destination Unreachable error message can have one of seven codes:

- 0 – No route to destination.
- 1 – Communication with destination administratively prohibited.
- 2 – Beyond scope of source address.
- 3 – Address unreachable.
- 4 – Port unreachable.
- 5 – Source address failed ingress/egress policy.
- 6 – Reject route to destination. [15]

Of these seven codes, three are of particular interest to us when talking about active topology mapping of the IPv6 Internet. Codes 1, 5, and 6 indicate filtering of our traffic by firewalls and routers. This can affect the amount of infrastructure we are able to discover by filtering

our probes to certain destinations and not allowing us to probe any further.

Table 4.5 shows the number of these three codes received within our Yarrp6 scans. The absence of code 5 in Table 4.5 is due to the fact that we do not receive any code 5 messages in our scan results. Of the remaining two codes for ICMPv6 Destinations Unreachable error messages received by the different probe methods, we see that the reject route code is the most prevalent, accounting for 18%–20% of each method's ICMPv6 Destination Unreachable error messages and ~ 5% of each methods total probes.

Table 4.5. ICMPv6 Destination Unreachable Codes in Yarrp6 Data.

Probe Method	Total	Administratively Prohibited (1)	Reject Route (6)
ICMPv6	19,301	688	3,849
UDP	21,911	2,212	3,824
TCP SYN	19,703	2,216	3,788
TCP ACK	18,712	1,039	3,804

The final ICMPv6 Destination Unreachable code we will discuss is code 1 which is sent when a packet is administratively prohibited and indicates the filtering of a specific type of traffic rather then all traffic coming from a certain prefix. Table 4.6 shows the results of our analysis of the percentage of ICMPv6 Destination Unreachable code 1 error messages. We see that ICMPv6 probes are filtered substantially less than the three remaining methods. ICMPv6 Destination Unreachable code 1 messages account for only ~ 4% of the ICMPv6 Destination Unreachable error messages and ~ 0.8% of ICMPv6's total probes. Of the three remaining methods, TCP ACK receives only slightly more ICMPv6 Destination Unreachable code 1 messages with ~ 5% of all ICMPv6 Destination Unreachable error messages and ~ 1% of all probes. UDP and TCP ACK both experience similar numbers of ICMPv6 Destination Unreachable code 1 message with ~ 10% and ~ 11% of all ICMPv6 Destination Unreachable error messages respectively and ~ 1% of total probes. The higher rate of filtering on UDP and TCP SYN may be a direct result of our choice of ports for these protocols. These specific ports may be filtered by destinations that do not have a web server running (TCP port 80) and do not allow for external DNS queries (UDP port 53). Additionally, a large number of TCP SYN packets in a short period of time might be flagged as a SYN flood by some destinations, resulting in higher filtering of our TCP SYN packets. We will discuss methodology for evaluating port selection in Section 5.1.

Table 4.6. Percentage of ICMPv6 Destination Unreachable Code 1 in Yarrp6 Data.

Probe Method	Percentage of Destination Unreachable	Percentage of Total Probes
ICMPv6	~ 4%	~ 0.8%
UDP	~ 10%	~ 1%
TCP SYN	~ 11%	~ 1%
TCP ACK	~ 5%	~ 1%

The final reason for halting is the detection of a routing loop, where the same IP address is seen for a second or more times in a trace. Routing loops account for the halting reason in ~ 10% of all probes for each method. Routing loops are indicative of routing problems and are not dependent on the method of the probing as can be seen by their relative consistency among all methods. While it is unlikely that duplicate addresses on different hops is due to real routing loops, we have not identified conditions in which the same IPv6 source address responds to probes to a given destination address with different hop limit values. The presence of routing loops requires more in depth analysis, but we leave that for future work.

4.2.2 Complete IP Paths, Unique IP Links, and Final Analysis

Next, we examine the complete IP paths metric. As discussed in Section 3.3, a complete IP path is a path where the destination is reached and a response is received from all intermediate hops within the path. We again exclude the TCP SYN and TCP ACK methods from these results due to the inability of Yarrp6 to currently retrieve the state information from the destinations response packets. We also exclude hop 3 of all paths in our analysis of complete IP paths based on results of the analysis conducted in Section 4.1.1. We feel that due to the large amount of missing addresses at this hop and relatively small number of addresses found at these hops in CAIDA's data, the results presented here are still accurate when excluding these hops. We see from Table 4.3 that ICMPv6 probes discover nearly 4 times as many complete IP paths as UDP probes. Of all ICMPv6 probes, ~ 5% are complete IP paths compared to ~ 1% of UDP probes.

The last metric we examine is the Unique IP Link metric. Recall from Section 3.3 that a Unique IP Link is any pair of adjacent hops within a trace. Looking again at Table 4.3, we see that ICMPv6 probes find ~ 1.3 times the amount of unique IP Links over UDP, TCP

SYN, and TCP ACK probes.

Based on our above analysis, we conclude, that for active topology mapping of the IPv6 Internet, ICMPv6 probes are the best method. ICMPv6 probes discover more unique IP links, find more complete IP paths, reach the most destinations, and are the least filtered of the four methods. While TCP SYN and TCP ACK probes were excluded from some of the results, we feel that the number of metrics collected on them support our conclusion that they will not perform better then ICMPv6.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusion

In this thesis, we develop a new tool for high-speed, active topology mapping of the IPv6 Internet. This new tool is based on Yarrp, a tool and traceroute method, developed at The Naval Postgraduate School (NPS), which is stateless in nature and enables scanning at speeds of up to 100K PPS. Yarrp currently only supports IPv4. We extend the functionality of Yarrp to IPv6 and name this new tool Yarrp6. To the best of our knowledge, and based on our evaluation, Yarrp6 redefines the state-of-the-art in high-speed discovery of IPv6 routing infrastructure.

After development, we evaluate Yarrp6 by comparing its performance against existing state-of-the-art tools and production mapping systems. In particular, we compare against results from CAIDA's IPv6 active topology mapping scans which employ the scamper tool for probing. We demonstrate Yarrp6's ability to produce topologically similar results to those from CAIDA, even when run at high speeds. Our evaluation shows that Yarrp6 discovers similar numbers of unique interface addresses and more links between these interfaces. We also evaluate the edit distance (the number of interface level address insertions, deletions, and substitutions) between CAIDA's traces and ours. We find that only $\sim 50\%$ of the paths match exactly, but find that similar differences between successive CAIDA's scans, even when separated by a short 15-hour period.

After validation of our tool's results, we use Yarrp6 to conduct a study of the impact of transport layer protocol on forward IP path discovery. We start by probing the same list of destination addresses used by the validation portion of our study using four different probe methods. These four methods differ in the transport layer protocol used for the probes and consist of ICMPv6, UDP, TCP SYN, and TCP ACK-based probes. We then analyze the data using three metrics for comparison of these different protocols. We evaluate each protocol based on the number of destinations reached, the number of complete IP paths, and the number of unique IP links found by each method. We find that in all three metrics, ICMPv6 based probes preform better than the remaining three methods. ICMPv6 probes reach over two times the number of destinations, reach thousands of destinations not reached by any

other method, discover four times as many complete IP paths, and discover ~ 1.3 times as many unique IP links. These findings suggest that ICMPv6 traffic is filtered by firewalls and routers far less frequently than other probe transport layer protocols. We conclude from these results that ICMPv6 based probes are the best method for active topology mapping of the IPv6 Internet.

5.1 Future Work

In this section, we present suggestions for possible future work in improvements to Yarrp6 and further research into active topology mapping of the IPv6 Internet with Yarrp6.

5.1.1 Improvements to Yarrp6

We provide six specific ways to improve our tool:

- As discussed in Section 3.1.3, Yarrp6 currently does not follow the Paris traceroute method for ICMPv6 probes. This is due to the inclusion of the hop limit, which changes with every packet Yarrp6 sends, in the calculation of the checksum and the use of the checksum for load balancing by per-flow load balancing routers. Future version of Yarrp6 should include a function which adds a value to the payload in order to ensure that the checksum matches for all packet, regardless of what header fields within the packet change.
- In order to control all of the fields within the IPv6 packet, Yarrp6 uses a packet socket. While this allows control of all fields within the packet, it comes at the cost of requiring the developer to include the layer 2 header with their packets. This requires them to determine the destination MAC address and source MAC address in order to build and send the packets. This responsibility is offloaded to the user for initial development of Yarrp6. Future versions of Yarrp6 should have the capability to determine the source MAC and destination MAC without input from the user. Potential solutions to this problem might include the addition of the capability to use Neighborhood Discovery to determine the MAC addresses or use an already developed library to leverage the already existent routing tables within the system to determine the MAC addresses.
- The use of the packet sockets in order to control the headers for Paris traceroute restricts the use of Yarrp6 to systems running Linux. Many active measurement

platforms have a mix of Linux and other operating systems (i.e., CAIDA's Ark active measurement infrastructure uses a mix of Linux and FreeBSD nodes [45].) In future versions of Yarrp6, a more portable solution might be found, allowing for its use on as many systems as possible.

- As discussed in Section 3.1.4, the RFC-required rate-limiting of ICMPv6 error message responses by routers is likely to be triggered by the high rates at which Yarrp6 is capable of scanning. In section 4.1.1, our analysis show some indications of rate-limiting within hops 1–4. We accounted for this by excluding hop 3 from some of our results, but this is clearly a compromise. In later versions of Yarrp6, the "neighborhood" list method that Yarrp uses should be added to Yarrp6. By tracking new addresses within those hops, and ceasing probing to them if no new addresses are seen within the last 30 seconds, we can reduce the over-probing of hops close to our VP and potentially reduce the amount of rate-limiting we experience.
- In this study, Yarrp6 only achieved scan rates of $\sim 4,000$ PPS. While this is a substantial increase in speed when compared to state-of-the-art tools, it is not anywhere near the 100K PPS speed achieved by Yarrp on IPv4 [16]. In future development of Yarrp6, improvements should be made to allow for scanning at 100K PPS speeds.
- Finally, we currently have no mechanism in place to determine if the information returned in the ICMPv6 Error Message quote is the same information we initially placed in the packet when we sent it. The fact that we are encoding all of the information needed to reconstruct the path within the payload of the packet requires us to ensure that the data in the payload was not modified after being sent. In future versions of Yarrp6, a Hashed Message Authentication Code (HMAC) should be added to the payload in order to allow us to determine if the information has been modified in any way and ensure that our offline reconstruction is accurate.

5.1.2 Scanning with Yarrp6

Besides improvements to Yarrp6 itself, we suggest the following procedural enhancements:

- The number of prefixes that were scanned for this study is still quite modest relative to the vast address space of IPv6. In this study we only performed an initial performance evaluation of Yarrp6. Future studies should scan a much larger number of prefixes. Scanning of the prefixes used in Rohrer et al. [10] is a logical continuation of this

study. Additionally, using a technique to reduce the target space by finding the active portions of the IPv6 Internet (i.e., Entropy/IP [11]) and generating a target list from this for active scans using Yarrp6 present an interesting potential direction for future work.

- As discussed in Section 4.1.1, there appears to be evidence of rate-limiting occurring within our scan data. In this study, we do a simple analysis of missing hops in an attempt to detect rate-limiting. A more rigorous analysis of rate-limiting is needed to determine its effect on active topology mapping of the IPv6 Internet. Guo et al. have done a more rigorous study on rate limiting of ICMP [22]. The methodology for this study might be extended to ICMPv6.
- The impact probing rate has on topology discovery is currently unknown. A rigorous study should be done to better understand how the speeds at which Yarrp6 can scan will impact overall topology discovery.
- Additionally, the impact of the load Yarrp6 imposes on nodes at the edge of the IPv6 Internet is also unknown. A study should be completed to fully understand the impact Yarrp6 will have on the load that is placed on nodes at the edge of the IPv6 Internet.
- One application of rapid topology discovery is the study of short-lived dynamics within the Internet as discussed in [16]. Conducting successive scans to the same destinations within a short period of time allows us to observe these short-lived dynamics and gain a better overall understanding of the Internet. An experiment similar to the one conducted in [16] should be conducted to study these short-lived dynamics within the IPv6 Internet.
- Finally, in this study, we only use one destination UDP port (53) and one destination TCP port (80). In our scan data, we see significant filtering of UDP and TCP which may be related to our choice of ports. A study of the effect of the port number on active topology mapping of the IPv6 Internet should be conducted. Scans conducted with many different well-know ports as well ephemeral ports should be analyzed to determine how port selection might minimize filtering of active scanning.

List of References

[1] American Registry for Internet Numbers. (n.d.). IPv4 addressing options. [Online]. Available: https://www.arin.net/resources/request/ipv4_countdown.html. Accessed Jan. 23, 2017.

[2] Réseaux IP Européens. (n.d.). RIPE NCC IPv4 available pool -- graph. [Online]. Available: https://www.ripe.net/publications/ipv6-info-centre/about-ipv6/ipv4-exhaustion/ipv4-available-pool-graph. Accessed Jan. 23, 2017.

[3] Asia-Pacific Network Information Center. (n.d.). APNIC's IPv4 pool status. [Online]. Available: https://www.apnic.net/community/ipv4-exhaustion/graphical-information. Accessed Jan. 23, 2017.

[4] Latin America and Caribbean Network Information Centre. (n.d.). Available IPv4 space. [Online]. Available: http://opendata.labs.lacnic.net/ipv4stats/graphs/ipv4avail.html. Accessed Jan. 23, 2017.

[5] African Network Information Center. (n.d.). IPv4 exhaustion. [Online]. Available: http://www.afrinic.net/en/services/statistics/ipv4-exhaustion. Accessed Jan. 23, 2017.

[6] S. Kawamura and M. Kawashima, "A recommendation for IPv6 address text representation," RFC 5952 (Proposed Standard), Internet Engineering Task Force, Aug. 2010, Aug. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5952.txt

[7] J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil, and M. Bailey, "Measuring ipv6 adoption," in *Proceedings of the 2014 ACM Conference on SIGCOMM* (SIGCOMM '14), Chicago, Illinois, USA, 2014, pp. 87–98.

[8] Internet Society. (n.d.). World IPv6 launch. [Online]. Available: http://www.worldipv6launch.org/. Accessed Jan. 23, 2017.

[9] A. Dhamdhere, M. Luckie, B. Huffaker, k. claffy, A. Elmokashfi, and E. Aben, "Measuring the deployment of ipv6: Topology, routing and performance," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference* (IMC '12), Boston, Massachusetts, USA, 2012, pp. 537–550.

[10] J. Rohrer, B. LaFever, and R. Beverly, "Empirical study of router ipv6 interface address distributions," *IEEE Internet Computing*, vol. 20, no. 4, pp. 36–45, Jul. 2016.

[11] P. Foremski, D. Plonka, and A. Berger, "Entropy/ip: Uncovering structure in ipv6 addresses," in *Proceedings of the 2016 ACM on Internet Measurement Conference* (IMC '16), Santa Monica, CA, 2016, pp. 167–181.

[12] D. Plonka and A. Berger, "Temporal and spatial classification of active ipv6 addresses," in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference* (IMC '15), Tokyo, Japan, 2015, pp. 509–522.

[13] M. Gray, "Discovery of ipv6 router interface addresses via heuristic methods," M.S. thesis, Dept. Comput. Sci., Naval Postgraduate School., Monterey, CA, 2015.

[14] B. LaFever, "Methods for intelligent mapping of the ipv6 address space," M.S. thesis, Dept. Comput. Sci., Naval Postgraduate School., Monterey, CA, 2015.

[15] A. Conta, S. Deering, and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," RFC 4443 (Draft Standard), Internet Engineering Task Force, Mar. 2006, updated by RFCs 4884. [Online]. Available: http://www.ietf.org/rfc/rfc4443.txt

[16] R. Beverly, "Yarrp'ing the internet: Randomized high-speed active topology discovery," in *Proceedings of the 2016 ACM on Internet Measurement Conference* (IMC '16), Santa Monica, CA, 2016, pp. 413–420.

[17] Center for Applied Internet Data Aanalysis. (2016, Jun. 9). The CAIDA UCSD IPv4 routed /24 topology dataset. [Online]. Available: http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml

[18] M. Luckie, Y. Hyun, and B. Huffaker, "Traceroute probe method and forward ip path inference," in *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement* (IMC '08), Vouliagmeni, Greece, 2008, pp. 311–324.

[19] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with paris traceroute," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement* (IMC '06), Rio de Janeriro, Brazil, 2006, pp. 153–158.

[20] V. Giotsas, G. Smaragdakis, B. Huffaker, M. Luckie, and k. claffy, "Mapping peering interconnections to a facility," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies* (CoNEXT '15), Heidelberg, Germany, 2015, pp. 37:1–37:13.

[21] A. Broido and k. claffy, "Internet topology: Connectivity of ip graphs," in *Proc. SPIE*, 2001, vol. 4526, pp. 172–187.

[22] H. Guo and J. Heidemann, "Detecting ICMP rate limiting in the Internet (extended)," USC/Information Sciences Institute, Los Angeles, CA, Tech. Rep. ISI-TR-717, Feb. 2017.

[23] F. Chen, R. K. Sitaraman, and M. Torres, "End-user mapping: Next generation request routing for content delivery," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (SIGCOMM '15), London, United Kingdom, 2015, pp. 167–181.

[24] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister, "Census and survey of the visible internet," in *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement* (IMC '08), Vouliagmeni, Greece, 2008, pp. 169–182.

[25] B. Augustin, T. Friedman, and R. Teixeira, "Measuring load-balanced paths in the internet," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement* (IMC '07), San Diego, California, USA, 2007, pp. 149–160.

[26] K. Claffy. (2015, Nov. 13). Cartographic capabilities for critical cyberinfrastructure ("C4"): Internet topology and performance analytics for mapping critical network infrastructure. [Online]. Available: https://www.caida.org/publications/presentations/2015/cartographic_capabilities_dhs/cartographic_capabilities_dhs.pdf

[27] Google. (n.d.). IPv6 Google. [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html. Accessed Mar. 17, 2017.

[28] Center for Applied Internet Data Aanalysis. (2016, Jun. 9). The CAIDA UCSD IPv6 topology dataset. [Online]. Available: http://www.caida.org/data/active/ipv6_allpref_topology_dataset.xml

[29] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 4291 (Draft Standard), Internet Engineering Task Force, Feb. 2006, updated by RFCs 5952, 6052, 7136, 7346, and 7371. [Online]. Available: http://www.ietf.org/rfc/rfc4291.txt

[30] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460 (Draft Standard), Internet Engineering Task Force, Dec. 1998, updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, and 7112. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[31] Van Jacobson. (1989). *traceroute*. [Online]. Available: ftp://ftp.ee.lbl.gov/traceroute.tar.gz. Accessed Feb. 19, 2017.

[32] Matthew Luckie. (2016). *Scamper*. [Online]. Available: https://www.caida.org/tools/measurement/scamper/. Accessed Dec. 12, 2016.

[33] M. Luckie, "Scamper: A scalable and extensible packet prober for active measurement of the internet," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement* (IMC '10), Melbourne, Australia, 2010, pp. 239–245.

[34] Robert Beverly. (2016). *Yarrp*. [Online]. Available: https://www.cmand.org/yarrp. Accessed Sep. 14, 2016.

[35] R. L. Rivest, "The rc5 encryption algorithm," in *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings* (Lecture Notes in Computer Science), 1994, vol. 1008, pp. 86–96.

[36] IAB and IESG, "IAB/IESG Recommendations on IPv6 Address Allocations to Sites," RFC 3177 (Informational), Internet Engineering Task Force, Sep. 2001, obsoleted by RFCs 6177. [Online]. Available: http://www.ietf.org/rfc/rfc3177.txt

[37] T. Narten, G. Huston, and L. Roberts, "IPv6 Address Assignment to End Sites," RFC 6177 (Best Current Practice), Internet Engineering Task Force, Mar. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6177.txt

[38] J. Postel, "Internet Control Message Protocol," RFC 792 (Internet Standard), Internet Engineering Task Force, Sep. 1981, updated by RFCs 950, 4884, 6633, and 6918. [Online]. Available: http://www.ietf.org/rfc/rfc792.txt

[39] SANS Institute. (2006, Apr. 19). Egress filtering FAQ. [Online]. Available: https://www.sans.org/reading-room/whitepapers/firewalls/egress-filtering-faq-1059

[40] J. Black and P. Rogaway, "Ciphers with arbitrary finite domains," in *Cryptographers' Track at the RSA Conference*, 2002, pp. 114–130.

[41] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers, "The simon and speck lightweight block ciphers," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, June 2015, pp. 1–6.

[42] W. Stevens, M. Thomas, E. Nordmark, and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6," RFC 3542 (Informational), Internet Engineering Task Force, May 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3542.txt

[43] Ubuntu. (n.d.). Ubuntu manpage: packet, AF_PACKET - packet interface on device level. [Online]. Available: http://manpages.ubuntu.com/manpages/precise/man7/packet.7.html. Accessed Mar. 04, 2017.

[44] W. Willinger, D. Alderson, and J. C. Doyle, "Mathematics and the internet: A source of enormous confusion and great potential," *Notices of the AMS*, vol. 56, no. 5, pp. 586–599, May 2009.

[45] Center for Applied Internet Data Aanalysis. (2016, Nov. 21). Archipelago measurement infrastructure. [Online]. Available: http://www.caida.org/projects/ark/

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California