NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IPV6 ALIAS RESOLUTION VIA INDUCED ROUTER FRAGMENTATION

by

William D. Brinkmeyer Jr.

June 2013

Thesis Advisor:	Robert Beverly
Second Reader:	Geoffrey Xie

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
27–3–2013	Master's Thesis	2011-03-28—2013-06-21

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
IPv6 Alias Resolution via Induced Router Fragmentation	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
William D. Brinkmeyer Jr.	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Naval Postgraduate School Monterey, CA 93943	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
Department of the Navy	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis 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

14. ABSTRACT

IPv4 addresses are a scarce resource with available allocations nearing exhaustion. DoD and government agencies were mandated to transition to IPv6 for greater security and flexibility. The transition to IPv6 faces a series of challenges associated with protecting the network. Among many defensive challenges associated with IPv6 is the inability to accurately identify and understand the network's router-level topology. Providing an accurate IPv6 topology map is needed for security, situational awareness, and understanding the operational deployment and evolution of IPv6. To better understand IPv6 networks, this thesis focuses on the alias resolution problem whereby we seek to identify multiple interfaces belonging to a single IPv6 router. Alias resolution is critical to developing an accurate router-level topology map. This thesis presents a fingerprint-based IPv6 alias resolution technique that induces fragmented responses from IPv6 router interfaces. We demonstrate perfect alias resolution accuracy in a controlled environment, and on a small subset of the production IPv6 Internet for which ground-truth is known. Internet-wide testing finds that over 70% of IPv6 interfaces probed respond to the method.

15. SUBJECT TERMS

IPv6, Alias resolution

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	65	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

IPV6 ALIAS RESOLUTION VIA INDUCED ROUTER FRAGMENTATION

William D. Brinkmeyer Jr.
Lieutenant , United States Navy
B.S. Computer Information Systems, Chapman University, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

from the

NAVAL POSTGRADUATE SCHOOL
June 2013

Author: William D. Brinkmeyer Jr.

Approved by: Robert Beverly
Thesis Advisor

Geoffrey Xie
Second Reader

Cynthia Irvine
Chair, Cyber Academic Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

IPv4 addresses are a scarce resource with available allocations nearing exhaustion. DoD and government agencies were mandated to transition to IPv6 for greater security and flexibility. The transition to IPv6 faces a series of challenges associated with protecting the network. Among many defensive challenges associated with IPv6 is the inability to accurately identify and understand the network's router-level topology. Providing an accurate IPv6 topology map is needed for security, situational awareness, and understanding the operational deployment and evolution of IPv6. To better understand IPv6 networks, this thesis focuses on the alias resolution problem whereby we seek to identify multiple interfaces belonging to a single IPv6 router. Alias resolution is critical to developing an accurate router-level topology map. This thesis presents a fingerprint-based IPv6 alias resolution technique that induces fragmented responses from IPv6 router interfaces. We demonstrate perfect alias resolution accuracy in a controlled environment, and on a small subset of the production IPv6 Internet for which ground-truth is known. Internet-wide testing finds that over 70% of IPv6 interfaces probed respond to the method.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**
1.1 Motivation 2
1.2 Department of Defense Applicability 3
1.3 Research Questions 4
1.4 Thesis Structure 4

2 Background and Related Work **5**
2.1 Background 5
2.2 Related Work 13

3 Methodology **21**
3.1 Eliciting Fragmented Responses 22
3.2 Ground-Truth Testing 24
3.3 IPv6 Alias Resolution Algorithm 26
3.4 Controlled Alias Resolution 28

4 Analysis **29**
4.1 Efficacy of TBT 29

5 Conclusions **35**
5.1 Limitations 35
5.2 Future Work 36
5.3 IPv6 Load Balancing 38

List of References **40**

Initial Distribution List **47**

List of Figures

Figure 2.1 Alias vs. Non-alias. 5

Figure 2.2 IPv6 Header Format. 7

Figure 2.3 IPv6 Extension Headers. 9

Figure 2.4 IPv6 Fragment Header. 10

Figure 2.5 IPv6 Destination Options Header. 10

Figure 2.6 IPv6 Routing Header. 11

Figure 2.7 ICMPv6 Message Format. 12

Figure 2.8 Identifying Aliases with Atlas. 14

Figure 2.9 Identifying Non-aliases with Atlas. 14

Figure 2.10 Route Positional Method. 15

Figure 2.11 Using the Destination Options Header Method with IPv6-in-IPv4 Tunnels. 15

Figure 2.12 Destination Options Header After the IPv6 Routing Header. 16

Figure 2.13 Example Ground-truth Topology. 17

Figure 2.14 Traceroute Topology with False Links and Missing Nodes. 17

Figure 3.1 TBT, the "Too-Big Trick" 23

Figure 3.2 GNS3 Test Topology with Asymmetric MTU Paths Inducing ICMPv6 Packet Too Big. 24

Figure 4.1 Histogram of IPv6 Fragment Identifiers Occurring $\geq 0.3\%$ 32

Figure 4.2 Histogram of IPv6 Fragment Identifiers Occurring $\geq 0.3\%$ 32

List of Tables

Table 2.1 IPv6 Next Header Values. 6

Table 2.2 ICMPv6 Error Message Codes. 13

Table 4.1 TBT Response Characteristics 30

Table 4.2 Operating System Identifiers for Alias Resolution 33

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

USG	United States Government
DoD	Department of Defense
IPv6	Internet Protocol version 6
IPv4	Internet Protocol version 4
IP	Internet Protocol
IPID	Internet Protocol Identifier
ICMPv6	Internet Control Message Protocol for IPv6
ICMP	Internet Control Message Protocol
CIDR	Classless Inter-Domain Routing
NAT	Network Address Translation
IANA	Internet Assigned Numbers Authority
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
BGP	Border Gateway Protocol
TTL	Time-To-Live
MTU	Maximum Transmission Unit
PMTU	*path* MTU
DNS	Domain Name Service
GNS3	Graphical Network Simulator
IOS	Internetwork Operating System
OS	Operating System
CDN	Content Distribution Network
CAIDA	Cooperative Association for Internet Data Analysis
AS	Autonomous System

ARP	Address Resolution Protocol
BSD	Berkeley Software Distribution
TBT	Too-Big Trick
SSRR	Strict Source and Record Route
LSRR	Loose Source and Record Route
CIO	Chief Information Officer
IHL	Internet Header Length
ISP	Internet Service Provider
RIR	Regional Internet Registry
DoS	Denial of Service
MIDAR	Monotonic ID-Based Alias Resolution

Acknowledgements

First and foremost, I would like to thank my wife, Edith, for all of her love and support. Thank you for putting up with me spending countless hours on the computer and for keeping me focused on what is most important. Without your continued sacrifice and support, I would not be where I am today. I would like to thank our children, Brandon, Lance and Noelani for reminding me to take breaks and enjoy the small things in life. I could't have done this without the support of my family.

I would also like to thank my second reader, Professor Geoff Xie, for his constructive feedback. The ability to leverage your expertise was beneficial throughout this research. Finally, I would like to thank my advisor Professor Rob Beverly for his mentorship, advice and encouragement throughout this entire process. Without your guidance and insight, this work would not have been possible.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

The address space in Internet Protocol version 4 (IPv4), 2^{32}, or approximately four billion addresses, is nearing exhaustion. The last available addresses have been allocated to the Regional Internet Registries (RIRs) [1]. While there are *approximately* 230 million (<0.01%) unique IPv4 addresses remaining [2], it is projected that IPv4 addresses for four of the five RIRs will be exhausted by 2014 and complete IPv4 exhaustion by 2020 [3]. The exhaustion of IPv4 addresses presents the problem of not having enough globally unique addresses to meet demand. Scarcity has made IPv4 addresses an economic commodity; IPv4 address blocks are sold and companies have been purchased solely for their address allocations [4], [5]. Various methods were introduced to delay the exhaustion of IPv4 addresses including Classless Inter-Domain Routing (CIDR), to reduce the rate at which IPv4 address space would be consumed [6], and Network Address Translation (NAT), to enable private address space to be used on internal networks [7]. However, these methods were only a short term solution to a larger problem and did not solve the the IPv4 address exhaustion problem as the last available IPv4 address blocks were allocated by Internet Assigned Numbers Authority (IANA) in February 2011 [1]. In addition to not solving the IPv4 address exhaustion problem, NAT has the disadvantage of taking away the end-to-end significance of an Internet Protocol (IP) address which can lead to the unanticipated failures in end-to-end communications [7], [8].

As a result, Internet Protocol version 6 (IPv6), standardized 15 years ago [9] as the successor to IPv4, is experiencing commercial deployment–primarily due to economic and business constraints, rather than any technical impetus [10]. Modern systems and hardware support IPv6, service and content providers are deploying IPv6 [11], and United States Government (USG) networks are mandating IPv6 [12]. Additionally, Google indicates that the percentage of users that access Google over native IPv6 has exceeded 1% [13].

The number of global IPv6 Border Gateway Protocol (BGP) routing prefixes is growing exponentially [14]. More than 6,000 autonomous systems, approximately 15%, now announce IPv6 reachability [15]. As IPv6 experiences an increased global adoption, the size of the IPv6 Internet will continue to grow. Amid IPv6 measurement efforts underway [16], [17], understanding the structure, and evolution, of the IPv6 router-level topology is an ongoing challenge. This thesis is a step forward in understanding IPv6 topology.

1.1 Motivation

Department of Defense (DoD) and USG agencies were mandated to transition from IPv4 to IPv6 in an effort to provide greater security and flexibility to the networks under its cognizance [12]. With the transition to IPv6 comes a series of challenges associated with defending the network. One challenge in securing and defending a network is the ability to accurately identify and understand the network's topology, specifically from a router-level topology mapping perspective.

Providing an accurate IPv6 topology map is needed for network security, situational awareness, and will aid in the identification of potential threats. Additionally, accurately identifying the IPv6 topology can improve the general knowledge of IPv6 and is necessary in tracking its evolution, for network research and routing protocol development. To better understand and characterize IPv6 network topologies, this thesis focuses on the alias resolution problem as alias resolution is critical in developing an accurate router-level topology map.

This research investigates IPv6 *alias resolution*: the process of determining if two IP addresses are assigned to *different* interfaces of the *same* physical router [18]. Alias resolution reduces the interface level graph, e.g., discovered via active probing, into a router-level graph [19], thereby permitting a better understanding of the resilience and robustness properties of the network [20].

As IPv6 alias resolution provides the ability to construct accurate router-level topology maps, the ability to compare both the IPv6 and IPv4 Internet topologies is desired in order to determine similarities of the two topologies. The ability to identify dual stacked interfaces, i.e., interfaces that are configured with both an IPv4 and IPv6 address, is crucial to the identification and protection of critical infrastructure and recognizing potential impacts of attacks. On dual stacked links the potential exists for an attack utilizing IPv4 to impact the performance of IPv6 as well as an IPv6 attack to affect IPv4 performance. For example, a Denial of Service (DoS) attack carried out on an IPv4 network where traffic flows over a dual stacked link can result in the degradation of IPv6 data flow as well due to both IPv4 and IPv6 data flows being on the same transmission medium (Ethernet, fiber optics, etc.).

Taking inspiration from prior IPv4 alias resolution work, this thesis presents a fingerprint based IPv6 alias resolution technique that relies on eliciting fragmented responses from IPv6 router interfaces. Although IPv6 has no in-network fragmentation [9], sources can send large IPv6 packets in fragments. This research finds that, as with IPv4 routers, the IPv6 fragment identifier counter is frequently common across a router's interfaces. While all IPv4 control-plane packets

sourced by a router require a unique fragment identifier, IPv6 fragment identifiers increase only when the router must source a fragmented packet. Thus, as Chapter 3 will describe, in contrast to fragmentation-based IPv4 alias resolution that is prone to false positives due to background control-plane traffic incrementing a small 16-bit counter, this IPv6 technique is highly accurate because control-plane messages are rarely fragmented, thus this traffic does not increment the IPv6 32-bit counter.

This thesis seeks to detail and validate a new IPv6 alias resolution algorithm for performing Internet-wide IPv6 alias resolution. Four primary contributions are made in this thesis:

1. Development of a fingerprint-based IPv6 alias resolution technique, termed the Too-Big Trick (TBT).
2. Validation on a large virtualized testbed of common commercial routers.
3. Internet-wide probing of more than 49,000 distinct live IPv6 router interfaces of which approximately 70% respond to tests.
4. Validation of the TBT technique on a small subset of the production IPv6 network for which alias ground-truth is known, where perfect accuracy is obtained.

1.2 Department of Defense Applicability

Offensive and defensive cyber operations are inherently dependent on the structure of the Internet. As the DoD transitions to IPv6, the need to understand the structure of the networks under its responsibility, and more importantly the IPv6 Internet in general, becomes increasingly important. This work seeks to provide a tool that can be used by the DoD to develop accurate router-level maps for use in cyber operations.

In developing a router-level topology map, the benefits of IPv6 alias resolution are not limited to defensive or security measures. In an offensive cyber operation, having an accurate router-level map can allow commanders to focus their resources appropriately. For example, given a pool of IPv6 addresses as targets, understanding if a given address is an alias of any other addresses will prevent duplication of effort, thus minimizing resource allocation to conduct an operation. However, if the topology is unknown, an operation may require allocating resources for each address, thus delaying the execution of other operations. Additionally, having an accurate representation of the topology allows planners to assess the potential impact of an operation ahead of time.

1.3 Research Questions

The questions that this research aims to answer are:

1. What are the limitations of current alias resolution techniques?
2. Are there any IPv4 alias resolution techniques that can be used in IPv6 alias resolution?
3. Is there a reliable identifier that can be used as a fingerprint for IPv6 alias resolution?

1.4 Thesis Structure

This remainder of this thesis is organized as follows:

- Chapter 2 provides and overview of the fundamentals of IPv6, highlights the major differences between IPv4 and IPv6 and describes prior efforts including the techniques developed for use in both IPv4 and IPv6 alias resolution.
- Chapter 3 discusses the methodology used in this research to include the virtual testbed components and configuration, topology and router configurations, preliminary findings and ground-truth testing.
- Chapter 4 details the results of both virtual and live Internet ground truth alias resolution testing, and provides initial results on the ability to perform end host alias resolution.
- Chapter 5 provides conclusion based on this research and provides recommendations for future areas of research, including research to address the areas where TBT falls short.

CHAPTER 2:
Background and Related Work

Alias resolution is the process of determining if two IP addresses are assigned to *different* interfaces of the *same* physical router [18].

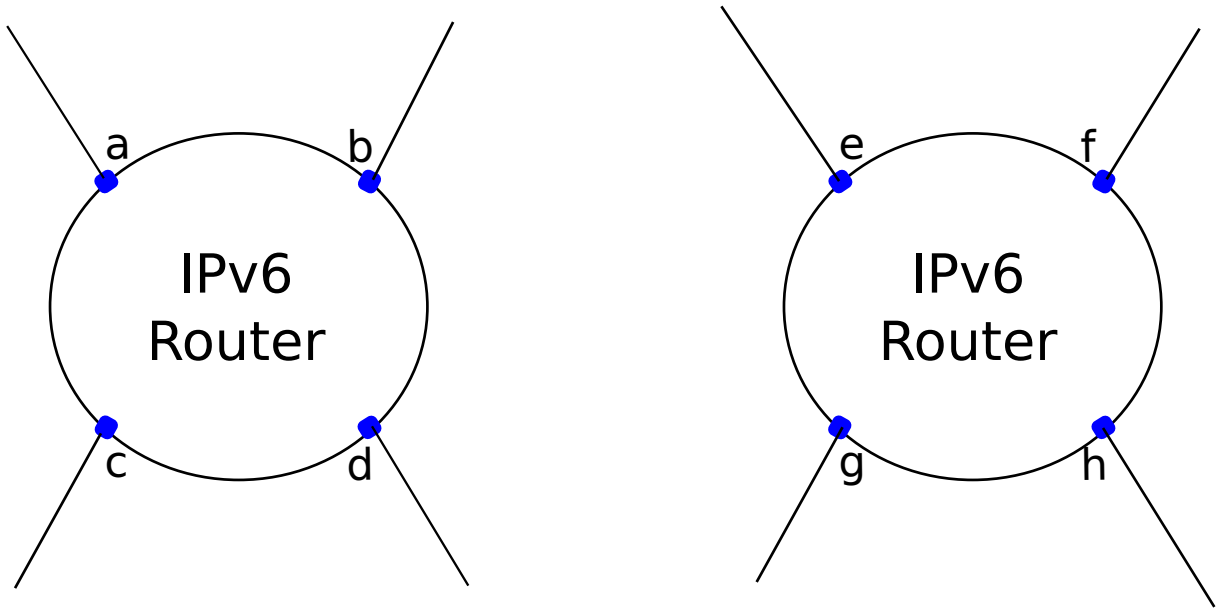

Figure 2.1: Interfaces located on the *same* physical router (i.e., a, b, c and d) are referenced as aliases whereas interfaces on *different* physical routers (i.e., a and e) are not aliases.

Design differences between IPv4 and IPv6 obsolete some of the alias detection techniques used in IPv4, while enabling new ones. This chapter covers IPv6 and relevant changes from IPv4 (§2.1), highlighting the technologies used in this research, and reviews existing alias resolution techniques for both IPv4 and IPv6 (§2.2).

2.1 Background

Nearly 17 years its inception, IPv6 is finally gaining momentum partly due to IPv4 address exhaustion. In January 2011, the last of the available IPv4 addresses were allocated to the RIRs by IANA [1]. In September 2010, the Federal Chief Information Officer (CIO) mandated that the USG transition to IPv6, outlining milestones to upgrade external facing servers and services, and internal client applications supporting enterprise networks to operationally use native IPv6 [21]. Comcast has started issuing IPv6 addresses to home users [22], and as the

population of Internet capable device users increases, the growth of the Internet is reliant on the adoption of IPv6.

As IPv6, the successor to IPv4, gains popularity and is more widely implemented, the ability to infer accurate IPv6 topologies increases, motivating the need for alias resolution. Alias resolution is an important tool when attempting to create accurate views of a network. These maps can provide the ability to identify critical points within a network allowing an organization to focus its resources when attempting to identify and secure critical components and information paths within a larger network.

In order to accurately identify and track the adoption of IPv6 while monitoring the growth of the IPv6 Internet, there must be a mechanism that can be used to establish an accurate baseline of nodes and infrastructure in use. Additionally, the ability to identify relationships among paths becomes increasingly important when attempting to creating accurate views of IPv6 networks. As a result, IPv6 alias resolution allows measurement researchers to track the changes of the IPv6 Internet infrastructure as IPv6 is more widely deployed.

2.1.1 IPv6 Header

The IPv6 header was completely revamped from IPv4. As shown in Figure 2.2, some IPv4 header fields were removed or replaced by fields located in optional IPv6 extension headers. A description of the IPv6 header fields is provided below [9]:

Version–specifies the Internet Protocol version, with IPv6 identified by the value of six. *Traffic Class*–identifies the priority and class of service of this packet. *Flow Label*–is used to identify packets that are part of a unique flow [23]. *Payload Length*–the payload length, in octets, of the data that follows the IPv6 header. *Next Header*–identifies the type of header that immediately follows the IPv6 header. Table 2.1 lists some of the next header values referenced in this work.

Table 2.1: IPv6 Next Header Values. From [24].

Decimal	Keyword	Protocol
41	IPv6	IPv6 encapsulation
43	IPv6-Route	Routing Header for IPv6
44	IPv6-Frag	Fragment Header for IPv6
58	IPv6-ICMP	ICMP for IPv6
59	IPv6-NoNxt	No Next Header for IPv6
60	IPv6-Opts	Destination Options for IPv6

Hop Limit–similar to the Time-To-Live (TTL) field of the IPv4 header, an 8-bit counter that specifies the number of hops a packet can traverse en route to a specified destination. The

originating source of the packet is responsible for assigning the hop limit value. Example default hop limit values are 64 (Cisco and Juniper routers) [25], [26] and 128 (Windows) [27]. As a packet is processed by the router, the router must decrease the hop limit value by one before forwarding the packet. If the hop limit value reaches zero, the packet is discarded and an Internet Control Message Protocol for IPv6 (ICMPv6) Time Exceeded message is returned to the originating host in order to prevent routing loops. ICMPv6 is further described in § 2.1.3

Source Address–the 128-bit address of the node that originated the packet. *Destination Address*–the 128-bit address of the recipient that the packet is destined for.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.2: IPv6 Header Format. From [9].

When comparing the IPv4 and IPv6 headers, one notices that the IPv6 header has fewer fields and is aligned to support the current processors found in most host computers and devices by using 4 and 8 bit boundaries that make header disassembly more efficient. Another difference between the headers is that the IPv6 header has a fixed size of 40 bytes whereas the IPv4 header size can vary from 20 to 60 bytes based on the Options field length. The IPv6 header has also been simplified by removing all the fields (from IPv4) that have little to no use in the v6 version

of the protocol. Below are some examples of the field differences between the headers [9]:

In IPv4, the *Internet Header Length (IHL)* field was necessary to identify the size of the header. The IHL field was removed due to the IPv6 header being a fixed size. Since IPv6 does not permit in-path fragmentation, the *Identification*, *Flags* and *Fragment Offset* were moved to Fragment header. The *Header Checksum* field was removed from the IPv6 header. *Options* are no longer defined in the IPv6 header and have been moved to the Destination Options and Hop-by-Hop Options extension headers.

To improve performance, the IPv6 header contains only essential data with everything else (e.g., fragmentation) handled by extension headers. Overall, the IPv6 header has been streamlined for simplicity and performance.

2.1.2 Extension Headers

In addition to the changes in the IPv6 header, one of the most notable changes in IPv6 is the use of extension headers to extend protocol functionality. Extension headers follow the IPv6 header and are a sequential list of optional internet layer information that are encoded in separate headers that may be placed between the IPv6 header and the transport layer header in the packet [9]. Extension headers contain an 8-bit option type that indicates the next extension header, an 8-bit unsigned integer that tells how long the header is, and the option data payload that is of variable size. Extension headers can be combined, where several appear concatenated (or "chained") in a single packet, but typically only a few are included [28]. The structure and arrangement of extension headers is illustrated in Figure 2.3.

An IPv6 packet can contain any number of extension headers which provides additional internet-layer information. Extension headers must be processed in the order that they are received. Each extension header is identified by the Next Header field of the preceding header (e.g., a next header value of 44 identifies the fragment header).

Extension headers are not examined or processed by any node along a packet's delivery path until the packet reaches the destination address identified in the IPv6 header, with the exception of the Hop-by-Hop options [9] and the destination options [29] headers.

Since IPv6 extension headers must be processed by the destination node in the order that they are received, RFC 2460 [9] recommends that those headers appear in the following order if more than one extension header is used in the same packet:

```
+---------------+------------------------
|  IPv6 header  | TCP header + data
|               |
| Next Header = |
|      TCP      |
+---------------+------------------------

+---------------+----------------+------------------------
|  IPv6 header  | Routing header | TCP header + data
|               |                |
| Next Header = | Next Header =  |
|    Routing    |      TCP       |
+---------------+----------------+------------------------
+---------------+----------------+-----------------+-----------------
|  IPv6 header  | Routing header | Fragment header | fragment of TCP
|               |                |                 |  header + data
| Next Header = |  Next Header = |  Next Header =  |
|    Routing    |    Fragment    |       TCP       |
+---------------+----------------+-----------------+-----------------
```

Figure 2.3: IPv6 extension headers are arranged with the next header value identifying subsequent extension headers or protocols. From [9].

1. IPv6 Header
2. Hop-by-Hop Options header
3. Destination Options header
4. Routing header
5. Fragment header
6. Authentication header
7. Encapsulation Security Payload header
8. Destination Options header
9. Upper-layer header

The order of precedence for processing IPv6 extension headers determines the manner in which an IPv6 packet will be processed by in-path routers, as used in prior techniques described in §2.2. For example, if the Destination Options header is placed before the Routing header, only the destination processes the Destination Options header; however, if placed after the routing header, each in-path router that supports source routing and the final destination process the Destination Options header [29].

Fragment Header

IPv6 does not allow in-network fragmentation, therefore all fragmentation is pushed to the source node to reduce overhead associated with packet fragmentation by in-path routers. The Fragment Header, Figure 2.4, is used by an IPv6 source node when sending a packet that exceeds the path Maximum Transmission Unit (MTU) to its destination. Each packet receives a Fragment Identification value that must be different from any other fragmented packet, during the maximum lifetime of the packet, with the same source and destination addresses. The fragment header is identified by the value 44 in the preceding Next Header field [9].

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |   Reserved    |      Fragment Offset    |Res|M|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Identification                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.4: IPv6 Fragment Header. From [9].

Destination Options Header

The destination options header, Figure 2.5 is used to carry optional information that need be examined only by a packet's destination node(s) [9]. As described by Qian et al. [29], the placement of the Destination Options header in relation to a routing header will determine how it will be processed. If the destination options header is placed before the routing header each in-path router will process it, otherwise it is processed only by the destination node.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |  Hdr Ext Len  |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
.                                                               .
.                            Options                            .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.5: IPv6 Destination Options Header. From [9].

Routing Header

The Routing header, identified by a next header value of 43, is used by an IPv6 source to list one or more intermediate nodes that a packet should traverse en route to the destination. In IPv4

there are two forms of source routing that allow a collection of nodes to be specified that the packet must traverse before reaching the destination: Strict Source and Record Route (SSRR), which requires the entire path to be defined in advance; and Loose Source and Record Route (LSRR), which only requires part of the path is set in advance [30].

The IPv6 Routing Header Type 0 is provides an extended version of IPv4 LSRR, thus using a similar method of source routing in IPv6 widely disabled in IPv4 [31]. Packets containing a routing header, Figure 2.6, that exceeds the size of a path's specified MTU are discarded by the receiving node following the processing of the routing header with the receiving node sending a ICMPv6 Packet Too Big message [9].

While source routing is widely disabled in IPv4 due to security concerns and the similarities between IPv4 and IPv6 source routing it is likely that IPv6 source routing will follow suit with IPv4 and be disabled, thus rendering current source routing based alias resolution techniques unusable.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |  Hdr Ext Len  |  Routing Type | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                       type-specific data                      .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.6: IPv6 Routing Header. From [9].

2.1.3 ICMPv6

ICMPv6, Figure 2.7, is used by IPv6 nodes to report errors encountered in processing packets, and to perform other internet-layer functions, such as diagnostics and testing (e.g., traceroute6). ICMPv6 operates on top of IPv6 as an extension header but actually works in conjunction with IPv6 for protocol operations. ICMPv6 is an integral part of IPv6, and must be fully implemented by every IPv6 node [32]. ICMPv6 is preceded by the IPv6 header and extension headers, if used, and is identified by a Next Header value of 58, whereas in IPv4, Internet Control Message Protocol (ICMP) is identified by the Protocol value of one. ICMPv6 combines functions previously subdivided among different protocols. For example, ICMPv6 Neighbor

Discovery combines Address Resolution Protocol (ARP), ICMP Router Discovery, and ICMP Redirect used in IPv4 [33].

ICMPv6 error messages are structured to include as much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU value of 1280 bytes [32], compared to the Internet Header plus the first 64 bits of original data in ICMP, [34].

Since IPv6 does not allow in-path fragmentation, if the size of a packet exceeds the MTU of the outgoing link, an ICMPv6 Packet Too Big message is sent to the originating node indicating the MTU of the outgoing link [32]. This research relies on the ability for nodes to send and receive Packet Too Big messages.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         Message Body                          +
|                                                               |
```

Figure 2.7: ICMPv6 Message Format. From [32].

ICMPv6 messages contain a type and a code that relate the details of the message to the type of message, as well as a checksum and a payload of variable size. ICMPv6 error messages relay useful information back to the source of the packet about any error that may have occurred along the path.

Type indicates the type of message [32].

There are two general types of ICMPv6 messages:

- ICMPv6 error messages
  - Destination Unreachable (1)
  - Packet Too Big (2)
  - Time Exceeded (3)
  - Parameter Problem (4)
- ICMPv6 informational messages
  - Echo Request (128)
  - Echo Reply (129)

Code used to create an additional level of message granularity. Table 2.2 summarizes the different ICMPv6 codes.

Table 2.2: ICMPv6 Error Message Codes. After [32].

Type	Code	Code Description
1	0	no route to destination
	1	communication with destination administratively prohibited
	2	beyond scope of source address (draft)
	3	address unreachable
	4	port unreachable
2	0	Set by sender and ignored by the receiver
3	0	hop limit exceeded in transit
	1	fragment reassembly time exceeded
4	0	erroneous header field encountered
	1	unrecognized next header type encountered
	2	unrecognized IPv6 option encountered

Checksum used to detect data corruption in the ICMPv6 message and parts of the IPv6 header.

2.2 Related Work

Significant research has been performed in the area of alias resolution. Various techniques exist today to provide the ability to resolve alias pairs in an attempt to reduce interface-level network maps to the more useful router-level maps [35]. Features such as source routing, previously disabled in IPv4 due to security reasons, have been enabled in IPv6 allowing researchers to develop additional alias resolution techniques using previously obsolete technologies.

2.2.1 Source Routing

Source routing provides the ability to transmit a packet along a set path, either completely or partially, rather than relying on routers to provide path determination. Source routing is largely disabled in IPv4 due to the potential security risks such as spoofing. With LSRR, an attacker can send a packet to the destination with a spoofed source address, enabling the attacker to see all of the return traffic by placing his address as a hop in the specified route.

These source routing methods can be used for remote network discovery, to bypass firewalls and to achieve packet amplification for the purposes of generating denial-of-service traffic [30].

The reintroduction of source-routing, using the routing header in IPv6 has enabled several new approaches including Atlas [36], the Route Positional Method [37], and the option header method [29].

Atlas

Given a potential alias pair (x, y), Atlas [36] performs a User Datagram Protocol (UDP) traceroute to y via x with the hop limit set to expire at x and relies on the fact that routers will generally process the routing header before checking the hop limit. Figures 2.8 and 2.9 show prober, P, and IPv6 routers, A and B. If x and y are aliases, *both* "hop limit exceeded" *and* "port unreachable" ICMPv6 messages will be generated. If x and y are not aliases, only an ICMPv6 "hop limit exceeded" message will be generated.

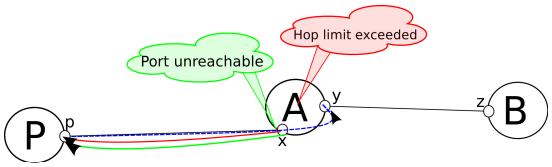

Figure 2.8: Hop limit exceeded and port unreachable messages indicate x and y are aliases. After [36].

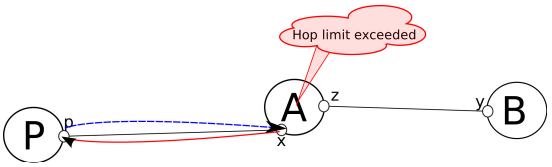

Figure 2.9: Non-aliases will only result in a hop limit exceeded message being returned. After [36].

By using source routing Atlas was more effective at discovering nodes compared to using tools such as traceroute which only discovered 617 nodes, just 45% of those identified by Atlas. Additionally, results showed that Atlas discovered a large number of prefixes compared to those previously identified in the 6Bone registry [36]. The authors identify the inefficiency of Atlas in performing alias resolution on large networks, but propose using Atlas in combination with other techniques to further reduce redundant probing required by the system.

Route Positional Method

The route positional method [37] makes use of the observation that the source address of ICMPv6 "hop limit exceeded" messages for packets that are *not* destined to the router at which the expiration occurred is frequently the address of the ingress port. To discover aliases for address y, probes are sent from p *via* x and y destined to p, with the hop limit set to expire at y. Performing this probe for a large enough set of addresses x will result in ICMPv6 "hop limit exceeded" messages originating from aliases of y.

Destination Options Header Method

This method uses the mandatory source routing features of IPv6 to direct packets through specific in-path routers, viarouters, and through IPv6-in-IPv4 tunnels using a strategically placed destination options header. The destination options header is processed by each viarouter, in-path routers that support source routing, if placed before the IPv6 routing header. If the destination options header is placed after the IPv6 routing header, only the final destination will

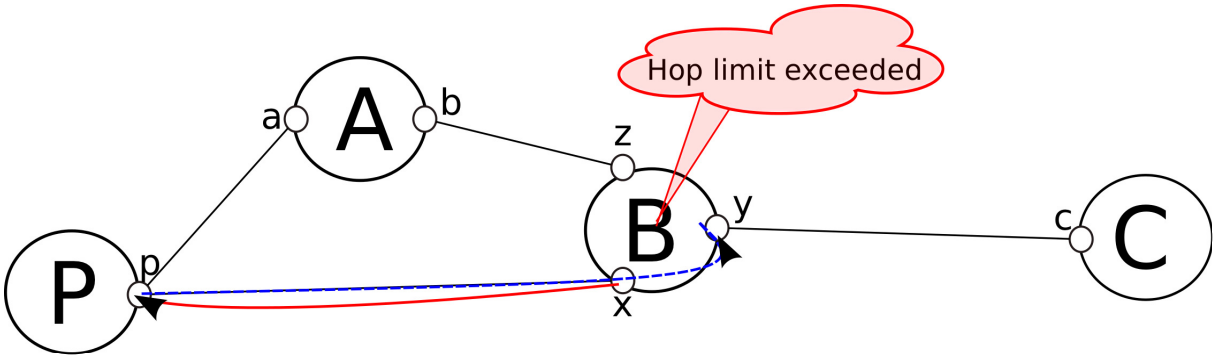

Figure 2.10: The Route Positional Method observes the source address of ICMPv6 "hop limit exceeded" messages. After [37].

process it [29]. The destination options header is placed before the IPv6 routing header when sending probes through tunnels, whereas placing the destination options header after the IPv6 routing header is used when tunnels are not expected to be encountered.

The Destination Options header method, proposed by the same authors as Route Positional Method, makes use of the fact that setting an invalid bit sequence in the IPv6 options header will result in an ICMPv6 "parameter problem" message being generated, originating from the ingress interface of the packet generating the response. By probing via multiple intermediate routers (similar to the Route Positional Method), multiple aliases of the target address may be discovered.

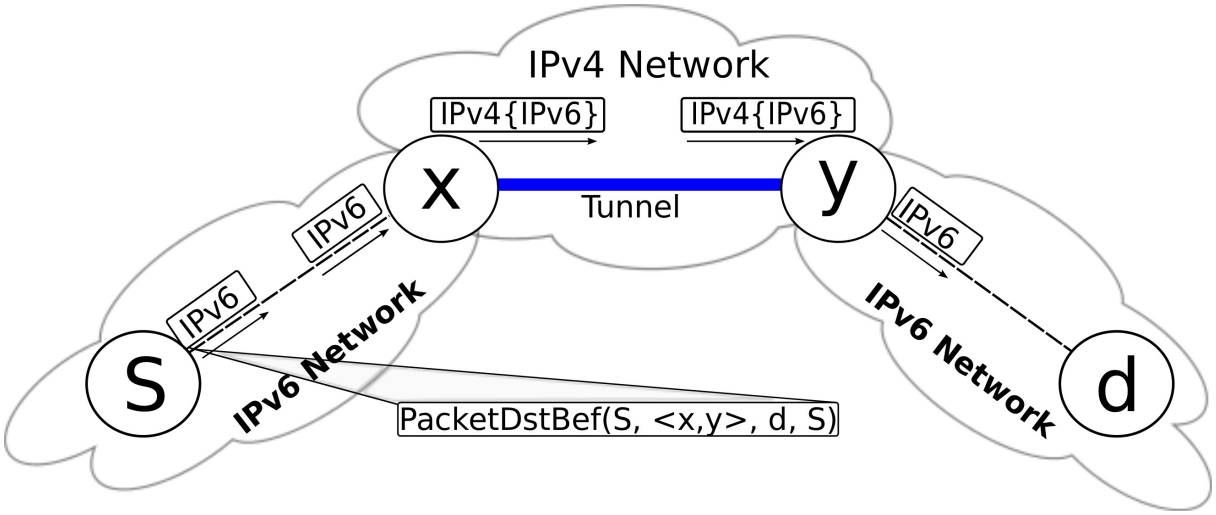

Figure 2.11: To discover aliases using IPv6-in-IPv4 tunnels place the destination options header before the IPv6 routing header. After [29].

Figure 2.11 illustrates an example where the destination options header placed before the IPv6

routing header identifying source host S sending a probe to itself through an IPv6-in-IPv4 tunnel by way of a viarouter, d. The first two bits of the destination options header are "10", which causes the viarouter d to discard the packet and send an ICMPv6 parameter problem message to the probing host S. If the source address of the ICMPv6 parameter problem message is not equal to the destination address, d, then d and and the source of the ICMPv6 parameter problem message are aliases.

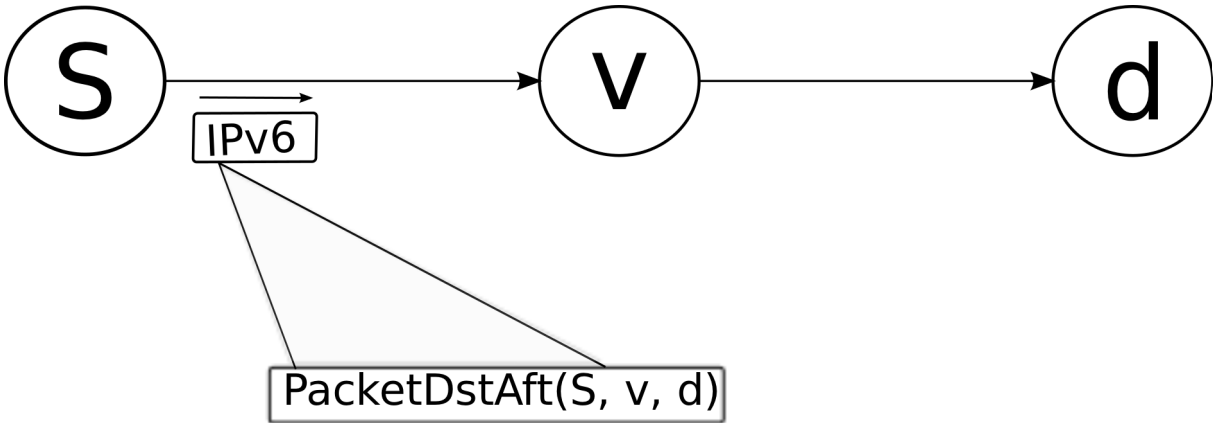

Figure 2.12: To force only the destination to process the destination options header, place it after the IPv6 routing header. After [29].

Figure 2.12 illustrates the second technique where the destination options header placed after the IPv6 routing header identifying source host S sending a probe to itself by way of a viarouter, v. By placing the destination options header after the IPv6 routing header, only the destination d will process the destination options header and send an ICMPv6 parameter problem message back to prober S. If the source of the ICMPv6 parameter problem message is not equal to the destination d, then d and and the source of the ICMPv6 parameter problem message are aliases.

Qian et al. [29] compared the destination options header method to the address-based method described [38]. Test results showed that the non-tunneled method identified 3426 alias pairs and the tunneled method identified 4544 alias pairs. When both methods are combined, a total of 5566 alias pairs were identified with 84.5% accuracy compared to 3383 identified with address-based technique.

2.2.2 Inference Methods

Traceroute-based Methods

Traceroute-based methods send TTL limited probes to determine interface-level network maps. Traceroute send a series of TTL limited probes to determine the path a packet would travel to

a given destination. As the traceroute probe is processed by in-path routers, the TTL value is decremented by one until reaching zero, where an ICMP "Time Exceeded" message is generated, with a source address of the router's inbound interface that generated address as the and sent back to the originating source. This returns the addresses for the inbound interface

Factors such as per-packet load balancing can complicate traceroute methods, causing missing nodes and links or false links [39]. For example, given the topology shown in Figure 2.13, comprised of probing source, S, and routers A, B, C, D and E. A traceroute is performed from S to E, with the traceroute probes traveling via L, TTL=1, ATTL=2, and DTTL=3, to destination E, TTL=4. The resulting topology, shown in Figure 2.14, indicates a direct path of L, A, D, E, which indicates a false link between A and D.

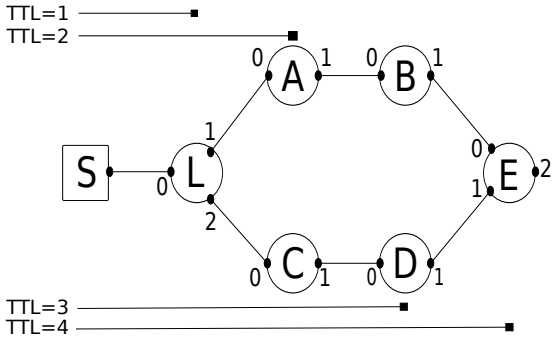

Figure 2.13: Example Ground-truth Topology. After [39].

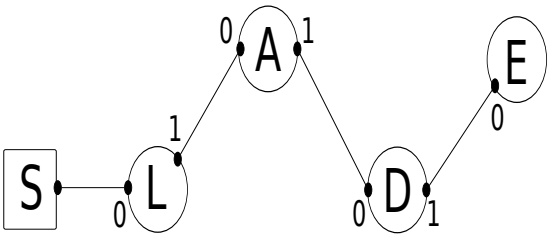

Figure 2.14: Traceroute Topology with False Links and Missing Nodes. After [39].

Paris-traceroute

Paris-traceroute [39] is the current state of the art used to correct the deficiencies of Traceroute for IPv4 networks. Paris traceroute provides the ability to accurately identify load-balanced IPv4 routes, that traceroute can sometimes misrepresent, and provide an accurate interface-level topology map for IPv4 networks by utilizing per flow load balancing and simultaneously manipulating the Identifier and Sequence number fields of the ICMP routing header and the sequence number field of the Transmission Control Protocol (TCP) header. The authors have identified the need for future work to develop an IPv6-based method of Paris traceroute, but at the time of this work is not yet implemented.

2.2.3 Internet Protocol Identifier (IPID)

Design differences between IPv4 and IPv6 obsolete some of the alias detection techniques used in IPv4, while enabling new ones. As described in §2.1.1, IPv6 does not permit in-network fragmentation, hence the IPID field was removed from the IPv6 header. Additionally, as shown

Chapter 3, the implementation of IPv6 fragmentation reduces the amount of fragmented traffic generated by routers, thus the IPv6 equivalent of IPID, the fragment identification counter, can increment little, if any, over a relatively long period of time. This hinders a direct application of the velocity modeling used by methods such as RadarGun.

RadarGun

RadarGun [40] introduces velocity modeling as a method for conducting alias resolution. It compares the rate at which a target interface's IPID counter increases with respect to a group of continuous probes sent to the target interface. RadarGun reduced the number of probes required, and the number of false positives generated, as compared to the Rocketfuel method [35]. There is not a guaranteed solution however, due to the approximation characteristics of this method.

The small IPID counter space is impacted by the pigeon hole principle. In the pigeonhole principle, given n pigeons and m holes, if $n > m$, it is guaranteed that at least one hole will have two pigeons in it.

When attempting to perform large scale alias resolution, for example across one million addresses, the limited 16-bit IPID counter space will result in many addresses will producing a similar IPID value. Applying the pigeonhole principle and using the relatively small IPID space can lead to collisions, thus causing potential false positives during large scale alias resolution [41].

Another complication of using the IPv4 IPID counter to infer aliases is raised from the birthday paradox. As the number of target interfaces exceeds the 65,536 possible IPID values, the chances that two interfaces will have the same IPID value increases greatly [41].

Additionally, RadarGun attempts to identify aliases by comparing the distance of slopes, based on a threshold between the IPID time series of two interfaces, to determine whether the interfaces share a common counter [40]. Due to the approximation characteristics of this method and the complications caused by the pigeon hole principle, birthday paradox, and low entropy of IPID velocity, false alias positives can exist for any chosen threshold.

Monotonic ID-Based Alias Resolution (MIDAR)

Monotonic ID-Based Alias Resolution (MIDAR) improves upon RadarGun by introducing monotonic bounds testing and comparing velocities in which given IPID counters increase to determine whether two given addresses use a shared counter [41]. This technique reduced the

number of false positives compared to the RadarGun implementation. Despite the improvements introduced in MIDAR, the technique does not solve the IPv6 alias resolution problem as the IPv6 fragment counter does not have a natural velocity, as observations will show in Chapter 4.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3: Methodology

Fingerprinting is a method to uniquely identify a device, or class of devices, based on externally observable common features and characteristics. Fingerprints may exist at many granularities. *Operating System (OS) fingerprinting* uses variations in the implementation of standards and guidance, e.g., the recommended default TTL value is 64 [42] however, the default initial TTL value can vary based on the OS type and version. *Physical device fingerprinting* uses variations unique to a physical machine in order to create a signature that can reliably used to identify a given machine, for example identifying the skew of a given machine's internal clock to accurately identify a remote machine [43]. As another example, *wireless fingerprinting* can use a combination of variables such as clock skew, received signal strength and 802.11 sequence numbers in order to develop a fingerprint for identifying a wireless device [44].

One means of router-level alias resolution is to find a unique identifier that is common to all interfaces such that it can serve as a router fingerprint. In determining a technique to perform router-level alias resolution, inspiration was taken from prior work in IPv4 alias resolution that used fragmentation identifiers as the unique identifier to determine the relationship between router interfaces.

Two factors complicate IPv4 identifier-based alias resolution. First, the natural rate that the IPID counter increments, which is caused by both control and data plane traffic, implies that observed counters from two true aliases may have large gaps due to a large volume of IPv4 incrementing the counter. In IPv6, however, as observations will show in Chapter 4, the IPv6 fragment identification counter does not increment at the same rate as the IPv4 identifier due to routers rarely being the source of fragmented traffic. Second, the 16-bit IPv4 identifier space is small relative to the number of possible Internet router interface aliases, yielding false positives. The 32-bit IPv6 fragment identifier serves a similar purpose as the IPv4 identifier, but the expanded identifier space can provide a more reliable identifier minimizing the likelihood of false positives caused by counter rollover. The increased IPv6 fragment identifier counter space limits problems that may be caused by the pigeonhole principle as discussed in § 2.2.3.

This chapter describes a fingerprinting-based IPv6 alias resolution technique used to induce fragments from a remote router that will be used in determining the relationship between frag-

ment identifiers and the interfaces with which the fragment identifiers are associated. Next, the controlled testing environment is described to show that the technique does not produce false positives, in contrast to existing IPv4 alias resolution approaches. Finally, this chapter describes the algorithm developed to perform IPv6 alias resolution and the controlled testing used to verify the accuracy of this alias resolution technique.

3.1 Eliciting Fragmented Responses

IPv6 does not permit in-network fragmentation, and the IPv6 header does not include any identifier field akin to IPv4. Instead, IPv6 requires that all fragmentation be completed by the source node. If a router's forwarding table entry for a packet is via an interface with a MTU smaller than the size of the packet, the router drops the packet and sends an ICMPv6 "packet too big" message to the source of the packet [32]. Since the MTU field value in the ICMPv6 packet too big message is less than the existing *path* MTU (PMTU), the router updates the PMTU with the specified value. It is then the responsibility of the end-host to maintain state ([45] states the node must maintain state for a minimum of five minutes, although it is recommended that state be maintained for a minimum of ten minutes), typically in the destination cache, of the path MTU feasible for a particular destination. This end-host then sends packets smaller than the PMTU, or can fragment large packets by using the IPv6 fragment header.

This research provides a technique, TBT, that forces a remote router to originate fragmented packets. Figure 3.1 is a timing diagram of TBT between a prober and an IPv6 interface. The prober first sends a 1300 byte ICMPv6 echo request, i.e., a ping request, to a candidate IPv6 router interface. The ping request is 1300 bytes as this is small enough to pass most tunnels, but larger than the IPv6 minimum MTU of 1280 bytes. The prober receives an 1300 byte ICMPv6 echo response and then sends an ICMPv6 packet too big message with its own source IPv6 address destined for the IPv6 router interface under test. The ICMPv6 packet too big message contains an MTU of 1280 along with the first 1184 bytes of the original ICMPv6 echo request sent from the prober. (RFC 4443 [32] states that an ICMPv6 packet too big message should include "as much of the invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU."). This "false" too big message sent by the prober mimics a PMTU constraint coming from a router along the reverse path from the interface to our prober. While the prober's source IPv6 address is used for the ICMPv6 packet too big message rather than an intermediate router, test results show that the receiving router is indifferent.

Test results also show that the candidate routers update the PMTU for an interface regardless

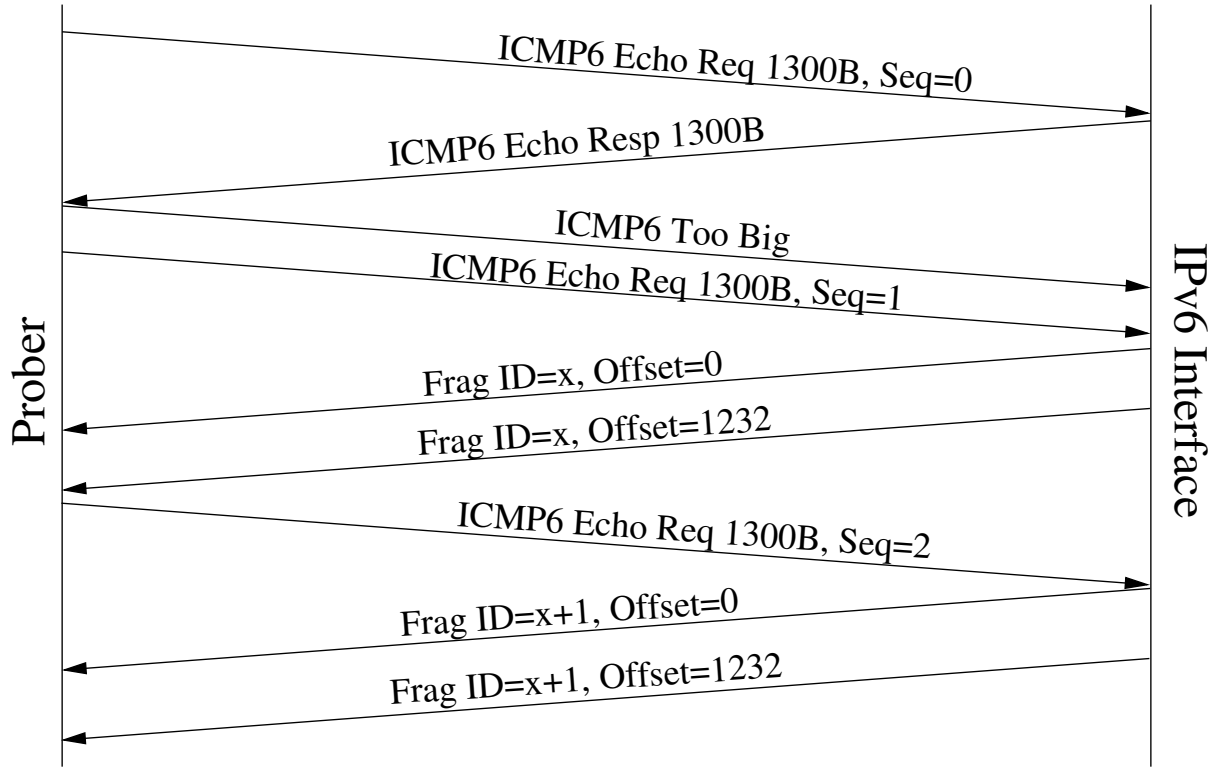

Figure 3.1: TBT, the "Too-Big Trick:" A prober spoofs an ICMPv6 too big message such that subsequent large ping responses are fragmented.

of the contents or size of the invoking packet included in the ICMPv6 packet too big message. However, beds the question of whether or not a stateful node will reject an ICMPv6 packet too big message if the contents of the invoking packet do not match a previously sent message. To prevent such occurrences, the TBT technique includes the exact contents, up to 1184 bytes, of the initial ICMPv6 echo request.

Following the ICMPv6 packet too big message, a series of 1300 byte ICMPv6 echo requests are sent to the candidate router interface. These ICMPv6 echo requests arrive at the router interface without fragmentation, but the router forwarding table now has a cached PMTU of 1280 bytes for packets destined to the prober. Each received ping causes the router to send two fragments per ICMPv6 echo reply, each with the same fragment identifier, but different offsets. As we will show next (§3.2), popular commercial routers use a common counter for the fragment identifier, regardless of the physical interface. Further, §4.1 shows that this counter frequently is monotonic and sequential.

A natural question is whether the ICMPv6 packet too big message is required. The prober could

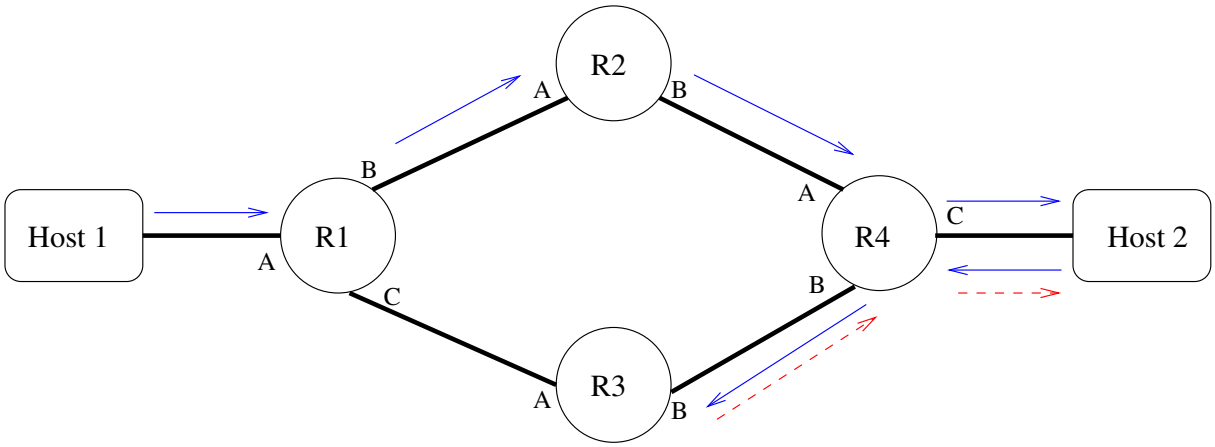

Figure 3.2: GNS3 Test Topology with Asymmetric MTU Paths Inducing ICMPv6 Packet Too Big.

instead send a larger than typical MTU echo request packet, e.g., 2000 bytes. Once received and reassembled, the remote router should respond in-kind with a 2000 byte reply that would be fragmented. Thus, the echo packets would be fragmented in both the forward and reverse direction. However, as we find in our real-world testing in §4.1, such fragmented requests are frequently either blocked or not processed by the receiving router. Using TBT results in $\approx 6\%$ more interfaces successfully identified than when sending large request packets. This is most likely due to destination hosts only being required to accept fragments with a reassembled size of 1500 bytes [9].

3.2 Ground-Truth Testing

This section will describe the efforts to develop, test and validate the functionality and accuracy of TBT. Graphical Network Simulator (GNS3) [46] was used to build virtual test topologies of routers and virtual hosts, while Internet ground-truth testing was conducted on three different sets of routers for which ground-truth was known.

3.2.1 Virtual Topologies

TBT emulates a normal operational mode whereby the forward path from the prober to an interface can carry full 1500 byte packets, while the reverse path is asymmetric and has a smaller, 1280 byte MTU. To understand the behavior of commercial routers in such situations, we implement the topology of Figure 3.2 in GNS3. In this test, static IPv6 routes pin traffic from Host 1 to Host 2 to take the upper path from $R1 \rightarrow R2 \rightarrow R4$. Reverse traffic from Host 2 to Host 1 is statically configured to take $R4 \rightarrow R3 \rightarrow R1$. We set the MTU of all links to 1500 bytes, except for the $R1C \leftrightarrow R3A$ link which is set to 1280 bytes.

A secondary goal of creating the topology in GNS3 was to verify that the composition of an ICMPv6 Packet Too Big message was in accordance with RFC 1981 [45] to craft a correct and realistic packet too big message. Once each static route was verified, the Maximum Transmission Unit (MTU) value of the link between $R1$ and $R4$ was reduced to the minimum IPv6 MTU value of 1280 [9]. A 1400 byte ICMPv6 echo request was sent from $Host1$ to $Host2$ with the goal eliciting an ICMPv6 echo reply that would exceed the MTU threshold, thereby forcing $R4$ to send an ICMPv6 Packet Too Big message to $Host2$.

Following the successful host-based experiment, a 1400 byte ICMPv6 echo request was sent from $Host1$ to $R3$'s inbound interface to force $R3$, now the source of the 1400-byte ICMPv6 echo reply, to elicit a ICMPv6 Packet Too Big message resulting in $R3$ storing the learned PMTU of 1280 bytes in its destination cache.

The attempt to force the router to automatically fragment all traffic exceeding the specified 1280 byte PMTU was successful based on verifying the source address of the ICMPv6 echo reply as $R3$'s inbound interface. Additionally, the resulting fragments with an initial fragment identifier value of one, with subsequent fragment identifier values that were sequential. Additional tests conducted for each router resulted in sequential fragment identifier values for subsequent fragmented traffic. Finally, the test sequence was repeated with $Host2$ sending a series of 1400 byte ping requests to each router producing the identical results. As a result of rebooting the routers, the fragment identifier counter was reset resulting in an initial fragment identifier value of one.

Identifying the behavior of the fragment identifier field provided a common identifier that could be used in the alias resolution algorithm. Given the findings from the fragmentation test, a similar experiment was conducted to determine if a similar result would occur against an end-host. To facilitate sending ICMPv6 Packet Too Big messages to any host on any path, we reset the virtual topology to its original configuration with all MTU values a standard Ethernet MTU of 1500 bytes [47], static routes were removed restoring routing in all directions.

Following the reconfiguration of the virtual topology, traffic flow was verified to be unrestricted by sending 1400 byte ICMPv6 echo requests to each interface while monitoring the traffic with Wireshark. An ICMPv6 Packet Too Big message was sent with a MTU value of 1280 bytes from $Host1$ to $Host2$, observing delivery to $Host2$ as expected. The expected response was that $Host2$ would drop the packet and update its destination cache (or routing table), as previously observed in routers. Following the ICMPv6 Packet Too Big message, a series of 1400 byte ICMPv6 echo requests were sent from $Host1$ to $Host2$ from which $Host2$ generated

fragmented ICMPv6 echo replies, confirming that this technique can work against end-hosts.

3.2.2 End-host testing

End-host probing was also conducted against various OSs to determine if the observed behavior of the host changed based on the OS. Virtual hosts were configured using Virtualbox and were then connected to the virtual topology to accommodate TBT testing. Each end-host was sent an initial ICMPv6 echo request to ensure the end-host was responsive. Then an ICMPv6 packet too big message was sent followed by a series of 1300 byte ICMPv6 echo requests to elicit fragmented packets to verify the behavior of the fragment identification counter. Once the TBT testing completed, the end-host was restarted and another round of TBT testing was completed, for a total of three rounds per end-host. This additional test was to performed to determine the behavior of the initial fragment identifier following a restart of the OS. The expected result from restarting the end-host was that the end-host fragment identifier counter would be reset to its initial state, e.g., for Windows systems, the fragment identifier counter would reset to one, or zero for Windows 7, regardless of how many fragments were sent prior to the restart. Additionally, for Linux systems, the initial fragment identifier would result in a different random value, and subsequent fragment identifiers would be sequential.

Another observation is that the initial fragment identifiers could also be used as proxy for remotely measuring the uptime of a router or end-host. Since the fragment counters are only incremented when the router or end-host is the source of fragmented traffic, a fragment identifier value of one would lead to the conclusion that the router or end-host has not sourced any fragmented traffic. However, if the destination is known to have sourced prior fragmented traffic, one would conclude that the router or end-host has recently been restarted or has not been the source of any fragmented traffic since last restarted.

3.3 IPv6 Alias Resolution Algorithm

Given the success in the controlled test environment, we develop an IPv6 alias resolution algorithm. There are two points of note:

- First, as we will detail in §4.1, more than 28% of live Internet interfaces we probed had sequential identifiers that start at either zero or one. In other words, prior to our probing these routers had sourced no fragmented IPv6 traffic. The presence of multiple interfaces starting with a low fragment identifier, such as zero or one, could lead to false positives during alias resolution if the algorithm fails to adequately cause non-aliases

Algorithm 1 $v6aliases(A,B)$: Determine whether A and B are IPv6 aliases

```
    send(A, TooBig)
 2: send(B, TooBig)
    for i in range(5) do
 4:     ID[0] ← echo(A)
        ID[1] ← echo(B)
 6:     if (ID[0]+1) ≠ ID[1] then
            return False
 8:     ID[2] ← echo(A)
        if (ID[1]+1) ≠ ID[2] then
10:         return False
    return True
```

with relatively close fragment identifiers to diverge. For example, given two interfaces A, with an initial fragment identifier of one, and B, with an initial fragment identifier of three. If each interface is probed twice in the pattern A, A, B, B, the result would be 1, 2, 3, 4. Based on sequential identifiers, A and B would appear to be aliases, when in fact they are not. Additionally, the algorithm must sufficiently prevent two non-aliases from converging due to a probing pattern. Therefore the alias algorithm must be careful to avoid false positives.

- Second, because the counter only increases when sending fragmented IPv6 traffic, which is a rare event, we can reasonably expect, in the absence of our probing, the counter to remain static.

Algorithm 1 provides the alias resolution pseudocode [48]. To determine whether two IPv6 addresses (A and B) are aliases, an initial echo request probe is sent to each destination, then the fake ICMPv6 packet too big messages are sent. Next, a probe is sent to A. Once the fragment ID from A is received, B is probed (each step proceeds synchronously; therefore, no race condition exists). The fragment identifiers from A and B are compared. If at anytime the fragment identifiers are not sequential, the test returns false to avoid generating needless traffic. Note that when performing $O(n^2)$ alias comparisons between all pairs of discovered interfaces, the common case will be a true negative where our algorithm quickly exits, thus the algorithm has a small constant factor. Only if the fragment identifiers are sequential are further probes sent to ensure no false positives. Based on the above observations, we ensure that, in each round of execution through the for loop, address A is probed a different number of times than B to avoid potential counter synchronization issues in the case that the addresses are not true aliases.

3.4 Controlled Alias Resolution

Controlled alias resolution testing was conducted on three commercial grade topologies, for which ground-truth was known; the first, comprised entirely of Cisco routers, the second, consisting of all Juniper routers [49], and the third, containing a combination of Cisco and Juniper routers. As results will show in Chapter 4, each topology contained routers that responded to TBT, however, as discussed in §3.1, algorithm 1 testing is conducted on routers with sequential counters as the algorithm is based on analyzing fragment identifiers that increment sequentially.

Isolating Candidate Interfaces

Selection of candidate IPv6 router interfaces consisted of two phases: an initial phase to determine the responsiveness of each interface to TBT, and a second phase for isolating interfaces with sequential fragment identifiers from randomized fragment identifiers. As Chapter 4 will show, Juniper routers produce randomized fragment identifiers that can lead to false-negative alias resolution results. As to not skew the results, interfaces that produce randomized fragment identifiers were eliminated from ground-truth testing.

The isolation process consisted of sending an initial ICMPv6 echo request to a candidate interface to verify that the interface was responsive. If the interface was not responsive to the ICMPv6 echo request, the interface was removed from subsequent testing. If the interface responded to the ICMPv6 echo request, an ICMPv6 packet too big message was sent to the candidate interface followed by a series of ten 1300 byte ICMPv6 echo requests.

Each ICMPv6 echo request was sent in succession, waiting either for an ICMPv6 echo reply or a timeout condition before sending the next ICMPv6 echo request. Once the series of ICMPv6 echo requests were finished, the returned fragment identifiers were analyzed to determine whether the series was sequential or randomized, with the interfaces with sequential fragment identifiers being grouped into candidate interfaces for subsequent alias resolution testing.

Alias Resolution of Candidate Interfaces

Following successful isolation of sequential fragment identifier producing interfaces, full-scale alias resolution was conducted. Alias resolution was executed for each interface against every other interface, then compared the results against the ground-truth list of true aliases. Alias resolution results for each set of candidate interfaces were analyzed to verify accuracy of both alias and non-alias identification. This analysis was performed to validate the accuracy of algorithm 1 and to determine the presence of false positives and negatives.

CHAPTER 4: Analysis

This chapter considers the real-world efficacy of TBT by performing Internet-wide probing. The performance of TBT is measured in terms of the fraction of Internet infrastructure responsive to the technique, as well as by testing against a subset of the topology for which ground truth is known.

Two traceroute datasets were used to determine candidate IPv6 router interfaces. The first dataset included 23,892 distinct IPv6 interfaces discovered via traceroutes from 33 vantage points belonging to a commercial Content Distribution Network (CDN) to approximately 12,300 destinations. Interestingly, nine link-local (`fe80::/10`) interfaces were found, suggesting that these non-public IPv6 addresses are being used for a small number of public links. The second data set was from the Cooperative Association for Internet Data Analysis (CAIDA) [50] with 38,300 distinct IPv6 interfaces, 25,174 of which were not also present in the CDN trace. For those traces that complete, the last hop IPv6 address of the target is ignored so as to only include router interfaces. Thus, a total of ≈49k distinct live Internet IPv6 router interfaces were probed.

It is important to note that the router interfaces we test are widely disbursed, and belong to multiple networks, spanning a total of 2,617 autonomous systems. The largest number of interfaces advertised by a single Autonomous System (AS) is 2,014 (ASN 3356, Level 3), and the median number of interfaces per AS is three. The CDN trace was collected on May 3 and 23, 2012, while the CAIDA traces were collected in August, 2012. Interfaces derived from the CDN trace were actively probed on August 28, 2012 , while the CAIDA interfaces were probed on August 29, 2012.

4.1 Efficacy of TBT

The goal of this research is two-fold, determine: i) *how many* live IPv6 interfaces respond to TBT; and ii) *in what way* these interfaces respond. All testing is performed from a single IPv6 vantage point. For each interface, a 1300 byte ICMPv6 echo request is first sent in order to determine if the interface is operational and responding to pings. If an interface is responsive to ICMPv6 ping requests, TBT is then used to send the ICMPv6 packet too big message that causes the router to update the interface's PMTU destination cache corresponding to the vantage point. Finally, ten 1300 byte ICMPv6 echo requests were sent to induce fragmented replies, from

Table 4.1: TBT Response Characteristics

	CDN		CAIDA	
ICMPv6 responsive	18486/23892	77.4%	18959/25174	75.3%
Post-TBT unresponsive	235/18486	1.3%	66/18959	0.4%
Post-TBT nofrags	5519/18486	29.9%	5800/18959	30.6%
TBT responsive	12732/18486	68.9%	13093/18959	69.1%
TBT sequential	8288/12732	65.1%	9183/13093	70.1%
TBT sequential (1)	3455/12732	27.1%	3496/13093	26.7%
TBT random	4320/12732	33.9%	3789/13093	28.9%

which the fragment identifier for each given response was captured to disk for analysis. The packet monitor, running on the same machine as the prober, did not experience any packet loss.

Table 4.1 summarizes the responsiveness of the sample of Internet IPv6 interfaces to TBT. 18,486 of 23,892 (77.4%) and 18,959 of 25,174 (75.3%) interfaces respectively were observed responding to "normal" ICMPv6 pings. The unresponsive interfaces may be due to router behavior, the delay between obtaining interfaces and subsequent probing, or ICMPv6 filtering. As these interfaces cannot be expected to respond to TBT, unresponsive interfaces are exclude from further analysis. Of the interfaces responding to the initial echo request, $\approx 70\%$ returned fragmented echo replies after a ICMPv6 packet too big was sent to the interface. Thus, this technique works for a significant fraction of Internet IPv6 routers probed.

Three primary conditions result from sending the TBT: subsequent ping responses from the router are sent fragmented, subsequent ping responses are sent unfragmented, or the router stops responding to ping requests. Approximately 29% of the interfaces probed continued to send unfragmented responses after sending TBT. This result could be caused by filtering of ICMPv6 Packet Too Big messages either by the destination router or by firewalls located in the path between the prober and destination. Additional probing using TBT should be performed against these interfaces from multiple vantage points to determine if a change in path results in the same unfragmented responses. A small fraction, between 1.3% and 0.4%, of interfaces respond to the initial echo request, but then cease to respond to subsequent echo requests following the TBT for a few minutes. We conjecture that these behaviors are due to paths that filter fragments or ICMPv6 packet too-big messages, routers incorrectly implementing IPv6, or other security measures.

Next, the sequence of returned fragment identifiers are characterized. Recall that ten ICMPv6 echo requests are sent after the TBT, therefore resulting in ten responses where each response consists of two fragmented packets, i.e., 20 total packets with identifiers. As shown in Table 4.1,

≥65% of interfaces that respond to TBT return sequential identifiers, e.g., `120, 121, ..., 130`. However, as many as 34% return random identifiers, a behavior consistent with BSD systems and BSD-based routers [51]. While TBT works for these interfaces, in so far as it induces routers to send fragmented responses, it does not admit a fingerprint for alias resolution. However, if the destination cache was maintained globally for all interfaces by a router, instead of per-interface, it would provide a useful fingerprint for alias resolution. For example, given two addresses A and B, after sending a Packet Too Big message to interface A, probes sent to A and B, which didn't receive the Packet Too Big message, would both result in fragmented packets of the PMTU size. Additionally A and B could then be confirmed to be aliases by sending a second Packet Too Big message to B with a different MTU value. If subsequent probes sent to both A and B result in fragmented packets that are the size of the new PMTU size, A and B are aliases, otherwise if the fragmented packet sizes are different, then A and B can be confirmed as non-aliases.

An interesting characteristic of those interfaces with sequential identifiers is that a significant fraction (27.1% and 26.7% respectively) had an initial fragment identifier of one. This suggests that, in the uptime of the router, it had sent zero fragmented IPv6 packets prior to this probing. As discussed in Chapter 3, non-alias interfaces that begin with correlated counters are taken into account; the algorithm advances the fragment counters at different rates to prevent false positives.

To understand the initial values of fragment counters in the wild, Figures 4.1 and 4.2 are histograms of initial fragment identifiers that occur with at least a 0.3% frequency. We see that one is the most common initial identifier for every sequence echoed and that all common identifiers are less than 50.

While this research presents and validates a new technique for IPv6 alias resolution, large-scale alias resolution on the IPv6 Internet is left for future work. However, it was observed that the second most common initial fragment identifier within a returned identifier sequence is 11, while there are modes at 21, 31, and 41. These modes are due to the probing naturally encountering aliases. Since each interface is probed 10 times, if an alias is later probed, the counter will have advanced to 10 and thus the new round of probing expects to receive an initial fragment identifier of 11.

Finally, a natural question is whether routers can be induced to send fragmented responses without TBT. Instead, large ICMPv6 echo requests are sent to an IPv6 router interface which are

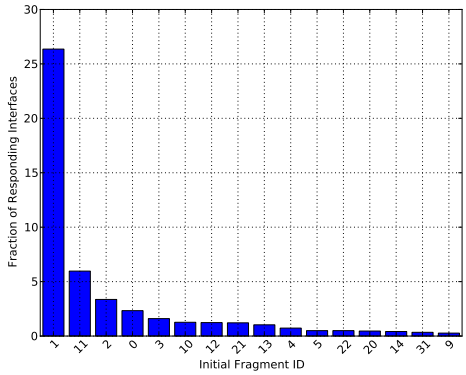

Figure 4.1: Histogram of IPv6 Fragment Identifiers Occurring $\geq 0.3\%$

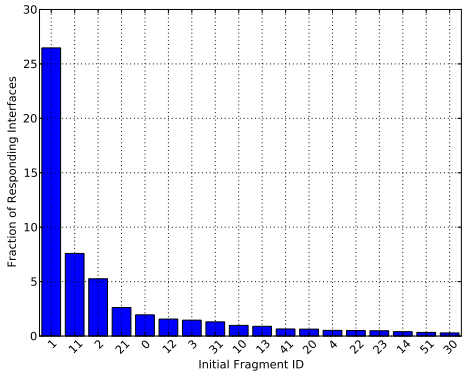

Figure 4.2: Histogram of IPv6 Fragment Identifiers Occurring $\geq 0.3\%$

themselves fragmented, such that the receiving IPv6 router interface must reassemble the fragments to respond, and then send a fragmented response. Again the two datasets of IPv6 interfaces are probed and find that this method results in 64.2% and 65.1% of interfaces successfully responding. However, using TBT results in over 5% more responses, which can equate to significantly more absolute interfaces. More importantly, sending large, fragmented probes results in much more traffic whereas the TBT technique is more efficient. Additionally, the increased traffic consumes available bandwidth, resulting in potential performance degradation during alias resolution testing. Measurement studies, including alias resolution solutions, should seek to minimize the impact on the networks being tested. For these reasons, this research focuses on TBT for alias resolution.

4.1.1 End-host Results

End hosts may have multiple IPv6 addresses assigned by their Internet Service Provider (ISP), so it is useful to determine if end host's interfaces respond to TBT in as similar fashion as router interfaces respond, and whether or not TBT correctly identifies aliases for end host interfaces. Various OSs were examined to identify how a given OS responds to TBT. Table 4.2 summarizes an end host's response to TBT based on the type of OS.

As Table 4.2 will show, all of the tested Linux OSs respond to TBT with a random initial fragment identifier followed by subsequent sequential fragment identifiers. An exception occurs when probing an Berkeley Software Distribution (BSD)-based OS, resulting in randomized fragment identifiers for both initial and subsequent fragments. Although the results from BSD-based systems support the explanation that Juniper routers respond to TBT with random

fragment identifiers, since Juniper OSs consist of a BSD underlying OS, this work was not able to confirm this hypothesis on Juniper OS images either using GNS3 or a lab test environment. Such testing is left for future work to confirm Juniper OS router behavior.

Windows-based OSs however, behave similar to Cisco in that the majority of Windows OSs respond to TBT with an initial fragment identifier of one, followed by sequential subsequent fragment identifiers. The exception in Windows systems is Windows 7, which responds with an initial fragment identifier of zero, followed by subsequent fragment identifiers being received in multiples of two (i.e., 0, 2, 4, 6, . . .). This behavior was unexpected, as no such behavior was observed by any other platform, and should be explored in future work to determine possible reasons of the observed results.

Table 4.2: Most operating systems provide a reliable identifier that can serve as a fingerprint for alias resolution.

Operating System	Initial Fragment ID	Subsequent Fragment IDs
Ubuntu	Random	Sequential
Fedora	Random	Sequential
FreeBSD	Random	Random
OpenSUSE	Random	Sequential
Windows XP	1	Sequential
Windows 2003 Server	1	Sequential
Windows 7	0	2,4,6,8,. . .

4.1.2 Accuracy of TBT Alias Resolution

Imperative to understanding the performance of the TBT alias resolution technique is having known ground-truth. In this subsection the inference accuracy of TBT is tested on both a virtual network topology in GNS3 [46], as well as on a small subset of the live IPv6 Internet for which ground-truth is available.

First, a virtual network topology is constructed in GNS3 [46] consisting of 26 Cisco routers, each containing up to four interfaces. Using the TBT tool, and algorithm as implemented in the publicly available ScaPy tool [48], a complete test is run comparing each interface to every other interface in the topology, i.e., the $O(n^2)$ all pairs testing that would be performed in the wild. All discovered aliases and non-aliases are then compared against known ground truth. The test results provide a count of identified aliases and identified non-aliases.

This controlled test results in 92 of 92 alias matches and 1584 of 1584 non-alias matches for a total accuracy of 100% with perfect precision and recall. The results, although constrained

by the virtual topology and simulation available in GNS3, help validate the ability of TBT in identifying IPv6 aliases and non-aliases.

Next, a list of IPv6 interfaces from eight physical production routers of a commercial IPv6 service provider was obtained. This small ground-truth dataset included 72 interfaces with each router having between 2 and 21 interfaces. Using TBT, pair wise alias resolution, i.e., each interface is compared against every other interface, was performed to determine the accuracy of TBT against IPv6 Internet production routers. TBT correctly identified 808 of 808 true alias pairs with zero false positives.

CHAPTER 5: Conclusions

This research developed and tested a new method for IPv6 alias resolution. TBT elicits a fragment identifier fingerprint from a significant fraction of production Internet IPv6 router interfaces. Further, this work has demonstrated an IPv6 alias resolution algorithm that is highly accurate among networks for which ground truth is known.

As IPv6 grows and gains importance, understanding its router-level topology and relationship to the IPv4 topology is increasingly important. In particular, this research examines how TBT can aid in developing router-level IPv6 topology maps. Comparing these topology maps to those previously inferred will yield valuable insights into the structure of the IPv6 network, and determine how it differs from the IPv4 topology. Additionally, topology comparisons can provide a way to examine how the IPv6 network evolves as its adoption grow.

5.1 Limitations

Although this research showed that *approximately* 70% of IPv6 router interfaces on the Internet respond to TBT, Cisco and Juniper routers were the only routers in which ground-truth was known. TBT correctly identifies IPv6 aliases for routers that use a sequential fragment counter; however an extensive list of fragment counter behavior for all router manufacturers and models was not determined. An understanding of fragment counter behaviors based on router manufacturer and model can provide a better understanding of observed router behavior and allow researchers to better utilize existing and future measurement techniques.

All Internet-based tests were conducted from a single vantage point. Although the test results were promising, *approximately* 30% of the probed interfaces either did not respond following the TBT or responded with unfragmented traffic. With only a single vantage point, this research was unable to determine the cause of these failures.

5.2 Future Work

This section serves as a consolidated list for future research efforts using TBT in IPv6 alias resolution.

5.2.1 Use of Multiple Vantage Points

To understand instances where TBT fails, future efforts will use multiple vantage points to help distinguish between path and host filtering of fragments, ICMPv6 Packet Too Big messages, or ICMPv6 as a whole. By using multiple vantage points, probes can be sent via different path, to better understand the variety of behaviors observed in Table 4.1. The results could explain the cause of some router interfaces to respond to TBT with non-fragmented responses. Additionally, testing of different routers models, both in hardware and within GNS3 could be used to better understand the variety of behaviors observed in Table 4.1.

5.2.2 Scalability and Efficiency

Throughout this research, scalability and efficiency were kept in mind. As the longevity of previous IPv6 techniques come into question due to security concerns of source routing, so does the longevity of TBT due to its inefficient algorithm. As IPv6 is more widely adopted, the number of interfaces will increase and so will the time and resources required to perform alias resolution. Currently, a complete test using TBT requires comparing each interface to every other interface in the topology, i.e., the $O(n^2)$ all pairs testing that would be performed in the wild, and verify the results against known truth. Given the performance of TBT against the test virtual topology in GNS3, to perform pair-wise alias resolution for the 49,000 IPv6 interfaces discovered in this research, it would take more than three thousand years. This provides an unacceptable method to be used for large scale IPv6 alias resolution. Therefore, an optimized algorithm is needed to improve efficiency and scalability of TBT.

5.2.3 Internet-wide Alias Resolution

In addition to the development of a more efficient algorithm, Internet-wide IPv6 alias resolution should be performed. An important step is making the algorithm robust to packet loss, or another TBT-like process causing the fragment counter to increase. Currently, algorithm 1 does not account for packet loss nor does it account for any other measurements or traffic that could increment the fragment identifier counter.

Accounting for Packet Loss

TBT does not account for packet loss as it waits a given period for a fragmented response before timing out and continuing with additional probing. An improvement to TBT could include sending an additional probe to the target, then verifying if the counter is sequential after accounting for the lost packet, such that if the last fragment identifier received was x, the expected fragment identifier following the lost packet would be $x+2$, then return to sequential checking.

Accounting for Additional Fragmented Traffic

TBT was not designed to account for the generation of additional fragmented traffic as it was observed that no such traffic had been generated by the IPv6 routers tested during this research. Due to this observation, algorithm 1 was designed only to account for sequential fragment identifiers. TBT will require a more robust algorithm that can account for additional fragmented traffic that might cause the fragment identifier counter to increase. This could be accomplished by introducing a threshold to account for such variations.

5.2.4 Effects of Fragment Counter Rollover in IPv6 Alias Resolution

IPv4 alias resolution techniques can suffer from the pigeonhole principle when using the IPID counter due the small size of counter space relative to the volume of traffic incrementing the counter. To determine the potential for encountering errors due to the pigeonhole principle, a 2^{32} Fragment counter rollover test should be conducted to understand the likelihood of encountering such errors during Internet-wide alias resolution.

5.2.5 Comparison with Current Techniques

To determine how TBT can enhance IPv6 alias resolution and topology measurement research, both ground-truth and Internet-wide comparison studies should be conducted. For instances where ground-truth is not known, Domain Name Service (DNS) could be used to verify alias resolution results. Although DNS may not provide a complete picture, it may however provide additional weight to alias inferences. This can be accomplished by comparing IPv4 A [52] and IPv6 AAAA [53] resource records to confirm the existence of inferred aliases. These tests can help determine the added benefits of TBT and how TBT complements existing alias resolution techniques. The findings from comparison studies may provide insight on improvements to TBT and existing alias resolution techniques.

5.2.6 Fingerprinting Congruent IPv4 and IPv6 Interfaces

Finally, the ability to identify dual stacked interfaces (i.e., a *single* interface may have *both* IPv4 and IPv6 addresses assigned) is crucial to the identification and protection of critical infrastructure and recognizing potential impacts of attacks. On dual stacked links the potential exists for an attack utilizing IPv4 to impact the performance of IPv6 as well as an IPv6 attack to affect IPv4 performance. For example, a DoS attack carried out on an IPv4 network where traffic flows over a dual stacked link may result in the degradation of IPv6 data flow as well due to both IPv4 and IPv6 data flows being on the same transmission medium (Ethernet, fiber optics, etc.). Future research should seek to understand the relationships between IPv4 and IPv6 and attempt to determine congruent IPv4/IPv6 fingerprinting techniques to aid in the production of an accurate topology of the Internet.

5.3 IPv6 Load Balancing

Early work was accomplished to determine what fields are used in IPv6 per-flow load balancing, a virtual test topology was designed and implemented using **gns3!** (**gns3!**) [46]. The virtual topology consisted of routers loaded with Cisco Internetwork Operating System (IOS) images and Lubuntu virtual hosts, integrated using VirtualBox. Wireshark was used to monitor traffic behavior within the virtual test topology. Each router interface was configured with dual stacked IPv4/IPv6 addresses. The dual stacked configuration provided to facilitate the verification of known IPv4 load balancing to verify the proper function and results of running a traceroute. Following the verification of excepted IPv4 load balancing behavior, the goal was to determine what fields of the IPv6 header were used in per-flow load balancing.

Motivated by prior IPv4 load balancing efforts [39], experiments were conducted to determine what effects changes to the IPv6 header had on IPv6 load balancing. Each experiment consisted of manually altering the values within the ranges of each IPv6 header field. The header fields were altered one field at a time, using the ScaPy packet manipulation library [54]. Three different protocols (ICMP, TCP and UDP) were used to send test probes across load balanced path and the routes were verified using Wireshark packet captures on each incoming virtual router interface.

Test results revealed that per-flow load balancing used the source and destination address, source and destination port, and protocol fields of the IPv6 header shown in Figure 2.2. The tests confirmed the expected results that per-flow load balancing would be accomplished using a seven-tuple consisting of source and destination addresses and ports, protocol, router ID and

ingress interface [55]. The results provided the ability to maintain a consistent path from a source to the destination for the duration of a set of test probes.

By understanding the load balancing schemes of IPv6 routers, test probes could be manipulated to travel along different paths to potentially identify possible alias candidates for future testing. With the ability to manipulate the path that a test probe traveled, the next step was to verify the process of fragmenting IPv6 traffic once an ICMPv6 Packet Too Big message was received by an end-host. Future work should determine the applicability of using load balancing in alias resolution testing using TBT, i.e., as an alternative solution to testing the 29/interfaces that responded with unfragmented traffic.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] L. Smith and I. Lipner, "The IANA IPv4 address free pool is now depleted," Feb. 2011. [Online]. Available: http://www.nro.net/news/ipv4-free-pool-depleted

[2] Hurricane Electric, "Hurricane Electric IPv4 exhaustion counters," 2013. [Online]. Available: http://ipv6.he.net/statistics

[3] G. Huston, "IPv4 address report," 2013. [Online]. Available: http://www.potaroo.net/tools/ipv4/index.html

[4] R. Miller, "IPv4 addresses now driving hosting deals," 2012. [Online]. Available: http://www.datacenterknowledge.com/archives/2012/07/16/ipv4-addresses-now-driving-hosting-deals

[5] C. Marsan, "Sales of unused IPv4 addresses gathering steam," 2012. [Online]. Available: http://www.networkworld.com/news/2012/052412-ipv4-resales-259588.html

[6] V. Fuller and T. Li, "Classless Inter-domain Routing (CIDR): the Internet address assignment and aggregation plan," RFC 4632, Internet Engineering Task Force, Aug. 2006.

[7] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, Internet Engineering Task Force, Jan. 2001.

[8] T. Hain, "Architectural implications of NAT," RFC 2993, Internet Engineering Task Force, Nov. 2000.

[9] S. Deering and R. Hinden, "Internet Protocol, version 6 (IPv6) specification," RFC 2460 (Draft Standard), Internet Engineering Task Force, Dec. 1998.

[10] K. Claffy, "Tracking IPv6 evolution: data we have and data we need," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 3, pp. 43–48, Jul. 2011.

[11] N. Sarrar, G. Maier, B. Ager, R. Sommer, and S. Uhlig, "Investigating IPv6 traffic: what happened at the world IPv6 day?" in *Proceedings of the 13th International Conference on Passive and Active Measurement*, 2012.

[12] R. Mohan, "Will U.S. government directives spur IPv6 adoption?" Sep. 2010. [Online]. Available: http://www.circleid.com/posts/20100929_will_us_government_directives_spur_ipv6_adoption/

[13] Google, Inc., "IPv6 adoption," 2013. [Online]. Available: http://www.google.com/intl/en/ipv6/statistics.html

[14] G. Huston, "IPv6 BGP statistics," 2012. [Online]. Available: http://bgp.potaroo.net/v6/as2.0/

[15] RIPE-NCC, "IPv6 enabled networks," 2012. [Online]. Available: http://v6asns.ripe.net/v/6

[16] 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 Internet Measurement Conference*, 2012, pp. 537–550.

[17] S. Zander, L. L. Andrew, G. Armitage, G. Huston, and G. Michaelson, "Mitigating sampling error when measuring Internet client IPv6 capabilities," in *Proceedings of the 2012 ACM Internet Measurement Conference*, 2012, pp. 87–100.

[18] K. Keys, "Internet-scale IP alias resolution techniques," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 50–55, Jan. 2010.

[19] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: from art to science," in *Conference For Homeland Security*, March 2009.

[20] 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, 2009.

[21] V. Kundra, "Transition to IPv6," 2010. [Online]. Available: http://www.whitehouse.gov/sites/default/files/omb/assets/egov_docs/transition-to-ipv6.pdf

[22] J. Brzozowski, "IPv6 home networking pilot market deployment technical details," Apr. 2012. [Online]. Available: http://corporate.comcast.com/comcast-voices/ipv6-deployment-technology-2

[23] S. Amante, B. Carpenter, S. Jiang, and J. Rajahalme, "IPv6 flow label specifications," RFC 6437, Internet Engineering Task Force, Nov. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6437.txt

[24] "Protocol numbers," Protocol Numbers, Internet Assigned Numbers Authority, Oct. 12012. [Online]. Available: http://www.iana.org/assignments/protocol-numbers/protocol-numbers.txt

[25] Cisco Systems, "Cisco IOS IPv6 command reference," 2011. [Online]. Available: http://www.cisco.com/en/US/docs/ios/ipv6/command/reference/ipv6_cr_book.pdf

[26] Juniper, "IPv6 hop-limit." [Online]. Available: http://www.juniper.net/techpubs/software/erx/junose700/swcmdref-a-m/html/i-commands285.html

[27] Microsoft, "TCP/IPv6 configurable registry settings." [Online]. Available: http://msdn.microsoft.com/en-us/library/aa915728.aspx

[28] S. Hogg and E. Vyncke, *IPv6 Security*. Cisco Systems, 2009.

[29] S. Qian, Y. Wang, and K. Xu, "Utilizing destination options header to resolve IPv6 alias resolution," in *GLOBECOM*, Dec. 2010, pp. 1–6.

[30] I. U. of Southern California, "Internet Protocol," RFC 791, Internet Engineering Task Force, Sep. 1981.

[31] P. Biondi and A. Ebalard, "IPv6 routing header security," 2007.

[32] A. Conta, S. Deering, and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol version 6 specification," RFC 4443, Internet Engineering Task Force, Mar. 2006.

[33] E. N. T. Narten and W. Simpson, "Internet Protocol, version 6 (IPv6) specification," RFC 2461, Internet Engineering Task Force, Dec. 1998.

[34] J. Postel, "Internet Control Message Protocol," RFC 792, Internet Engineering Task Force, Sep. 1981.

[35] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," *SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 133–145, August 2002.

[36] D. G. Waddington, F. Chang, R. Viswanathan, and B. Yao, "Topology discovery for public IPv6 networks," *SIGCOMM Comput. Commun. Rev.*, vol. 33, pp. 59–68, July 2003.

[37] S. Qian, M. Xu, Z. Qiao, and K. Xu, "Route positional method for IPv6 alias resolution," in *Computer Communications and Networks (ICCCN)*, Aug. 2010.

[38] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, mar 2000, pp. 1371–1380 vol.3.

[39] 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*, ser. IMC '06. New York, NY, USA: ACM, 2006, pp. 153–158. [Online]. Available: http://doi.acm.org/10.1145/1177080.1177100

[40] A. Bender, R. Sherwood, and N. Spring, "Fixing Ally's growing pains with velocity modeling," in *Proceedings of the 8th Internet Measurement Conference*, 2008.

[41] K. Keys, Y. Hyun, M. Luckie, and K. Claffy, "Internet-scale IPv4 alias resolution with MIDAR," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, p. 1, 2012.

[42] A. Malis, C. Graff, D. Estrin, D. Farinacci, G. Finn, and J. Postel, "IP option numbers," nov 2012. [Online]. Available: http://www.iana.org/assignments/ip-parameters/ip-parameters.xml

[43] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," in *Proceedings of IEEE Security and Privacy*, 2005, pp. 211–225. [Online]. Available: http://dx.doi.org/10.1109/SP.2005.18

[44] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee, "Identifying unique devices through wireless fingerprinting," in *Proceedings of the first ACM Conference on Wireless Network Security*, ser. WiSec '08. New York, NY, USA: ACM, 2008, pp. 46–55. [Online]. Available: http://doi.acm.org/10.1145/1352533.1352542

[45] J. McCann, S. Deering, and J. Mogul, "Path MTU discovery for IP version 6," RFC 1981, Internet Engineering Task Force, Aug. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1981.txt

[46] J. Grossman, B. Marsili, C. Goudjil, and A. Eromenko, "GNS3 graphical network simulator," 2012. [Online]. Available: http://www.gns3.net/

[47] C. Hornig, "A standard for the transmission of IP datagrams over Ethernet networks," RFC 894, Internet Engineering Task Force, Apr. 1984.

[48] W. Brinkmeyer and R. Beverly, "Too-Big Trick Python prototype," 2012. [Online]. Available: http://www.cmand.org/tbt/

[49] "Internet2 network NOC–visible network," 2012. [Online]. Available: http://noc.net.internet2.edu/i2network/live-network-status/visible-network.html

[50] "The CAIDA UCSD IPv6 topology dataset," 2012. [Online]. Available: http://www.caida.org/data/active/ipv6_allpref_topology_dataset.xml

[51] M. J. Silbersack, "Improving TCP/IP security through randomization without sacrificing interoperability," in *Proceedings of BSDCan*, 2006.

[52] P. Mockapetris, "Domain names - concepts and facilities," RFC 1034, Internet Engineering Task Force, Nov. 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1034.txt

[53] V. Jacobson, R. Braden, and D. Borman, "IPv6 DNS extensions," RFC 1886, Internet Engineering Task Force, Dec. 1995. [Online]. Available: http://www.ietf.org/rfc/rfc1886.txt

[54] P. Biondi, "ScaPy." [Online]. Available: http://www.secdev.org/projects/scapy/

[55] Cisco Systems, "How does load balancing work?" 2005. [Online]. Available: http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094820.shtml

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