# IPv6 Alias Resolution via Induced Fragmentation

Robert Beverly[1], William Brinkmeyer[1], Matthew Luckie[2],
and Justin P. Rohrer[1]

[1] Naval Postgraduate School, Monterey, CA
[2] CAIDA, University of California, San Diego, CA
{rbeverly,wdbrinkm,jprohrer}@nps.edu, mjl@caida.org

**Abstract.** Discovering router-level IPv6 topologies is important to understanding IPv6 growth, structure, and evolution and relation to IPv4. This work presents a fingerprint-based IPv6 alias resolution technique that induces fragmented responses from IPv6 router interfaces. We leverage the way in which IPv6 implements fragmentation to provide reliable inferences. We demonstrate perfect alias resolution accuracy in a controlled environment, and on a small subset of the production IPv6 Internet for which we have ground-truth. Internet-wide testing finds that over 70% of IPv6 interfaces probed respond to the test. Our promising results suggest a valuable technique to aid IPv6 topology discovery.

## 1 Introduction

IPv6, standardized nearly 15 years ago [6] as the successor to Internet Protocol version 4 (IPv4), is experiencing commercial deployment – primarily due to economic and business constraints, rather than any technical impetus [4]. Modern systems and hardware support IPv6, service and content providers are deploying IPv6 [17], and government networks are mandating IPv6 [13].

The number of global IPv6 BGP routing prefixes is growing exponentially [11]. More than 6,000 autonomous systems, approximately 15%, now announce IPv6 reachability [16]. Amid IPv6 measurement efforts underway [7] [22], understanding the evolution of the IPv6 router-level topology is an ongoing challenge.

This paper investigates IPv6 *alias resolution* – the process of determining if two IP addresses are assigned to different interfaces of the same physical router [12]. Alias resolution reduces an interface-level graph, e.g. discovered via active probing, into a router-level graph [3], thereby permitting a better understanding of the resilience and robustness properties of the network [21].

Taking inspiration from prior IPv4 alias resolution work, we present a fingerprint based IPv6 alias resolution technique that relies on eliciting *fragmented* responses from IPv6 router interfaces. Although IPv6 has no in-network fragmentation, sources can send large IPv6 packets in fragments. We find 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, 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, our IPv6 technique is highly accurate because control-plane messages are rarely fragmented.

This paper seeks to detail and validate a new IPv6 alias resolution algorithm; we leave Internet-wide alias resolution, scaling, and comparison against other techniques to future work. We make four primary contributions:

1. Development of a fingerprint-based IPv6 alias resolution technique[3].
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 where we discover approximately 70% respond to our test.
4. Validation of the technique on a small subset of the production IPv6 network for which we have alias ground-truth, where we obtain perfect accuracy.

## 2 Related Work

Significant prior research investigates IPv4 alias resolution; see [12] and [9] for an overview of major techniques. Design differences between IPv4 and IPv6 obsolete some techniques used in IPv4, while enabling new ones. For instance, the elimination of in-network fragmentation and the simplification of the IPv6 header prevents the trivial reapplication of IPv4 techniques that utilize the IPID field [19]. Alias resolution through IPv6 source-routing has been explored in Atlas [20], RPM [15], and the "option header method" [14]. Given a potential alias pair $(x, y)$, Atlas performs a 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 extension header before checking the hop limit. If $x$ and $y$ are aliases, "hop limit exceeded" and "port unreachable" ICMP6 messages are generated.

RPM finds that the source address of "hop limit exceeded" ICMP6 messages for packets that are not destined to the router at which the expiration occurs is frequently the ingress address. 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 ICMP6 "hop limit exceeded" messages originating from aliases of $y$. The option header method makes use of the fact that setting an invalid bit sequence in the IPv6 options header will generate an ICMP6 "parameter problem" message, originating from the ingress interface of the packet generating the response. By probing via multiple intermediate routers (similar to RPM), multiple aliases of the target address may be discovered. Our alias resolution method is distinct from those listed here in that it does not depend on IPv6 source routing and therefore is not defeated on hosts where source routing is disabled due to security concerns, as has become the norm in IPv4.

Lastly, the THC IPv6 [10] toolkit employs false ICMP6 packet too big messages (discussed next) as part of its attack suite. However, the tool's goal is to maliciously reduce the MTU of a target rather than to resolve IPv6 aliases.

[3] A freely licensed prototype Python implementation is available from: [2].

# 3 Methodology

Our technique is fingerprint-based: we require some identifier or signature that is both common to all interfaces on an IPv6 router, and is unique across routers such that we do not make false inferences. Further, it must be possible for a remote probing host to obtain the identifier without any privileged access.

We take inspiration from prior work in IPv4 alias resolution that relies on fragment identifiers [19]. The IPv4 header contains a 16-bit identifier that is used by an end-host receiving fragmented packets such that it can reconstruct the original packet. Prior research [19] has shown that packets originated by IPv4 routers often use a common counter, irrespective of physical interface, for the identifier field. Since this counter increases sequentially, it is possible to infer whether two interfaces are aliases by querying the router, e.g. via ping.

Two factors complicate IPv4 identifier-based alias resolution. First, the natural rate of counter increase as the router sends other control plane traffic implies that observed counters from two true aliases may have large gaps. Second, the 16-bit identifier space is small relative to the number of possible Internet router interface aliases, yielding false positives.

This section describes our IPv6 alias resolution technique and how we induce a remote router to send fragmented packets. We then describe our controlled environment for ground-truth testing where we show that our technique does not suffer from the false positive problems inherent in similar IPv4 approaches.

## 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. However, IPv6 supports end-host fragmentation. If a router's forwarding table entry for a packet is via an interface with a Maximum Transmission Unit (MTU) smaller than the size of the packet, the router drops the packet and sends an ICMP6 "packet too big" message to the source of the packet [5]. It is then the responsibility of the end-host to maintain state, typically in the destination cache, of the *path* MTU (PMTU) feasible for a particular destination. The host then sends packets smaller than the PMTU, or can fragment large packets by using the IPv6 fragment header [6].

Our approach, which we term the "Too-Big Trick" (TBT) induces a remote router to originate fragmented packets. Figure 1 is a timing diagram of TBT between a prober and an IPv6 interface. The prober first sends a 1300 byte ICMP6 echo request to a candidate IPv6 interface. The request is 1300 bytes – larger than the IPv6 minimum MTU of 1280 bytes, but small enough to pass most tunnels. The prober receives an 1300 byte ICMP6 echo response and then sends an ICMP6 packet too big message with its own source IPv6 address to the interface under test, and includes an MTU of 1280 along with the first 1184 bytes of the original ICMP6 echo request ([5] states that the packet-too-big message include "as much of the invoking packet as possible without the ICMP6 packet exceeding the minimum IPv6 MTU."). This "false" too big message mimics a PMTU constraint coming from a router along the reverse path from the interface to our prober. While we use the prober's source IPv6 address for the too big message rather than an intermediate router, the receiving router is indifferent.

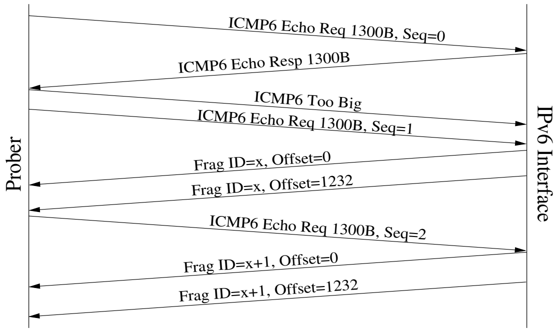


**Fig. 1.** TBT, the "Too-Big Trick:" A prober spoofs an ICMP6 too big message such that subsequent large ping responses are fragmented.

We then send a series of 1300 byte ICMP6 echo requests. These arrive at the interface without fragmentation, but the end IPv6 stack now has a cached PMTU of 1280 for packets destined to the prober. Each ping causes the router to send two fragments, 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, in §4 we show that this counter frequently is monotonic and sequential.

A natural question is whether the ICMP6 too big packet is required. The prober could 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, 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, most likely due to destination hosts only being required to accept fragments with a reassembled size of 1500 bytes [6].

### 3.2 Ground-Truth Testing

To develop, test, and validate TBT, we employ the Graphical Network Simulator (GNS3) [8] to build virtual test topologies of routers and virtual hosts.

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 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.

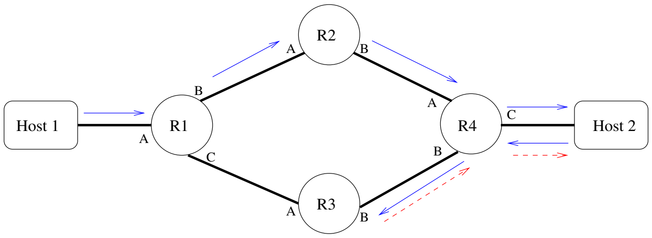


**Fig. 2.** GNS3 Test topology with asymmetric MTU paths inducing ICMP6 too big.

A 1300 byte ICMP6 ping request from Host 1 to Host 2 induces a 1300 byte ping response (blue arrows). However, $R3$ sends an ICMP6 too big message to Host 2 (red arrows). Host 1 receives no reply to this first ping since the packet is dropped at $R3$. Host 2 records a new PMTU for traffic destined to Host 1 and maintains soft-state resulting in Host 2 fragmenting future responses to Host 1.

Next, we send 1300 byte ICMP6 ping requests from Host 1 to the router interface $R4A$. $R4$ receives the ICMP6 packet too big message from $R3$ upon sending the ping response to Host 1, and $R4$ updates its destination cache PMTU value. We observe that subsequent pings to $R4A$ results in fragments with sequential identifiers, with the first identifier after router boot being 1.

We then send ICMP6 ping requests to $R4B$ and $R4C$. Critically, we observe that identifiers come from a common counter, i.e. the fragment identifier is one more than the last identifier received from the other interface. Specifically, for a large ICMP6 echo request to $R4A$ that returns fragments with identifier $x$, a subsequent probe to $R4B$ returns $x+1$, and a third probe to $R4C$ returns $x+2$. Probing $R4A$ again returns $x+3$. Thus, with the Cisco images we test, these routers use a *fragment identifier counter that is common across interfaces.*

Based on these findings, we reset all links to the standard Ethernet 1500 byte MTU. Here we seek to determine whether we can masquerade as an in-path router instructing the probed router to update its PMTU for traffic sent to Host 1. We first verify that large 1300 byte echo requests traverse the network to and from the target without fragmentation. We then repeat testing, but send an ICMP6 too big message with Host 1's source IPv6 address to the target. We verify that the ICMP6 too big message arrives at Host 2 and that Host 2 fragments subsequent echo replies, confirming that our technique is indeed able to induce remote interfaces to send fragmented traffic.

Lastly, we find that while the routers use a common fragment counter, the destination cache appears to be per-interface. After sending an ICMP6 too big message from host 1 to e.g. $R4A$, a large probe to $R4C$ does not return fragmented responses to host 1. In our testing, the ICMP6 too big message must be sent to each interface to reliably induce fragmentation.

### 3.3 IPv6 Alias Resolution Algorithm

Given the success in the controlled test environment, we develop an IPv6 alias resolution algorithm. There are several points of note. First, as we will detail in §4, more than 28% of live Internet interfaces we probed had sequential identifiers

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

```
$send(A, TooBig)$
$send(B, TooBig)$
for $i$ in range(5) do
    ID[0] $\leftarrow echo(A)$
    ID[1] $\leftarrow echo(B)$
    if (ID[0]+1) $\neq$ ID[1] then
        return $False$
    ID[2] $\leftarrow echo(A)$
    if (ID[1]+1) $\neq$ ID[2] then
        return $False$
return $True$
```

that start at either zero or one. In other words, prior to our probing these routers had sourced no fragmented IPv6 traffic. 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 [2]. To determine whether two IPv6 addresses ($A$ and $B$) are aliases, an initial echo request probe is sent to each destination, then the fake ICMP6 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; no race condition exists). The fragment identifiers from $A$ and $B$ are compared. If at anytime the fragment IDs 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. Only if the fragment IDs 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.

## 4 Results

To understand the real-world efficacy of our technique, we perform Internet-wide probing. For candidate IPv6 router interfaces, we utilize two traceroute datasets. The first dataset includes 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, we find nine link-local (`fe80::/10`) addresses included in this dataset, suggesting that these non-public IPv6 addresses are being used for a small number of public links. The second data set is from CAIDA [1] with 38,300 distinct IPv6 interfaces, 25,174 of which are not present in the CDN trace. For those traces that complete, we ignore the last hop IPv6 address of the target so as to only use router interfaces.

**Table 1.** TBT Response Characteristics

| | **CDN** | | **CAIDA** | |
|---|---|---|---|---|
| ICMP6 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% |

Thus, we probe a total of $\approx$ 49k distinct live Internet IPv6 router interfaces, belonging to networks advertised by 2,617 different autonomous systems. The largest number of interfaces belonging to a single AS is 2,014 (from ASN 3356, Level 3), and the median number of interfaces per AS is 3. The CDN trace was collected on May 3 and 23, 2012, while the CAIDA traces were collected in August, 2012. We actively probed interfaces derived from the CDN trace on August 28, 2012, while the CAIDA interfaces were probed on August 29, 2012.

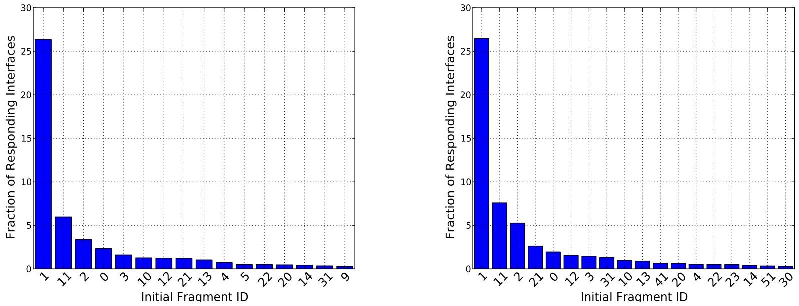
### 4.1 Efficacy of TBT

Our goal is two-fold, determine: i) *how many* live IPv6 interfaces respond to TBT; and ii) *in what way* these interfaces respond. We perform all testing from a single IPv6 vantage point. For each interface, we first send a 1300 byte ICMP6 echo request in order to determine if the interface is live and responding to pings. We then use TBT to send the ICMP6 message too big that will update the interface's PMTU to our vantage point. Finally, we send ten 1300 byte ICMP6 echo requests. Contemporaneous to our probing, we capture all IPv6 packets to disk for analysis; our packet monitor did not experience any packet loss.

Table 1 summarizes the responsiveness of our sample of Internet interfaces to TBT. We observe 18,486 of 23,892 (77.4%) and 18,959 of 25,174 (75.3%) interfaces respectively responding to "normal" ICMP6 pings. The unresponsive interfaces may be due to router behavior, or ICMP6 filtering. As these interfaces cannot be expected to respond to TBT, we exclude them from our analysis. Of the interfaces responding to our initial echo request, we find $\approx$ 70% returning fragmented echo replies after we send a packet too big to the interface. Thus, our technique works for a significant fraction of Internet IPv6 routers we probe.

Three primary conditions result from sending the TBT: subsequent ping responses are sent fragmented, subsequent ping responses are sent unfragmented, or the router stops responding to ping requests. We observe approximately 29% of the interfaces we probe continuing to send unfragmented responses after we send TBT. Between 1.3 and 0.4 percent of interfaces respond to the initial echo request, but then respond to no subsequent echo requests after the packet too big for a few minutes. We conjecture that these behaviors are due to paths that filter fragments or ICMP6 too-big messages, routers incorrectly implementing IPv6, or security measures. In future work we plan to more precisely understand the root causes of such non-responsive behavior.

Next, we characterize the sequence of returned fragment identifiers. Recall that we send ten ICMP6 echo requests after the TBT, therefore we expect to



(a) CDN

(b) CAIDA

**Fig. 3.** Histogram of IPv6 Fragment Identifiers Occurring $\geq 0.3\%$

receive ten responses where each response consists of two fragmented packets, i.e. 20 total packets with identifiers. As shown in Table 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 [18]. While TBT works for these interfaces, it does not admit a fingerprint for alias resolution.

An interesting characteristic of those interfaces with sequential identifiers is that a significant fraction (27.1% and 26.7% respectively) had an initial identifier of one. This suggests that, in the uptime of the router, it had sent no fragmented IPv6 packets prior to our probing. As discussed in §3, we take into account non-alias interfaces that begin with correlated counters; our algorithm advances them at different rates to prevent false positives.

To understand the initial values of fragment counters in the wild, Figures 3(a) and 3(b) 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 paper presents and validates a new technique for IPv6 alias resolution, we leave large-scale alias resolution on the IPv6 Internet for future work. However, we observe that the second most common initial identifier within a returned identifier sequence is 11, while there are modes at 21, 31, and 41. These modes are due to our probing naturally encountering aliases. Since we probe each interface 10 times, if we happen to later probe an alias, the counter will have advanced to 10 and we expect to receive 11.

Finally, a natural question is whether we can induce routers to send fragmented responses without TBT. Instead, we experiment with sending large ICMP6 echo requests that are themselves fragmented, such that the receiving IPv6 router interface must reassemble the fragments to respond, and then send a fragmented response. We again probe our two datasets of IPv6 interfaces 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 our technique is more efficient. For these reasons, we focus on TBT for alias resolution.

### 4.2 Accuracy of TBT Alias Resolution

Imperative to understanding the performance of our TBT alias resolution technique is having known ground-truth. In this subsection we test the inference accuracy of our tool on both a virtual network topology in GNS3 [8], as well as on a small subset of the live IPv6 Internet for which we have ground-truth.

First, we construct a virtual network topology in GNS3 [8] consisting of 26 Cisco routers, each containing up to four interfaces. Using our TBT tool, and Algorithm 1 as implemented in our publicly available ScaPy tool [2], we run a complete test 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. The test results provide a count of identified aliases and identified non-aliases. This controlled test results in 92/92 alias matches and 1584/1584 non-alias matches for a total accuracy of 100 percent with perfect precision and recall. The results, although constrained by the virtual topology and simulation available in GNS3, help validate the ability of our tool in identifying IPv6 aliases and non-aliases.

Finally, we obtain a list of IPv6 interfaces from eight physical production routers of a commercial IPv6 service provider. This small ground-truth dataset includes 72 interfaces with each router having between 2 and 21 interfaces. Using TBT we correctly identify 808/808 true alias pairs with no false positives. Given this encouraging result, we plan more extensive probing in the future.

## 5 Conclusion

This research develops and tests a new method for IPv6 alias resolution. Our technique, the “Too-Big Trick” (TBT), elicits a fragment identifier fingerprint from a significant fraction of production IPv6 router interfaces. We demonstrate that our alias resolution algorithm, a prototype of which is publicly available, is highly accurate among networks for which we have ground truth.

To understand instances where TBT fails, we plan to use multiple vantage points to help distinguish between path and host filtering of fragments. We plan to test additional routers, both in hardware and within GNS3 to better understand the variety of behaviors we observe in Table 1.

We leave to future work the task of leveraging TBT to perform Internet-wide IPv6 alias resolution. An important step is making the algorithm robust to packet loss, or another TBT-like process causing the fragment counter to increase. Toward this goal, we are investigating sequential hypothesis detection to provide a bounded confidence in the alias pair. Further, at scale, we must modify the algorithm to be more intelligent than pair-wise resolution.

As IPv6 grows and gains importance, understanding its router-level topology and relationship to the IPv4 topology is increasingly important. In particular, our current research examines how TBT compares with and compliments existing resolution schemes, while generating router-level IPv6 topologies. Comparing these topologies to those previously inferred will yield valuable insights into the structure of the IPv6 network, and how it differs from the IPv4 topology.

## Acknowledgments

We thank Arthur Berger and Geoff Xie for invaluable early feedback, and Ítalo Cunha for shepherding. Special thanks to Aaron Hughes and 6connect for operational support and insight. This work supported by collaborative NSF grant CNS-1111445 and CNS-1111449. Views and conclusions are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government.

## References

1. The CAIDA UCSD IPv6 Topology Dataset (2012), `http://www.caida.org/data/active/ipv6_allpref_topology_dataset.xml`
2. Brinkmeyer, W.: Too-Big Trick prototype (2012), `http://www.cmand.org/tbt/`
3. Claffy, K., Hyun, Y., Keys, K., Fomenkov, M., Krioukov, D.: Internet mapping: From art to science. In: Conference For Homeland Security (March 2009)
4. claffy, k.: Tracking IPv6 evolution: data we have and data we need. SIGCOMM Comput. Commun. Rev. 41(3), 43–48 (Jul 2011)
5. Conta, A., Deering, S., Gupta, M.: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 Specification. RFC 4443 (Mar 2006)
6. Deering, S., Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard) (Dec 1998)
7. Dhamdhere, A., Luckie, M., Huffaker, B., claffy, k., Elmokashfi, A., Aben, E.: Measuring the deployment of ipv6: topology, routing and performance. In: Proceedings of the 2012 ACM Internet measurement conference. pp. 537–550 (2012)
8. Grossman, J., Marsili, B., Goudjil, C., Eromenko, A.: GNS3 Graphical Network Simulator (2012), `http://www.gns3.net/`
9. Gunes, M.H., Sarac, K.: Resolving ip aliases in building traceroute-based internet maps. IEEE/ACM Trans. Netw. 17, 1738–1751 (December 2009)
10. Heuse, M.: THC-IPv6 tool suite (2012), `http://www.thc.org/thc-ipv6/`
11. Huston, G.: IPv6 BGP Statistics (2012), `http://bgp.potaroo.net/v6/as2.0/`
12. Keys, K.: Internet-scale IP alias resolution techniques. SIGCOMM Comput. Commun. Rev. 40, 50–55 (Jan 2010)
13. Mohan, R.: Will U.S. Government Directives Spur IPv6 Adoption? (Sep 2010)
14. Qian, S., Wang, Y., Xu, K.: Utilizing Destination Options Header to Resolve IPv6 Alias Resolution. In: GLOBECOM. pp. 1 –6 (Dec 2010)
15. Qian, S., Xu, M., Qiao, Z., Xu, K.: Route Positional Method for IPv6 Alias Resolution. In: Computer Communications and Networks (ICCCN) (Aug 2010)
16. RIPE-NCC: IPv6 Enabled Networks (2012), `http://v6asns.ripe.net/v/6`
17. Sarrar, N., Maier, G., Ager, B., Sommer, R., Uhlig, S.: Investigating IPv6 traffic: what happened at the world IPv6 day? In: Proceedings of PAM (2012)
18. Silbersack, M.J.: Improving TCP/IP security through randomization without sacrificing interoperability. In: Proceedings of BSDCan (2006)
19. Spring, N., Mahajan, R., Wetherall, D.: Measuring ISP topologies with rocketfuel. SIGCOMM Comput. Commun. Rev. 32, 133–145 (August 2002)
20. Waddington, D.G., Chang, F., Viswanathan, R., Yao, B.: Topology discovery for public IPv6 networks. SIGCOMM Comput. Commun. Rev. 33, 59–68 (July 2003)
21. Willinger, W., Alderson, D., Doyle, J.C.: Mathematics and the internet: A source of enormous confusion and great potential. Notices of the AMS 56(5) (2009)
22. Zander, S., Andrew, L.L., Armitage, G., Huston, G., Michaelson, G.: Mitigating sampling error when measuring internet client ipv6 capabilities. In: Proceedings of the 2012 ACM Internet measurement conference. pp. 87–100 (2012)