Performance Implications of Unilateral Enabling of IPv6

Hussein A. Alzoubi[1], Michael Rabinovich[1], and Oliver Spatscheck[2]

[1] Case Western Reserve University
[2] AT&T Research Labs

Abstract. While some IPv6-enabled Web sites such as Google require an explicit opt-in by IPv6-enabled clients before serving them over the IPv6 protocol, we quantify performance implications of *unilateral* enabling of IPv6 by a Web site. In this approach, the Web site enables dual-stack IPv4/6 support and resolves DNS queries for IPv6 addresses with the IPv6 addresses of its Web servers, and legacy DNS queries for IPv4 addresses with the IPv4 addresses. Thus, clients indicating the willingness to communicate over IPv6 are allowed to immediately do so. Although the existence of the end-to-end IPv6 path between these clients and the Web site is currently unlikely, we found no evidence of performance penalty (subject to 1sec. granularity of our measurement) for this unilateral IPv6 adoption. We hope our findings will help facilitate the IPv6 transition and prove useful to the sites considering their IPv6 migration strategy.

1 Introduction

The address space of IPv4 is practically exhausted: the last block was allocated to regional Internet registries in February 2011. While registries can still distribute their allocated addresses internally, the last allocation brought the issue of IPv6 transition into stark focus. With the revived efforts for IPv6 transition, many clients are now dual-stack, that is, are capable to using both IPv4 and v6 protocols. High profile Web sites, e.g., Google, started to likewise deploy IPv6 platforms to serve these clients [6]. However, as the overall Internet transition to IPv6 is lagging, the network paths between these clients and the Web site commonly do not support IPv6, in which case the two end-hosts cannot communicate over IPv6 even if they both are IPv6-enabled. Despite a recent recommendation on how end-hosts should handle this situation [19], in practice the lack of end-to-end IPv6 path may expose the user to excessive delays or outright connectivity disruption. The possibility of these delays can influence the Web site's IPv6 transition strategy – e.g., Google only directs clients to its IPv6 servers if they have verified the end-to-end IPv6 connectivity and explicitly opted in [6].

This paper quantifies the basis for such a conservative strategy. In other words, it asks an important question: what are the implications of a Web site *unilaterally* switching to a dual-stack mode, whereby it would simply send IPv6-enabled clients to an IPv6 server, and IPv4 clients to an IPv4 server? We found

no evidence of any performance penalty (subject to 1 sec. granularity of our measurement) and an extremely small increase in failure to download the object (from 0.0038% to 0.0064% of accesses). This suggests the feasibility of the unilateral IPv6 deployment, which could in turn spur a speedier overall IPv6 transition.

2 Background

A user access to a Web site is usually preceded with a DNS resolution of the site's domain name. An IPv6-enabled client would issue a DNS query for an IPv6 address (an AAAA-type query) while an IPv4 client would send an A-type query for an IPv4 address. Our goal is to assess the implications of a unilateral enabling of a dual-stack IPv4/6 support by the Web site. In this setup, the Web site would deploy both the IPv6 and IPv4 HTTP servers. It would then resolve AAAA DNS queries to the IPv6 address of the IPv6 server, and A-type queries to the IPv4 address of the IPv4 server. Thus, clients indicating the willingness to communicate over IPv6 are allowed to immediately do so.

The danger of this approach is that, given the current state of IPv6 adoption in the core networks, the likelihood of the valid end-to-end IPv6 path or tunnel between any host-pair is low, even if both end-points are IPv6-enabled. When the IPv6 path does not exist, plausible scenarios for IPv6-enabled clients can be grouped into two categories. In the first scenario, the client follows the recent IETF recommendation [19] to avoid any delay in attempting to use an unreachable IPv6 Web server. Basically, clients would issue both AAAA and A queries to obtain both IPv6 and IPv4 addresses, then establish both the IPv4 and IPv6 HTTP connections at the same time using both addresses; if the IPv6 connection advances through the TCP handshake, the IPv4 connection is abandoned through an RST segment. The other scenario is that the client attempts to use IPv6 first and then, after failure to connect, resorts to IPv4, which leads to a delay penalty.

The macro-effects of dual IPv4/6 Web site deployment are the result of complex interactions between behaviors of browsers, operating systems, and DNS resolvers, which differ widely, leading to drastically different delay penalties (see [5] for an excellent survey of different browser and OS behaviors). Consequently, to avoid the possibility of high delay penalty, high-profile Web sites, such as Google, only resolve AAAA DNS queries to IPv6 addresses for clients that have verified the existence of an end-to-end IPv6 path between themselves and Google and explicitly opted-in for IPv6 service. This procedure is valuable as a demonstration and testbed for IPv6 migration but it does not scale as making a client network to duplicate this procedure for every Web site is infeasible.

Note that the client typically resolves DNS queries through a client-side DNS resolver (commonly referred to as "local DNS server", or "LDNS"), which is often shared among multiple clients. It is possible that the resolver submits AAAA queries even if some (or all) of its clients are not IPv6-enabled. A Web site that unilaterally deploys IPv6 as described above has no way of knowing the status

of IPv6 support of the actual client when the AAAA query arrives - it simply responds with the IPv6 address. Our measurement methodology captures any possible effects of this uncertainty. Thus, unless it may cause confusion, we refer to all clients behind the resolver that sends AAAA queries as IPv6-enabled or, interchangeably, dual-stack.

3 Methodology

We used the following methodology to measure the performance implications of unilateral IPv6 deployment when the client cannot reach the IPv6 server due to the lack of the end-to-end IPv6 path or tunnel. We registered the domain `dns-research.com` and built a specialized DNS server to act as its authoritative DNS server (ADNS) as well as a specialized Web server to host a single object (a one-pixel image) from subdomain *sub.dns-research.com*. Assume for a moment that we can reliably associate a given DNS query with the subsequent HTTP request (we describe the approach we use to accomplish this shortly). We configured our specialized DNS server to respond to IPv6 queries (type-AAAA requests) for any hostname from domain *sub.dns-research.com* with a non-existent IPv6 address, and to any IPv4 DNS queries (type-A requests) with the valid IPv4 address of our Web server. Responding to IPv6 queries with a non-existent IPv6 address mimics the situation where there is no end-to-end IPv6 path between the client and server and thus the server is unreachable[3]. We then measure the delay penalty as the time between when we send the unreachable IPv6 address to the client's DNS resolver and when the client falls back to IPv4 (i.e., the corresponding HTTP request from the client arrives over IPv4). We deployed both our ADNS and Web servers on the same host so that we could measure time intervals between DNS and HTTP events without clock skew.

Our setup is illustrated in Figure 1. As mentioned, we need to associate a given DNS query with the subsequent HTTP request. To do so, we first associate a DNS query with the originating client using the approach from [14]. Our Web server hosts a special image URL, `dns-research.com/special.jpg`. When a user accesses this image, their browser first sends a DNS query for dns-research.com (we refer to this as a "base domain" and "base query") to the user's DNS resolver (step 1 in the figure), which then sends it to our ADNS server (step 2). An IPv6-enabled client network is likely to send both A and AAAA DNS queries. Since the base DNS queries can not be reliably associated with the clients, our ADNS responds with NXDOMAIN ("Non-Existent Domain") to the AAAA query and with the proper IPv4 address of our Web server to the A query (step 3). The resolver forwards the response to the client (step 4), which then sends the HTTP request for this image to our server (step 5). Our Web server returns an HTTP 302 ("Moved") response (step 6) redirecting the client

[3] Indeed, attempting to communicate with our non-existent address has the same effect as an attempt to communicate with an existing IPv6 destination over a non-existent path, which is the same as a path that is not end-to-end IPv6-enabled.

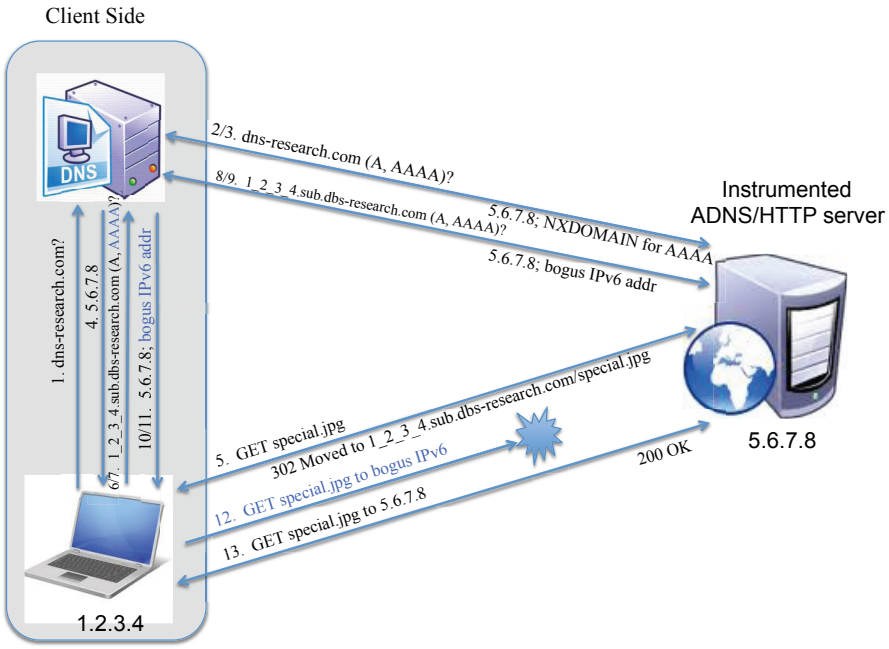

Fig. 1. Measurement Setup. Presumed interactions are marked in blue font.

to another URL in the `sub.dns-research.com` domain[4], but with the host name that embeds the client's IP address (we refer to these queries as "sub" requests). The client needs to resolve this name through its resolver again (steps 6-9). This time the DNS query can be reliably attributed to the HTTP client through the client's IP address embedded in the hostname.

Having associated the DNS query to the originating client, we measure the delay between the arrival of AAAA query in step 8 and the first subsequent HTTP request from the same client in step 13 as the delay penalty for unilateral IPv6 deployment. To eliminate HTTP requests that utilized previously cached DNS resolutions (as their time since the preceding DNS interaction would obviously not indicate the delay penalty) we measure the incidents of IPv6 delays for an HTTP request only if it was immediately preceded (i.e., without another interposed HTTP request) by a full DNS interaction, including both A and AAAA requests, for that client. We contrast these delays with the delays for non-IPv6 enabled clients, whose resolvers did not send AAAA queries. We use the same technique to associate these HTTP clients with DNS queries, and measure the delays as the time between a type-A DNS query in step 2 and the first subsequent HTTP request in step 13.

[4] In reality our setup involved more redirections to enable other measurements; we omit these details for clarity as they are unrelated to the present study.

Table 1. High-level dataset characterization

LDNS IP addreses	278,559
Client IP addresses	11,378,020
Unique Client/LDNS IP Pairs	21,432,181

We have collaborated with a major consumer-oriented Website to embed our image starting URL into their home page. Whenever a Web browser visits the home page, the browser downloads the linked image and the interactions in Figure 1 take place. We used a low 10 seconds TTL for our DNS records. This allowed us to obtain repeated measurements from the same client without overwhelming our setup. Further, our Web server adds a "cache-control:no-cache" header field to its HTTP responses to make sure we receive every request to our special image. Unfortunately, the conditions for this collaboration prevent us from releasing the datasets collected in the course of our experiment.

4 The Dataset

We have collected the DNS logs (including the timestamp of the query, LDNS IP, query type, and query string) and HTTP logs (request time, User-Agent and Host headers) resulting from the interactions described in the previous section. Our experiment lasted 28 days, from Jan 5th, 2011 to Feb 1st, during which we collected over 34.4 million DNS *"sub"* requests and around 56 million of the HTTP downloads of the final image (step 13 in Figure 1).[5]

Table 2. The basic IPv6 statistics

	Base DNS Requests	"Sub" DNS Requests
# Requests	19,945,037	2,398,367
LDNS IP addrs	59,978	32,291
Client IP addrs	No data	1,134,617

Table 1 shows high-level characteristics for our dataset. We have collected over 21M client/LDNS associations between 11.3M unique client IP addresses from 17,778 autonomous systems (ASs) and almost 280K LDNS resolvers from 14,627 ASs.

[5] While one could have expected the number of final HTTP downloads to be roughly equal to the number of "sub" DNS requests since each client access would normally generate a "sub" DNS request and one final HTTP download, the number of these HTTP downloads in our dataset is much greater than that. We verified that this is due to clients and LDNSs caching our replies to "sub" queries for much longer than our specified TTL value. These wide-spread TTL violations were first reported in [15].

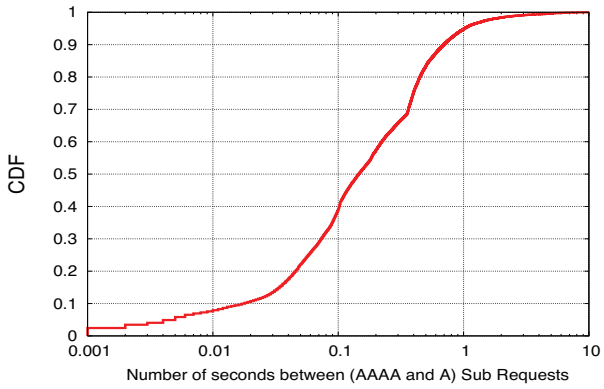

Fig. 2. Time difference between A and AAAA "sub" requests

Table 2 summarizes the general statistics about IPv6 requests, as well as clients and LDNSs behind them. Out of the 278,559 LDNSs we observed during our experiment, almost 22% were IPv6-enabled (i.e., sent some AAAA queries). However, only around 54% of the latter sent AAAA "sub" requests, and the number of "sub" requests was much lower than that of the base queries. This is because some LDNS servers seem to cache the NXDOMAIN response (which, as discussed earlier, our DNS server returns to the IPv6 queries for the base domain) and not issue queries for subdomains of the base domain, while other LDNS servers seem to not cache NXDOMAIN responses at all and send repeated base queries even when serving subsequent "sub" requests from the cache.

5 The Results

We now present our measurement results. We first consider if unilateral IPv6 enabling entails any penalty clients' DNS resolution, and then report our measurements of the overall delays.

5.1 DNS Resolution Penalty

Our first experiment investigates any potential delays in obtaining the IPv4 DNS resolution given that our IPv6 Web server is unreachable. If clients failover to IPv4 only after being unable to connect to the IPv6 Web server, then it could be that the DNS A-type query would only arrive after the corresponding timeout. To test for this behavior, we consider the time between A and AAAA "sub" request arrivals from the same client. Our immediate observation is that almost 88% out of the 2.3 Million AAAA "sub" requests were received *after* their corresponding A request. This says that not only do these clients/LDNSs perform both resolutions in parallel but, between the two wall-to-wall DNS requests, they most likely send the A query first. For the remaining 12% of requests, Figure

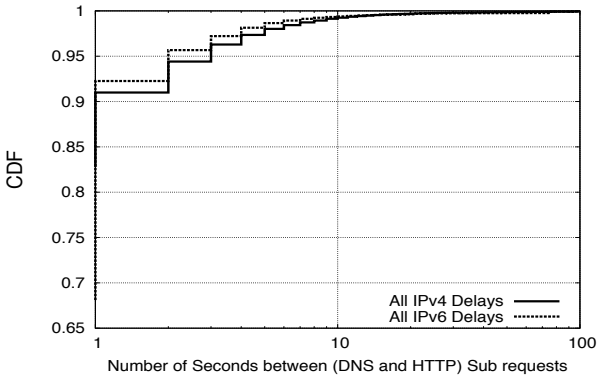

Fig. 3. Comparison of all IPv6 and IPv4 delays.

2 shows the CDF of the time difference between A and AAAA "sub" requests. The figure indicates that even among these requests, most clients did not wait for a failed attempt to contact the IPv6 Web server before obtaining the IPv4 address. Indeed, even assuming an accelerated default connection timeout used in this case by Safari and Chrome (270ms and 300ms respectively [5] - as opposed to hundreds of seconds for regular TCP timeout [1]), roughly 70% of type-A queries in these requests came within this timeout value. We conclude that a vast majority (roughly $88 + 0.7 \times 12 \approx 95\%$) of requests do not incur extra DNS resolution penalty due to IPv6 deployment.

5.2 End-to-End Penalty

Our first concern is to see whether unilateral IPv6 enabling can lead to disruption of Web accesses, that is, whether the IPv6-enabled clients successfully fail over to IPv4 for HTTP downloads. We compare the rate of interactions where HTTP request fails to arrive following the AAAA DNS query, either until the next DNS interaction from the same client or until the end of the trace. For IPv6-enabled clients, these lost HTTP requests amounted to 154 out of 2,398,367 total interactions, or 0.0064%. For IPv4-only clients, this number was 1217 lost requests out of the total (34.4M-2.4M), or 0.0038%. Although the rate of lost requests in IPv6-enabled clients is higher, both rates are so extremely low that they can both be considered insignificant.

Turning to assessing the upper bound on the overall delay for IPv6-enabled clients, we measure the time between the arrivals of the AAAA "sub" DNS request (a conservative estimate of when the client receives the unreachable IPv6 address) and the actual subsequent HTTP request by the client. As a reminder, to eliminate HTTP requests that utilized previously cached DNS resolutions, we measure the incidents of IPv6 delays for an HTTP request only if it was immediately preceded (i.e., without another interposed HTTP request) by a full DNS interaction, including both A and AAAA sub requests for that client. Apply-

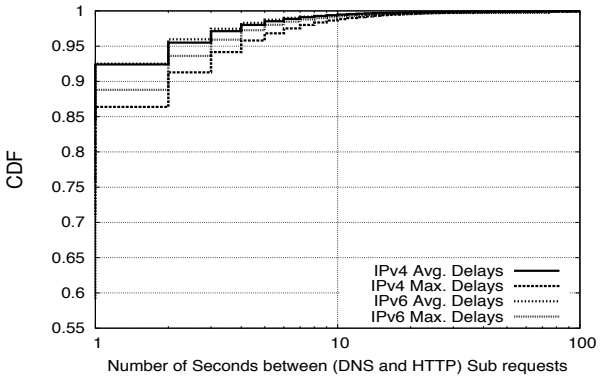

Fig. 4. IPv4 and IPv6 delays per client.

ing this condition resulted in 1,949,231 instances of IPv6 delays from 1,086,323 unique client IP addresses. Our HTTP logs provide timestamps with granularity of one second; thus we can only report our delays at this granularity.

Figures 3 and 4 compare delays incurred by IPv6-enabled and IPv4-only clients. Figure 3 shows CDFs of all delays across all clients in the respective categories (i.e., multiple delay instances from the same client are counted multiple times) and Figure 4 shows the CDFs of average and maximum delays observed per client. Both figures concentrate on delays within 100s. There were 0.063% of IPv6 delays and 0.076% of IPv4 delays exceeding 100s, with the maximum IPv6 delay of 1.2M sec and IPv4 delay of 1.8M sec. We attribute exceedingly long delays to a combination of clients commonly violating DNS time-to-live (as first observed in [15]) with corner cases such as duplicate DNS requests resulting from a single client interaction (the behavior that we directly observed in a different study). For instance, one HTTP request on January 7 was surrounded by 6 DNS queries, two of which arrived after the HTTP request; since there were no more DNS requests until the next HTTP request on January 27 (presumably due to a TTL violation), this scenario contributed a delay of 1.7M sec.

Neither figure shows significant differences in delay between the two categories of clients. In fact, where one can discern the difference, the delay distributions actually show *lower* delay penalty for IPv6-enabled clients. The maximum per-client delays shows the most discernible difference; this could be explained by the fact that there are an order of magnitude more IPv4-only interactions, thus there is a higher chance of an outlier value of maximum delay. While the one-second measurement granularity is clearly a limitation of this experiment, our study finds no evidence of delay penalty and in any case provides the upper bound of 1 sec. for any penalty that could not be measured.

6 Related Work

Much effort has been devoted to IPv6 transition. A number of transition technologies have been proposed that help construct end-to-end IPv6 paths without the need for ubiquitous deployment of IPv6 network infrastructure (see, e.g., [2, 18, 12, 4, 8]). We look at another aspect of IPv6 migration, namely, the penalty for unilateral IPv6 enabling when the end-to-end path does not exist. A number of studies have reported on the extent of IPv6 penetration from a variety of vantage points. In particular, Shen et al. [17] used netflow data from a Chinese tier-1 ISP, Savola [16] and Hei and Yamazki [7] analyzed data collected on 6to4 relays, Kreibich et al. [11] employed user-launched measurements, Malone [13] and Huston [9] studied IPv6 traffic attracted to IPv6-connected Web sites, and Karpilovsky et al. [10] considered IPv6 penetration from several vantage points including netflows in core networks, address allocations, and BGP route announcements. A general conclusion of these studies is that IPv6 deployment remains low. For example, Huston found that in 2009, end-to-end IPv6 connectivity was only available to around 1% of the clients of the two Web sites he considered. These findings motivate our study by showing that most clients receiving an IPv6 address from a unilaterally IPv6-enabled Web site would have no end-to-end IPv6 connectivity to the site.

Several studies considered the performance of the current IPv6 network infrastructure. Zhou and Van Mieghem [20] compared the end-to-end delay of IPv6 and IPv4 packets between selected end-hosts and observed that IPv6 paths had higher variation in delay. Colitti et al. [3] compared latency experienced by clients accessing the Google platform over IPv4 and IPv6 and found little difference once the effect of processing at tunnel termination points is factored out (otherwise the IPv6 latency was slightly higher). While this study considered performance of the IPv6 clients that had the end-to-end IPv6 path to their platform, we focus on the performance implications for IPv6-enabled clients that do not have this connectivity.

7 Conclusion

While many end-hosts have become IPv6-enabled, the overall IPv6 adoption in the network is lagging and thus it is common for two IPv6-enabled hosts to have no end-to-end IPv6 network path between them. Consequently, Web sites such as Google that pioneer IPv6 adoption only direct those clients to their IPv6 servers that previously verified their end-to-end IPv6 connectivity to the IPv6 servers in question and explicitly opted in. This paper studies the performance implications of a *unilateral* enabling of IPv6 by a Web site, without requiring any verification or opt-in from the clients. We found no evidence of performance penalty for such unilateral IPv6 adoption and an extremely small increase in failure to download the object (from 0.0038% to 0.0064% of accesses). While the one-second measurement granularity is clearly a limitation of our study, it in any case provides the upper bound of 1 sec. for any penalty that could not

be measured. We hope these findings will help sites as they consider their IPv6 migration strategy.

References

1. Z. Al-Qudah, M. Rabinovich, and M. Allman. Web timeouts and their implications. In *Passive and Active Measurement Conf.*, pages 211–221, 2010.
2. B. Carpenter and K. Moore. Connection of IPv6 domains via IPv4 clouds. RFC 3056, 2001.
3. L. Colitti, S. Gunderson, E. Kline, and T. Refice. Evaluating IPv6 adoption in the Internet. In *Passive and Active Measurement Conf.*, pages 141–150, 2010.
4. J. De Clercq, D. Ooms, S. Prevost, and F. Le Faucheur. Connecting IPv6 islands over IPv4 MPLS using IPv6 provider edge routers (6PE). *RFC 4798*, 2007.
5. Dual stack esotropia. http://labs.apnic.net/blabs/?p=47.
6. Google over IPv6. http://www.google.com/intl/en/ipv6/.
7. Y. Hei and K. Yamazaki. Traffic analysis and worldwide operation of open 6to4 relays for ipv6 deployment. In *IEEE Int. Symp. on Applications and the Internet*, pages 265–268, 2004.
8. C. Huitema. Teredo: Tunneling IPv6 over UDP through network address translations (NATs). *RFC 4380*, 2006.
9. G. Huston. IPv6 Transition. http://www.potaroo.net/presentations/2009-09-01-ipv6-transition.pdf, 2009. Presentation at the 3d Meeting of the Australian Network Operators Group.
10. E. Karpilovsky, A. Gerber, D. Pei, J. Rexford, and A. Shaikh. Quantifying the extent of IPv6 deployment. *Passive and Active Measurement Conf.*, pages 13–22, 2009.
11. C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: illuminating the edge network. In *The 10th ACM Conf. on Internet Measurement*, pages 246–259, 2010.
12. Y. Lee, A. Durand, J. Woodyatt, and R. Droms. Dual-Stack Lite broadband deployments following IPv4 exhaustion. *RFC 6333*, 2011.
13. D. Malone. Observations of IPv6 addresses. *Passive and Active Meas. Conf.*, pages 21–30, 2008.
14. Z. M. Mao, C. D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang. A precise and efficient evaluation of the proximity between web clients and their local DNS servers. In *USENIX Annual Technical Conference*, pages 229–242, 2002.
15. J. Pang, A. Akella, A. Shaikh, B. Krishnamurthy, and S. Seshan. On the responsiveness of DNS-based network control. In *The 4th ACM Conf. on Internet measurement*, pages 21–26, 2004.
16. Pekka Savola. Observations of IPv6 traffic on a 6to4 relay. *SIGCOMM Comput. Commun. Rev.*, 35(1):23–28, January 2005.
17. W. Shen, Y. Chen, Q. Zhang, Y. Chen, B. Deng, X. Li, and G. Lv. Observations of IPv6 traffic. In *ISECS Int. Colloq. on Computing, Communication, Control, and Management*, volume 2, pages 278–282. IEEE, 2009.
18. M. Townsley and O. Troan. IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)–Protocol Specification. *RFC 5969*, 2010.
19. D. Wing and A. Yourtchenko. Happy eyeballs: Success with dual-stack hosts. IETF draft. http://tools.ietf.org/html/draft-ietf-v6ops-happy-eyeballs-05, October 2011.
20. X. Zhou and P. Van Mieghem. Hopcount and E2E delay: IPv6 versus IPv4. *Passive and Active Measurement Conf.*, pages 345–348, 2005.