A CASE FOR FASTER MOBILE WEB IN CELLULAR IPV6 NETWORKS

Abstract

The transition to IPv6 cellular networks creates uncertainty for content providers (CPs) and content delivery networks (CDNs) of whether and how to follow suit. Do CPs that update their CDN contracts to allow IPv6 hosting achieve better, or worse performance in mobile networks? Should CDNs continue to host mobile content over IPv4 networks, or persuade to their CP customers the performance benefits of IPv6 content delivery?

In this paper we answer these questions through a comprehensive comparison of IPv4 and IPv6 mobile Web performance in cellular networks in the US from the point of view of Akamai's content delivery infrastructure. Our data show that IPv6 hosting outperforms legacy IPv4 paths in mobile Web. Our analysis leads to clear recommendations for CPs to transition to IPv6-hosted mobile Web. Finally, we propose new mechanisms, through which CDNs can safely transition mobile content to IPv6-enabled servers for improved content delivery.

Introduction

Despite many years of research to improve Web performance in mobile and wireless networks, users remain dissatisfied with lengthy webpage load times [263]. As Internet Service Providers (ISPs) upgrade their network infrastructure from IPv4 to IPv6, understanding the performance of mobile content delivery in cellular IPv6 networks is crucial. In this study, we take a novel approach to characterize the dynamically changing IPv6 ecosystem from the point of view of Akamai's content delivery infrastructure for cellular networks [224]. We argue that, unlike PlanetLab

and Amazon EC2 datacenters [2, 45], Akamai's content delivery servers are so deeply deployed inside several cellular ISPs' networks that the end-to-end communication between mobile devices and Akamai's servers need not, strictly speaking, touch the wired public Internet outside the cellular network. As a result, Akamai's unique content delivery infrastructure enables us to view the end-to-end cellular ecosystem between mobile devices and cellular gateways and evaluate how content is delivered over cellular IPv6 networks from the perspective of content providers (CPs), ISPs, and other content delivery networks (CDNs) [28, 55].

CPs, such as Facebook and others, care about the experience of users with their respective applications. To deliver application content from datacenters to users in a timely manner, CPs make contractual agreements with CDNs to ensure content has high availability, is secure, and is delivered to users through low latency connections. Some CPs sign contracts with CDNs for content delivery only over a cellular ISP's IPv4 network, while other CPs sign contracts for content delivery over IPv6 networks. Although, CPs are aware that ISPs are deploying IPv6 in their networks and CDNs are offering IPv6 hosting of mobile content, CPs remain uncertain whether upgrading to IPv6 will improve or worsen the performance of mobile content delivery.

CDNs, such as Akamai and others, care about the performance of content delivery to their mobile users. Although CDNs strive to upgrade their infrastructure to overcome IPv4 address scarcity and to make content available over IPv6 networks [34, 110, 113, 222, 262], one of their goals is to ensure that the performance of content delivery over an ISP's IPv6 network is as good as over that ISP's IPv4 network. CDNs generally act as a surrogate infrastructure for its many CPs, and for stability, reliability, and contractual implications, the configurations for each CP are often only changed with respective permission granted from the CP. Thus, adoption of new content delivery techniques, such as IPv6, is often a multi-year process as

its performance implications become better-understood in real-world conditions over time.

As cellular network operators adopt IPv6 addressing to resolve the challenges imposed by IPv4 address scarcity [113], the research community has shown interest in understanding different IPv6 deployment strategies within ISPs worldwide, and the adoption rate of IPv6 among mobile users and content providers [110,222,274]. Consequently, CPs and CDNs remain reluctant to embrace a wide-scale transition to IPv6, as they remain unaware of performance of IPv6 networks deployed by cellular carriers.

In this paper, we take a novel approach to investigate and expose performance of IPv6 and IPv4 ecosystems from a CDN's perspective – in between cellular ISPs and Content Providers. Our goal is to improve awareness within different networking communities about performance benefits (if any) of serving mobile content over IPv6, as opposed to IPv4. Our work precisely describes the current role of a CDN in connecting mobile users to content servers in the evolving cellular ecosystem in the US. To the best of our knowledge, our work is the most detailed investigation to compare mobile Web performance over IPv4 and IPv6 networks. We classify the four major contributions of this work as follows:

Dataset Richness: We conducted a large scale, comprehensive study to measure IPv6 performance in four major cellular carriers in the US to compare its native and NAT64/DSLite deployments. Using Akamai CDN infrastructure, we collected a rich dataset consisting of millions of data points of measured IPv6 and IPv4 performance, during the months of January - August in 2015.

Measurement: Our study investigates IPv6 performance across multiple factors that influence Web performance on cellular networks.

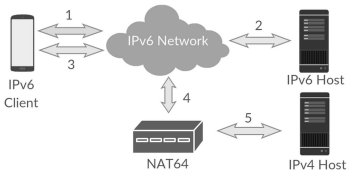

Figure 6.1: T-Mobile's IPv6 network.

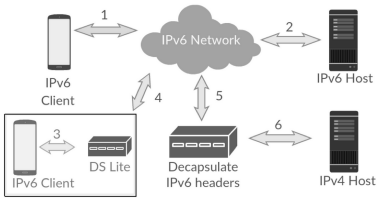

Figure 6.2: Verizon's IPv6 network.

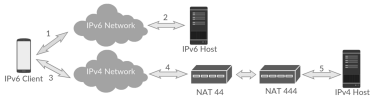

Figure 6.3: AT&T and Sprint's IPv6 net.

- We compare IPv6 and IPv4 networks through 1) round trip time between clients and CDN servers; 2) time to resolve domain names from cellular DNS; and 3) webpage load time.

- We extend Akamai's Real User Monitoring System (RUM) [41] to accurately extract Web performance metrics for mobile content hosted on IPv6-enabled content servers in US cellular networks.

Inferences Drawn: Our experience with Akamai's content delivery infrastructure shows that IPv6 networks deployed by cellular ISPs outperform their IPv4 networks.

- Our analysis includes recommendations for CPs to host mobile content on IPv6 for improved user experience.

- We also recommend that CDNs deliver mobile content over IPv6 to avoid in-path middleboxes for IP address translation deployed by cellular carriers.

- And finally, we suggest cellular network operators upgrade their network infrastructure to support IPv6, instead of continuing to deploy legacy IPv4 technologies in their network.

Problems Discovered: During our study, we discovered the following three problems related to how IPv6 content is delivered in cellular networks. We also propose several solutions we adopted to address these problems.

- We discovered that the DNS lookup process takes longer on Android devices with IPv6 capability than Android devices with IPv4-only capability. The lookup time is high because IPv6-capable devices wait for both `Type A` and `Type AAAA` (pronounced 'quad A') DNS queries to finish before establishing a TCP connection. Because of higher DNS lookup time for IPv6-capable clients, we observe that IPv6 clients in the Sprint network often experience slower webpage loads when connecting to IPv4 servers, than IPv4-only clients connecting to same IPv4 servers.

  Solution: To address this problem, we made four recommendations to the Google Android team to reduce long Round Trips Times (RTTs) to cellular resolvers. First, if the DNS lookup process on IPv6-capable Android devices could be modified to send `AAAA` and `A` DNS queries in parallel, lookup times could potentially become twice as fast. Second, we suggested that additional speedup in DNS lookup could be achieved by letting client browsers indicate to the mobile OS that they do not need to wait to get the `A` lookup back if they get an `AAAA` answer in DNS response. Third, in cases where mobile clients are on an IPv6-only network, the mobile OS could be modified to only send DNS `AAAA` queries, instead of both `AAAA` and `A`. Finally, the existence of `A` or `AAAA` only answers could be cached per-name for some time on the device, not withstanding the caching of the actual answers, which would enable the device to make a smarter query the following time that an IP address is needed.

- In the case of T-Mobile (and applicable to other major IPv6-only networks worldwide), we discovered that when IPv6-capable clients resolve an IPv4-only domain name, the cellular DNS introduces an extra round trip to the DNS Authorities. We discuss details in Section 6.

Solution: We develop and prototype OneTrip, a technique for DNS Authorities to eliminate the extra round trip in DNS lookups when resolving IPv4-only domains. Through in-lab simulations we show that DNS lookup times reduce significantly when DNS Authorities use OneTrip. We also experimentally verify that in T-Mobile's production cellular network, OneTrip maintains end-to-end connectivity.

- In a cellular network outside of the US, IPv6 packets were being routed via the US, which resulted in latency of end-to-end connections on IPv6 to be higher than IPv4 by 200 ms. While it is possible that IPv6 latency may be higher than IPv4 in some networks, we argue that it could be due to misconfigured routing policies.

  Solution: Based on our findings, Akamai's network team is actively working with that cellular carrier to resolve misconfiguration in its IPv6 routing.

The rest of the paper is organized as follows. In the next section, we offer a discussion on how IPv6 is deployed by different cellular ISPs in the US. An overview of different IPv6 deployment strategies will support our measurement techniques and research findings. In Section 6, we describe our data collection methodology. In Section 6, 6, and 6, we investigate the component differences of Web performance in IPv6 and IPv4 networks through measuring the round trip latency between clients and CDN servers, the DNS lookup time, and the webpage load time. In Section 6, we introduce OneTrip as a technique for DNS Authorities to eliminate unnecessary round trips from DNS lookups in IPv6-only networks. In Section 6, we discuss related work. Finally, we conclude in Section 6.

Overview of IPv6 Deployment in Cellular Networks

A cellular carrier that supports IPv6 addressing must provide a way for its IPv6 devices to connect with IPv4 and dual-stacked (both IPv4 and IPv6 addresses) Internet servers. Cellular network architectures are influenced by a variety of factors such as the capabilities of the existing infrastructure hardware to support IPv6 addressing, urgency to upgrade networks to IPv6, number of users with IPv6-capable devices, number of available IPv4 addresses, etc., which result in ISPs taking different approaches to IPv6 deployment [63]. In this section we provide an overview of how the different cellular carriers in the US have upgraded their IPv4 networks to provide IPv6 addressing to their users, based on the publicly available information from those carriers.

T-Mobile is an IPv6-only network for all phones with support for `464XLAT`, which includes all phones with Android version 4.3 and above [90]. For older versions of Android, as well as iPhone, Blackberry, and Windows devices, T-Mobile is an IPv4-only network. As a result, IPv6 devices in T-Mobile network always transmit IPv6 packets, whereas, IPv4 devices always transmit IPv4 packets. Thus, the choice of addressing (IPv4/IPv6) for devices is based on whether the device supports `464XLAT`.

In Figure 6.1 we depict the high level infrastructure of the IPv6 network deployed by T-Mobile. We show that when an IPv6 device communicates with an IPv6 server the packets are routed directly to the server through the IPv6 network without any NAT-based stateful middleboxes (Steps 1 and 2). However, when an IPv6 device communicates with an IPv4 server, the IPv6 packets generated by the device are routed to a stateful NAT64 middlebox (Steps 3 and 4). The NAT64 middlebox translates IPv6 packets to IPv4 address family and forwards the translated packets to the IPv4 server (Step 5). The NAT64 middlebox also converts reply IPv4

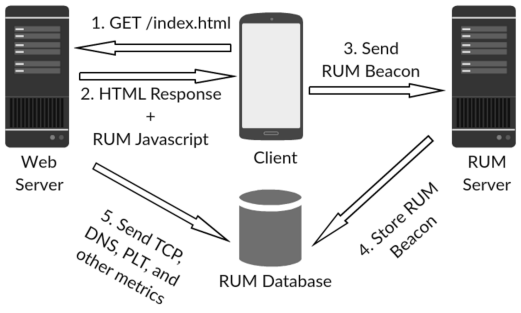

Figure 6.4: Sequence of Akamai's RUM interactions with client's browser.

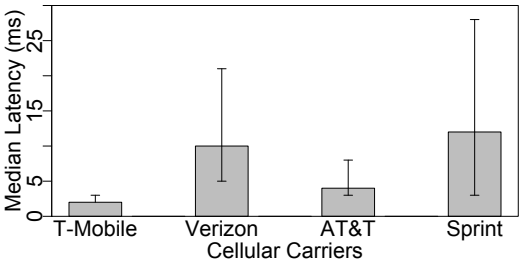

Figure 6.5: Round trip latency between Akamai CDN servers and cellular TCP Split proxies.

packets (from the IPv4 server) to IPv6 packets (forwarded to the mobile device), as shown in Steps 5, 4, and 3.

In summary, there are two ways in which T-Mobile routes packets from IPv6 devices: 1) via IPv6 network with no stateful middleboxes, and 2) via NAT64 middlebox, where IPv6 network is used between clients and NAT64 and IPv4 network is used between NAT64 and IPv4 servers.

Verizon Wireless (Verizon) provides both IPv6 and IPv4 addressing to all of its devices connected to its LTE network [60]. For devices with no IPv6 support or not connected to the LTE network, Verizon provides only IPv4 addressing – resulting in the devices using Verizon's IPv4 network. Thus, in the Verizon network, the choice of addressing on the device is based on whether the device is connected to the LTE network [260].

In Figure 6.2, we depict a high level infrastructure of Verizon's IPv6 network. Similarly to T-Mobile, when an IPv6 device communicates with an IPv6 server, the packets are routed to the server through Verizon's IPv6 network without any stateful NAT middleboxes (Steps 1 and 2). However, when an IPv6 device communicates with

an IPv4 server, the device uses the Gateway-Initiated Dual-Stack Lite (DS Lite), a software installed on the phone, to encapsulate IPv4 packets inside IPv6 headers (Step 3) [79,130]. The encapsulated packets are then forwarded to a middlebox that decapsulates the packet contents and strips out the IPv6 header (Steps 4 and 5). Finally, the IPv4 packets are forwarded to the IPv4 server (Step 6). One of the benefits of using a gateway initiated DS Lite is that encapsulating IPv4 packets in an IPv6 header allows Verizon to use IPv6 addressing for routing packets within the cellular network.

In summary, there are two different ways in which Verizon routes packets from IPv6 devices: 1) via IPv6 network with no stateful middleboxes, and 2) via IPv4-in-IPv6 tunnels using DS Lite software on the phone.

AT&T Mobility (AT&T) and Sprint provide both IPv6 and IPv4 addressing to only some of their IPv6-capable devices [5,64]. For other IPv6-capable and IPv4-only devices, both these networks provide only IPv4 addressing. Therefore, the choice of addressing for devices in these networks is neither dependent on `464XLAT` nor on the cellular technology and is rather likely to be decided by the carrier's respective network configurations.

In Figure 6.3, we depict a high level infrastructure of the IPv6 network deployed by AT&T and Sprint. We show that when an IPv6 device communicates with an IPv6 server, the packets are routed directly to the server, through the IPv6 network without any stateful middleboxes (Steps 1 and 2) [5,64]. However, when an IPv6 device communicates with an IPv4 server, unlike in T-Mobile and Verizon networks, the packets route through the IPv4 network, which consists of several stateful NAT middleboxes, such as NAT 44 and NAT 444, to convert private IPv4 addresses to public IPv4 addresses (Steps 3, 4, and 5).

In summary, there are two different ways in which both AT&T and Sprint route packets from IPv6 devices: 1) via IPv6 network with no stateful middleboxes, and 2) via IPv4 network with several stateful NAT middleboxes.

Data Collection Methodology

Recent studies on measuring CDN adoption rate among websites show that out of the most popular 1,000, 10,000, and 100,000 websites listed on Alexa [44], 77%, 35%, and 19% are hosted on different CDN infrastructures, respectively [15]. Further, a study on understanding the CDN market share indicates that Akamai CDN infrastructure leads in delivering content for majority of the popular websites, including several e-commerce, media, government, news, and social media websites [28] [55]. Specifically, out of the most popular 1,000, 10,000 websites listed on Alexa, as well as, top 500 websites listed on Fortune [17], Akamai delivers content for over 23%, 16.4%, and 32% websites, respectively [31]. Based on these results, we believe that our dataset on cellular network performance collected by globally distributed CDN servers of Akamai is representative of mobile Web performance in general.

We now shift our focus to organize our measurement data collection to accurately represent performance of native IPv6, legacy IPv4, NAT64, and DS-Lite sessions. First, we provide an overview of how we collect performance data from client devices and CDN servers. Next, we discuss techniques used to filter data generated by a number of independent sources, such as client's browser caching of content, Web proxies in the cellular network, and mobile device's operating system. Finally, we describe how we sanitize our measurement data to only contain RTTs, DNS lookup times, and webpage load times for pages loaded over end-to-end (E2E) IPv6 and

IPv4 sessions between clients and CDN servers, as well as pages loaded via NAT 64 middleboxes and DS-Lite.

Experimental Setup: To compare the Web performance perceived by end-users on IPv6 and IPv4 networks, we use Akamai's RUM system [41], as depicted in Figure 6.4. Akamai's Web servers inject JavaScript (RUM Javascript) into a small fraction of user requests for some of the customer-websites hosted on Akamai infrastructure (Steps 1 and 2). The injected JavaScript uses the browser exposed Navigation Timing API to capture the time to resolve domain names, time to establish TCP connections, and webpage load time, among several other metrics [36]. The JavaScript then sends the collected timing data in the form of a `RUM Beacon` to a dual-stacked RUM server after the page load completes (Step 3). The RUM server then sends the data in the `RUM Beacon` to the RUM database (Step 4). Finally, in Step 5, the Web server complements the data into the database with the TCP latency estimated by the Web server that served the webpage, the publicly routable IP addresses of the CDN server and the client, indicator of whether the webpage was available via IPv4-only, or dual-stacked GET requests, an indicator of whether the webpage was requested over IPv4 or IPv6, an indicator of whether the `RUM Beacon` was submitted over IPv4 or IPv6, and the cellular ISP name to which the client's IP address belongs as determined by Akamai's `EdgeScape` [1].

Note: Earlier CDN deployments only served some of the static webpage content. However, as the need for responsive Web performance increased, CPs moved their base pages and other page resources onto replica servers as well. DNS CNAMEing allows base page domain to resolve to a domain page hosted by a CDN [207].

Using Akamai's RUM, in Figure 6.5 we show the round trip TCP latency estimated by Akamai servers when connecting to TCP terminating proxies deployed by

cellular ISPs using measurement techniques developed by Goel *et al.* [148]. The height of each bar graph represents the median TCP latency, and the extreme ends of error bars represent the 25th and 75th percentile of the latency respectively. We show that the median latency between Akamai CDN servers and cellular gateways of T-Mobile and AT&T is only 2 ms and 4 ms, respectively. For Verizon and Sprint, the latency is less than 10ms. We argue that such a low latency is possible only when Akamai's CDN servers are in extreme proximity with the cellular gateways, as opposed to significantly higher latency estimated by Amazon EC2 and PlanetLab datacenters in the US [112]. Therefore, our view of the IPv6 ecosystem using Akamai's infrastructure allows us to isolate the performance differences if IPv4 and IPv6 within the cellular network (without introducing the confounding factor of the public Internet).

Data Sanitization: We filter our dataset to include performance numbers that pertain only to webpages loaded on Google Chrome browser on Android devices. Our choice to eliminate any influence of iOS was based on lack of IPv6 support on iOS devices at the time of this study. During this study, IPv6 support on iOS devices was not available and was only made available in late 2015 with the release of iOS 9 [185]. Therefore, we choose to consider measurement data for Android devices only. Next, in our dataset, we observe that the Web browsers, from which the webpages were mostly requested were Chrome Mobile on Android and Safari on iOS devices. The lack of support for Navigation Timing API on Safari browser installed on iOS version 8 and below motivated us to eliminate any RTT, DNS, and webpage load time related measurement data from our dataset [37].

Recent study on detecting Performance Enhancing Proxies (PEPs) in cellular networks has revealed that cellular networks in the US do not use PEPs for HTTPS traffic [148]. Therefore, to remove any influence of PEPs (in terms of Web content

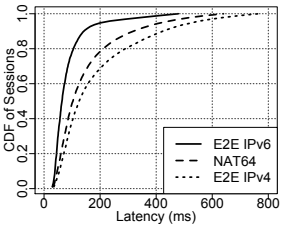

Figure 6.6: RTT distribution for T-Mobile clients.

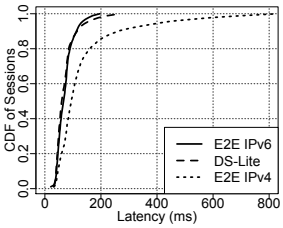

Figure 6.7: RTT distribution for Verizon clients.

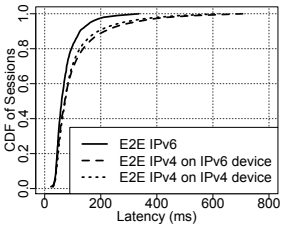

Figure 6.8: RTT distribution for AT&T clients.

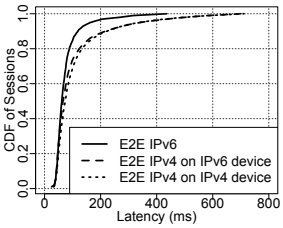

Figure 6.9: RTT distribution for Sprint clients.

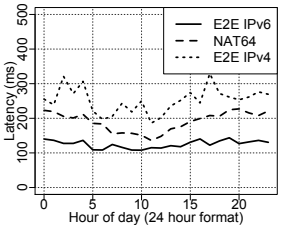

Figure 6.10: 24-hour RTT distribution for T-Mobile.

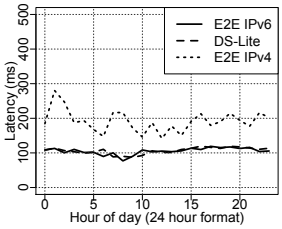

Figure 6.11: 24-hour RTT distribution for Verizon.

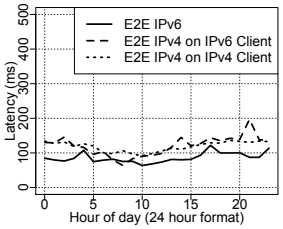

Figure 6.12: 24-hour RTT distribution for AT&T.

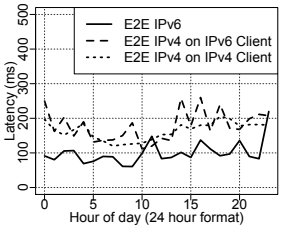

Figure 6.13: 24-hour RTT distribution for Sprint.

caching and TCP split connections) in our dataset, we consider latency for only HTTPS sessions. Latency for HTTPS sessions allows us to accurately estimate the latency between CDN servers and client devices and ensure that the estimated latency is **not** between servers and PEPs in cellular networks. In this work we focus on Web performance over native IPv6 and legacy IPv4 networks, and so eliminate factors, such as presence of PEPs that may confound our measurement data. Analysis of how PEPs in cellular networks impact Web performance is potential future work of this study.

Finally, to remove any influence of content caching in Web browsers on the measurement data, we consider data for only newly created TCP connections. We extract data for connections whose setup time is more than 20ms, which ensures that a TCP socket was created over the cellular network and that an existing connection was not used [38]. We employ a similar technique to extract DNS resolution times

that were resolved at the time of webpage load, thus eliminating the influence of any cached DNS resolutions in the browser.

Our sanitized dataset consists of measurement data for RTT and DNS lookup time for several million sessions between clients and Akamai CDN servers, and webpage load time from several hundred page loads.

Data Analysis: To record the latency over an IPv6 connection we use RTT to clients estimated by CDN servers for webpages requested over IPv6 network. To get the latency over connections via NAT64 (in T-Mobile) or DS-Lite (in Verizon), or by IPv6 clients using IPv4 network (in AT&T and Sprint), we use latency estimated by CDN servers for connections, where webpages were requested over IPv4 network and the `RUM Beacon` was submitted over IPv6 – indicating that the webpage loaded on an IPv6 client. To record the latency over IPv4 connections, we use latency estimated by CDN servers for webpages requested over IPv4 and where `RUM beacon` was also submitted over IPv4 – indicating that the webpage is loaded on an IPv4 client. We apply similar techniques to extract webpage load time.

To record DNS lookup times that pertain to domain names resolved by IPv6 clients, we extract data points for clients on which either websites were loaded over IPv6, NAT 64, or DS-Lite connectivity, or the RUM beacon was submitted to the RUM server over an IPv6. Similarly, to get DNS lookup times that pertain to domain names resolved by IPv4 clients, we extract data points for clients on which either websites were loaded over IPv4 or the `RUM Beacon` was submitted to the RUM server over IPv4 network.

Round Trip Latency over IPv6 and IPv4 Cellular Networks

The Round trip time (RTT) between clients and servers plays an important role in influencing Web performance [151]. In this section, we investigate whether serving mobile content over IPv6 results in lower latency between mobile clients and content servers. We also investigate whether the performance of IPv6 network as well as the performance gap between IPv6 and IPv4 remains same at peak and non-peak traffic hours of a day.

In Figures 6.6–6.9, we show the overall distribution of RTT between clients and CDN servers over IPv6 and IPv4 networks of different cellular carriers, collected in five months of 2015. The solid CDF lines in these graphs show the RTT when IPv6 clients connect to IPv6 servers, over the IPv6 network. The dashed CDF lines show the RTT when IPv6 clients connect to IPv4 servers via NAT64 middleboxes (in T-Mobile), via IPv4-in-IPv6 tunnel (in Verizon), or via the IPv4 network (in AT&T and Sprint). The dotted CDF lines show the RTT when IPv4 clients connect to IPv4 servers, over the IPv4 network. Additionally, in Figures 6.10–6.13, we show the RTT distribution for 24-hour period, averaged over two months (June and July in 2015).

In the case of T-Mobile in Figure 6.6, we observe that the RTT for sessions over IPv6 network is lower than the RTT over the IPv4 network. For example, for median and 80% of sessions, the RTT over IPv6 network is about 49% and 64% faster than RTT over IPv4 network, respectively. Even for sessions via NAT64 middlebox, the IPv6 RTT is lower than RTT over IPv4 network. For example, for connections that go through NAT64 middleboxes, the latencies for median and 80% of sessions are about 18% and 27% faster than RTT over the IPv4 network, respectively.

Further, to eliminate any effects of provisioning differences between IPv4 and IPv6 networks, we compare performance over IPv6 and IPv4 networks at peak and

non-peak traffic hours in Figure 6.10. We observe that the RTT over T-Mobile's IPv6 network outperforms latency over its IPv4 network at all times. In fact, NAT64 sessions also experience lower latency than IPv4 sessions at all times of the day. Although we see that the performance gaps between all three distributions is not same at all times, native IPv6 connectivity always provides least possible latency among others. Based on our observations, we argue that such differences in round trip latency in T-Mobile network arise from the elimination of overhead of NAT middleboxes deployed in the IPv4 network to perform address translation of client sessions. With no middleboxes in case of end-to-end IPv6 connectivity or one middlebox in case of NAT64 sessions, users experience lower latency.

In the case of Verizon in Figure 6.7, we observe that RTT for IPv6 sessions is similar to RTT for DS Lite sessions. We expect the two RTTs to be similar since in case of DS Lite (IPv4-in-IPv6 tunneling) all packets are sent from the device over the IPv6 network, resulting in similar RTT as end-to-end IPv6 sessions. The RTT over IPv4 network however is influenced by Carrier Grade NATs and Large Scale NATs and thus experience significantly higher latency than end-to-end IPv6 or IPv4-in-IPv6 tunneled sessions. For example, for median and 80% of sessions on Verizon, the RTT over IPv6 network is about 29% and 44% faster than RTT over its IPv4 network, respectively. Additionally, when comparing performance over 24-hour periods in Figure 6.11, we observe that IPv6 latency inside Verizon's network outperforms latency over its IPv4 network in both peak and non-peak traffic hours. Based on our observations, we argue that the reduced RTT over Verizon's IPv6 network is due to two major factors: 1) no stateful middleboxes in its IPv6 network, and 2) use of IPv6 connectivity only over LTE network, as opposed to use of IPv4 connectivity over 3G network.[1]

[1]We could not disambiguate our measurement data specific to sessions over LTE and 3G networks, because Akamai RUM uses JavaScript to collect client-side performance and at the time of our

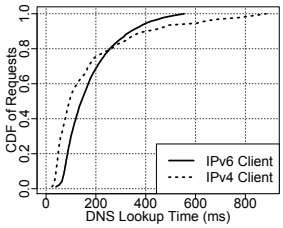

Figure 6.14: DNS Lookup time for T-Mobile clients.

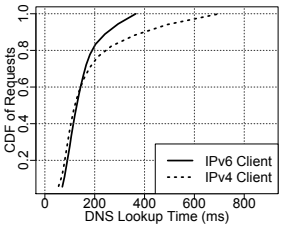

Figure 6.15: DNS Lookup time for Verizon clients.

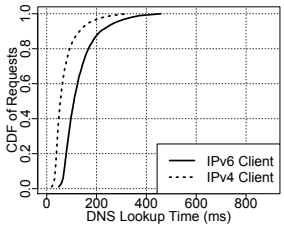

Figure 6.16: DNS Lookup time for AT&T clients.

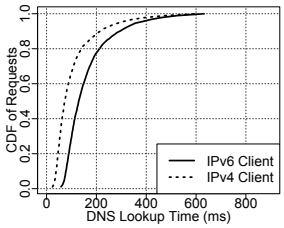

Figure 6.17: DNS Lookup time for Sprint clients.

Finally, in the case of AT&T and Sprint in Figures 6.8 and 6.9 respectively, we observe that RTT over IPv6 network is lower than RTT over their respective IPv4 networks, especially in the long tail. For example, for median and 80% of sessions on AT&T network in Figure 6.8, the RTT for IPv6 network is 17% and 24% faster than RTT over its IPv4 network respectively. Further, the RTT for sessions established by IPv6 clients with IPv4 servers is similar to RTT for sessions established by IPv4 clients with IPv4 servers. We expect the two RTTs to be similar because both AT&T and Sprint are dual-stacked networks and therefore both IPv6 and IPv4 clients must connect to IPv4 servers over their respective legacy IPv4 networks with NAT 44 and NAT 444 middleboxes – resulting in similar latency. Additionally, when comparing AT&T and Sprint's IPv6 network performance with their respective IPv4 networks over 24-hour periods in Figures 6.12 and 6.13, we observe that IPv6 sessions on both AT&T and Sprint experience lower RTT than latency experienced by IPv4 sessions. Therefore, based on our observations we argue that similarly to T-Mobile, IPv6 networks of both AT&T and Sprint outperform their respective IPv4 networks because there are no stateful middleboxes in their IPv6 networks.

measurement (Jan - Aug 2015), Chrome browser did not capture the cellular technology to which the client is connected when loading a webpage [114].

Discussion: Although, we observe that providing mobile content over IPv6 offers reduced latency for end-users, we identified a cellular network outside the US where, during our study, RTT over IPv6 network was higher than RTT over its IPv4 network by almost 200 ms. To investigate, we ran traceroutes from CDN servers to several IPv6 client IP addresses in that network and identified that IPv6 packets were being routed through another country, resulting in higher RTT over its IPv6 network. To investigate whether similar routing was applicable to IPv4 packets in that network at the time of our measurement, we ran traceroutes from the same CDN servers to IPv4 client IP addresses in that network and found that packets were **not** being routed via another country. While it is possible that IPv6 could be slower than IPv4 in some networks, it could be related to how IPv6 packets are forwarded on the Internet. Therefore, proximity of Akamai servers to cellular gateways eliminates the effects of misconfigured routing in the public Internet.

DNS Lookup Time for IPv6 and IPv4 Clients

in addition to RTT, DNS lookup time is another important factor which influences the Web performance in cellular networks [307]. In this section, we measure the DNS lookup time for both IPv6 and IPv4 clients resolving dual-stacked domain names (domains which can be resolved to both IPv4 and IPv6 addresses). In Figures 6.14–6.17, we show the distribution of DNS lookup times when IPv6 and IPv4 clients resolve dual-stacked domain names. The dotted CDF lines represent the lookup time when domains are resolved for IPv6 clients, whereas, the solid CDF lines represent lookup time when domains are resolved for IPv4 clients.

In general and contrary to the trends in the previous section, we see that the DNS lookup takes longer for IPv6 clients than IPv4 clients in T-Mobile, AT&T, and

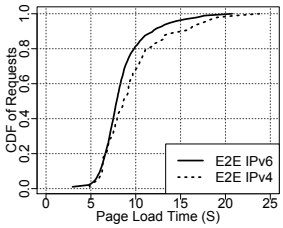

Figure 6.18: Dual-Stack webpage PLT for T-Mobile.

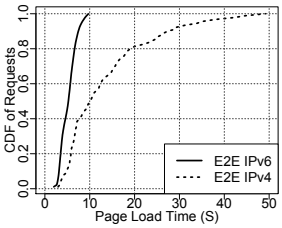

Figure 6.19: Dual-Stack webpage PLT for Verizon.

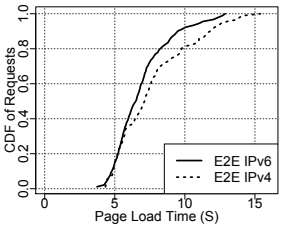

Figure 6.20: Dual-Stack webpage PLT for AT&T.

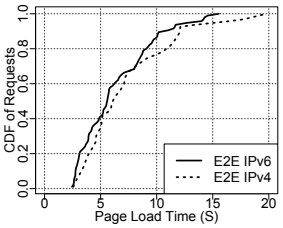

Figure 6.21: Dual-Stack webpage PLT for Sprint.

Sprint networks. However, for Verizon's IPv6 and IPv4 clients, the DNS lookup times are similar. DNS lookup times for IPv6 clients are influenced by their technique of DNS resolution. Client devices are unaware of whether content from a domain is available over IPv4 network or over IPv6 network. Therefore, clients must send both `AAAA` (IPv6) and `A` (IPv4) DNS queries to their local resolvers to resolve domain names. If an IPv6 address for the requested domain is available, Android clients prefer to connect with the IPv6 address, instead of the IPv4 address of the server [317]. Although the two DNS requests could be sent in parallel to reduce the time to perform the DNS lookups, we observe that regardless of the type of domain (IPv4-only or dual-stack), IPv6 Android clients always issue both `AAAA` and `A` DNS queries serially. Further, before returning the DNS response to the application the IPv6 clients wait until responses for both queries arrive, or the resolutions times out. Therefore, the DNS resolution on IPv6 Android clients require two round trips between clients and DNS server, whereas IPv4 Android clients wait for only one round trip for resolving the domain via type `A` query.

In the case of T-Mobile in Figure 6.14, we observe that the median DNS lookup time for IPv6 clients is 25.7% slower than IPv4 clients, because IPv6 clients wait for both type `AAAA` and `A` queries to finish, whereas IPv4 clients wait for only type `A`

queries. However, for about 20% of DNS requests, IPv6 clients experience faster resolution time than IPv4 clients. Since T-Mobile's IPv6 clients can only transmit IPv6 packets into its network, DNS lookups for IPv6 clients take place over T-Mobile's IPv6 network. Our earlier observation from Figure 6.6 shows that RTT over T-Mobile's IPv6 network is lower than IPv4 network, which likely helps about 20% of IPv6 DNS lookups to complete faster in spite of the additional RTT. Therefore, DNS lookups for some IPv6 clients outperform the lookup time for IPv4 clients, even though DNS lookup process for IPv6 clients waits for an additional DNS query to finish.

In the case of Verizon in Figure 6.15, we observe that about 60% of the DNS queries by IPv6 clients take same time as queries by IPv4 clients. Similarly to T-Mobile, IPv6 clients in Verizon network transmit IPv6 packets into the network, which results in DNS lookups over the IPv6 network. From Figure 6.7, we know that RTT over Verizon's IPv6 network is significantly lower than its IPv4 network, therefore the DNS lookup time for IPv6 clients is similar to lookup time for IPv4 clients.

In the case of AT&T and Sprint in Figures 6.16 and 6.17 respectively, we observe that the median DNS lookup times for IPv6 clients are about 38% slower than lookup times for IPv4 client. Since both IPv4 and IPv6 clients in these networks use their respective IPv4 networks to send DNS queries to local resolvers and that RTTs for IPv4 packets sent by IPv6 and IPv4 clients are similar (from Figures 6.8 and 6.9), IPv6 clients wait for responses for two DNS queries in serial, as opposed to IPv4 clients that wait for only one DNS lookup.

Discussion: We observe that the DNS lookup process takes longer for devices with IPv6 capabilities than devices with IPv4-only capabilities in AT&T, Sprint, and partly in T-Mobile networks. Following our observation, we argue that if the DNS lookup process on IPv6 capable Android devices could be modified to send `AAAA` and `A` DNS

queries in parallel, lookup time could be significantly reduced. Additionally, if the client browser could indicate to the mobile OS that they do not need to wait to get the `A` lookup back if they get an `AAAA` answer in DNS response, the DNS lookup time can be further reduced. And finally, for devices in T-Mobile network (and other IPv6-only networks such as Orange Poland and SK Telecom, Telenor [222]), if the OS installed on the device could identify whether the connected network is IPv6-only, the DNS lookup process could only resolve a `AAAA` DNS query, instead of the current implementation where both type `A` and `AAAA` DNS requests are resolved. Such a network specific DNS resolution process will allow clients connected to IPv6-only networks to speed up their DNS lookups. We are working with the Android team at Google to improve DNS resolution process.

Page Load Time over IPv6 and IPv4 Networks

Interactive webpages often require multiple DNS lookups and many round trips between clients and servers to download Web objects onto the client's browser. To investigate the overall impact on Web performance of IPv6's low RTTs and high DNS lookup times, we compare the webpage load time (PLT) over IPv6 and IPv4 networks, using the browser's Navigation Timing API. For this part of the study, we analyze measurement data for two types of webpages: 1) those available over both IPv6 and IPv4 networks (dual-stacked), and 2) those available over IPv4 network only. Our comparison of PLTs only includes page load requests, for which DNS lookups were performed by the client browser and the pages were loaded over newly established TCP connections. Our immediate goal is to eliminate any PLT data that includes DNS lookup from Web browser's cache and reuse of an existing TCP connection.

Table 6.1: Selected mobile device models with highest number of webpage load requests in different cellular networks.

Network	Device Model Name	Model ID
T-Mobile	Samsung Galaxy S5	SM-G900
Verizon	Samsung Galaxy S5	SM-G900V
AT&T	Samsung Galaxy S6 Edge	SM-G925
Sprint	Samsung Galaxy S6 Edge	SM-G925

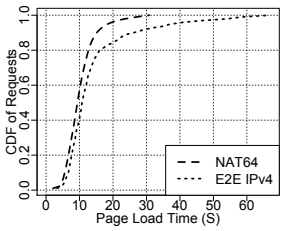

Figure 6.22: IPv4 webpage PLT for T-Mobile.

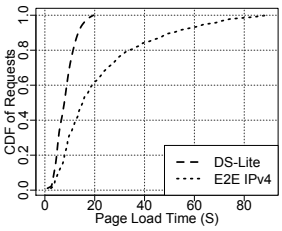

Figure 6.23: IPv4 webpage PLT for Verizon.

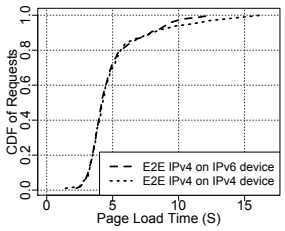

Figure 6.24: IPv4 webpage PLT for AT&T.

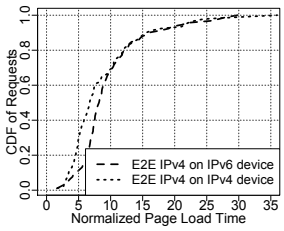

Figure 6.25: IPv4 webpage PLT for Sprint.

To mitigate the influence of mobile hardware on PLT, we consider PLTs from only one specific device model for each cellular network. For each carrier we selected the device model for which Akamai's RUM had highest number of requests for downloading webpage over both IPv4 and IPv6 networks of the corresponding carrier. We list the device models selected for each network from our dataset in Table 6.1.

Further, to accurately characterize the performance of webpage loads over IPv6 and IPv4 networks, we consider PLTs for only one Web URL loaded on one specific device model for each network. Similarly to our choice of device model, we selected the URL for which Akamai's RUM received highest number of requests from the selected device model. Interestingly, the URL for a major postal service website was the only dual-stacked webpage that was loaded the most number of times on all four cellular networks, with the page having 94 embedded Web objects in total and 0.83 MB in size. To prevent the influence of any in-network HTTP caches on PLTs, the page

Table 6.2: Details of IPv4-only webpages loaded over IPv4 networks of different cellular carriers.

Network	Webpage	#Object	Size
T-Mobile	Clothing	444	0.70 MB
Verizon	Internet Retailer	165	1.30 MB
AT&T	Home Improvement	63	0.67 MB
Sprint	Internet Retailer	165	1.30 MB

was loaded over Secure HTTP (HTTPS). Further, we identified different IPv4-only webpages for different carriers that had the most number of page load requests from the selected device. We list the details of IPv4-only URLs selected in each network for performance comparison in Table 6.2 respectively.

IPv6 webpage load time: In Figures 6.18–6.21, we show the distribution of page load time of one dual-stacked webpage loaded by IPv6 clients over networks and loaded by IPv4 clients over IPv4 network. The solid CDF lines show PLTs when the page was loaded over IPv6 network. The doted CDF lines show PLTs when the page was loaded over IPv4 network. In general, we observe that for all four US carriers, the PLTs of pages loaded by IPv6 clients over IPv6 networks are lower than PLTs of the same pages loaded by IPv4 clients over the respective carrier's IPv4 networks. Further, despite DNS lookup times being higher for IPv6 clients, we observe that PLTs are lower for IPv6 clients loading pages over IPv6 network. We argue that DNS lookup times for IPv6 clients influence the overall page load time by just one extra round trip and that the actual benefits of faster IPv6 network are observed when multiple Web objects are loaded over the IPv6 network in several round trips.

In case of T-Mobile in Figure 6.18, we observe that for median and 80% of page loads by IPv6 clients, the PLTs over IPv6 network are 9% and 14% faster than PLTs over T-Mobile's IPv4 network. In Figure 6.19, we observe similar reductions in PLTs

for pages loaded by Verizon's IPv6 clients over Verizon's IPv6 network. Specifically, we show that the median and 80% of the PLTs by IPv6 clients over Verizon's IPv6 network are 48% and 64% faster than PLTs over its IPv4 network, because of the differences in RTTs between Verizon's IPv6 and IPv4 networks as shown in Figure 6.7. For AT&T and Sprint as well, we observe that PLTs are lower over the IPv6 network.

IPv4 webpage load time: In Figures 6.22–6.25, we show distribution of PLTs of an IPv4 webpage, loaded by IPv6 and IPv4 clients. The dashed CDF lines show PLTs when IPv6 clients load an IPv4 webpage via NAT64 (in T-Mobile), via IPv4-in-IPv6 tunnel (in Verizon), or via IPv4 network (in AT&T and Sprint). The dotted CDF lines show PLTs when IPv4 clients load an IPv4 webpage over the IPv4 network. In general, we observe that T-Mobile and Verizon IPv6 clients experience reduced PLTs for IPv4 webpages, due to reduced RTT when using NAT64 in T-Mobile (as shown in Figure 6.6) and the use of LTE and IPv6 network in Verizon (as shown in Figure 6.7). For example, in Figure 6.23, we show that the median and 80% of the IPv4 page loads in Verizon, are about 49% and 67% faster than page loads over IPv4 network. In the case of IPv6 clients in AT&T and Sprint network in Figures 6.24 and 6.25 respectively, we observe that IPv4 webpage PLTs are similar (in AT&T), or occasionally slower (in Sprint) than PLTs experienced by IPv4 clients. We expect the two PLTs to be similar since both IPv6 and IPv4 clients use the IPv4 network to load IPv4 websites. However, in some cases we expect the PLTs by IPv6 clients to be slower than PLTs by IPv4 clients, since such clients wait for an additional DNS query for each domain in the webpage.

Although we only show performance gains with IPv6 for only one URL loaded on one type of device model, we also looked at other URLs loaded from other device models as well. We identified that IPv6 connectivity also improves the page load time

of other URLs, though due to space constraints we do not show PLT distributions for other URLs and devices.

Discussion: Based on our findings on webpage load times, we show that for IPv6 clients in T-Mobile and Verizon networks, both IPv4-only and dual-stacked websites will load faster than IPv4 clients loading the same websites over the carrier's respective IPv4 network. Although, we observe DNS lookups are slower for IPv6 clients in T-Mobile, AT&T, and Sprint networks, we observe their impact on the overall PLT over these networks to be minimal, since the RTT over IPv6 network is lower than RTT over IPv4 networks, which accounts for most of the round trips when loading a webpage [151].

For dual-stacked networks, such as AT&T and Sprint, IPv6 clients in such networks experience faster page loads for only dual-stacked websites. Further, websites available over IPv4 only may occasionally experience poor performance on IPv6 clients, likely due to slower DNS lookups that lower IPv6 latency cannot offset. Finally, we argue that since users are likely to visit websites from different networks, providing mobile content over IPv6 will not hurt the Web performance and in fact in some cases the Web performance will improve if pages are served over IPv6.

DNS Lookups in T-Mobile's IPv6-only Network

As cellular carriers upgrade their network infrastructure to IPv6 to mitigate IPv4 address scarcity, we find that the DNS protocol starts to introduce avoidable performance overhead. We observe an extra round trip in cellular ISPs, such as T-Mobile, Orange Poland, SK Telecom, and others [222], which use IPv6-only

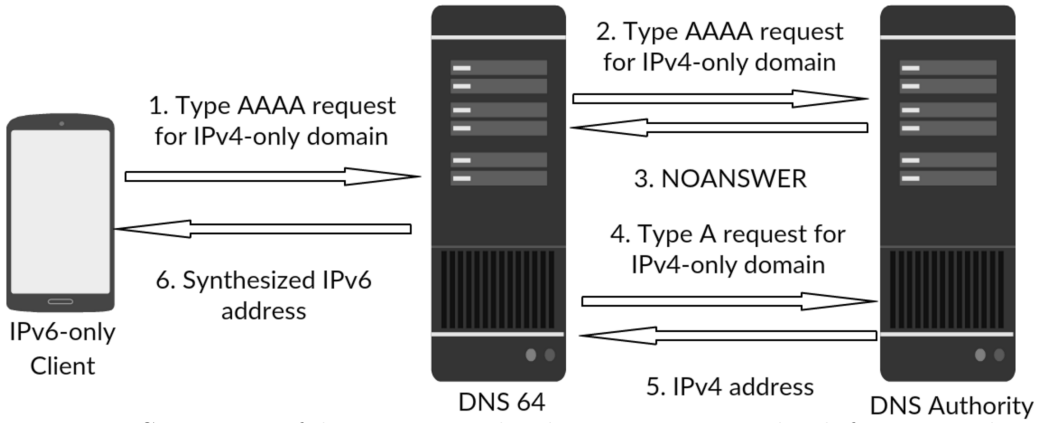

Figure 6.26: Sequence of how IPv4-only domains are resolved for IPv6-clients in IPv6-only networks.

addressing in their network [90]. In effect, ISPs that fully embrace IPv6 technology are penalized with slower domain name resolution.

We illustrate the problem scenario in Figure 6.26, which shows a sequence of DNS messages exchanged between a user device, T-Mobile's DNS 64 server, and a DNS Authority. An IPv6-only environment requires a mobile client to send a `AAAA` DNS query to the cellular DNS server to resolve an IPv4-only domain (Step 1). The DNS request then travels to the DNS Authority (Step 2). The DNS Authority replies with `NOANSWER` flag in the DNS response (Step 3), because an IPv6 address is not available for the IPv4 domain in question. Instead of returning the DNS response with `NOANSWER` back to the client, T-Mobile's DNS server sends a subsequent `A` DNS query for the same domain (Step 4), to which the DNS Authority replies with an IPv4 address (Step 5). After receiving the IPv4 address, T-Mobile's DNS server synthesizes an IPv6 address corresponding to the IPv4 address and sends the synthesized address to the client in response to the client's DNS request (Step 6). At this point, the mobile client is unaware whether the returned IPv6 address is a synthesized address, or a real address returned by the DNS Authority. Since T-

Mobile employs NAT 64 middleboxes to translate synthesized IPv6 addresses back to real IPv4 addresses (Figure 6.1), the synthesized address that the clients receive does not alter their end-to-end connectivity [67, 90].

Thus, if IPv6 addresses are available for a domain, the DNS lookup will finish in Step 3. However, when a domain has only IPv4 addresses available, the DNS lookup requires an extra round trip between the cellular DNS and the DNS Authority (Steps 4 and 5) [90]. The latency of this extra round trip could significantly influence the PLT when cellular DNS servers are not in proximity to DNS Authorities and the webpage requires multiple DNS lookups.

To address this overhead in DNS lookup process, we design OneTrip – a technique for DNS Authorities to eliminate the extra round trip from DNS lookup process in IPv6-only mobile networks. We show that for mobile clients in IPv6-only networks, a DNS Authority can proactively synthesize IPv6 addresses from IPv4 addresses, for all domains it holds mappings between domains and IPv4 addresses. The DNS Authorities could then reply with a synthesized IPv6 address, instead of a `NOANSWER`, to any IPv4-only domain name lookup from IPv6-only networks.

ONETRIP's Approach

To synthesize an IPv6 address from an IPv4 address, similarly to how T-Mobile's DNS 64 servers synthesize [90], OneTrip requires two types of datasets that identify the `/64` prefix used to synthesize an IPv6 address by cellular DNS servers. The first are mappings between client and cellular DNS server IP addresses. The second are mappings between cellular DNS server IP addresses and the `/64` prefix used by the NAT 64 middlebox to which their clients connect. The two datasets collectively allow OneTrip to map a client IP address to `/64` prefix, used by the NAT 64 middlebox associated with each DNS (DNS 64) server. OneTrip makes these mappings available

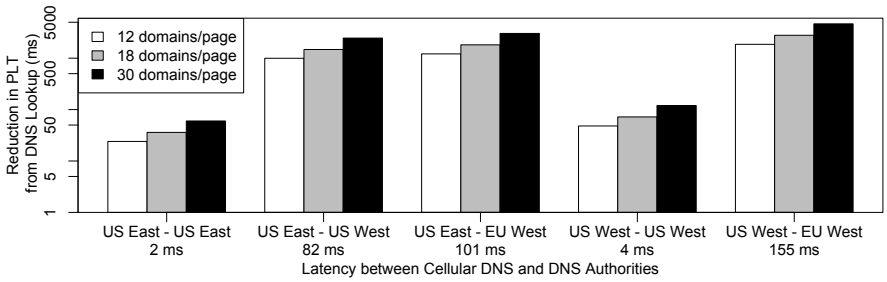

Figure 6.27: Reduction in DNS Lookup time when using OneTrip on DNS Authority.

to DNS Authorities so that when a cellular DNS server sends a request to the Authority, the Authority already knows (based on cellular DNS IP in the DNS query) the `/64` prefix of the NAT 64 middlebox to which the client is connected. The Authority then uses the corresponding `/64` prefix to synthesize the IPv6 address from the IPv4 address associated with the IPv4-only domain name in question.

Next, we describe OneTrip's approach to collect the above mappings using Akamai's global infrastructure of content delivery.

Collecting DNS IP – NAT 64 Prefix mappings: Using the crowd-sourced dataset collected by Netalyzr in over 11 months in 2013-2014 [183], we extracted mappings between cellular DNS server IP addresses and the `/64` prefixes used by `NAT 64` middleboxes deployed in T-Mobile's network [297]. The Netalyzr mobile application sends a `AAAA` DNS request to a cellular DNS server for resolving an IPv4-only domain name whose DNS Authority replies with the IP address of the DNS server that made the request. From the DNS response received by the client, we extract the IP address of the cellular DNS server by converting the last (least significant) 32 bits of the synthesized IPv6 address format from HEX to DEC. We also extract the `/64` prefix from the first (most significant) 64 bits of the synthesized IPv6 address.

Collecting Client IP – DNS IP mappings: For the purpose of this work, we modified our measurement system to make clients resolve a unique hostname that allows us to map the client IP addresses to the cellular DNS IP addresses in a manner similar to the above.

Finally, from the above two mappings we generate the associations between client IP addresses and `/64` prefixes that enable DNS Authorities to identify the NAT 64 middlebox, to which the clients connect. These mappings are also useful when DNS requests use the `EDNS0` extension (containing the IP address of the mobile client) for CDNs to perform server selection based on the client IP instead of the DNS server IP [98]. Specifically, when ISP resolvers send IPv6 client addresses in the DNS request, DNS Authorities search for the NAT 64 address, to which the client is currently connected, and then perform mapping based on the location of NAT64 address in the network. DNS Authorities may also combine NAT64 location information with the (limited) information about the location of client's IPv6 address.

Additionally, the mappings between client IP and NAT64 prefixes enable Web/proxy servers to reduce latency for HTTP transactions, as opposed to using synthetic IPv6 addresses only for the purposes of DNS lookups. Specifically, servers that currently embed static IPv4 addresses in HTTP headers or payload [90], can now embed synthesized IPv6 addresses to sidestep `464XLAT` software on resource limited mobile devices. This procedure reduces latency perceived by end users because the `464XLAT` software on the client first converts the IPv4 address (returned in HTTP response) into an IPv6 address. Next, IPv6 packets are forwarded to the NAT64 middlebox that converts them back to IPv4 packets for the IPv4 host. By allowing servers to embed synthetic IPv6 addresses in HTTP headers, or payloads, latency introduced by 464XLAT software on the client is eliminated.

Discussion: Based on the mappings we collect between clients and DNS server IP addresses, we find that each DNS IP is used by thousands of mobile clients to resolve domain names. Therefore, OneTrip's implementation on DNS Authorities can improve DNS lookup time for thousands of cellular clients for every IPv4-only domain name in resolution. We also observe from the data collected by Netalyzr that the DNS server address and `/64` prefix mappings were stable and that there were no changes over a period of 11 months in 2013-2014. We argue that such demonstrated historical mapping stability supports OneTrip's approach to reliably synthesize IPv6 addresses on the DNS Authorities, however there is a serious risk that doing this mapping outside of operator control could impair their ability to operate their network. The OneTrip mechanism is however a good way to identify the performance improvements available by improving the DNS protocol.

Speeding DNS Lookups with ONETRIP

We discussed OneTrip's approach for DNS Authorities to eliminate the extra round trip during DNS resolutions. However, it still remains unclear whether T-Mobile's cellular DNS would honor, or reject, the IPv6 addresses synthesized by an Authority outside of T-Mobile's network. To answer this question, we configured IPv6 clients in T-Mobile's network to send `AAAA` DNS requests to their respective cellular DNS servers. We also setup a DNS Authority, outside of T-Mobile's network, maintaining DNS records for a domain name with only IPv4 address. Using the two mappings collected by OneTrip, we configured the DNS Authority to reply with a synthesized IPv6 address to the DNS request for the IPv4-only domain name, instead of a `NOANSWER`.

Our experiments show that T-Mobile's DNS servers do not discard any DNS replies with IPv6 addresses synthesized by a DNS Authority outside of the T-Mobile's

network. Further, the mobile clients successfully connect to the IPv4 servers, using the synthesized IPv6 address.

Finally, we perform several experiments to understand the performance improvements that OneTrip brings to DNS content delivery when DNS Authorities are hosted at different proximity from cellular DNS servers. Specifically, our evaluation of OneTrip is based on several possible latency values that exist between cellular DNS servers and DNS Authorities. We expect geographically-distributed DNS Authorities to have lower latency to cellular DNS servers than a centralized DNS Authority. For example, a previous study has shown that Internet backbone round-trip time is around 82 ms between the East and West coast in the US [286]. The same study also provides latencies between different regions in the US and the EU. We use these latency values as representative of latency between cellular DNS servers and DNS Authorities hosted at these different locations.

A recent study by Varvello *et al.* shows that the 25th, 50th, and 75th percentile of websites (out of the top 9000 Alexa Websites that support HTTP/2) consists of over 12, 18, and 30 unique domain names, respectively [299]. We therefore argue that depending on the number of unique domain names in a given webpage and the latency between cellular DNS servers and DNS Authorities, the webpage load time could be significantly improved through OneTrip's removal of one round trip per resolution.

In Figure 6.27, using simulations we show the absolute reduction in DNS lookup time when using OneTrip on DNS Authorities. To measure the effectiveness of OneTrip, we perform in-lab simulations to emulate the behavior of T-Mobile's DNS 64 servers, where we perform DNS lookups from an emulated cellular DNS issuing name resolutions to a DNS Authority. On the x-axis, we represent five different latency values between cellular DNS and DNS Authority. On the y-axis, we show the

latency reduction in the DNS lookup time achieved when OneTrip is used on the DNS Authority.

In general, we see that as the network latency between cellular DNS and Authority increases, the gains increase as well because OneTrip shaves off the extra round in the DNS lookup process. The overall gain further increases when the number of unique domain names associated with different Web objects embedded in a webpage increases, because OneTrip reduces the latency for resolving each of these unique domain names. For example, when the latency between cellular DNS server and DNS Authority is about 82 ms and the number of unique domain names (that needs to be resolved from that Authority) on a webpage is 18, OneTrip reduces the overall DNS lookup time by about 1.4 seconds (18 * 82 ms). The latency reduction shown in Figure 6.27 is the maximum saving OneTrip offers when all domain name lookups are critical for webpage rendering; the minimum benefit is just one RTT saved on the lookup of the base page. In general, as OneTrip reduces the number of round trips between the cellular DNS and DNS Authorities from two to one, OneTrip offers a reduction of about 50% in the DNS lookup time between cellular DNS servers and DNS Authorities in IPv6-only network environments for IPv4-only content.

Discussion on ONETRIP's Approach

OneTrip relies on identifying the `/64` prefix associated with the NAT 64 middlebox, to which a client connects. Similarly, previous studies have developed several techniques to detect the presence of NAT 64 middleboxes in the network [120], including techniques to resolve an IPv4-only domain name with a `AAAA` DNS request and checking if an IPv4 answer is available [277]. Other studies have developed techniques to identify IPv4 addresses from synthesized IPv6 packets [72]. Some other techniques showcase new extensions to DNS protocol and a new Resource Record that

DNS servers can adopt to let clients know what the original IPv4 address is for the domain name in question [82,182,316]. Previous studies also investigated application layer protocols such as STUN to detect the NAT 64 prefix [268]. A study by Ding *et al.* offers a detailed comparison of different techniques used to identifying NAT 64 prefix in IPv6 networks [121]. The same study also showcase how the `EDNS0` extension could be used by DNS resolvers to send NAT 64 prefixes in the DNS request, together with the the technique to calculate the prefix by resolving an IPv4-only domain name.

OneTrip, in contrast to the previous techniques, is unique in that it enables DNS Authorities to detect and effectively use NAT 64 prefixes to reduce DNS lookup time, without the need of any support from the cellular ISPs, allowing measurement of the performance difference. Our motivation for OneTrip is to eliminate the extra round trip present in DNS lookups in IPv6-only networks, with the goal of faster mobile Web for the end-users. Similar round trip elimination has also been proposed in QUIC [171] and TCP Fast Open [255]. Although for eliminating the extra round trip, we also recommend the use of the `EDNS0` option in the DNS query for the cellular DNS servers to pass along the NAT 64 prefix that they are using, such that in the absence of an `AAAA` record DNS Authorities could synthesize one from the `A` record using the contents of the `EDNS0` option. However, we argue that support for such an `EDNS0` option may not be appealing for some cellular ISPs as it introduces additional operational overheads in ISPs' functionality. Therefore, we designed OneTrip for DNS Authorities maintained by Content Providers, Content Delivery Networks, and any independent server operator, which they can implement without any support from cellular ISPs. A strictly better option for Content Providers is to make their content available over IPv6 as this reduces the round-trip by making the `AAAA` record available. Finally, although OneTrip does not address performance overhead introduced by Android DNS lookup process when IPv6 clients wait for both `AAAA` and `A` replies (as

discussed in Section 6), our recommendations to Google's Android team would address the issue of sequential lookups.

Related Work

IPv6 adoption and deployment challenges: Previous studies have measured the adoption rate of IPv6 across different ISPs worldwide and indicate a significant growth in IPv6 traffic over the recent years [29, 110, 180, 222, 274]. While understanding the adoption of IPv6 is important, several communities (including network operators, CDNs, and content providers) have shown interest in discussing challenges faced by mobile ISPs and content providers to adopt IPv6 in their infrastructures in a panel at the @Scale conference held in 2015 [34]. Several case studies conducted at Akamai and Fortinet provide experiences with IPv6 deployment on a global scale [221] and the current state of the IPv6 in terms of information security [30, 142].

IPv6 performance measurement: A study by Dhamdhere *et al.* investigates the impact of BGP route changes on the performance of IPv6 networks [119]. Plonka *et al.* investigate the flow bit rates of IPv4 and IPv6 traffic at different times of the day [254]. A study by Donley *et al.* investigates the impact of several NAT middleboxes on IPv4 latency and offers a performance comparison between IPv4 and native IPv6 connectivity in wired networks [124]. Other studies investigate throughput, RTT, packet loss, and hop counts provided by IPv6 networks [186, 208, 312, 325, 330].

Addressing IPv6 connectivity issues: At the 2015 @Scale conference, a representative from Verizon Wireless suggested that browsers should not fallback on IPv4 when IPv6 connectivity is slow, because a fallback to IPv4 can mask critical performance issues related to IPv6 connectivity [34]. Collectively, several mobile operators suggested that application developers should support IPv6 connectivity to

help kickstart a transition from IPv4 to IPv6. This suggestion complements Apple's recent announcement of IPv6 support in all iOS9 applications [185].

Redirecting clients from broken IPv6 links: Several studies suggest methods to improve Web performance by selectively handing out answers to `AAAA` queries based on the performance of the current IPv6 connectivity is behind the client's resolver network [109, 159, 181, 221].

Our work, in contrast to these studies, focuses on understanding how different IPv6 deployment strategies in cellular networks influence the mobile Web performance, from the perspective of CDNs deployed in the middle of cellular networks and content providers. Based on our study, we argue that hosting mobile content on IPv6-enabled networks and content servers is another direction that CDNs and content providers could adopt to improve the end-user experience.

Conclusions

Content Providers (CPs) and Content Delivery Networks (CDNs) are not fully aware of the differences in mobile Web performance in cellular IPv6 and IPv4 networks. In this paper, we provide our experience with the changing IPv6 ecosystem in major cellular networks from the perspective Akamai's global infrastructure for content delivery. We perform extensive measurement to understand how different IPv6 technologies deployed by cellular carriers impact mobile Web performance. Our results indicate that cellular IPv6 networks outperform their legacy IPv4 networks. We argue to CPs and CDNs that the transition from IPv4 to IPv6 is another milestone for improving mobile Web performance.