TECHNISCHE UNIVERSITÄT MÜNCHEN

DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Large-Scale Data Analysis of Netflix Performance over IPv6

Abhishek Kapoor

TECHNISCHE UNIVERSITÄT MÜNCHEN

DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Large-Scale Data Analysis of Netflix Performance over IPv6

Umfangreiche Datenanalyse der Netflix-Performance über IPv6

Author:	Abhishek Kapoor
Supervisor:	Prof. Dr. Jörg Ott
Advisor:	Dr. Vaibhav Bajpai
Submission Date:	15.06.2018

DECLARATION

I hereby declare that this thesis is the result of my independent work, has not been previously accepted in substance for any degree and is not concurrently submitted for any degree. This thesis is being submitted in fulfillment of the requirements for the degree of Master of Science in Informatics at Technical University of Munich, Germany.

Munich, 15.06.2018 Abhishek Kapoor

ACKNOWLEDGMENTS

This thesis revolves around Netflix performance over IPv6. I would like to thank my thesis advisor Dr. Vaibhav Bajpai from the Chair of Connected Mobility at the Technical University of Munich for constantly guiding me out in analyzing the dataset, for the continuous reviews, feedback, being the second reader of my thesis and for providing me the existing Jupyter notebooks, which helped by forming the building block for this study. I would like to thanks Prof. Dr. Jörg Ott and the research group of Connected Mobility for providing me the desired computing resources and support for carrying out the study. Special thanks to Trinh Viet Doan, Alemnew Sheferaw Asrese and Ermias Walelgne for providing their valuable inputs and for reviewing the paper. Finally, I would like to express my gratitude towards my family and friends for constantly motivating me to carry out my research.

ABSTRACT

We are analyzing a 22-months long (since July 2016) Netflix dataset to uncover the performance of the content delivery network over IPv6. As IPv4 address space is getting exhausted due to increase in the number of connected devices, IPv6 was proposed to counter this problem, but after two decades of its releasing, IPv6 is still not fully adopted by the network and content providers. The main aim of this study was to look into the performance over IPv6 with respect to different aspects. The Netflix dataset that we analyzed was collected using 100 SamKnows probes connected to dual-stacked networks representing 66 different origin ASes. The results show that a Happy Eyeballs (HE) race during initial TCP connection establishment leads to a strong (around 93%) preference over IPv6. However, even though clients prefer streaming videos over IPv6, worse performance over IPv6 than over IPv4 was observed, whereby consistent higher TCP connection establishment times and pre-buffering duration (40ms or more) were witnessed over IPv6. Similarly, consistent lower achieved throughput over IPv6 was also observed. Less than 10% stall rates over both address families were observed. ISP content caches do have an impact on the latency and throughput, and as per the results it was observed that content caches lead to reduced TCP connect times and Pre-buffering duration. Also, content caches achieved higher throughput (around 66% of the times) over both IPv4 and IPv6. We also analyze the Speedtest dataset, which is speedtest measurements to Measurement Lab (M-Lab) servers, collected using the same probes. The comparison between M-Lab and Netflix speedtest reveals that the speedtest is better towards M-Lab servers than towards Netflix OCA servers. We did find out that the path length (TTL) to M-Lab servers is comparatively less than the TTL to Netflix OCA servers, indicating that shorter path lengths correlate with a higher speed for M-Lab. From the traceroute data for Netflix, it was observed that content caches had reduced path lengths, TCP connect times and Pre-buffering durations, and can be reached within 5 hops and 21ms. The results from the analysis indicates good ongoing adoption of the IPv6 protocol.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Introduction

CHAPTER 1

INTRODUCTION

1.1. Motivation

Internet Protocol (IPv4) is the main network protocol of the present Internet for a very long time as specified in RFC 791 [85]. As the number of connected devices is increasing day by day, there is a shortage the address space for every device. With the advent of Internet-of-things devices, network engineers are using Network Address Translation (NAT) and other means to divert the address space exhaustion issue. As every device had to have an end-to-end address, the NAT resolution is breaking the initial concept regarding the Internet and acting as a workaround to a real problem.

In 1998, IPv6 was proposed as a solution to the address space issue [80]. In addition to the address space, IPv6 also improves header simplifications and extensions [35]. In comparison to IPv4, IPv6 uses 128-bit addresses, whereas IPv4 only uses 32-bit addresses, therefore, an increase from 4.3 Billion addresses to 340 undecillion addresses. After two decades of proposing IPv6, it is yet not fully adopted and supported by network content providers. Google statistics [48], reveals that IPv6 is only available to around 20% of the users, but the recent growth is exponential as many organizations and Internet Service Providers are promoting IPv6 after the launch of *World IPv6 Launch Day* in June 2012 [31].

Due to a strong dependency of other protocols on IPv4, its very difficult to modify the present network protocol stack, which is in an hourglass shape [5], this is one of the reasons of slow adoption of IPv6. Chapter 2 describes the studies which inform us about the transition of IPv4 to IPv6 and discusses the ongoing deployment and configuration of IPv6. Now, with IoT devices on the rise, increasing distributed computing and mobile networking [49], ISPs are also supporting IPv6 adoption to counter this problem, as IPv6 has a key role in the IoT network stack [53].

Rise in Netflix users membership (around 109 million [18]) in recent years and the Internet traffic that Netflix is generating (around 35% in North America [92]) made us look into Netflix dataset. The aim of this study is to better understand the present state of IPv6 for Netflix, and its role in content delivery. We will also be comparing

the performance of IPv6 with IPv4. The analyses are performed on a Netflix dataset collected by SamKnows probes [90] since July 2016.

1.2. Research Questions

The thesis revolves around Netflix Content Delivery Network (Open Connect Appliances), as content delivery forms a major part of present Internet traffic. We have looked into various research questions involving IPv6 current topology, its network stack and measuring its performance for Netflix. The thesis tries to answer the following research questions.

RQ 1. How does IPv6 performance compare with IPv4 for Netflix? Google statistics [48] reveals the exponential growth of IPv6 connectivity among internet users. Internet Service Provides, and Netflix is deploying advanced infrastructure over the different part of the world to support the growth and adoption of IPv6 [3]. Analyzing different performance indicators and metrics such as Success Rate, Stall Rate, Throughput, Connect Times, etc. over both the address family's will help us compare the performance difference.

RQ 2. How beneficial are the ISP content caches? How do the performance over IPv4 and IPv6 compare in accessing these caches? In order to improve the user experience, content providers bring the media content closer to the user by utilizing ISPs content caches [[98], [77]]. Thus, the content caches have become very important in the last decade with respect to serving digital content [[42], [29], [7]]. Thus, we wanted to identify such content caches and see their effect on latency, delay, and throughput over IPv4 and IPv6.

RQ 3. How do IPv4 and IPv6 Speedtest results compare for Netflix and Measurement Lab? Speedtest has now commercially become a very important metric for characterizing QoS for broadband [19]. Do users experience similar speedtest results when they try the Measurement Lab Network Diagnostic Tool [96] and the Netflix *Fast.com*. We would like to compare the throughput achieved when accessing an M-Lab server and when accessing a Netflix server.

RQ 4. How do path length, latency, and delay compare over IPv4 and IPv6 for Netflix? Studies [[74], [61]] reports that IPv6 performance is comparable to IPv4, and the largest difference arises because of the IPv6 control plane. The IPv6 control plane is responsible for handling the routing, and as the internet is becoming more *flat* [44], it

would be good to compare Time-To-Live (TTL), latency and delay over IPv4 and IPv6, to check if the network topology is similar as stated in [45].

1.3. Research Contribution

The main findings and the analysis that we did in chapter 4, 5, 6, and 7, contributes to the research about IPv6 performance and network stack. The study discusses different aspects such as IPv6 routing, IPv6 adoption, content cache performances, speedtest results towards M-Lab and Netflix, and the traceroute measurements towards Netflix. We have defined the limitations of this study in chapter 8, together with the conclusion, which also point out the potential future work for this research. Chapter 9 discusses the reproducibility considerations to produce the results that we presented here. The Appendices discuss further analysis and additional information about the study.

IPv4 versus IPv6 for Netflix

1. The median error occurrence rate for a single day lies mostly between 15-45% for the whole timeline, whereas the median success rate over IPv4 and IPv6 is comparable for the whole duration and is around 100%.

2. Residential probes are influencing most the analysis as out of 100 probes, 79 of them are Residential probes.

3. Happy Eyeballs (HE) race during initial TCP connection establishment leads to a strong (around 93%) preference over IPv6.

4. Worse performance over IPv6 than over IPv4 was observed, whereby consistent higher TCP connection establishment times and pre-buffering duration (around 40ms or more) were witnessed over IPv6. Similarly, consistent lower achieved throughput over IPv6 was also observed.

5. Less than 10% stall rates over both address families were observed.

Content Caches

1. ISP content caches do have an impact on the latency and throughput, and as per the results, it was observed that content caches lead to reduced TCP connect times and Pre-buffering duration.

2. Content caches achieved higher throughput (around 66% of the times) over both IPv4 and IPv6.

Speedtests over M-Lab and Netflix

1. The comparison between M-Lab and Netflix speedtest reveals that the speedtest is better towards M-Lab servers than towards Netflix OCA servers. Around 59% of the times, M-Lab achieved higher speedtest over Netflix for both the address families i.e. IPv4 and IPv6.

2. We did find out that the path length (TTL) to M-Lab servers is comparatively less than the TTL to Netflix OCA servers, indicating that shorter path lengths correlate with a higher speed for M-Lab.

Traceroute Analysis for Netflix

1. From the traceroute data for Netflix, it was observed that content caches have reduced path lengths, TCP connect times and Pre-buffering durations, and can be reached within 5 hops and 21ms.

2. The fraction of median TTL over IPv6 being shorter, equal or faster to IPv4 are somewhat same.

3. The prebuffering duration was around 1801 ms over IPv4 for a cache hit, and 2428 ms when there was a cache miss for 90% of the measurements. For IPv6, the pre-buffering duration was 1992 ms for a cache hit, and 2752 ms when there was a cache miss for 90% of the measurements.

Related Work

CHAPTER 2

RELATED WORK

IPv6

IPv6 was initially released in 1998 [80], with the motive of replacing its predecessor IPv4. As per the new specification of IPv6 [86], it brings in expanded addressing capabilities, simpler header format, improved support for extensions and options, flow label capabilities and authentication and privacy capabilities. However, in spite of these upgrades and benefits, IPv6 has not been fully adopted by Internet Service Providers (ISP) and Content Delivery giants like, Google, Netflix, etc. As per the Google IPv6 adoption statistics [48], IPv6 connectivity at present remains around 24% worldwide (refer fig. 2.1. Although IPv6 connectivity is low, its increasing exponentially after the *WORLD IPv6 LAUNCH* in June 2012 [31]. This was an outcome of *WORLD IPv6 DAY* on June 2011, on which major ISPs and content providers decided to enable IPv6, just to test IPv6 performance and reliability. But, the belief and prediction of having more than 50% IPv6 user connectivity by 2016 are not achieved [31]. Sarrar et al. [93] reported that although the adoption rate for IPv6 is not what was expected, it's still growing at a very good rate. The slow adoption rate and because of its importance, IPv6 has been the central focus of several studies, to get a better understanding of the issues that the ISPs are facing.

Colitti et al. from Google [28] studied the quality and quantity of IPv6 connectivity in 2010. The main findings of their results reveal that although IPv6 connectivity is growing significantly, it is not up to the mark. They also said that few larger deployments of IPv6 are influencing the adoption rate and that IPv6 connectivity varies by countries. They also found out that native IPv6 latency is comparable to IPv4. Dhamdhere et al. in [33], further confirmed the findings of [28]. They also found out that IPv6 network deployment is stronger in Europe and Asia, and that Hurricane Electric is the major player in contributing to the IPv6 network topology. One of the main reason is that Hurricane Electric offers free IPv6 tunneling service to their clients [94]. Wu et al. in [99] surveyed the popular solutions for the transition of IPv4 to IPv6. They reconsider the key problems and issues in IPv4-IPv6 transition and found out that

scalability, heterogeneous addressing, and application layer translation are the main issues for the IPv4-IPv6 transition. Studies by Nikkhah and Guerin [73] [72], further reports that disagreement among ISPs on migration to IPv6 or offering competing alternatives to IPv6, created a destabilizing effect as they couldn't converge to a common solution. Also, they reported that little coordination among ISPs in offering IPv6 could go a long way. Czyz et al. further explores the IPv6 adoption rate in [30], where they are using ten global-scale datasets and considers twelve metrics and analyzed the IPv6 adoption. According to their results, IPv6 adoption varies by two orders of magnitude depending on the metric being evaluated, and that isolated evaluations must be taken care of. Czyz et al. [30] also reported similar to other studies, that IPv6 has improved drastically in the past few years and that the IPv6 adoption rate is not globally uniform. Aben in [97] reported that despite the improvement of IPv6, many Internet Service Providers disabled IPv6 due to lack of clients interests. He surveyed the Local Internet Registries which stopped announcing IPv6 and found out that other reasons for disabling IPv6 were the business itself or the network infrastructure changes.

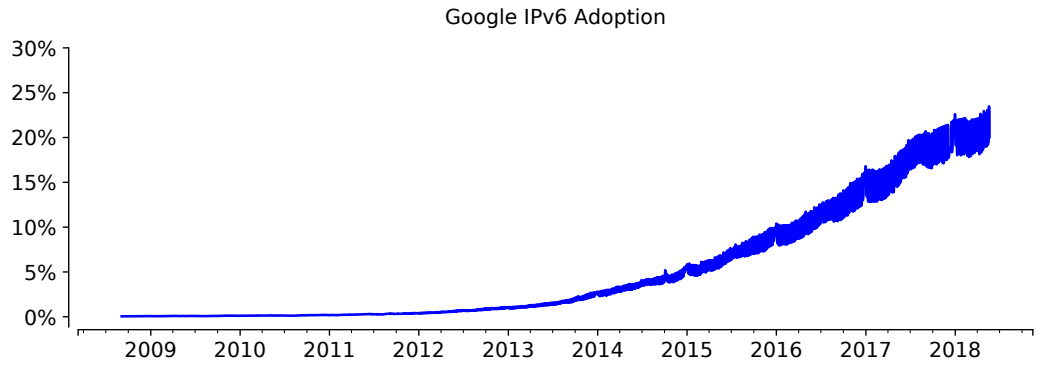

Fig. 2.1.: *Google IPv6 adoption statistics [48] for recent years, and its shows that IPv6 adoption has grown exponentially, being around 24% as of May 2018.*

Internet Topology

Dhamdhere et al. in [32], reported that the Internet infrastructure and topology is changing and evolving rapidly. They explained that the topology is transitioning from a multi-tier hierarchy of ISPs and transit providers to a horizontal mesh of peering links. Furthermore, Labovitz et al. [57] explains that the reason for this transition in the Internet topology is the shifting of Internet traffic to Content Delivery Networks (mostly video) or consumer networks, which are located within an ISP infrastructure or at Internet Exchange Points (IXPs). To find out these changes in the Internet topology,

many studies used different approaches in the past. Dong et al. [37], proposed an algorithm which takes into account multiple protocols, general internet protocols, and routing protocols, and helps discover the network topology. Donnet [38] further did a more concrete study on methods of discovering Internet topology, and one of the methods used *traceroute* to study the number of hops in an IP interface. *traceroute* is a very popular tool used all over the world for network analysis, CAIDA's Archipelago [25] and RIPE ATLAS [89] are such organizations which are using this for large-scale measurements. [38] further explains the limitations of traditional traceroute implementations. It reasons that the same network can result in different network topologies being discovered, because of routing dependency and load balancing issues of traditional traceroute. Knight et al. [55], covered these traceroute issues and asked the ISPs to provide their network data, which will be made public through Internet Topology Zoo, which is a store of network data consisting of around 200 topologies. Holbert et al. [51] further explain that non-cooperative routers towards traceroute can influence the measurements, and as a remedy they developed an algorithm, named iTop, to discover the network topology when only partial information is available.

Paris-Traceroute proposed by Augustin et al. in [10], is one of the improvements over the traditional traceroute, and it ignores incorrect path inferences by forcing the packets to follow the same route. This variant i.e. *Paris-traceroute* is also used in `SCAMPER` which is a general purpose and scaling active measurement tool supporting multiple traceroute variants, proposed by Luckie et al. [64]. Almeida et al. [6], implemented and applied the Multipath Detection Algorithm (MDA) which is also used in *paris-traceroute*. They discovered that around 74% of the IPv6 routes includes one load-balancer.

Content Caching

The load-balancers do play an important role in content delivery. As most of the Internet traffic comprises of content delivery (audio and video), content caches and CDNs like `Akamai` are necessary to provide Quality of Experience (QoE) to the users. Many studies were done to optimize cache deployments, allocation and throughput problems, and in one such study, Poese et al. in [77] reported that Domain Name System (DNS) controls the major content delivery traffic which leads to ISPs losing control of their content delivery. Wang et al. further reported in [98] that content popularity and network topology plays an important role in content delivery, as optimal cache allocation depends on these factors, even for content-centric networking. Trade-offs in deploying caches optimally were studied by Hasan et al. [50], where they discussed that to provide overlay networks for high cache proximity and to improve performance, CDNs place their caches within an Autonomous Systems (ASes). Also, the CDNs deploy additional caches to get the desired high performance. Cordero and Bonaventure [29]

showed that the path lengths to a more popular content are shorter, which supports the notion of "Internet Flattening".

Frank et al. [42], discussed that major content providers are doing strategic collaborations with large ISPs to provide better content delivery network solution. They further suggested a system which can bring content closer to the end-user which will reduce download time, increase capacity and will eventually improve CDN-ISP collaboration.

Dual-Stack

In the early 2000s, a prevalent solution for IPv6 adoption was to use dual-stack implementations, i.e. supporting both IPv4 and IPv6 in parallel within the same network infrastructure. In 2004, Cho et al. [27] reported that the response time of IPv6 connections on dual-stack implementations was bad, due to reasons such as network misconfiguration or other network problems. Pujol et al. [78], studied the state of IPv6 traffic share over a dual-stack residential broadband network. They elaborated the difference between IPv6 "connectivity" and IPv6 "traffic share", explaining that a major fraction of Internet users do have IPv6 connectivity but the IPv6 traffic share is pretty low. As already discussed, the current rate of IPv6 connectivity is around 24% as shown in fig. 2.1, while at Amsterdam Internet Exchange, the IPv6 traffic is around 90Gbps on an average, whereas the total traffic here around 3.5Tbps, i.e. IPv6 traffic is around 3% only, as depicted in fig. 2.2. Pujol et al. reasoned for this discrepancy that home routers were not supporting IPv6, applications were falling back to IPv4, and that service providers were lacking IPv6 support. They further reasoned that the IPv6 network hierarchy is completely different than IPv4 and that Hurrican Electric has a major role in contributing to IPv6 topology. Also, as the network path from the client to host doesn't completely support IPv6, it affects the fallback behavior.

As per RFC 6555, which is also referred to as `Happy Eyeballs` algorithm, IPv6 gets a slight preference over IPv4, and whenever the connectivity is bad, IPv6 can fallback to IPv4. Bajpai and Schönwälder [17] showed that with a timer value of 300ms, IPv6 connections are preferred 90% of the time in spite of being slower than IPv4. They further suggested reducing the timer value to 150ms and discussed that the current value of 300ms is too large as IPv6 performance has improved over the years and thus a reduced value of 150ms will maintain the same preference level for IPv6. Pujol et al. [78] also suggested the fallback behavior of IPv6 can be attributed to incompatibility issues rather than IPv6 slowness.

IPv4 versus IPv6

The dual-stack implementation discussed above, enable comparison of IPv4 and IPv6, and in a study by Zhou et al. [102], analyzed and compared the end-to-end delay

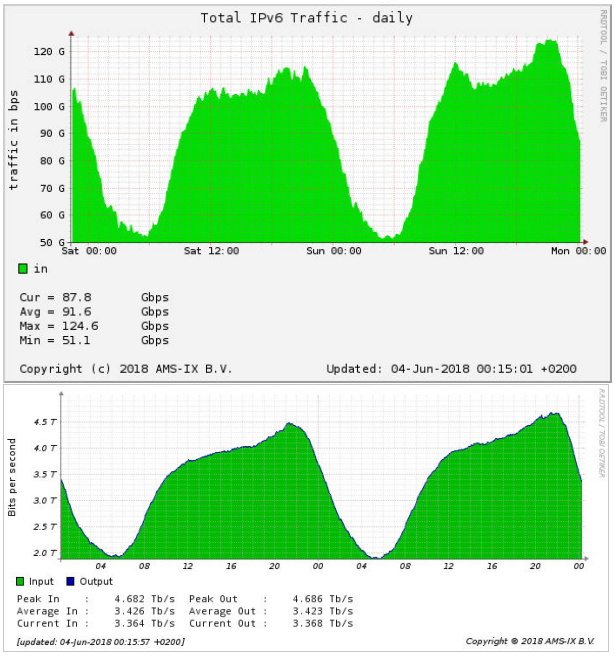

Fig. 2.2.: *Internet traffic at Amsterdam Internet Exchange on 4th June 2018, showing IPv6 traffic [8] and total traffic [9]. As per the graphs, IPv6 traffic accounts to around 3% of the total traffic on average.*

and hop count of IPv6 and IPv4. They published that bad configuration could be a reason for IPv6 to show larger delay times as compared to IPv4. Berger et al. in [20] further compare the latency and loss from Akamai networks over both IP versions. They established that tunneled services showed higher latency and loss depending on the geographical and topological locations. Wu and Zhou [100] evaluated the throughput over IPv4 and IPv6 and established that throughput over IPv6 is slightly higher than IPv4. Nikkhah et al. [74], monitored a list of websites from ALEXA's top 1 Million ranking and argued that the data planes for both address families performed comparatively and that routing is the reason for poor performance over IPv6. Li et al. [59] further establish that actual traffic features such as packet sizes, packet flows and traffic volume are quite different among both address families.

The previous studies discussed showed that IPv6 performed poorly as compared to IPv4, but recent studies indicate that IPv6 performance is improving with time. Giotsas et al. [46] studied the convergence of IPv4 and IPv6 topologies and found out that relationship between IPv6 ASes is forming the same trend as IPv4 and the routing paths are also becoming similar. They also pointed out that Hurricane Electric influenced these results. Bajpai and Schönwälder [16] further compared the IPv4 and IPv6 TCP

connect times towards dual-stacked websites and found out, that the performance was comparable but the difference could be attributed to the presence of content caches in ISP for IPv4 which improves the connect times for IPv4. Hyun et al. [52] further did a similar analysis for the mobile networks and reported that around 85% of the dual-stacked websites experienced better TCP connect times over IPv4 as compared to IPv6 and reasoned that this is due to lack of IPv6 infrastructure in the mobile network.

Livadariu et al. [62], measured the routing dynamics in the control plane of IPv4 and IPv6 and found out that IPv4 is more stable here, whereas for the data plane both address families show comparable performance. They attributed that network congestion leads to low performance, as IPv4 and IPv6 share the same infrastructure as well.

IPv6 Address Space

Understanding the network topology is very important to figure out the growth of the Internet. Lutu et al. [66] and Beverly et al. [23] did a study to understand the reachability of prefixes and availability of routers for IPv6. Similarly, Li et al. [58] found out that even though when the traffic is skewed, around 60% of the IPv6 prefixes are reachable from multiple paths. Beverly and Berger [22] developed an active measurement technique to find out "server siblings" i.e. both IPv4 and IPv6 addresses corrosponds to one physical machine. They further established that understanding the sibling and non-sibling relation gives insight about the co-relations, enhance IPv6 geolocation lookups, and helps compare paths.

Netflix

Netflix has become an Internet giant in providing content and generating Internet traffic all over the world (around 190 countries), it is reportedly having around 109 million customer base [18]. In 2015, The Washington Post published an article [43] stating that Netflix accounts for 37% of Internet downstream traffic in North America. Given the immense growth of Netflix as a content provider, few studies have analyzed Netflix Open Connect Appliance (OCA) architecture, its traffic, and its caching architecture. Adhikari et al. [2] published their study in early 2012, where they focused on Netflix architecture and its service strategy. As per their findings, Netflix uses a blend of cloud services such as Amazon AWS, CDNs, other public services and little infrastructure of their own for the content delivery. They also propose their strategy to improve video content delivery using multiple CDNs, which is choosing the best performing CDN based on a small measurement conducted in the beginning of the video playback claims to improve 12% bandwidth over a single CDN, and more than 50% improvement in

case of using multiple CDNs. In 2013, Martin et al. presented a study [67], where they considered Netflix as a Dynamic Adaptive Streaming over HTTP (DASH) application and characterized the bandwidth consumption of Netflix. They further suggested that during stable network congestion, Netflix adaptation defaults to the TCP control, whereas in case of unstable network congestion, the algorithm is twisted with TCP control. Adhikari et al. further did a study on the Netflix and Hulu CDNs in [1], here they reveal that both Netflix and Hulu are heavily using third-party infrastructures such as Amazon AWS or Akamai. Both the platforms consider the video request to select the CDN, without considering the network conditions. Furthermore, the available bandwidth for Netflix and Hulu CDN varies according to the geographical regions. Adhikari et al. measured the available bandwidth by conducting light-weighted measurements at the beginning of the playback and found the similar results they published in [2] in 2012, i.e. an improvement of 12% bandwidth when a single CDN is used and more than 50% improvement in bandwidth when using multiple CDNs. Netflix in 2012 started its own CDN named "OpenConnect" [**netflixoca**] and a recent study conducted by Böttger and et al. discussed the Open Connect Appliances (OCAs) in [24], where they studied the infrastructure deployment of Netflix and how Netflix is using Internet Exchange Points(IXPs) to deliver its video content all over the world. The authors described how Netflix OCAs help to peer with ISPs for free because of these IXPs, saving transit and content delivery costs. Furthermore, they showed in [24] that Netflix is present in nine of the top ten largest IXPs of the world, Netflix infrastructure presence in major parts of the world (signifying global presence), and their analysis about the internet traffic generated by Netflix is consistent over different geographical regions. Böttger et al. further claims that all these factors make Netflix a hypergiant as a content distribution provider.

Netflix OCA Architecture

The working of Netflix Open Connect CDN infrastructure [71] is the foundation of the `netflix` test (explained in chapter 3) that we are using to get the data. The building blocks of the Open Connect CDN is the Open Connect Appliance (OCA) which are purpose-built server hardware appliances and store the Netflix video content [71]. These OCAs doesn't store client data and are installed at Internet Exchange Points (IXPs) or within the ISPs.

As depicted in fig. 2.3, the clients or end users interact with the Netflix application which acts as a control plane and is hosted on Amazon AWS. The OCAs also communicate health and routability to this control plane. When the Netflix application verifies the user authorization, the control plane determines the characteristics such as user location, network conditions, server utilization and content distribution to select the

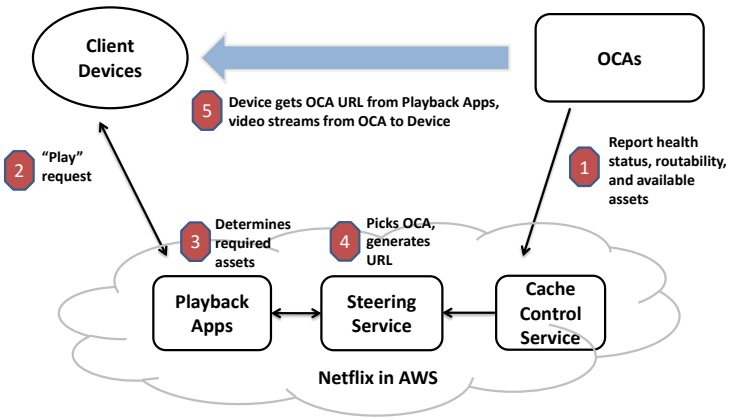

Fig. 2.3.: *Netflix playback process depicting the process of OCA selection and Content Delivery to the customers (figure referred from [71]). Variables that define the particular OCA selection is based on location or load on the server, number of TCP connections, and content distribution [41].*

OCAs which will be used to stream the content to the user. The Netflix control plane generates URLs for these OCAs and deliver it to the client, and then the client redirects to these OCAs and the video streaming starts.

Datasets

CHAPTER 3

DATASETS

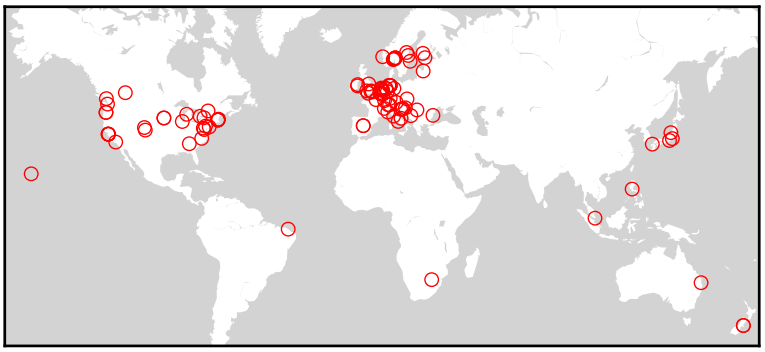

Fig. 3.1.: *Geographical Distribution of 100 dual-stacked SamKnows Probes. The probes takes hourly measurements of IPv4 and IPv6 towards Netflix CDN. The metadata for each probe is available online: https://goo.gl/NN81ij.*

3.1. Methodology

In this section, we will summarize the methodology of collecting the Netflix dataset. The data collection is not a contribution of this thesis, we will be just using the dataset to perform our analysis. We will explain the data collection technique in this section and will explain the dataset metrics and aggregation strategies in more detail in the next section.

Bajpai and Schönwälder in their study [15] explained the installation of SamKnows probes [90], which are hardware-based probes used to take Internet measurements. There are around 100 dual-stacked probes which are geographically deployed (refer fig. 3.1) all over the world and are obtaining measurements over both IPv4 and IPv6. The probes are running hourly Paris-traceroute [11] measurements towards Netflix OCA servers using *scamper* [64]. SamKnows [90] in direct cooperation with Netflix

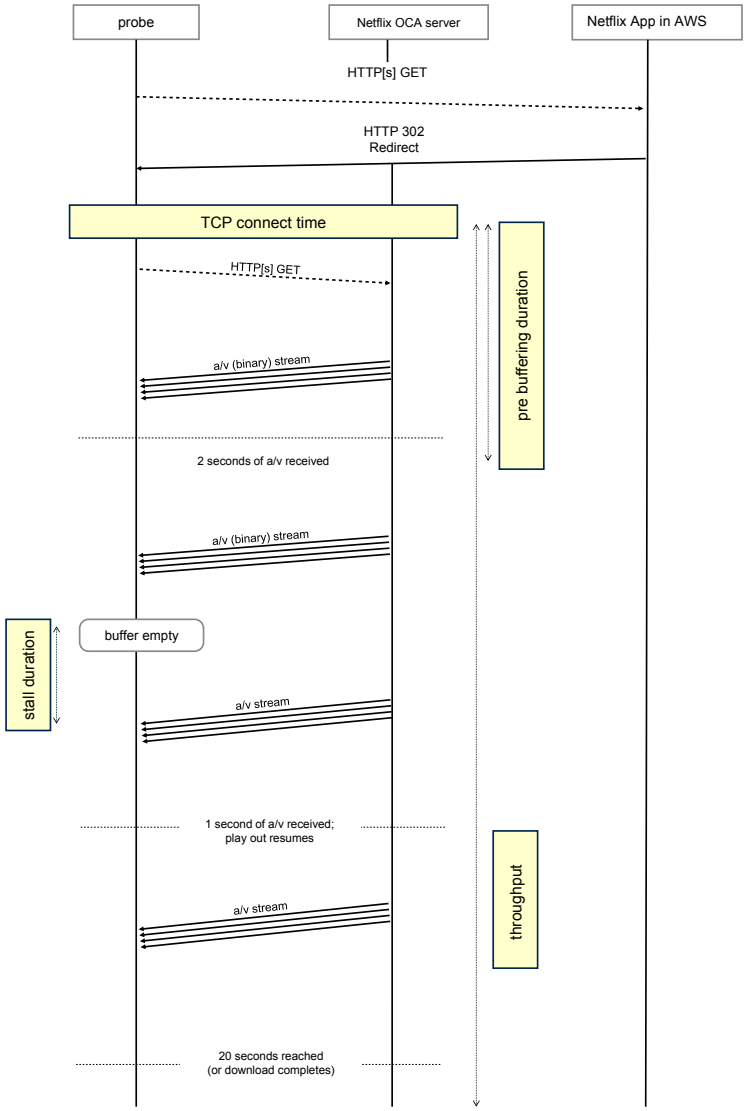

Fig. 3.2.: *A sequence diagram depicting the operation of the* `netflix` *test and the stages where different performance metrics are collected.*

[70] developed the Netflix Test, which is an application-specific test [91]. The test measures the TCP Connect times, Prebuffering Duration, achieved throughput, the number of stalls and stall duration metrics which help us evaluate the performance when streaming a Netflix video. The fig. 3.2 shows the sequence diagram of the operation of this Netflix test, it also shows the stages where different performance

metrics are collected. As per [91] the `netflix` test is designed in such a way that it allows the streaming of binary data from the Netflix OCA servers, also it uses the same OCA server selection logic as their real client uses when a registered user tries to access the content on their website. The functionality is similar to what Netflix is using in *Fast.com* [69]. Refer the official Netflix blog post related to the architecture of *Fast.com* in [41].

The explanation of the test is majorly based on [91] and as per that the test starts by calling the Netflix web-based API which is hosted on AWS [71]. Client source IP address, Traffic load, Network distance, and Location determines the selection of the OCA server, and the API provides this Netflix OCA server which will serve the content to the client. This server can be a Netflix OCA server or can be an ISP content cache (if the ISP participates in Netflix OCA programme). The client receives an HTTP 302 redirect to a 25 MB binary file which is hosted on the Netflix OCA server, or on the ISP content cache. After this, the client will connect to the received OCA server and will try to fetch the 25 MB binary file. The Netflix test runs for 20 seconds, and HTTP pipelining is used to fetch multiple copies of the binary file, which helps ensure receiving more data if the test runtime of 20 seconds is exhausted [91].

Samknows [91] further explains that the binary file is not an audio-video file, but with the knowledge of bitrates at which Netflix streams content, the binary data can be treated as an audio-video data of a fixed bitrate. Furthermore, because of this, if there is a stall occurring, the download doesn't start again but the playback characteristics of the same network are simply recomputed at a different bitrate. The rest of the study explains the meaning of different metrics and the analysis we perform over those metrics.

3.2. Netflix Dataset Overview

Description

The Netflix dataset contains multiple tables (*Netflix* and *traceroute*), and each table consists of attributes which help us evaluate performance metrics such as *Success Rate, Throughput, Error Rate, TCP Connect Times, Pre-Buffering Durations, Stall Rate, Path Length (TTL)* and so on. The dataset collection was started from July 2016 and is running till date, so it's almost a two years dataset when this study got finished. The database consist of few tables which aren't relevant for this study and we haven't used them either, these are *msmpoint*, *netflix_status*, and *traceroute_status*. The Netflix database can be found in /data/akapoor/netflix.db and the schema of the *netflix* table is described in table 3.1.

Table 3.1.: *Schema of the Netflix table*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) of the measurement as "YYYY–MM-DD HH:MM:SS"
target	Netflix OCA hostname
address	Netflix OCA IP address
bitrate	Bitrate of emulated video in bytes/sec
max_bitrate	The maximum bitrate supported by Netflix (currently always 15.6 Mbps) in bytes/sec
stall_events	How many times has it stalled at this bitrate
stall_duration_total	Sum off the durations (usec) of all the stalls
connect_time	Time it took to establish the TCP connection with the content server in usec
download_duration	Duration of the test in usec
prebuffering_duration	How long it took to fetch 2 seconds of video at the specified bitrate in usec
bytes_sec	Download speed in bytes/sec after the prebuffering finished
error	Detailed error code
error_msg	Detailed error message if a problem occurred fetching the content from Netflix
successes	1 if the test runs for the full duration (may have stalls though) i.e. it was not aborted and stepped down
failures	1 if the test was aborted for some reason

The *error* and *error_msg* field in the *netflix* table describes the errors and the their actual reasons for occurring. We have inferred the meanings of these errors from *error_msg* field and from our own research [81] and [87]. There are nine different kinds of errors including the "NO_ERROR" value and table 3.2 describes these errors and the possible reasons for their occurrence.

Table 3.2.: *Overview of Errors (error and error_msg field)*

Error	Possible reasons
NO_ERROR	The test was executed properly.
DNS_RESOLUTION_CONTENT_ERROR	"getaddrinfo: Name or service not known", "getaddrinfo: ai_family not supported", "getaddrinfo: Address family for hostname not supported",
DNS_RESOLUTION_API_ERROR	"curl_easy_perform: Couldn't resolve host 'api-global.netflix.com'(IP:)", "curl_easy_perform: name lookup timed out (IP:)",
STALLED_WILL_STEP_DOWN	When the client is configured to start testing at the highest supportable bitrate but step down to a lower bitrate when stalls occur.
NETWORK_CONTENT_ERROR	"The speed was too low to finish the prebuffering" "Server returned error 307", "recv: Connection reset by peer", "send: Resource temporarily unavailable", "Read timeout", "The server closed the connection", "Server returned error 500", "send: Broken pipe",
CONNECTION_CONTENT_ERROR	"Could not connect to host"
NETWORK_API_ERROR	When there the server is not able to handle the requests (Error Code 503) or is temporarily unavailable (Error Code 500). Other reasons could be are connection timed out or operation timed out.
CONNECTION_API_ERROR	When the Network is unreachable, or the client is not able to connect to the host.
STALLED_ON_FINAL	Stall happening for the lowest bitrate as well i.e. can't step down after this.

As Viet [35] did in his thesis, we also created additional tables which assisted us in our study. We looked up the AS holder names from the RIPEstat [88] APIs for the respective destination i.e. *address* field. We used the scripts by Bajpai et al. [40], which were part of Measurement Research within the Python3 Ecosystem study. There are about 6000 distinct IP (both IPv4 and IPv6) addresses in the dataset and their AS holder names are depicted in table 3.3 and table 3.4. We further segregated this table for IPv4 and IPv6 and the results can be found in the Appendices.

Table 3.3.: *ASN Holder Names*

AS Name	**#IPs**
AS-SSI - Netflix Streaming Services Inc.	5612
BSKYB-BROADBAND-AS - Sky UK Limited	93
BT-UK-AS - British Telecommunications PLC	77
TELENOR-NEXTEL - Telenor Norge AS	60
TEKSAVVY - TekSavvy Solutions	46
BELGACOM-SKYNET-AS - Proximus NV	43
EIRCOM - Eircom Limited	14
GET-NO - Get AS	14
COMHEM-SWEDEN - Com Hem AB	13
XS4ALL-NL - Xs4all Internet BV	12
ALTIBOX_AS - Altibox AS	12
AS-VOBIZ - vanoppen.biz LLC	12
PLUSNET - British Telecommunications PLC	12
INTERNODE-AS Internode Pty Ltd	11
SNAP-NZ-AS Snap Internet Limited	11
NEXTGENTEL - NextGenTel AS	11
BEANFIELD - Beanfield Technologies Inc.	10
AUREON-5056 - Aureon Network Services	10
ALGAR TELECOM S/A	8
SUNRISE - Sunrise Communications AG	8
ASBRUTELE - Brutele SC	8
BREDBAND2 - Bredband2 AB	7
ALLST-15290 - Allstream Corp.	6
ASN-CATCHCOM - Broadnet AS	6
CALLPLUS-NZ-AP CallPlus Services Limited	5
RCN-AS - RCN	5
MOBILEONELTD-AS-AP MobileOne Ltd.	4

Table 3.4.: *Continuing table 3.3*

AS Name	#IPs
MNET-AS - M-net Telekommunikations GmbH	4
INIT7 - Init7 (Switzerland) Ltd.	4
NORDUNET - NORDUnet	4
PENNREN - KINBER	4
CENTURYLINK-US-LEGACY-QWEST - Qwest Comm. Co.	3
WAIA-TRANSIT-AP Western Australian Internet Association	3
UUNET - MCI Communications Services	3
TEKSAVVY-WEST - TekSavvy Solutions	2
ASN-IINET iiNet Limited	2
TPG-INTERNET-AP TPG Telecom Limited	2
PREMIER-COMMUNICATIONS - Premier Communications	2
EWETEL - EWE TEL GmbH	2
NMAX - Nmax	1
ASSOCIAÇÃO NACIONAL PARA INCLUSÃO DIGITAL - ANID	1
PSC-EXT - Pittsburgh Supercomputing Center	1
T-MOBILE-AS21928 - T-Mobile USA	1
WEBFOCO TELECOMUNICACOES LTDA	1
G8 NETWORKS LTDA	1
TBNet Informatica LTDA	1
Rede Connect Telecom	1

The Netflix dataset has the *traceroute* table which consists of TTL and RTTs values to the Netflix CDN hosts. The schema of the *traceroute* table can be found in table 3.5.

Table 3.5.: *Schema of the traceroute and traceroute-filtered table*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) as "YYYY–MM-DD HH:MM:SS"
version	Scamper version used for the measurement
source	IP address of the source
destination	IP address of the destination
method	Traceroute variant
status	Measurement status
ttl	TTL of the packet
endpoint	IP address of the intermediate hop
rtt	RTT of the packet

The status column in the above table tells us whether a measurement was completed or not. It also defines the reasons if the measurement was not completed. The possible reasons are depicted in table 3.6 and their meanings were inferred from the dataset and from our own research about scamper [65], and [63].

Table 3.6.: *Overview of possible stop reasons (status values) for scamper*

Reasons	Description
COMPLETED	Destination reached, traceroute measurement successful
GAPLIMIT	Destination reached, traceroute measurement successful
UNREACH	ICMP destination unreachable message received
LOOP	Specific IP address appeared twice in the same trace
ERROR	Error occurred when sending messages on socket (sendto() error)

We further followed the same approach of aggregation which Viet followed in his Master's Thesis [35]. Therefore, most of the table described below are based on his work. We collected the public source IPs of the probes from the SamKnows [90] dashboard

and queried the RIPEstat [88] API to get the AS number and holder name for these public *source* IPs, *endpoint* IPs, and *destination* IPs. The public source IPs we collected can be found at /metadata/sources.csv and we were only able to collect the IPv4 addresses. We used the IPv6 addresses available from the *traceroute* table. The resulted tables can be found in table table 3.7, table 3.8, and table 3.9.

Table 3.7.: *Schema of the dst_asn_mapping table*

Field	Description
ip	IP address of the destination
asn	AS Number of the Source
holder	Entity name which owns the AS

Table 3.8.: *Schema of the src_asn_mapping table*

Field	Description
unit_id	The ID of the measurement probe
src_ip	Public IP address of the source
asn	AS Number of the Source
holder	Entity name which owns the AS

Table 3.9.: *Schema of the endpoint_asn_mapping table*

Field	Description
endpoint	Endpoint or Intermediate hop IP address
asnnum	AS Number of the Endpoint
holder	Entity name which owns the AS

We wanted to find out the AS type for the individual endpoints along the destination paths. CAIDA [26] does classify every AS according to three types i.e. *Transit, Content, and Enterprise*. These CAIDA classifications are based on their machine learning algorithm and the PeeringDB classifications [76]; table 3.10 contains the schema for the CAIDA dataset.

Table 3.10.: *Schema of the as_types table*

Field	Description
asn	AS number, assigned by Internet Assigned Numbers Authority (IANA)
source	Used classifier i.e. CAIDA_class or peerDB_class
type	Business type of the AS

Reverse DNS lookup of the destination IPs are contained in table 3.11, to help us further classify the IPs. We again used RIPEstat [88] to get this information, and as result table *hostnames* is created. Although there are around 6689 destination IPs only 5154 IPs could be looked up.

Table 3.11.: *Schema of the hostnames table*

Field	Description
ip	IP address of the destination
hostname	Reverse DNS hostname

Aggregation

We aggregated the original *traceroute* table to help us in our analysis. Our motivation for doing this aggregation was to get a detailed overview of the dataset and to avoid unnecessary repetition of steps required for dataset filtering. All the below-mentioned tables were pushed onto the same database i.e. at /data/akapoor/netflix.db. We will briefly explain the steps that we followed in this aggregation and filtering. As already informed, we followed the same strategy that Viet [35] followed in his study. The reason for this is that Viet used the *Youtube* dataset which has the similar schema as *Netflix* dataset, and both datasets are collected using SamKnows probes [90].

We started out by filtering the *traceroute* table, and we first removed the rows which contained NaN values or the probes which don't contain measurements for both the IP address families i.e probes with *unit_id* 525884 and 658929. We further removed measurements from Hurricane Electric Probes (refer [79], [47], [34], [21] and [101]), due to the reasons discussed in chapter *Related Work*. Therefore, we removed the rows which contained measurements from probes which belonged to Hurricane Electric i.e. AS 6939 (refer table 3.8 for such probes). The dataset contains a lot of duplicated rows,

originated due to various reasons and we had to drop these rows to remove redundancy. The original *traceroute* table had 34002477 rows and after performing all these operations the rows were reduced to 8768279. We named the resulting table *traceroute-filtered* which contains the non-duplicate rows with measurements from dual-stacked probes only, and its schema is depicted in table 3.5.

We used an additional table which we didn't explicitly pushed to our database, but it contains the *COMPLETED* i.e. status == 'COMPLETED' results (refer table 3.6 for more information). We named this table *completed-traceroute* and it contains the whole TTL measurements i.e. where TTL increments by one for the desired path sequence. We further rounded the *dtime* field to hours and dropped the columns which were not relevant for our analysis. The schema for the *completed-traceroute* table can be found in table 3.12.

Table 3.12.: *Schema of completed-traceroute table*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) of the measurement as "YYYY–MM-DD HH:00:00"
source	IP address of the source
destination	IP address of the destination
ttl	TTL for this path
rtt	RTT of the packet

After considering the "COMPLETED" results, we further filtered the data based on matching IP addresses in the *destination* and *endpoint* fields. We then selected the maximum TTL values based on the grouping of *unit_id, dtime, source and destination* fields, as done by Viet [35]. We did this filtering to make sure that we only analyze the traceroute results which were completed and actually reached the destination. Also, multiple rows for a single path were reduced to only one row with the maximum TTL value and the RTT value for the whole path. The grouping fields that we selected above ensures to filter only one traceroute measurement per path i.e. for a specific *source* and a *destination*.

We split-up this table for IPv4 and IPv6 addresses and the schema for these tables are depicted in table 3.13.

Table 3.13.: *Schema of traceroute_v4 and traceroute_v6 tables*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) of the measurement as "YYYY–MM-DD HH:MM:SS"
source	IP address of the source
destination	IP address of the destination
max(ttl)	Maximum TTL for this path
rtt	RTT of the packet

To further compare the two address family's i.e. IPv4 and IPv6, we merged both the tables table 3.13. As the data was collected every hour, concurrently for IPv4 and IPv6, *unit_id and dtime* could be used to identify traceroute measurements pairs for both the address families.

We rounded the time in *traceroute_v4* and *traceroute_v6* to nearest hour, to identify such measurement pairs. After doing this, we then merged the two tables group by *unit_id* and *dtime(rounded)* fields, and computed the deltas for the max(TTL) and RTT fields. The resulting schema can be found in table 3.14, where fields with the same description are shown side-by-side.

Table 3.14.: *Schema of deltas table (difference of traceroute_v4 and traceroute_v6 tables)*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) of the measurement as "YYYY–MM-DD HH:MM:SS"
source_v4 and source_v6	IP address of the source
destination_v4 and destination_v6	IP address of the destination
max(ttl)_v4 and max(ttl)_v6	Maximum TTL for this path
rtt_v4 and rtt_v6	RTT of the packet
ttl_delta	ΔTTL
rtt_delta	ΔRTT

We wanted to analyze data irrespective of the time, and therefore we removed the *dtime* field to achieve this. The *traceroute_v4 and traceroute_v6* tables were further grouped by *unit_id, source, and destination* fields, which denote recurring paths from the source IP address to the destination IP address for the whole period, and we computed

the median TTL and median RTT values for this group. The tables *path_medians_v4* and *path_medians_v6* achieved after this operation are shown in table 3.15.

Table 3.15.: *Schema of path_medians_v4 and path_medians_v6 tables*

Field	Description
unit_id	The ID of the measurement probe
source	IP address of the source
destination	IP address of the destination
median(ttl)	Median TTL for this path
median(rtt)	Median RTT of the packet

The *deltas* table was also filtered and we removed the *dtime* field to analyze the data irrespective of the time factor. We considered the *unit_id, destination_v4, and destination_v6* grouping to achieve the measurements pairs and calculated the median TTL and median RTT for these measurement pairs. We further calculated the deltas for the median TTL and median RTT, and the resulting schema can be seen in table 3.16.

Table 3.16.: *Schema of pair_medians table*

Field	Description
unit_id	The ID of the measurement probe
source_v4 and source_v6	IP address of the source
destination_v4 and destination_v6	IP address of the destination
max(ttl)_v4 and max(ttl)_v6	Median TTL for this path
rtt_v4 and rtt_v6	Median RTT of the packet
ttl_delta	Median ΔTTL
rtt_delta	Median ΔRTT

We further created a table containing the metadata from the additional tables table 3.7, table 3.8, and table 3.11. The ASN mappings were joined based on an inner join, and the reverse DNS lookup hostnames were joined based on an outer join. Here, the measurements from probes with no public source IP were dropped due to the inner join. The resulting table is *pair_medians_meta* and the schema can be found in table 3.17.

Table 3.17.: *Schema of pair_medians_meta table*

Field	Description
unit_id	The ID of the measurement probe
src_v4 and src_v6	IP address of the source
src_asn_v4 and src_asn_v6	AS Number of the Source IP address
src_holder_v4 and src_holder_v6	Entity owning the AS number
dst_v4 and dst_v6	IP address of the destination
hostname_v4 and hostname_v6	Hostname of destination
dst_asn_v4 and dst_asn_v6	AS Number of the Destination IP address
dst_holder_v4 and dst_holder_v6	Entity owning the AS number
m_ttl_v4 and m_ttl_v6	Median TTL to reach the destination
m_rtt_v4 and m_rtt_v6	Median RTT of the packet
m_ttl_delta	Median ΔTTL
m_rtt_delta	Median ΔRTT

3.3. SpeedTest Dataset Overview

Description

We further investigated and analyzed the Speedtest dataset which is measurements of metrics towards Measurement Lab servers, therefore we will call it M-Lab Dataset from now onwards. Similar to Netflix, the M-Lab dataset also contains tables which help us evaluate the *throughput and TTL* measurements. We will compare the performance of both Netflix and M-Lab servers, and the dataset collection was started in September 2014 and is running till date. The dataset consists of few tables which aren't relevant to this study, these are namely *httppostmt, httppostmt6, httpgetmt_status, httpgetmt6_status, httppostmt, httppostmt6_status, msmpoint, and traceroute_status*. The M-lab dataset can be found at /data/akapoor/speedtest.db and the schema of the used tables are mentioned below. Note, we aren't specifying all the fields present in the table, but only the ones we are using for our analysis to maintain readability and context. The *httpgetmt and httpgetmt6* tables are used to compare the throughput of M-Lab media servers with Netflix OCA servers. Here, the relevant field is *bytes_sec*, and the schema can be found in table 3.18 and table 3.19 for IPv4 and IPv6 address family's respectively.

Table 3.18.: *Schema of the httpgetmt table*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) of the measurement as "YYYY–MM-DD HH:MM:SS"
target	M-Lab IPv4 hostname
address	M-Lab IPv4 address
bytes_sec	Download speed in bytes/sec after the prebuffering finished
successes	1 if the test runs for the full duration (may have stalls though) i.e. it was not aborted and stepped down
failures	1 if the test was aborted for some reason

Table 3.19.: *Schema of the httpgetmt6 table*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) of the measurement as "YYYY–MM-DD HH:MM:SS"
target	M-Lab IPv6 hostname
address	M-Lab IPv6 address
bytes_sec	Download speed in bytes/sec after the prebuffering finished
successes	1 if the test runs for the full duration (may have stalls though) i.e. it was not aborted and stepped down
failures	1 if the test was aborted for some reason

We also wanted to compare the path lengths of Netflix and M-Lab for the corresponding throughputs, thus we used the M-Lab's *traceroute* table for this. Note here that we are following the same strategy [35] did in his study and thus, the following tables and aggregation is based on his work. Also to note, as we already explained these tables and aggregation strategy in section 3.2. Therefore, the reader can skip the following section as it provides the same description and information.

The M-Lab dataset has the *traceroute* table which consists of TTL and RTTs values to the M-Lab media servers. The traceroute measurements were started in May 2016 and are running till date. The schema of the *traceroute* table can be found in table 3.20.

Table 3.20.: *Schema of the traceroute and traceroute-filtered table*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) of the measurement as "YYYY–MM-DD HH:MM:SS"
version	Scamper version used for the measurement
source	IP address of the source
destination	IP address of the destination
method	Traceroute variant
status	Measurement status
ttl	TTL of the packet
endpoint	IP address of the intermediate hop
rtt	RTT of the packet
location_id	Internal ID denoting geographical location of the probe

We collected the public source IPs of the probes from the SamKnows [90] dashboard and queried the RIPEstat [88] API to get the AS number and holder name for these public *source* IPs, *endpoint* IPs, and *destination* IPs. The public source IPs we collected can be found at /metadata/sources.csv and we were only able to collect the IPv4 addresses. We used the IPv6 addresses available from the *traceroute* table. The resulted M-Lab tables can be found in table 3.21, table 3.22, and table 3.23.

Table 3.21.: *Schema of the dst_asn_mapping table*

Field	Description
ip	IP address of the destination
asn	AS Number of the Source
holder	Entity name which owns the AS

Table 3.22.: *Schema of the src_asn_mapping table*

Field	Description
unit_id	The ID of the measurement probe
src_ip	Public IP address of the source
asn	AS Number of the Source
holder	Entity name which owns the AS

Table 3.23.: *Schema of the endpoint_asn_mapping table*

Field	Description
endpoint	Endpoint or Intermediate hop IP address
asnnum	AS Number of the Endpoint
holder	Entity name which owns the AS

We wanted to find out the AS type for the individual endpoints along the destination paths. CAIDA [26] does classify every AS according to three types i.e. *Transit, Content, and Enterprise*. These CAIDA classifications are based on their machine learning algorithm and the PeeringDB classifications [76]; table 3.24 contains the schema for the CAIDA dataset.

Table 3.24.: *Schema of the as_types table*

Field	Description
asn	AS number, assigned by Internet Assigned Numbers Authority (IANA)
source	Used classifier i.e. CAIDA_class or peerDB_class
type	Business type of the AS

Reverse DNS lookup of the destination IPs of the M-Lab media servers are contained in table 3.25, to help us further classify the IPs. We again used RIPEstat [88] to get this information, and as result table *hostnames* is created. Although there are around 158 destination IPs only 45 IPs could be looked up.

Table 3.25.: *Schema of the hostnames table*

Field	Description
ip	IP address of the destination
hostname	Reverse DNS hostname

Aggregation

We aggregated the original *traceroute* table to help us in our analysis. Our motivation for doing this aggregation was to get a detailed overview of the dataset and to avoid unnecessary repetition of steps required for dataset filtering. All the below-mentioned tables were pushed onto the same database i.e. at /data/akapoor/speedtest.db. We will briefly explain the steps that we followed in this aggregation and filtering. As already informed, we followed the same strategy that Viet [35] followed in his study. The reason for this is that Viet used the *Youtube* dataset which has the similar schema as *M-Lab* dataset, and both datasets are collected using SamKnows probes [90].

We started out by filtering the *traceroute* table, we first removed the rows which contained NaN values or the probes which don't contain measurements for both the IP address families i.e probes with *unit_id* 525884 and 658929. We further removed measurements from Hurricane Electric Probes (refer [79], [47], [34], [21], and [101]), due to the reasons discussed in chapter *Related Work*. Therefore, we removed the rows which contained measurements from probes which belonged to Hurricane Electric i.e. AS 6939 (refer table 3.8 for such probes). The dataset contains a lot of duplicated rows, originated due to various reasons and we had to drop these rows to remove redundancy. The original *traceroute* table had 12365994 rows and after performing all these operations the rows were reduced to 3522311. We named the resulting table *traceroute-filtered* which contains the non-duplicate rows with measurements from dual-stacked probes only, and its schema is depicted in table 3.20.

We used an additional table which we didn't explicitly pushed to our database, but it contains the *COMPLETED* i.e. status == 'COMPLETED' results (refer **??** for more information). We named this table *completed-traceroute* and it contains the whole TTL measurements i.e. where TTL increments by one for the desired path sequence. We further rounded the *dtime* field to hours and dropped the columns which were not relevant for our analysis. The schema for the *completed-traceroute* table can be found in table 3.26.

Table 3.26.: *Schema of completed-traceroutemlab table*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) of the measurement as "YYYY–MM-DD HH:00:00"
source	IP address of the source
destination	IP address of the destination
ttl	TTL for this path
rtt	RTT of the packet

After considering the "COMPLETED" results, we further filtered the data based on matching IP addresses in the *destination* and *endpoint* fields. We then selected the maximum TTL values based on the grouping of *unit_id, dtime, source and destination* fields, as done by Viet [35]. We did this filtering to make sure that we only analyze the traceroute results which were completed and actually reached the destination. Also, multiple rows for a single path were reduced to only one row with the maximum TTL value and the RTT value for the whole path. The grouping fields that we selected above ensures to filter only one traceroute measurement per path i.e. for a specific *source* and a *destination*.

We split-up this table for IPv4 and IPv6 addresses and the schema for these tables are depicted in table 3.27.

Table 3.27.: *Schema of traceroute_v4 and traceroute_v6 tables*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) of the measurement as "YYYY–MM-DD HH:MM:SS"
source	IP address of the source
destination	IP address of the destination
max(ttl)	Maximum TTL for this path
rtt	RTT of the packet

To further compare the two address family's i.e. IPv4 and IPv6, we merged both the tables table 3.27. As the data was collected every hour, concurrently for IPv4 and IPv6, *unit_id and dtime* could be used to identify traceroute measurements pairs for both the address families.

We rounded the time in *traceroute_v4* and *traceroute_v6* to nearest hour, to identify such measurement pairs. After doing this, we then merged the two tables group by *unit_id* and *dtime(rounded)* fields, and computed the deltas for the max(TTL) and RTT fields. The resulting schema can be found in table 3.28, where fields with the same description are shown side-by-side.

Table 3.28.: *Schema of deltas table (difference of traceroute_v4 and traceroute_v6 tables)*

Field	Description
unit_id	The ID of the measurement probe
dtime	Timestamp (UTC) of the measurement as "YYYY–MM-DD HH:MM:SS"
source_v4 and source_v6	IP address of the source
destination_v4 and destination_v6	IP address of the destination
max(ttl)_v4 and max(ttl)_v6	Maximum TTL for this path
rtt_v4 and rtt_v6	RTT of the packet
ttl_delta	ΔTTL
rtt_delta	ΔRTT

We wanted to analyze data irrespective of the time, and therefore we removed the *dtime* field to achieve this. The *traceroute_v4 and traceroute_v6* tables were further grouped by *unit_id, source, and destination* fields, which denote recurring paths from the source IP address to the destination IP address for the whole period, and we computed the median TTL and median RTT values for this group. The tables *path_medians_v4* and *path_medians_v6* achieved after this operation are shown in table 3.29.

Table 3.29.: *Schema of path_medians_v4 and path_medians_v6 tables*

Field	Description
unit_id	The ID of the measurement probe
source	IP address of the source
destination	IP address of the destination
median(ttl)	Median TTL for this path
median(rtt)	Median RTT of the packet

The *deltas* table was also filtered and we removed the *dtime* field to analyze the data irrespective of the time factor. We considered the *unit_id, destination_v4, and destination_v6* grouping to achieve the measurements pairs and calculated the median TTL and median RTT for these measurement pairs. We further calculated the deltas for

the median TTL and median RTT, and the resulting schema can be seen in table 3.30.

Table 3.30.: *Schema of pair_medians table*

Field	Description
unit_id	The ID of the measurement probe
source_v4 and source_v6	IP address of the source
destination_v4 and destination_v6	IP address of the destination
max(ttl)_v4 and max(ttl)_v6	Median TTL for this path
rtt_v4 and rtt_v6	Median RTT of the packet
ttl_delta	Median ΔTTL
rtt_delta	Median ΔRTT

We created a table containing the metadata from the additional tables table 3.21, table 3.22, and table 3.25. The ASN mappings were joined based on an inner join, and the reverse DNS lookup hostnames were joined based on an outer join. Here, the measurements from probes with no public source IP were dropped due to the inner join. The resulting table is *pair_medians_meta* and the schema can be found in table 3.31.

Table 3.31.: *Schema of pair_medians_meta table*

Field	Description
unit_id	The ID of the measurement probe
src_v4 and src_v6	IP address of the source
src_asn_v4 and src_asn_v6	AS Number of the Source IP address
src_holder_v4 and src_holder_v6	Entity owning the AS number
dst_v4 and dst_v6	IP address of the destination
hostname_v4 and hostname_v6	Hostname of destination
dst_asn_v4 and dst_asn_v6	AS Number of the Destination IP address
dst_holder_v4 and dst_holder_v6	Entity owning the AS number
m_ttl_v4 and m_ttl_v6	Median TTL to reach the destination
m_rtt_v4 and m_rtt_v6	Median RTT of the packet
m_ttl_delta	Median ΔTTL
m_rtt_delta	Median ΔRTT

Data Analysis

CHAPTER 4

IPV4 VS IPV6 - WHICH PERFORMS BETTER?

The previous section consists the description of the Netflix dataset and tables schema associated to it. It also talks about the methodology used in collecting the data. This chapter covers the analysis and the findings associated to the Netflix dataset. The Appendices consists of more details about the analysis and the used jupyter notebooks are described in the chapter Reproducibility.

4.1. Error and Success Rate

Error Rate is an important aspect when considering the amount of errors in the dataset. The Netflix table consists of attributes like *error* which gives us the error name and *error_msg* which describes the reasons for it. table 3.2 gives a detail description about the errors and their actual reasons. We computed the number of different errors and found out that 74.6% cases in the dataset are without errors and the rest 25.4% cases had different errors. table 4.1 gives us the percentage of each error in the dataset, and we can see that "STALLED_WILL_STEP_DOWN" is the most occurring error (around 13.50%), followed by "CONNECTION_API_ERROR" and the others.

Error Occurrence Rate

We wanted to see are these errors distributed along the whole timeline. For calculating the error occurrence rate along the whole time period, we created a new field name *Occurrence*, which consists of a single value, which is, '1'. We calculated the number of errors occurring per day, that is, sum of every error occurrence in a day upon the total number of cases i.e. error plus non-errors for that day.

For example, There are 100 data points for a day, 70 of them had NO_ERROR, 30 of them had different errors. So, error occurrence rate for that day will be 30%. fig. 4.1 depicts the error occurrence rate during the whole time period. We can see that the occurrence rate for error mostly lies between 15%-45% for a single day. We also calculated the individual error occurrence rate for each error, and figures for each

Table 4.1.: *Precentage of Different Errors*

Errors	**Presence Percentage**
NO_ERROR	74.60
STALLED_WILL_STEP_DOWN	13.50
CONNECTION_API_ERROR	4.50
NETWORK_CONTENT_ERROR	4.30
NETWORK_API_ERROR	2.30
DNS_RESOLUTION_CONTENT_ERROR	0.40
DNS_RESOLUTION_API_ERROR	0.30
CONNECTION_CONTENT_ERROR	0.20
STALLED_ON_FINAL	0.01

one of them is in Appendices. We did similar analysis to check how these errors are distributed across each probe, and we find out from fig. 4.4 that the errors are evenly distributed across each probe and it's not that only some probes are generating errors.

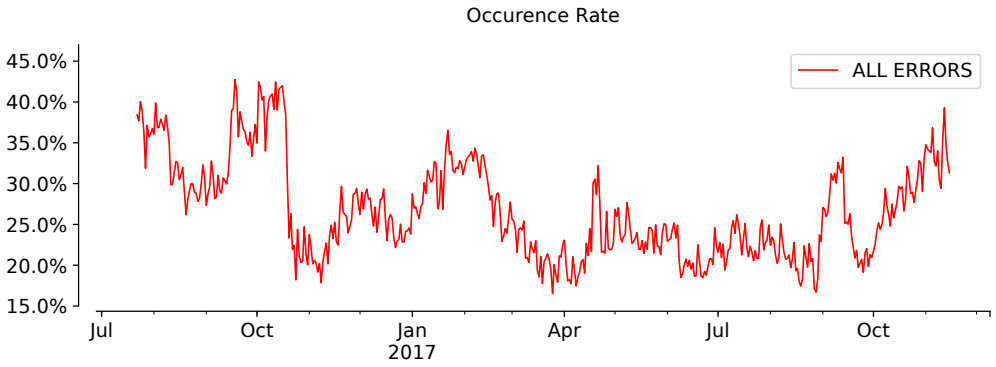

Fig. 4.1.: *Error occurrence rate for all errors during the whole timeline. The occurrence rate mostly lies between 15%-45% for a single day.*

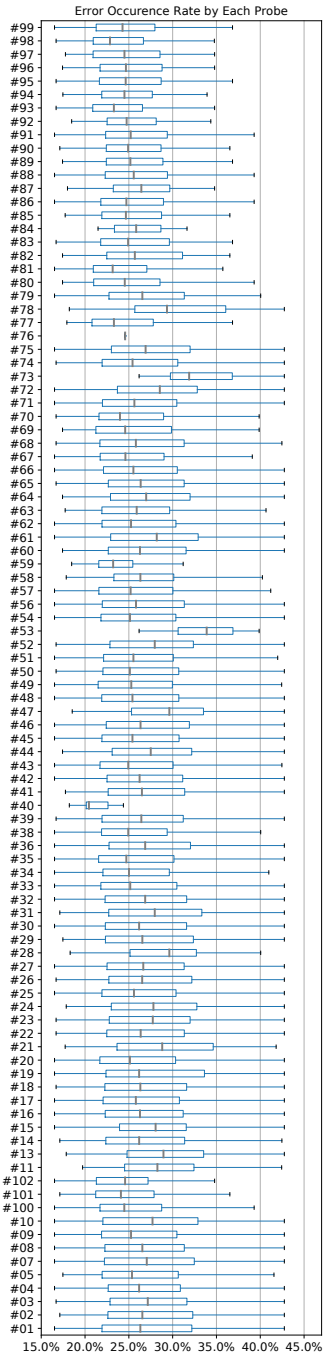

Fig. 4.2.: *(a)*

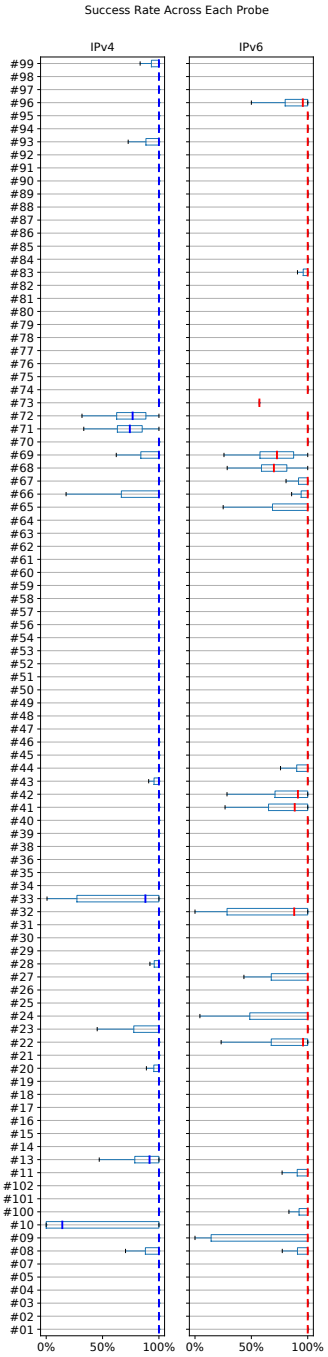

Fig. 4.3.: *(b)*

Fig. 4.4.: *(a) Boxplot of error occurrence rate for each probe. It signifies that the error occurrence rate is evenly distributed across each probe. (b) Boxplots of success rate across each probe over IPv4 and IPv6. Around 30 probes have varying success rate.*

Success Rate

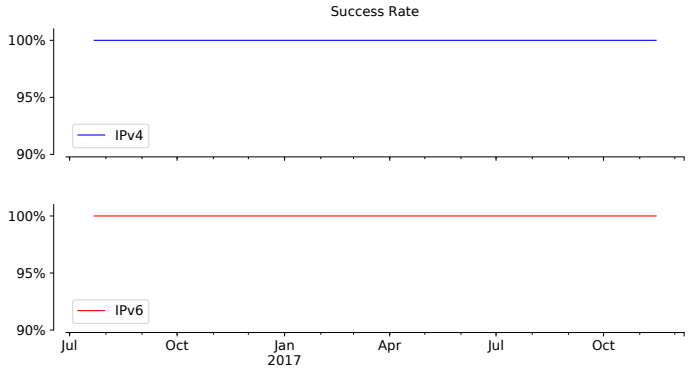

Fig. 4.5.: *Time series of success rate over IPv4 and IPv6. We are taking the median aggregate of success rate across all the probes on each day since July, 2016. Median Success rate for both the address families remains constant around 100% over the entire time duration.*

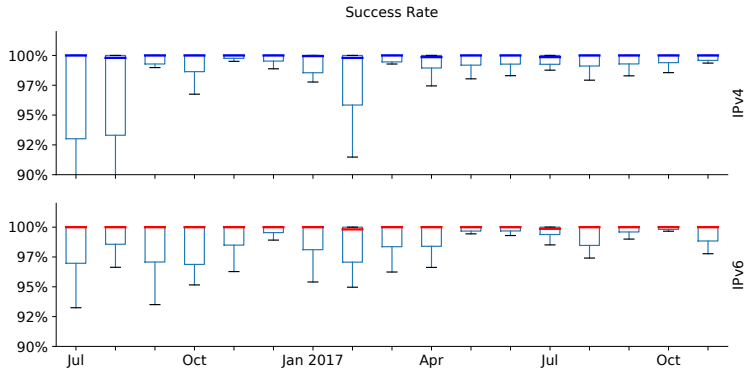

Fig. 4.6.: *Boxplot of success rate over IPv4 and IPv6. The median line is close to 100% for both the address families and resembles the fig. 4.5. Variation among different months can be seen here.*

After considering the error rate, it would be goood to look at the success of the test. We will now consider the success rate over both the address families i.e. IPv4 and IPv6. For success rate we will be considering the *successes* field, described in the table 3.1 in chapter 3. We will start with the similar analysis Vaibhav Bajpai et al. did in [14]. They define success rate as the successful cases upon the total number of cases in the dataset. Downloading a stall free version of the video indicates a successful test, and if a stall occurs the test restarts by stepping down to a lower bitrate and reports an error. Thus, we are not considering the cases where this is a stall i.e. we are excluding cases when the *error* is "STALLED_WILL_STEP_DOWN" and "STALLED_ON_FINAL".

fig. 4.5 shows us the time series of median success rates across all the probes on each day since July, 2016, when there isn't a stall. As we can see from the plot, median success rate remain constant around 100% for both the address families.

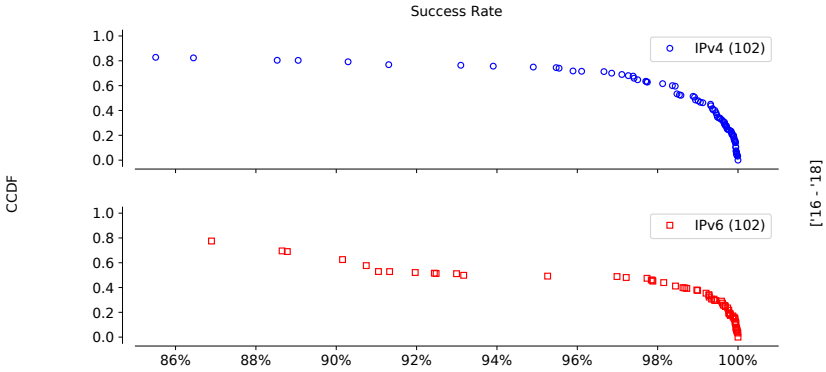

Fig. 4.7.: *CCDF of success rate over IPv4 and IPv6, when there isn't a stall. Around 79% of the probes achieve success rate of more than 90% for IPv4, while for IPv6, only around 63% of the probes were able to achieve this success rate.*

Median aggregate ensures that the results doesn't get biased due to a specific probe or specific vantage point. We also plotted all the datapoints and not just the medians, and the results can be seen in Appendices. To further investigate the success rate we plotted the boxplots to see the variation among different months. We have rounded the time to the nearest month, and we can see in fig. 4.6 that the median success rate is close to 100% for both the address families but the individual distribution varies across time. We further investigate the distribution of success rate over IPv4 and IPv6, when there isn't a stall. We calculated the success rate and then took the CCDF of it. fig. 4.7 shows the CCDF of success rate over IPv4 and IPv6. Around 79% of the probes achieve success rate of more than 90% for IPv4, while for IPv6, only around 63% of the probes were able to achieve this success rate. It can be seen that probes achieve a lower success rate over IPv6. On careful investigation, we found out that the reason for lower success rates over IPv6 is due to DNS Resolution Error and Network Errors. Moving forward we will work on cases where both the address families reports success. Appendices also discusses the CDF for two individual months i.e. August 2016 and October 2017. We also calcualted the success rate across each probe. fig. 4.4(b) gives us the bocplot of success rate over IPv4 and IPv6 across each probe. We can see from the median values that there are around 30 probes which have varying success rate over the time line. We did time series analysis of success rate across IPv4 and IPv6 removing these outliers and the results can be seen in the Appendices.

Success Rate by Region

Table 4.2.: *Regions with Number of Probes*

Regions	#Probes
European Region	60
Region of the Americas	32
Western Pacific Region	9
African Region	1

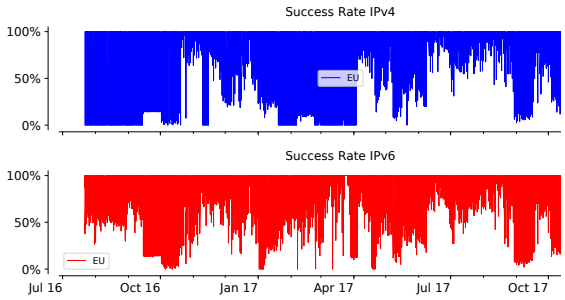

Fig. 4.8.: *(a)*

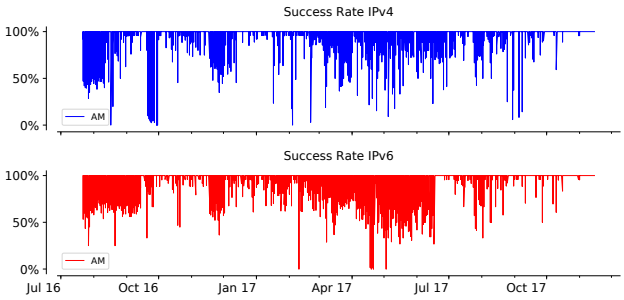

Fig. 4.9.: *(b)*

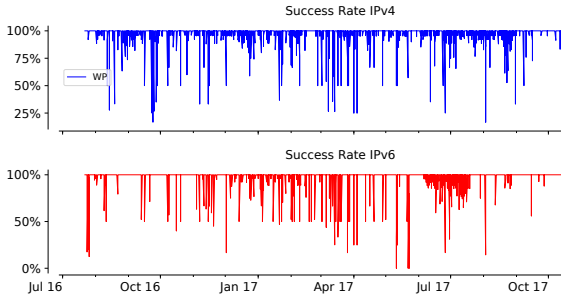

Fig. 4.10.: *(c)*

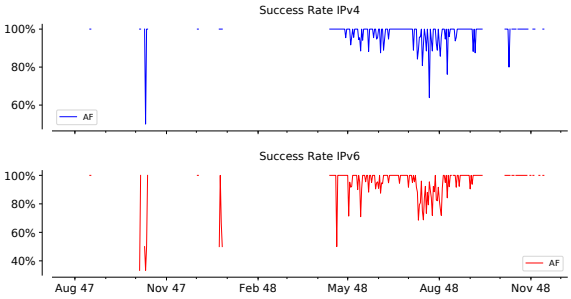

Fig. 4.11.: *(d)*

Fig. 4.12.: *(a) Time series of success rate over IPv4 and IPv6 for European Region, (b) Time series of success rate over IPv4 and IPv6 for Region of Americas, (c) Time series of success rate over IPv4 and IPv6 for Western Pacific Region, (d) Time series of success rate over IPv4 and IPv6 for African Region. We can see the variation of success rate among different regions. Success rate in Region of Americas seems to quite better compared to European Region.*

We wanted to see how success rate over IPv4 and IPv6 performs by different geographical regions of the world. The metadata of the probes is listed at [12], provides

us the geographical information of the probes as well. There is a *location* field in the metadata which tells us the city or organisation where the probe is located. We did some manual analysis and collected the different regions based on this *location* field. We then created a new field named *region* containing the geographical region where the probe is located, based on the World Health Organization defination of different geographical regions of the world [75]. table 4.3 gives the number of SaKnows probes in different regions of the world. The probes are situated in four different regions. We

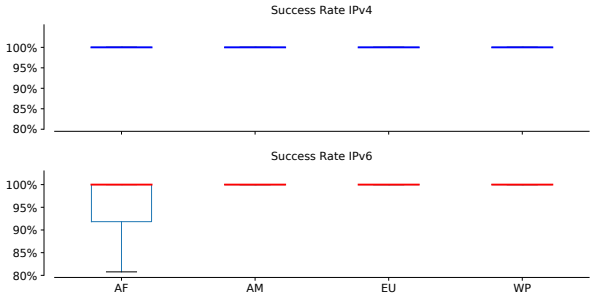

Fig. 4.13.: *Boxplot of success rate over IPv4 and IPv6 for different regions. The performance over IPv4 and IPv6 seems equivalent for different regions, except for African region where first quartile of success rate is around 80%.*

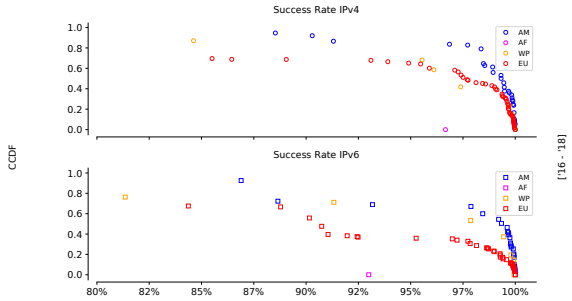

Fig. 4.14.: *CCDF of success rate over IPv4 and IPv6 for different regions. The success rate for Region of the Americas is better than European Region over both the address families.*

followed the same steps as what we did in the previous section. Here, we merged the *region* field based on the name of the probe. After merging we grouped the data based on *dtime*, *address family (af)*, *probe name* and *region*, aggregating the *success* field based on this grouping. We then simply calcualted the success rate for all these regions and then plotted them as a time series. fig. 4.12 gives the time series of success rate over IPv4 and IPv6 for different regions, and as can be seen Success rate in Region of Americas seems to quite better compared to European Region. To further investigate we plotted

the boxplot of success rate over IPv4 and IPv6 for different regions in fig. 4.13.The performance over IPv4 and IPv6 seems equivalent for different regions, except for African region where first quartile of success rate is around 80%. We wanted to see the distribution of success rate over IPv4 and IPv6 for these regions. fig. 4.14 gives the CCDF of success rate over IPv4 and IPv6 for different regions. We can see that the success rate for Region of the Americas is better than European Region over both the address families.

Success Rate by Network Type

Table 4.3.: *Network Types with Number of Probes*

Network Type	**#Probes**
RESIDENTIAL	79
NREN	8
RESEARCH	2
BUSINESS	5
DATACENTER	3
OPERATOR LAB	4
IXP	1

After going through the success rate for different geographical regions, we also wanted to see the success rate over IPv4 and IPv6 for different network types. There is a *Type* field in the metadata of the probes [12], which gives us the network type of the probe. There are seven different types listed in the metadata and **??** shows us these seven network types and the number of probes in each type. As can be seen most of the probes are "RESIDENTIAL" probes. Also, we grouped the "NREN" and "RESEARCH" together into "RESEARCH", and "BUSINESS" and "DATACENTER" into "BUSINESS" as per [14]. We created a new field named *type* and merged this field based on the *name* of the probe. Here, we are grouping on *dtime*, *address family (af)*, *probe* and *type* field. We then simply calculated the *success rate* over IPv4 and IPv6 for different network types and plotted them as a time series. fig. 4.20 shows the timeseries of success rate over IPv4 and IPv6 for different network types, and we can see that the "RESIDENTIAL" probes are influencing the success rate over IPv4 and IPv6 across all probes fig. 4.5.

To further investigate, we plotted the boxplot of success rate over IPv4 and IPv6 for different network types in fig. 4.21. It can be seen that boxplot of usccess rate is around 100% over IPv4 and IPv6, except "NREN" and "RESEARCH" probes where the success rate is distributed. We also wanted to see the distribution of success rate over both the

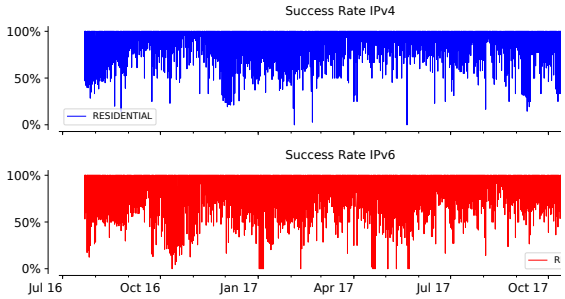

Fig. 4.15.: *(a)*

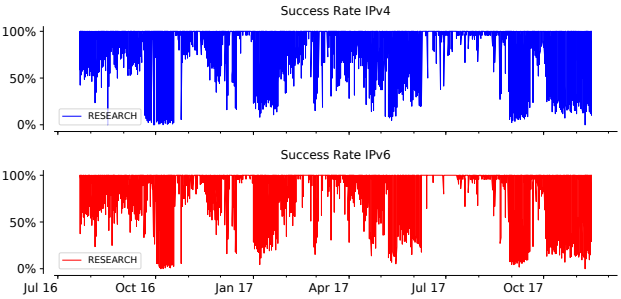

Fig. 4.16.: *(b)*

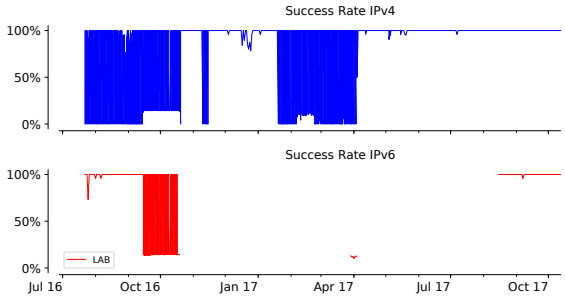

Fig. 4.17.: *(c)*

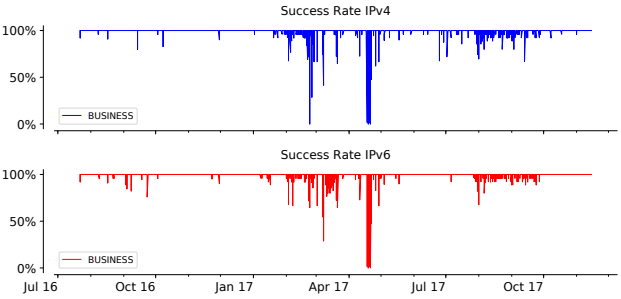

Fig. 4.18.: *(d)*

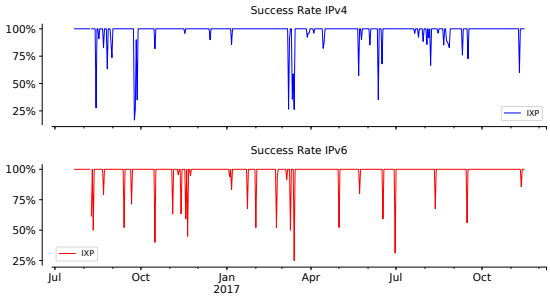

Fig. 4.19.: *(e)*

Fig. 4.20.: *(a) Time series of success rate over IPv4 and IPv6 for RESIDENTIAL Probes, (b) Time series of success rate over IPv4 and IPv6 for RESEARCH Probes, (c) Time series of success rate over IPv4 and IPv6 for LAB Probes, (d) Time series of success rate over IPv4 and IPv6 for BUSINESS Probes. (e) Time series of success rate over IPv4 and IPv6 for IXP Probes. We can see the variation of success rate among different regions. Success rate in Region of Americas seems to quite better compared to European Region.*

address families for different network types. fig. 4.22 gives us the CCDF of success rate over both families for the probes in different network types. CCDF of success rate

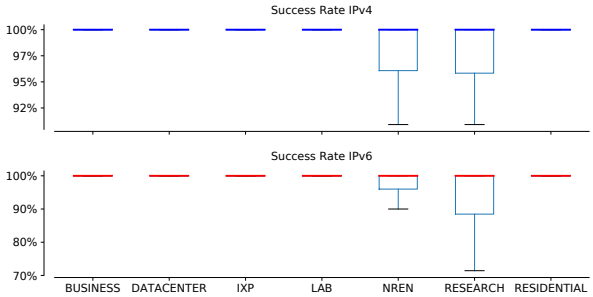

Fig. 4.21.: *Success Rate Boxplot per each Network Types of Probes*

for "RESIDENTIAL" probes resembles the CCDF of success rate across all probes as in fig. 4.7.

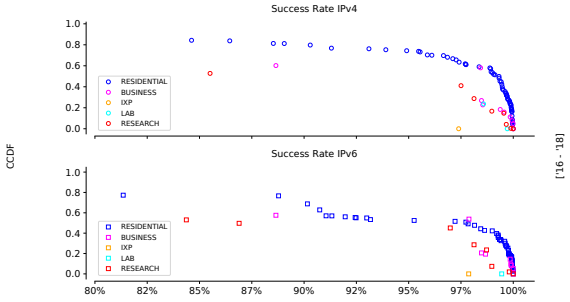

Fig. 4.22.: *Success Rate CCDF for probes over different Network Types*

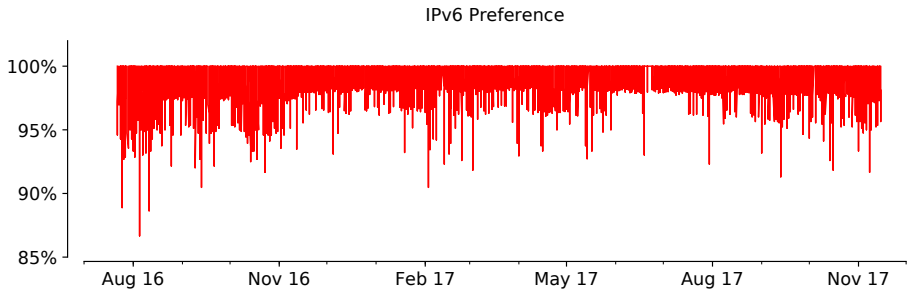

Fig. 4.23.: *The timeseries shows the effects of HE algorithm and as per this the TCP connections were preferred for more than 90% of the time during the whole time period of the dataset.*

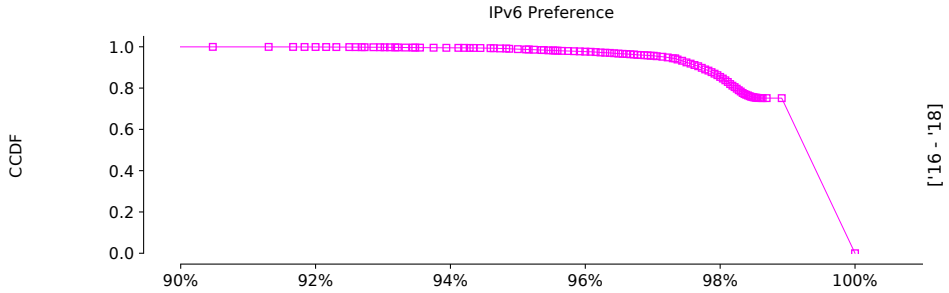

Fig. 4.24.: *CCDF of TCP connection establishment preference over IPv6. The TCP connections over IPv6 to the OCA server are preferred around 93% of the time.*

4.2. IPv6 Preference

We are measuring TCP connect times (*connect_time* field in the table **??**) to the Netflix Open Connect Appliance (OCA) server in (refer Sequence diagram here). Vaibhav Bajpai et al. in [14] measured the TCP connect times for Youtube, and we followed their analysis approach here. TCP connect times doesn't consider the DNS name resolution time, and is based on the connect() system call completion time. We are doing this as applications running on dual-stacked hosts should prefer connections over IPv6, as per the default address selection policy for IPv6 [84]. RFC 6724 makes getaddrinfo() resolve DNS names in an order that mandates an IPv6 upgrade path for the applications, whereas, Happy Eyeballs (HE) algorithm [83] allows these applications to resolve to IPv4 if the connectivity over IPv6 is not good. As per HE algorithm the connectivity is not considered good when the connection does not complete within 300 ms. We first filtered the data as per the IP address family and then merged them to calculated the preference. For calculating the preference we used the HE algorithm implementaiton by [14]. We clustered the columns by *dtime* and used the algorithm to calculate the

preference. fig. 4.24 shows the outcome of HE algorithm and as per this connections over IPv6 are preferred for around 93% of the time. We did the same steps to figure out how IPv6 preference is treated over the whole time period and for this we plotted fig. 4.23 where we plotted the preference against the time, this justify's the above results as it shows that IPv6 is being preferred more than 90% of the time.

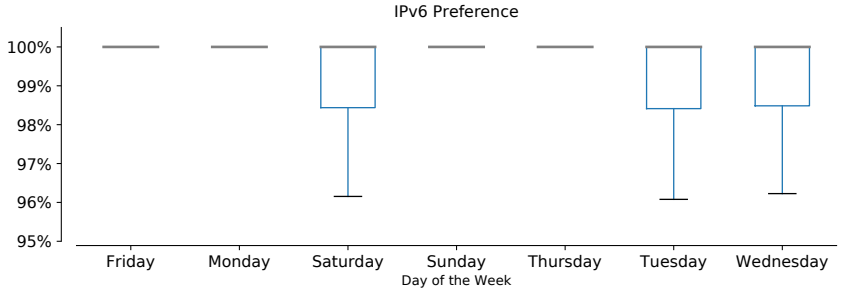

Fig. 4.25.: *Boxplot shows that IPv6 is preferred around 100% of the time, but not on Saturdays, Tuesdays, and Wednesdays.*

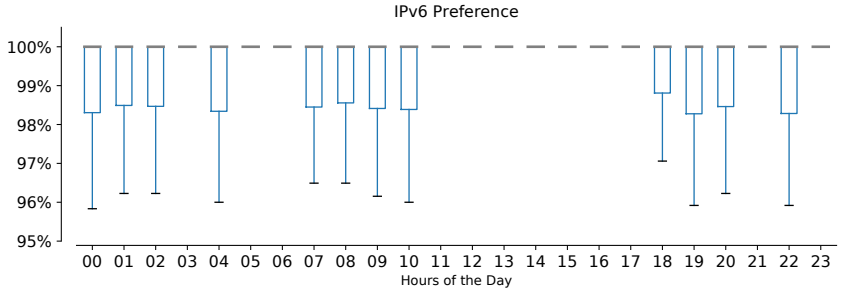

Fig. 4.26.: *Boxplot shows that TCP connect time over IPv6 is preferred around 100% during most hours of the day.*

To get more deeper insights into the TCP connect times over IPv6, we did daily and hourly analysis. For this we first created a new field *days* and got the days of the week from the *dtime* field. We performed the steps mentioned before for calculating the IPv6 preference and clustered by *dtime* field. As a result, **??** shows that IPv6 is preferred approximately 100%, but on Saturdays, Tuesday and Wednesday. The reason for this could be that these days are the peak days when users are requesting content form Netflix OCA servers. Similarly, the hourly boxplot in fig. 4.26 shows that there are few hours which can be considered peak hours for watching content on Netflix as the preference is not 100% for these hours of the day.

4.3. Latency and Delay

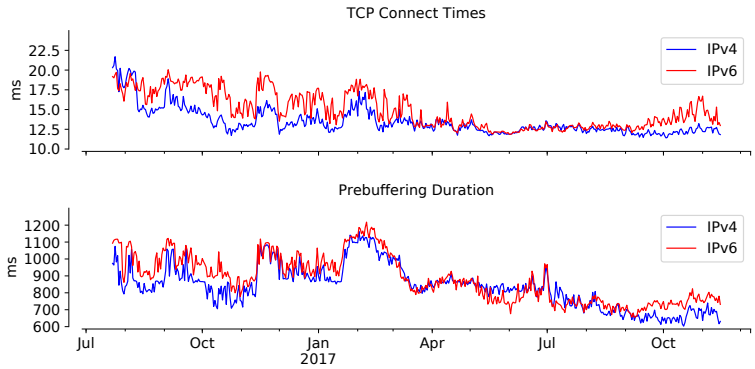

Fig. 4.27.: *Timeseries depicting the TCP connect times (connect_time field) and Prebuffering Duration (prebuffering_duration field) for IPv4 and IPv6. We are applying a median aggregate here and graph resembles similar curves over both the address families.*

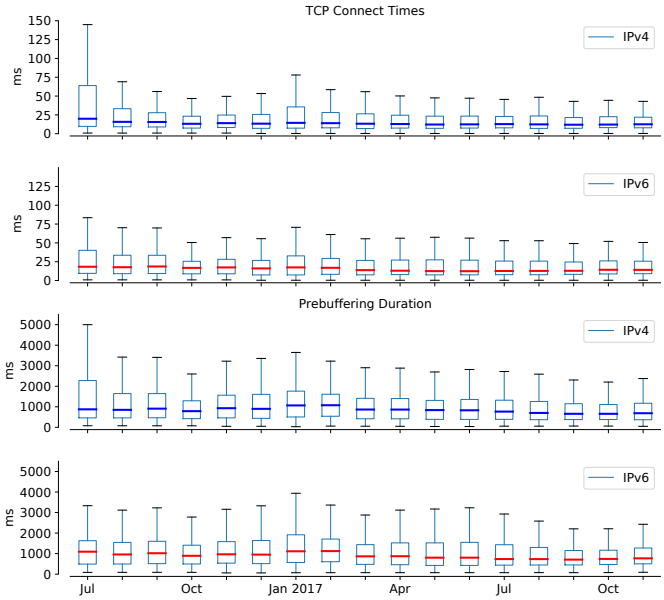

Fig. 4.28.: *Boxplot showing the TCP connect times and Prebuffering Duration over both the familys. It resembles the results from the timeseries fig. 4.27.*

After checking the IPv6 preference, we now know that clients prefer streaming videos over IPv6. It would be good to investigate how IPv6 performance compares with

IPv4. Here, we are considering the *connect_time* and *prebuffering_duration* fields that are defined in table 3.1. We first want to see how TCP connect times and prebuffering durations perform for both the address families. fig. 4.27 shows the timeseries for both the address families, and we have group by *dtime* field and are considering the median aggregate here, so that a single vantage point doesn't bias the results. As we can see, the graph resembles similar curves for both the Address Family's. We have converted the connect time and pre-buffering duration to 'ms' to get a better understanding. The median TCP connect times lies between 10-25 ms and the median pre-buffering duration lies between 600-1200 ms for both the address families. To get more clear view of the delays, we plot the boxplots for both the address familys. We filter the data with respect to the address family i.e. IPv4 or IPv6, and now we can see in the figure fig. 4.28 that the graphs resembles the timeseries fig. 4.27. We further investigate the distribution of delays, here also, we filter the data along IPv4 and IPv6 and then take the CDF of the desired attributes. fig. 4.29 shows the CDF of connect times and prebuffering durations. Although it resembles the timeseries and boxplots, but we cannot see much difference between IPv4 and IPv6 performance. Thus, it would be good to compare the deltas i.e. the difference between the two address family's.

Latency and Delay Deltas

Let *tc(y)* be the time taken to establish a TCP connection over IPv6 to a Netflix video and *tc(x)* be the time taken to establish a TCP connection over IPv4 to the same video. Similarly, *p(y)* be the prebuffering duration over IPv6 and *p(x)* be the prebuffering duration over IPv4 for accessing a Netflix video. Also, as already discussed in chapter 3, prebuffering duration takes into account DNS resolution times and TCP connect times. A user desires to get lower latency which can be achived through lower TCP connect times and lower prebuffering duration. To calculate latency over IPv4 and IPv6, we use the 4.1, where Δtc, and Δp are the differences in TCP connect times and Prebuffering duration respectively.

$$\Delta tc = tc(x) - tc(y)$$

$$\Delta p = p(x) - p(y) \tag{4.1}$$

To plot the deltas, we followed the same analysis [14] did. To plot the timeseries for this deltas, we calculate the median aggregate on the TCP connect times and the prebuffering duration across all probes on each day. fig. 4.30 shows the timeseries of median TCP connect times and prebuffering duration over IPv4 and IPv6 across all

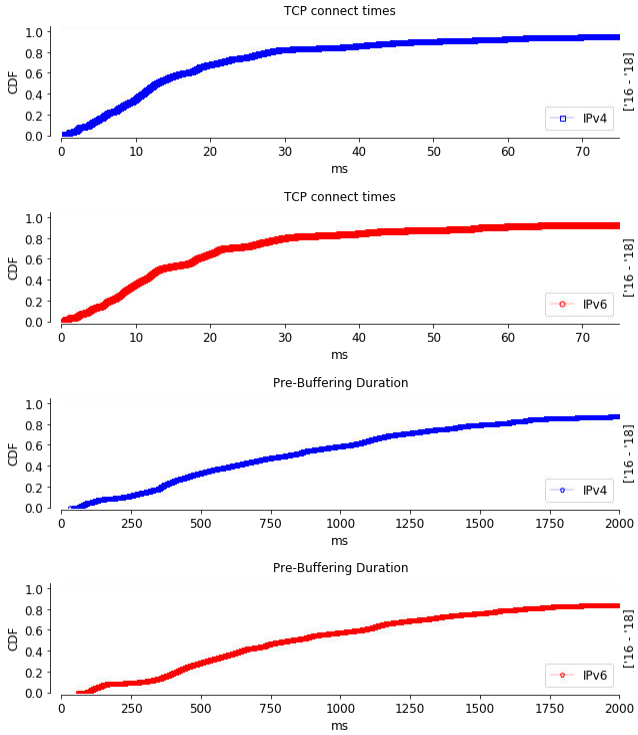

Fig. 4.29.: *CDF of TCP Connect Times and Prebuffering Duration for IPv4 and IPv6. We cannot see much difference between the address family's and it would be good to check the deltas here.*

probes. Here, it can be seen that the TCP connect times were initially higher for IPv4, but eventually decreased with time and remained consistent. Even though IPv6 tend to have higher TCP connect times, and only around 0.2 ms slower than IPv4, it plays very important role for the HE algorithm [83], as this the stage where HE algorithm decides on the address family which will be used for streaming the video. As the difference between IPv4 and IPv6 is less than 300ms, therefore IPv6 is preferred most of the time. Also, IPv6 has higher prebuffering durations (around 40ms or more). For boxplots in fig. 4.31, we have rounded the time to the nearest month. Here, we can see that the median resembles the time series fig. 4.30. To get more vivid analysis, fig. 4.32 shows us the distribution of difference in TCP connect times and prebuffering duration for the whole dataset. For calculating the CDF, we calcualted the deltas and then plotted their CDF. To compare the performance of IPv6 and IPv4, we are measuring the TCP connect times to the Netflix OCA (Open Connect Appliances) server, the distribution here shows that 57% of the connections are slower over IPv6 , with 13% of them at least 10 ms slower. For prebuffering duration which is the time to fetch 2 seconds of video at the specified bitrate from the Netflix OCA (Open Connect Appliances) server, the distribution shows that 69% of the connections are slower over IPv6.

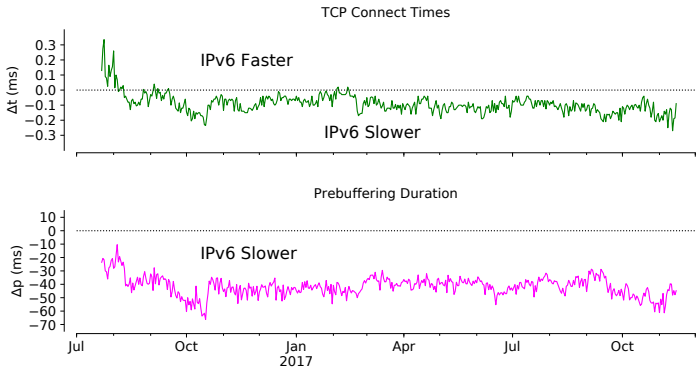

Fig. 4.30.: *Time series of difference for TCP connect times and prebuffering durations over IPv4 and IPv6 to Netflix. Latency of around 0.2 ms and higher prebuffering durations (aroun40 ms or more) are observed over IPv6.*

SKY UK Latency and Delays

We wanted to know how Netflix OCA performs compared to when the content server is in an Internet Service Provider (ISP). We already discussed in chapter 2 about the Open Connect Appliances, Netflix normally places them on the Internet Exchange Points (IXPs). We wanted to find out how these OCAs performs compares to when

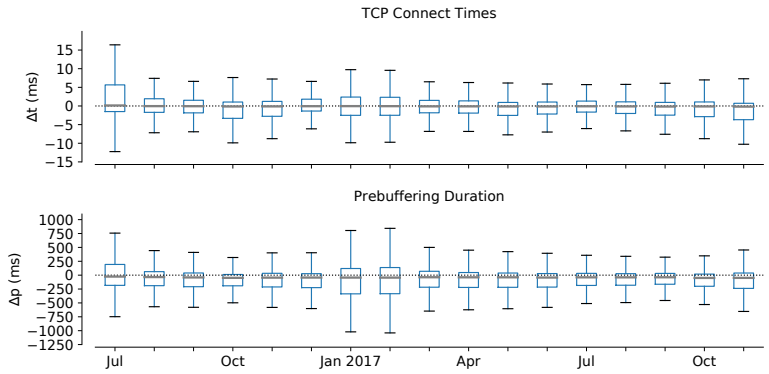

Fig. 4.31.: *Boxplots depicting the difference for TCP connect times and prebuffering duration. The median line here resembles the time series fig. 4.30.*

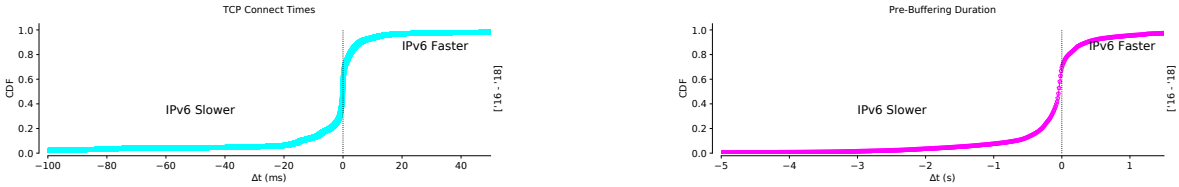

Fig. 4.32.: *CDF of difference of TCP connect times and Prebuffering Durations for IPv4 and IPv6. The distribution here shows that 57% of the connections are slower over IPv6 , with 13% of them at least 10 ms slower. The prebuffering duration distribution shows that 69% of the connections are slower over IPv6.*

the content is served from an ISP server. Netflix may be saving a lot of capital from placing OCAs at IXPs but are they compromising on the performance. fig. 4.33 gives us a boxplot of TCP connect times and prebuffering duration for Sky UK Limited. Here, we considered both the probe and the *target* from the same Autonomous System (AS) i.e. "BSKYB-BROADBAND-AS - Sky UK Limited", compared to the rest of the cases. As can be seen, fig. 4.33, the performance is pretty good when both the user and the *target* belongs to the same AS.

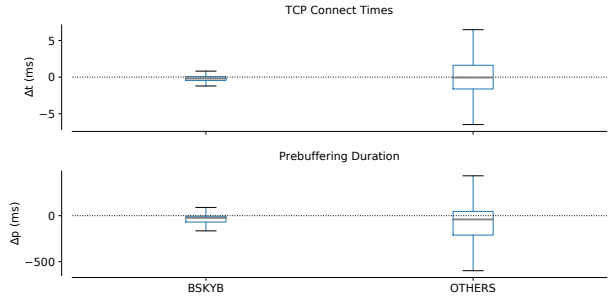

Fig. 4.33.: *Boxplot for TCP connect times and prebuffering duration for Sky UK Limited. We are comparing the case where both the probe and the target belongs to the "BSKYB-BROADBAND-AS - Sky UK Limited" AS compared to the rest of the cases. The performance over BSKYB is good compared to when the probe and target are in different ASes.*

4.4. Throughput

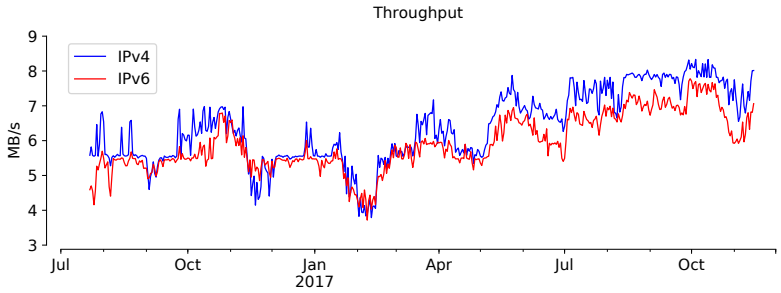

Fig. 4.34.: *Time series of Throughput for individual address family's i.e. IPv4 and IPv6. We are considering the bytes_sec field described in table 3.1. Both the families shows similar curves with throughput between 3-9 MB/s.*

After going through section 4.2 we know that clients prefer to stream videos on Netflix over IPv6. We also saw in section 4.3 that the clients take consistently higher TCP connect times and prebuffering durations (40ms or more) when compared to IPv4 (see fig. 4.30. To compare their performance, one important factor is achieved throughput, we will now look into this factor and compares the performance over both families. We first look into the individual performance of IPv4 and IPv6 and then will look into their deltas. fig. 4.34 gives the time series for both the address families, it shows that both the address families shows similar curves and that the throughput is increasing with time. We are considering the *bytes_sec* field from the Netflix **??**, and we are taking the median aggregate across all probes on each day. Also, the throughput

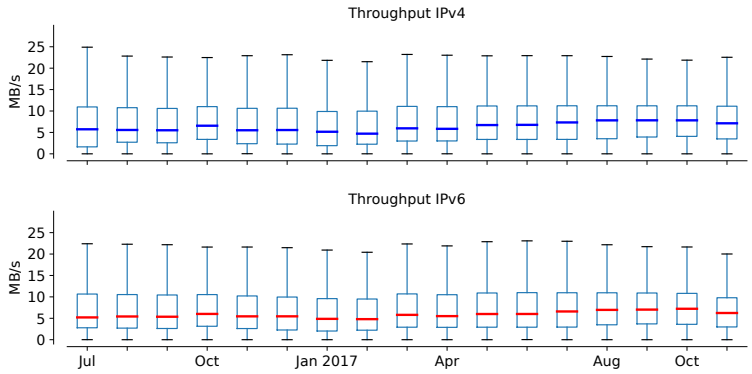

Fig. 4.35.: *Boxplot of throughput over IPv4 and IPv6. We can see the monthly variation here, also the median line resembles the time series in fig. 4.34.*

for IPv4 and IPv6 lies lies between 3-9 MB/s. To get more deeper insights we also did the boxplot for IPv4 and IPv6 throughput, and as can be seen in fig. 4.35 the monthly throughput variation over IPv4 and IPv6. The median line resembles the time series in fig. 4.34. We have converted the *bytes_sec* field into MB/s to get a more realistic view. fig. 4.36 shows the CDF of throughput over IPv4 and IPv6. We followed the same steps here, and plotted the CDF of *bytes_sec* field. The CDF for individual shows similar curves and does not give a clear difference in the performance of IPv4 and IPv6. We need to look into the deltas to see which address family performs better than the other.

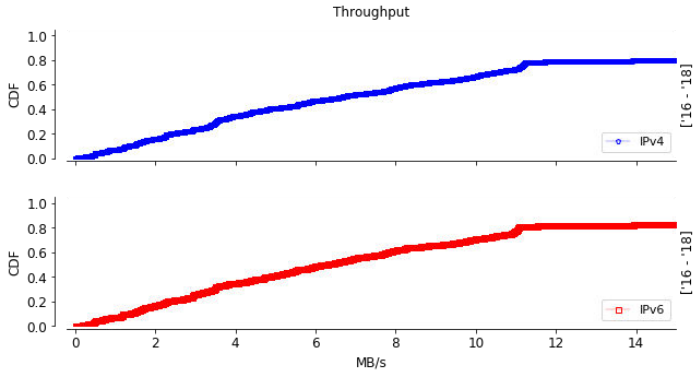

Fig. 4.36.: *CDF of throughput over IPv4 and IPv6, shows the similar curves over both the family's. We cannot see much difference in the performance here and thus would be good to plot the CDF for the difference betweent he two families.*

Throughput Deltas

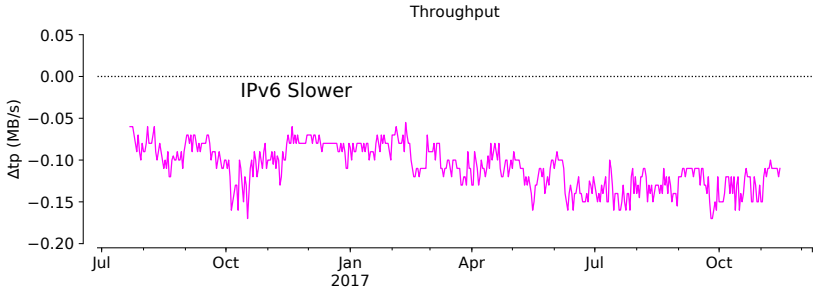

Fig. 4.37.: *Time series of difference of throughput over IPv4 and IPv6. We plot the median aggregate across all probes on each day. The achieved throughput is consistently lower over Ipv6.*

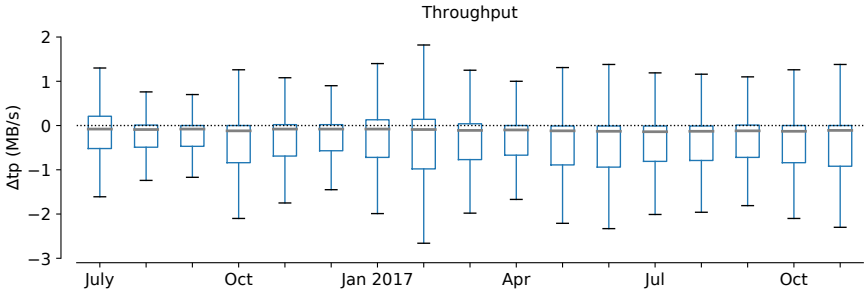

Fig. 4.38.: *Boxplot of difference of Throughput over IPv4 and IPv6. The figure also shows similar results as the time series fig. 4.37, depicting lower throughput over IPv6.*

Before starting with the deltas, we would live to define the terminologies we used to get the results. We are considering the *bytes_sec* field only and using the same terminilogy [14] used. We denote the throughput over IPv6 as *tp(y)* and throughput over IPv4 as *tp(x)*. We will use **??** where Δtp is the difference in achieved throughput over both the address families. To plot the deltas, we first start with the time series, and fig. 4.37 shows us the time series of median aggregae of throughput across all probes over each day. We can see that the difference is consistent over time and is lower for IPv6 over the whole duration. The negative value indicates a lower throughput over IPv6, also the difference liew between 0-0.2 MB/s. fig. 4.38 shows the boxplot of difference of throughput over IPv4 and IPv6. The median line shows similar curves as the time series fig. 4.37, depicting lower throughput over IPv6. To get a more clear picture about the performance, we plot the CDF of *bytes_sec* field. For CDF, we first compute the difference of throughput over IPv6 and IPv4 and then take the CDf of that difference.

fig. 4.39 shows the CDF of difference of throughput over IPv4 and IPv6. Around 73to a lower resolution video. This subsequently lowers the achieved throughput because th test the drops to the next highest bitrate and begin downloading from the start again. This enables the highest resolution bitrate streaming over a connection without any disruptions. Vaibhav Bajpai et al. in [14] further informs us that the test maintains a playout buffer and that it should wait before requesting any frames till the previous buffer is empty. The playout buffer can store a playable video of 40s only. fig. 4.40 further investigates the difference in throughput over IPv6 and IPv4 over different days of the week. The throughput over IPv6 is consistently lower on all the days.

$$\Delta tp = tp(y) - tp(x) \tag{4.2}$$

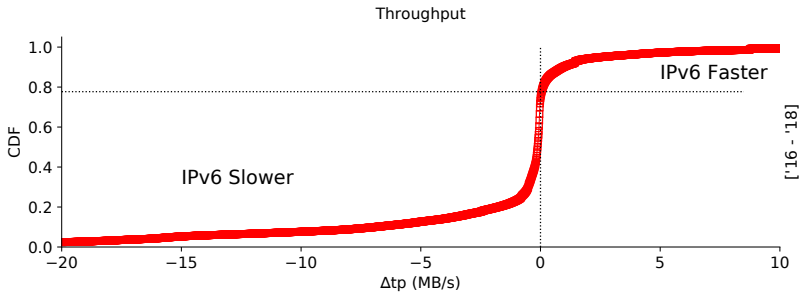

Fig. 4.39.: *CDF shows the distribution of difference of throughput over IPv6 and IPv4. Around 73% of the times, IPv6 achieved lower throughput.*

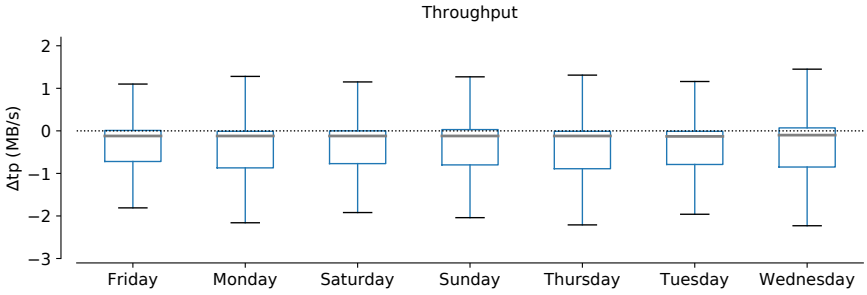

Fig. 4.40.: *Boxplot of difference of throughput over IPv6 and IPv4. The throughput is consistently lower over IPv6 on all days of the week.*

4.5. Stall Events

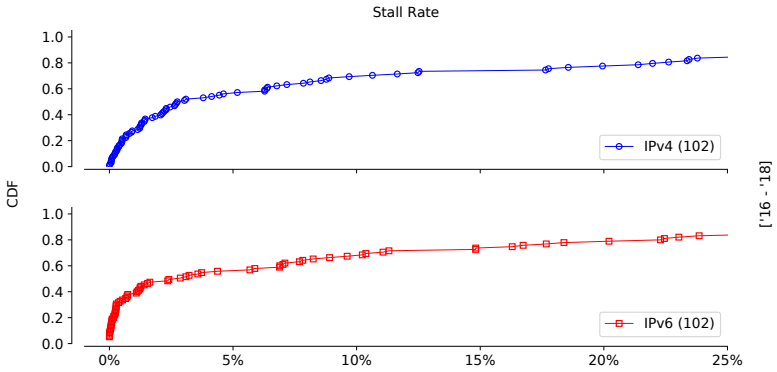

Fig. 4.41.: *CDF of stall rate over IPv4 and IPv6. 70% of the probes witness stall rate less than 10% over both IPv4 and IPv6.*

We now know that TCP connect times, prebuffering durations and througpput are worse over IPv6. We also know that clients prefer to stream videos over IPv6. We now look into stall events and stall durations over IPv4 and IPv6 to see which one performed better. We will look into *stall_events* and *stall_duration_total* field that we already defined in table 3.1. [14] defines a stall as an event that occurs during playback in situations when a frame is not recieved during its playout time. Throughput constraints leads to stall events which is caused by the bottlenecks between Netflix OCA and the client path. Results from SamKnows speed tests [4] are used to ignore unnecessary stalling, so as to limit maximum bit rate the the client will attempt to download. SamKnows test uses this concept and use the playout buffer of 40s. Thus, incase of a stall event rebuffering is done before resuming the playout timer **??**. To not exceed this playout buffer capacity, media downloads are paced. We first define the stall rates over both the address families i.e. Ipv4 and IPv6. *Stall Rate* is defined as the number of stall events to the total number of iterations of the test [14]. We are considering the *stall_event* field here, and after calculating the stal rate we take a CDF of it. fig. 4.41 depicts the distribution of stall rates over IPv4 and IPv6 across all probes. As can be seen, both the address families shows comparable stall rates. Around 70% of the probes witness stall rate less than 10% over both IPv4 and IPv6. *stall_durations_total* field gives us the total durations of stall as already discussed in table 3.1. We know measure the stall durations over IPv4 and IPv6. fig. 4.42 gives us the time series of stall durations over IPv4 and IPv6. We took the median aggregate of *stall_durations_total* across all probes on each day. Stall durations have been consistent over time. To see variation over different months fig. 4.43 gives us the boxplot of stall durations variation over different

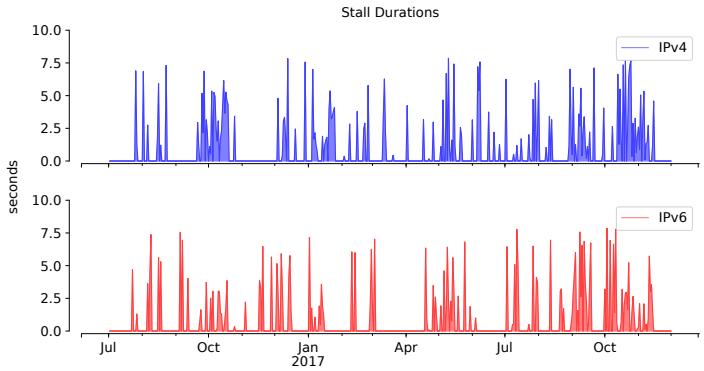

Fig. 4.42.: *Time series of Stall durations over IPv4 and IPv6. We took the median aggregate of stall_durations_total across all probes on each day. Stall durations have been consistent over time.*

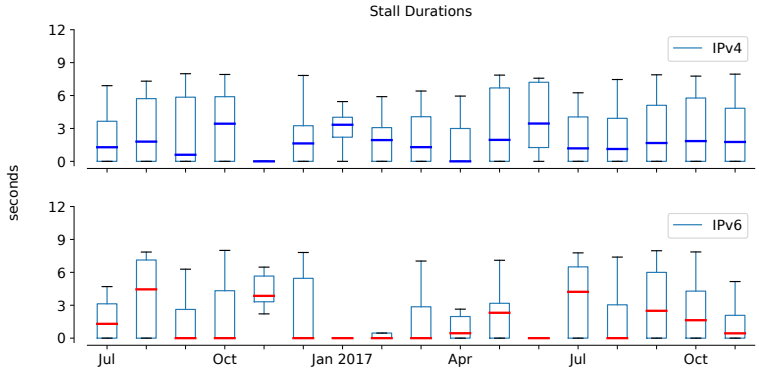

Fig. 4.43.: *Boxplot of Stall durations over IPv4 and IPv6. We have rounded the time to the nearest month, and we are measuring the stall durations over the period. The variation can be seen over different months.*

months. To get a clear picture fig. 4.44 shows us the CDF of stall durations over IPv4 and IPv6. Here, we cannot see much of difference as both the address families shows similar results. It would be good to check their deltas.

Stall Events Deltas

As individual CDFs doesn't give much information, we will calculate the difference of stall durations over IPv4 and IPv6 over both the families. We take the same terminology defined in [14]. We use eq. (4.3) to calculate the difference in stall durations over IPv4 and IPv6. Here, Δst gives the delta for both the address families. $st(y)$ is the

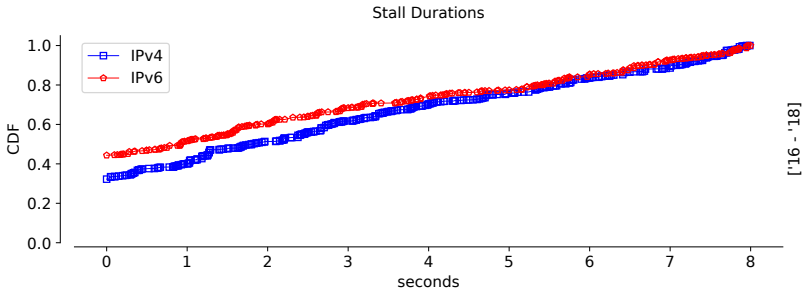

Fig. 4.44.: *CDF of stall durations over IPv4 and IPv6. Both the address families shows similar curves.*

stall duration over IPv6 and *st(x)* is the stall duration over IPv4. fig. 4.46 gives us the distribution of difference of stall duratiosn over IPv4 and IPv6. Here, we calculated the difference of stalldurations and took a CDF of it. Positive values indicates that stall durations are lower over IPv6. 41fig. 4.45 shows us boxplot of difference of stall durations over IPv4 and Ipv6. It shows the similar characteristics as in fig. 4.43 and we can see the variation between different months.

$$\Delta st = st(x) - st(y) \tag{4.3}$$

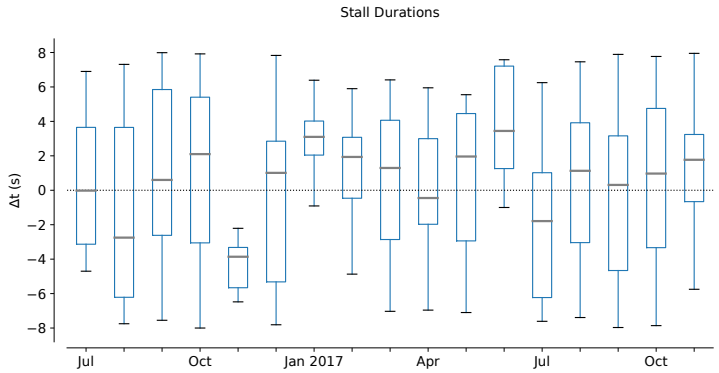

Fig. 4.45.: *Boxplot of difference of stall durations over IPv4 and Ipv6. It shows the similar characteristics as in fig. 4.43 and we can see the variation between different months.*

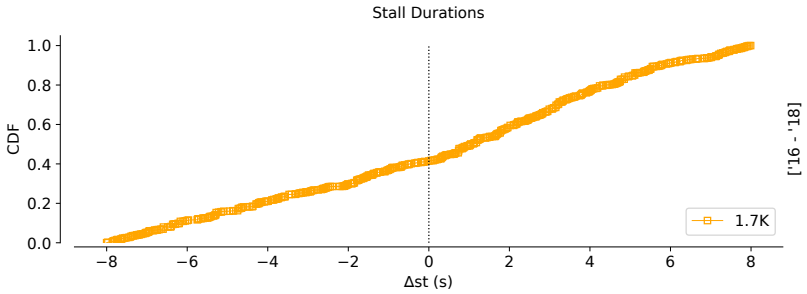

Fig. 4.46.: *CDF of distribution of difference in stall durations over IPv4 and IPv6. Positive values indicates that stall durations are lower over IPv6. 41% of the samples experience lower stall durations over IPv4, whereas, 59% of the samples experience lower stall durations over IPv6 with around 25% of them at least 5 seconds shorter.*

CHAPTER 5

BENEFITS OF ISP CONTENT CACHES

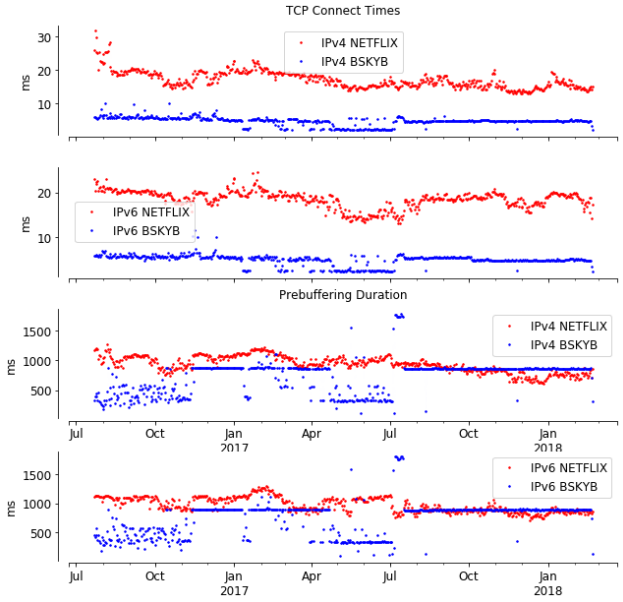

Fig. 5.1.: *Timeseries depicting the TCP connect times (connect_time field) and Prebuffering Duration (prebuffering_duration field) for SKY UK Limited against Netflix OCA servers. We are applying a median aggregate here over the absolute values ans as can be seen, the TCP connect times is better for SKY UK Limited for both the address families. Prebuffering duration also shows the similar results until August 2017, after that the performance is comparable to Netflix OCA servers.*

We wanted to check if there is any benefits from the ISP content caches. Therfore, we identified the ISPs from all the holder names (refer table 3.3) and tried to compare the performance of an ISP cache with that of Netflix OCA servers. Here, the criteria we are using to identify a cache is when the AS number of the source IP address is same as the AS number of the destination IP address. Therefore, the cache is identified when both the source and destination belongs to the same IP address. We started

with the ISP named *SKY UK Limited* [60], as this is the second ISP which contains the maximum number of distinct IPs (refer table 3.3). We calcualted the Latency, Delay and Thoughput for this ISP with respect to Netflix (AS Number 2906). After considering *SKY UK Limited*, we did the similar analysis for two more cases, one of them is for the ISP *British Telecommunications* [95] which is also a major ISP in UK, the results for this ISP can be found in Appendices. Also, *British Telecommunications* has the third most number of distinct IPs in the dataset (refer table 3.3). We then analysed the results for all ISPs content caches that were present in the dataset against Netflix OCA servers.

5.1. SKY UK Limted

TCP Latency and Delay

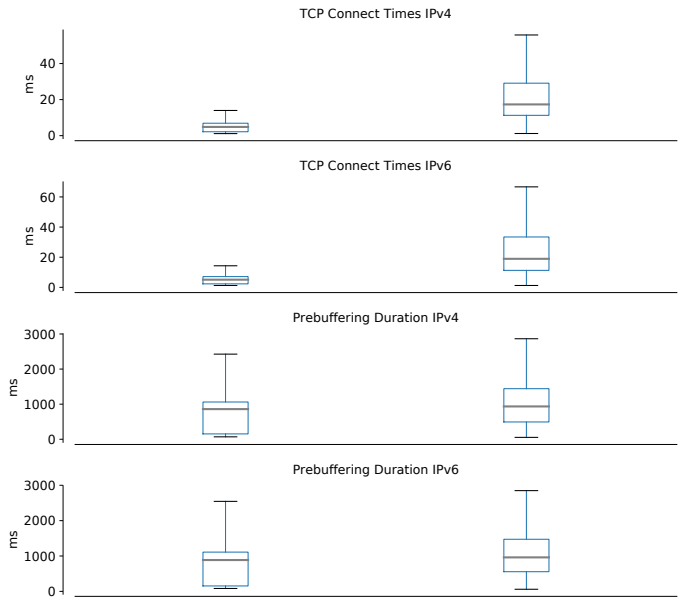

Fig. 5.2.: *Boxplot depicting TCP connect times and Pre-Bufferring Duration for SKY UK Limited and Netflix. The median resemembles the Time series graph in* **??**.

SKY UK Limited has the AS Number 5607, so for filtering out the rows which are using the criteria where both AS number of source IP address and AS number of destination IP address belongs to the same As number which is 5607 here. Here, we are considering the *connect_time* and *prebuffering_duration* fields that are defined in table 3.1. We first wanted to see how TCP connect times and prebuffering durations perform for both ISP caches and for Netflix (AS number 2906). fig. 5.1 shows the timeseries

for SKY UK Content caches and Netflix OCA servers over both the address families, and we have group by *dtime* field and are considering the median aggregate here, so that a single vantage point doesn't bias the results. As we can see, the TCP connect times is better for SKY UK Limited for both the address families, i.e. (0-10 ms), whereas for Netflix its around (0-30 ms). Prebuffering duration also shows the similar results until August 2017, after that the performance is comparable to Netflix OCA servers. We have converted the connect time and pre-buffering duration to 'ms' to get a better understanding. To get more clear view of the delays, we plot the boxplots for both SKY UK Limited and Netflix over Ipv4 and IPv6. The median line in fig. 5.2 graph resembles the timeseries in fig. 4.27. Furthermore, the thrid quartile is pretty low for *SKY UK Limited* content caches. We further investigate the distribution of latency and delays for SKY UK and Netflix, here also, we filter the data along IPv4 and IPv6 and then take the CDF of the desired attributes. fig. 5.3 shows the CDF of TCP connect times and prebuffering durations for SKY UK and Netflix. Around 80% of the probes require TCP connect times of 19.5 ms for SKY UK Limited over IPv4, whereas it'3 38.5 ms for the same number of probes for Netflix. Similarly, for IPv6, 80% of the probes require 15.5 ms for SKY UK, whereas it is 40.75 ms for Netflix. For pre-bufferind duration, SKY UK caches require around 1530 ms for 80% of the probes over IPv4, and for Netflix the number is 1632 ms for same number of probes. For IPv6, SKY UK requires 1560 ms for 80% of the probes and for Netflix the number is 1730 ms. It would be also good to compare the deltas i.e. the difference between the Latencies over IPv4 and IPv6 for SKY UK and Netflix.

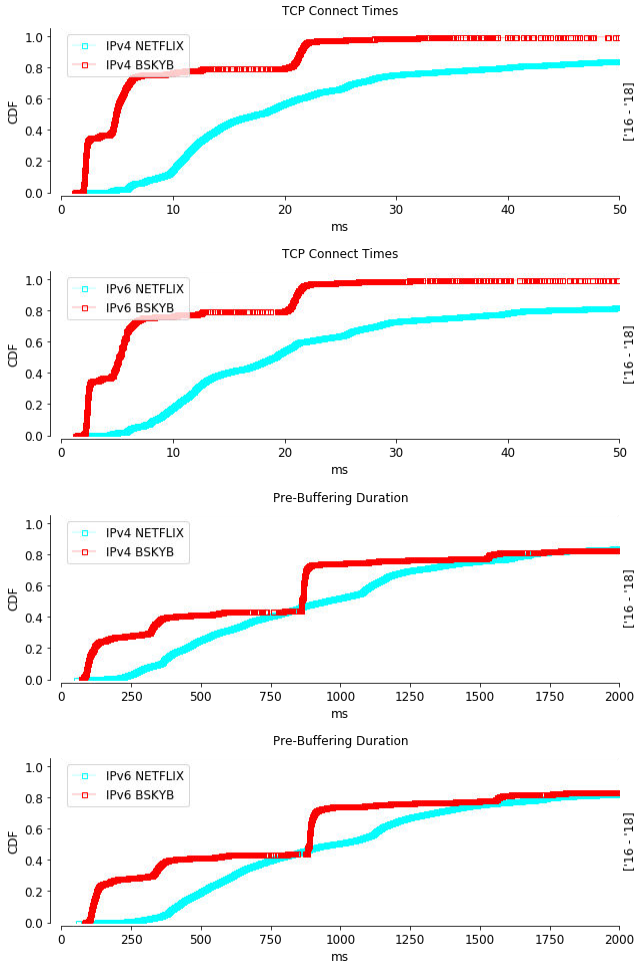

Fig. 5.3.: *CDF of TCP Connect times and Pre-Buffering duration for SKY UK Limited and Netflix over IPv4 and IPv6. As we can see, TCP COnnect times and Prebuffering duration for SKY UK Limited content caches is better than Netflix OCA servers.*

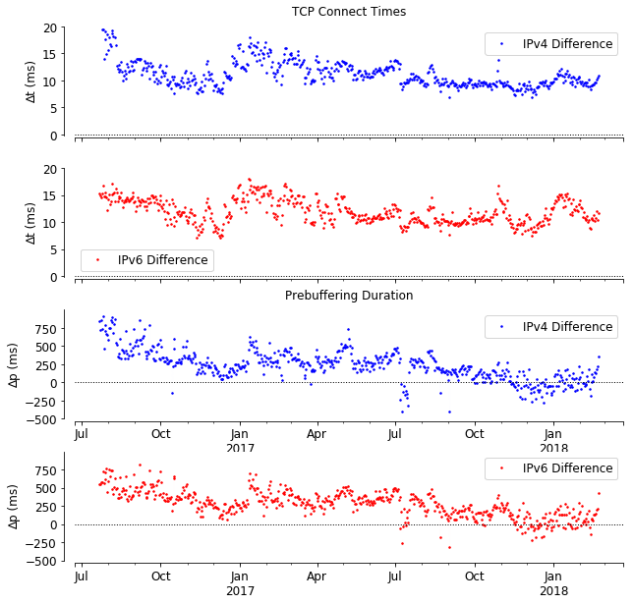

Fig. 5.4.: *Time series of difference for TCP connect times and prebuffering durations over IPv4 and IPv6 between Netflix and SKY UK Limited. Here, positive values indicate Netflix OCA server requires more connect time and pre-buffering duration as compared to SKY UK caches. Latency of around 10-15 ms and higher prebuffering durations (around 0-250 ms or more) are observed for Netflix over both IPv4 and IPv6.*

We will now compare the Deltas that is, difference between Netflix OCA server TCP connect times and SKY UK caches TCP connect times over IPv4 and IPv6. We will now define the terminology that we are using here, let us say, TCP connect time over IPv4 for SKY UK is tp(y) and TCP connect time over IPv4 for Netflix is tp(x), then the *delta* will be $\Delta tp = tp(x) - tp(y)$. We did calculate these deltas for TCP connect times and Prebuffering duration for SKY UK and Netflix over IPv4 and IPv6. Also, as already discussed in chapter 3, prebuffering duration takes into account DNS resolution times and TCP connect times. We will now analyse the deltas to check the benefits of ISP caches.

To plot the deltas, we followed the same analysis [14] did for the Youtueb dataset. To plot the timeseries for these deltas, we calculate the median aggregate on the TCP connect times and the prebuffering duration across all probes on each day for the difference between Netflix and SKY UK limited. fig. 5.4 shows the timeseries of median TCP connect times and prebuffering duration over IPv4 and IPv6 across all probes. Here, positive values indicate Netflix OCA server requires more connect time and pre-buffering duration as compared to SKY UK caches. Latency of around 10-15 ms and higher prebuffering durations (around 0-250 ms or more) are observed for Netflix

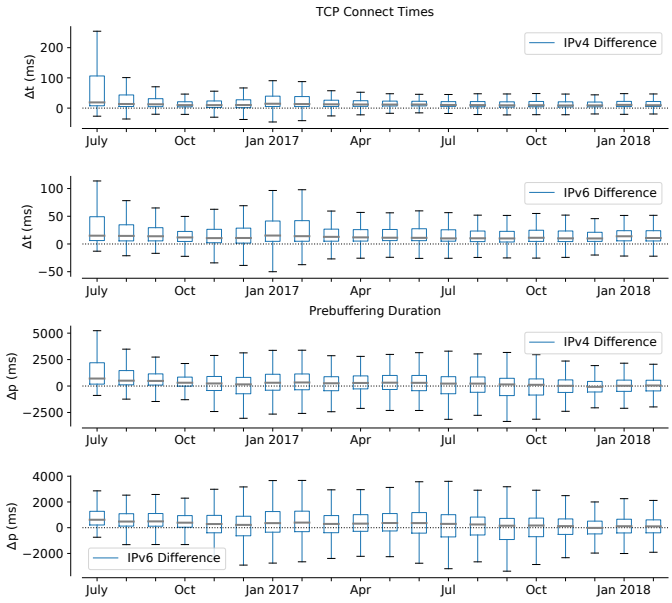

Fig. 5.5.: *Boxplot of difference for TCP connect times and prebuffering durations over IPv4 and IPv6 between Netflix and SKY UK Limited. The median line here represents the time series graph in fig. 5.4.*

over both IPv4 and IPv6. For boxplots in fig. 5.5, we have rounded the time to the nearest month. Here, we can see that the median resembles the time series fig. 5.4. To get more vivid analysis, fig. 5.6 shows us the distribution of difference in TCP connect times and prebuffering duration over the whole duration. For calculating the CDF, we calcualted the deltas and then plotted their CDF. To compare the performance, we are measuring the TCP connect times to the Netflix OCA (Open Connect Appliances) server and to the SKY UK content caches, the distribution here shows that around 85% of the connections are slower for Netflix OCA servers over IPv4 , and around 86% of the connections are slower for Netflix over IPv6. For prebuffering duration which is the time to fetch 2 seconds of video at the specified bitrate from the content server, here Netflix OCA (Open Connect Appliances) server nad SKY Uk caches, the distribution shows that 61% of the connections are slower for Netflix over IPv4, and around 63% of the connections are slower for Netflix over IPv6.

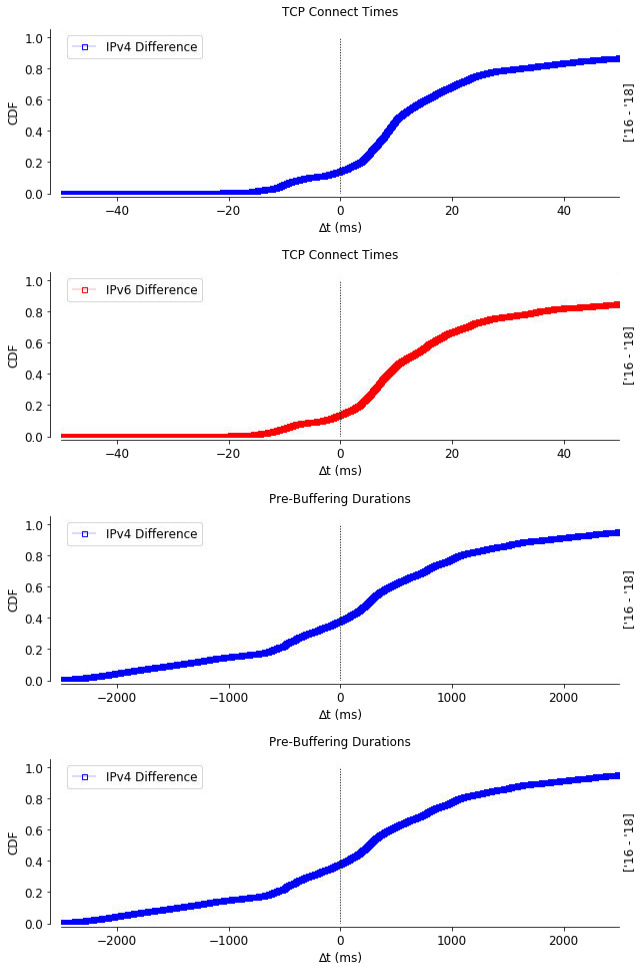

Fig. 5.6.: *CDF of difference of TCP connect times and Prebuffering Durations for Netflix and SKY UK over IPv4 and IPv6. The distribution here shows that around 85% of the connections are slower for Netflix OCA servers over IPv4 , and around 86% of the connections are slower for Netflix over IPv6. For prebuffering duration which is the time to fetch 2 seconds of video at the specified bitrate from the content server, here Netflix OCA (Open Connect Appliances) server nad SKY Uk caches, the distribution shows that 61% of the connections are slower for Netflix over IPv4, and around 63% of the connections are slower for Netflix over IPv6.*

Throughput

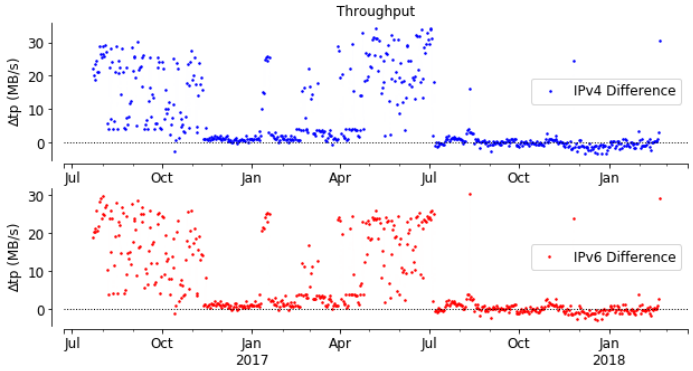

Fig. 5.7.: *Time series of Throughput for individual address family's i.e. IPv4 and IPv6 for Netflix and SKY UK. We are considering the bytes_sec field described in table 3.1. As can be seen, the achieved throughput for SKY UK caches is somwehat comparable or more than Netflix OCA server, except after August 2017, where it is lower.*

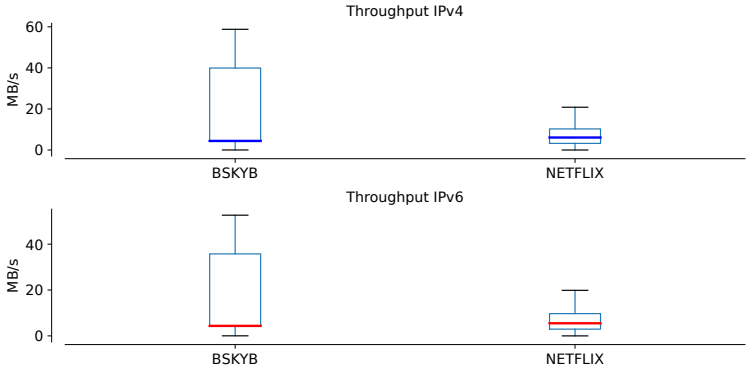

Fig. 5.8.: *Boxplot of Throughput for individual address family's i.e. IPv4 and Ipv6 for Netflix and SKY UK. The median line here shows that the achieved throguhput is comparable but the thrid quartile of SKY UK shows that the achieved throughput is higher for SKY UK over both address family's.*

After comparing the TCP connect times and Pre-Buffering duration for Netflix and SKY UK, we now know that clients take higher TCP connect times and prebuffering duration for Netflix as compared to the SKY UK caches over both the address family's. We will now be comparing the achieved throughput over IPv4 and IPv6 for Netflix OCA server and SKY UK caches. We will first look into the individual performance of IPv4 and IPv6 for Netflix and SKY UK and then will look into their deltas. fig. 5.7 gives

the time series for both the address families for SKY UK and Netflix,As can be seen, the achieved throughput for SKY UK caches is somwehat comparable or more than Netflix OCA server, except after August 2017, where it is lower. We are considering the *bytes_sec* field from the Netflix **??**, and we are taking the median aggregate across all probes on each day. Also, the throughput for IPv4 and IPv6 lies between 0-40 MB/s for SKY UK. To get more deeper insights we also did the boxplot for IPv4 and IPv6 throughput for SKY UK and Netflix, and as can be seen in fig. 5.8 the monthly throughput variation over IPv4 and IPv6. The median line here shows that the achieved throguhput is comparable but the thrid quartile of SKY UK shows that the achieved throughput is higher for SKY UK over both address family's. We have converted the *bytes_sec* field into MB/s to get a more realistic view. fig. 5.9 shows the CDF of throughput over IPv4 and IPv6. We followed the same steps here, and plotted the CDF of *bytes_sec* field. The CDF here shows that around 80% of the probes achieved a throughput of 42 MB/s for SKY UK over IPv4, whereas for Netflix this is only 11 MB/s for similar number of probes. For IPv6, 80% of the probes achieved the throughput of 36 MB/s, while for Netflix this was only 10 MB/s for similar number of probes. We will now look into the deltas to get better comparison between SKY UK and Netflix.

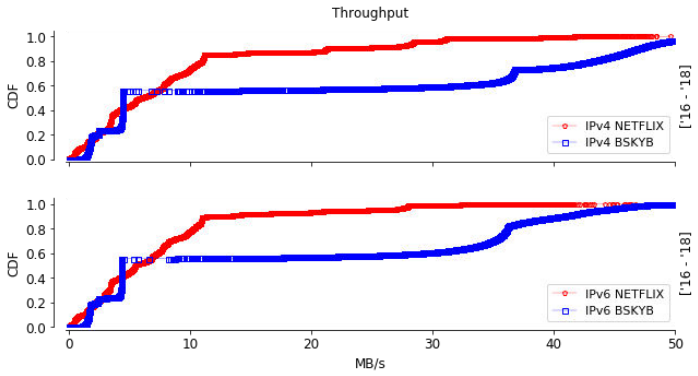

Fig. 5.9.: *CDF of Throughput over IPv4 and IPv6 for SKY UK and Netflix. The CDF here shows that around 80% of the probes achieved a throughput of 42 MB/s for SKY UK over IPv4, whereas for Netflix this is only 11 MB/s for similar number of probes. For IPv6, 80% of the probes achieved the throughput of 36 MB/s, while for Netflix this was only 10 MB/s for similar number of probes.*

Before starting with the deltas, we would live to define the terminologies we used to get the results. We are considering the *bytes_sec* field only and using the same terminilogy Bajpai et al. used in [14]. We denote the throughput over IPv4 for SKY UK as *tp(y)* and throughput over IPv4 for Netflix as *tp(x)*, then the *delta* will be Δtp = tp(y) - tp(x). To plot the deltas, we first start with the time series, and fig. 5.10 shows us the

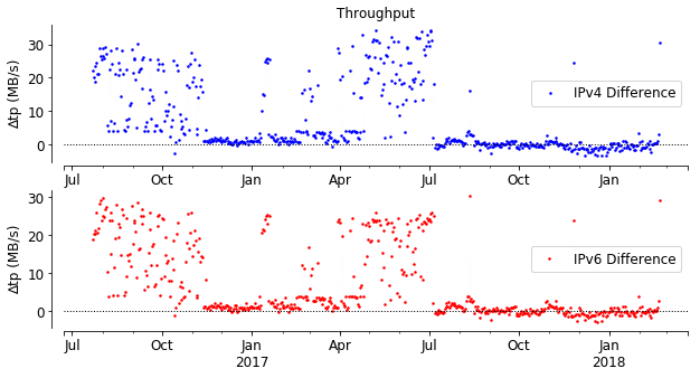

Fig. 5.10.: *Time series of deltas of Throughput over IPv4 and IPv6 between SKY UK and Netflix. We can see that the difference is around 0 or more for the whole duration. The posiitive value indicates a higher throughput for SKY UK, also the difference lies between 0-30 MB/s.*

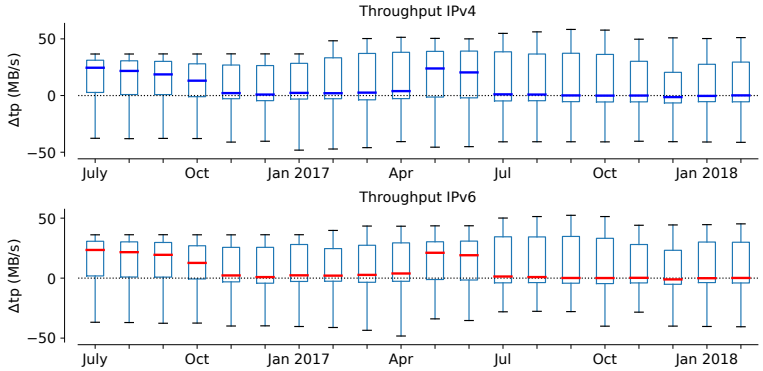

Fig. 5.11.: *Boxplot of difference of throughput over IPv4 and IPv6 between SKY UK and Netflix. The median line shows similar curves as the time series fig. 5.10, depicting higher throughput for SKy UK over both address families.*

time series of median aggregate of throughput across all probes over each day. We can see that the difference is around 0 or more for the whole duration. The posiitive value indicates a higher throughput for SKY UK, also the difference lies between 0-30 MB/s. fig. 5.11 shows the boxplot of difference of throughput over IPv4 and IPv6 between SKY UK and Netflix. The median line shows similar curves as the time series fig. 5.10, depicting higher throughput for SKy UK over both address families. To get a more clear picture about the performance, we plot the CDF of *bytes_sec* field. fig. 5.12 shows the CDF of difference of throughput over IPv4 and IPv6 for SKY UK and Netflix. Around 60% of the times, SKY UK caches achieved higher throughput than Netflix over IPv4 and IPv6.

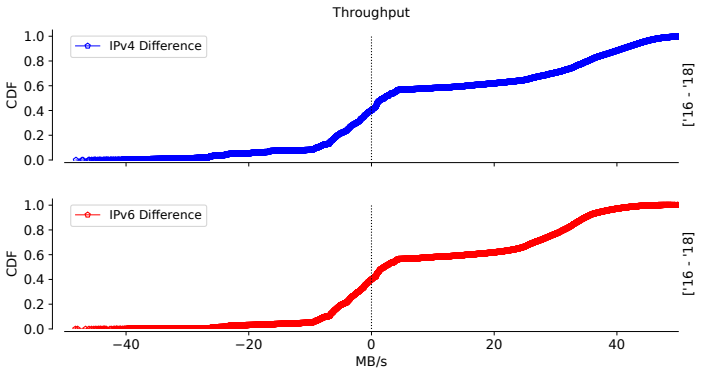

Fig. 5.12.: *CDF of difference of throughput over IPv4 and IPv6 for SKY UK and Netflix. Around 60% of the times, SKY UK caches achieved higher throughput than Netflix over IPv4 and IPv6.*

5.2. All ISPs Together

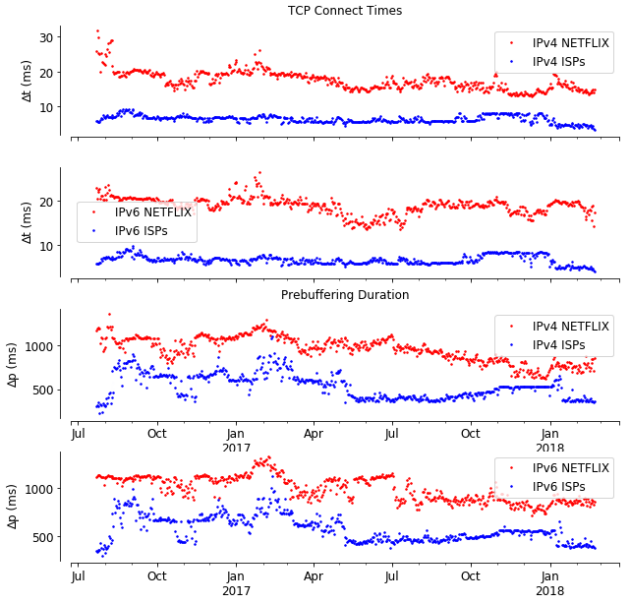

Fig. 5.13.: *Timeseries depicting the TCP connect times (connect_time field) and Prebuffering Duration (prebuffering_duration field) for all ISPs caches against Netflix OCA servers. We are applying a median aggregate here over the absolute values and as can be seen, the TCP connect times is better for ISPs for both the address families. Prebuffering duration also shows the similar results i.e. ISP caches performs better here as well.*

SKY UK was one of the ISP we considered till now, we wanted to see how Netflix compares against all the ISPs content caches. So, here we are grouping all the ISPs mentioned in table 3.3. We included all the ISPs belonging to this table. We know will compare All ISPs content caches against Netflix for the TCP connect times, Prebuffering duration and achieved throughput.

TCP Latency and Delay

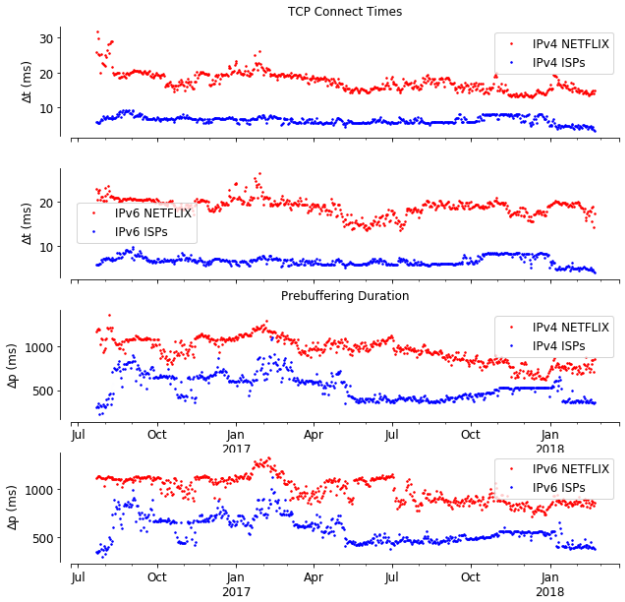

Fig. 5.14.: *Timeseries depicting the TCP connect times (connect_time field) and Prebuffering Duration (prebuffering_duration field) for all ISPs caches against Netflix OCA servers. We are applying a median aggregate here over the absolute values and as can be seen, the TCP connect times is better for ISPs for both the address families. Prebuffering duration also shows the similar results i.e. ISP caches performs better here as well.*

Here, all ISPs are grouped together and we are using the criteria where both AS number of source IP address and AS number of destination IP address belongs to the same AS number. We are considering the *connect_time* and *prebuffering_duration* fields that are defined in table 3.1. We first wanted to see how TCP connect times and prebuffering durations perform for both ISP caches and for Netflix (AS number 2906). fig. 5.14 shows the timeseries for all ISPs content caches and Netflix OCA servers over both the address families, and we have group by *dtime* field and are considering the median aggregate here, so that a single vantage point doesn't bias the results. As we can see, the TCP connect times is better for ISPs for both the address families, i.e.

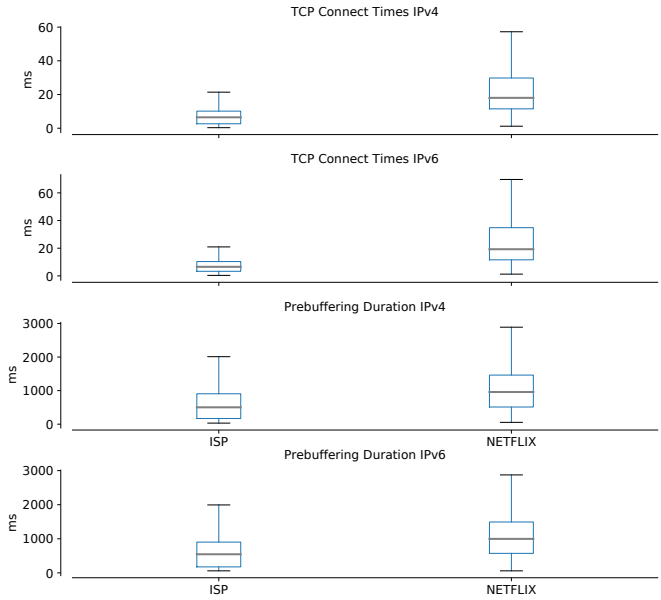

Fig. 5.15.: *Boxplot depicting TCP connect times and Pre-Bufferring Duration for ISPs and Netflix. The median resememble the Time series graph in fig. 5.14.*

(0-10 ms), whereas for Netflix its around (0-30 ms). Prebuffering duration also shows the similar results i.e. ISP caches performs better here as well. We have converted the connect time and pre-buffering duration to 'ms' to get a better understanding. To get more clear view of the delays, we plot the boxplots for both ISPs and Netflix over IPv4 and IPv6. The median line in fig. 5.15 graph resembles the timeseries in fig. 5.14. Furthermore, the thrid quartile is pretty low for *ISPs* content caches. We further investigate the distribution of latency and delays for ISPs and Netflix, here also, we filter the data along IPv4 and IPv6 and then take the CDF of the desired attributes. fig. 5.16 shows the CDF of TCP connect times and prebuffering durations for the ISPs and Netflix. Around 80% of the probes require TCP connect times of 13 ms for ISPs over IPv4, whereas it's 38 ms for the same number of probes for Netflix. Similarly, for IPv6, 80% of the probes require 14 ms for ISPs, whereas it is 41 ms for Netflix. For pre-buffering duration, ISPs caches require around 1243 ms for 80% of the probes over IPv4, and for Netflix the number is 1671 ms for same number of probes. For IPv6, ISPs caches requires 1275 ms for 80% of the probes and for Netflix the number is 1738 ms. It would be also good to compare the deltas i.e. the difference between the Latencies over IPv4 and IPv6 for ISPs and Netflix.

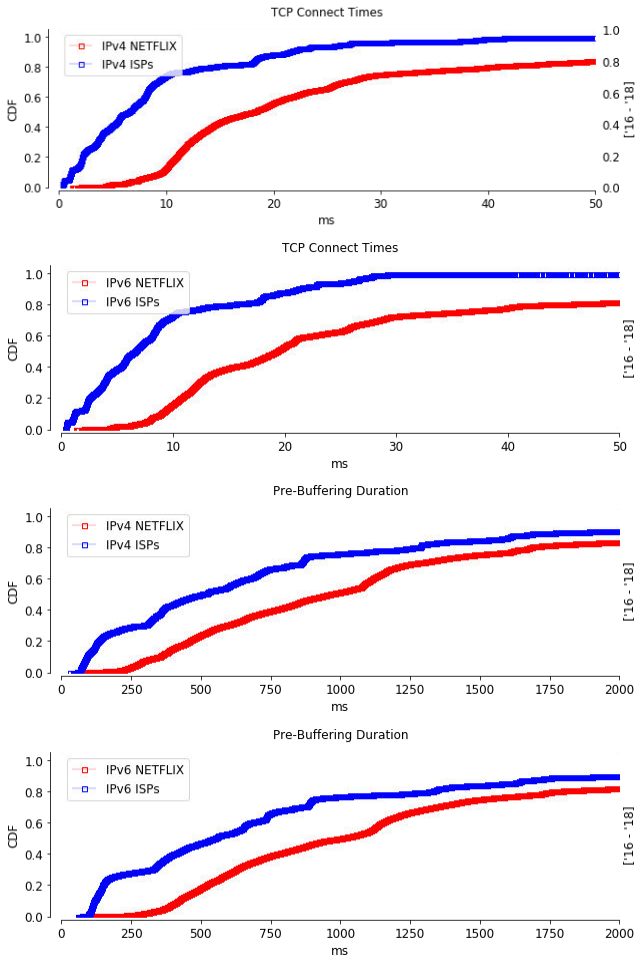

Fig. 5.16.: *CDF of TCP Connect times and Pre-Buffering duration for all ISPs and Netflix over IPv4 and IPv6. As we can see, TCP Connect times and Prebuffering duration for ISPs content caches is better than Netflix OCA servers.*

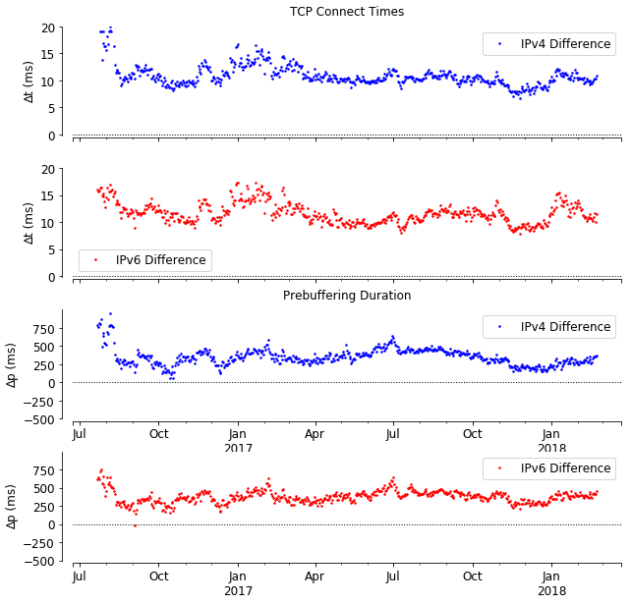

Fig. 5.17.: *Time series of difference for TCP connect times and prebuffering durations over IPv4 and IPv6 between Netflix and ISPs. Here, positive values indicate Netflix OCA server requires more connect time and pre-buffering duration as compared to ISPs caches. Latency of around 10-20 ms and higher prebuffering durations (around 0-750 ms or more) are observed for Netflix over both IPv4 and IPv6.*

We will now compare the Deltas that is, difference between Netflix OCA server TCP connect times and ISPs caches TCP connect times over IPv4 and IPv6. We will now define the terminology that we are using here, let us say, TCP connect time over IPv4 for ISPs is tp(y) and TCP connect time over IPv4 for Netflix is tp(x), then the *delta* will be Δtp = tp(x) - tp(y). We did calculate these deltas for TCP connect times and Prebuffering duration for ISPs and Netflix over IPv4 and IPv6. Also, as already discussed in chapter 3, prebuffering duration takes into account DNS resolution times and TCP connect times. We will now analyse the deltas to check the benefits of ISP caches.

To plot the deltas, we followed the same analysis [14] did for the Youtueb dataset. To plot the timeseries for these deltas, we calculate the median aggregate on the TCP connect times and the prebuffering duration across all probes on each day for the difference between Netflix and ISPs. fig. 5.17 shows the timeseries of median TCP connect times and prebuffering duration over IPv4 and IPv6 across all probes. Here, positive values indicate Netflix OCA server requires more connect time and pre-buffering duration as compared to ISPs caches. Latency of around 10-20 ms and higher prebuffering durations (around 0-750 ms or more) are observed for Netflix over

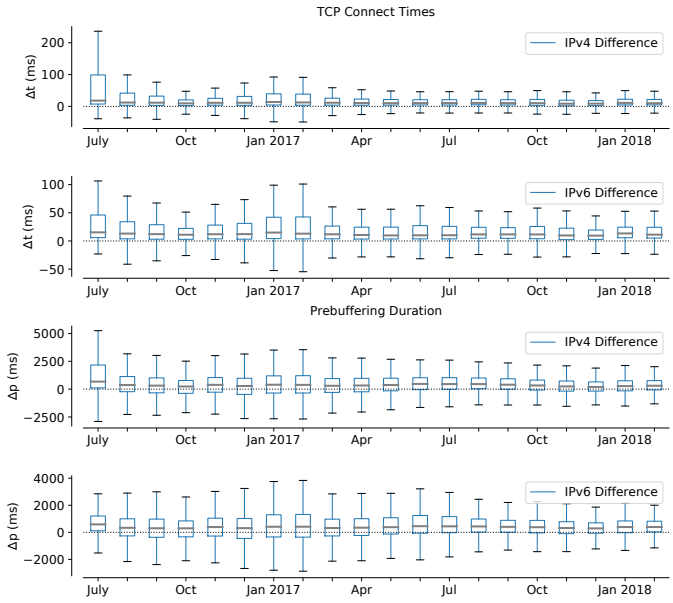

Fig. 5.18.: *Boxplot of difference for TCP connect times and prebuffering durations over IPv4 and IPv6 between Netflix and ISPs. The median line here represents the time series graph in fig. 5.17.*

both IPv4 and IPv6. For boxplots in fig. 5.18, we have rounded the time to the nearest month. Here, we can see that the median resembles the time series fig. 5.17. To get more vivid analysis, fig. 5.19 shows us the distribution of difference in TCP connect times and prebuffering duration over the whole duration.

For calculating the CDF, we calcualted the deltas and then plotted their CDF. To compare the performance, we are measuring the TCP connect times to the Netflix OCA (Open Connect Appliances) server and to the ISPs content caches, the distribution here shows that around 84% of the connections are slower for Netflix OCA servers over IPv4 , and around 85% of the connections are slower for Netflix over IPv6. For prebuffering duration which is the time to fetch 2 seconds of video at the specified bitrate from the content server, here Netflix OCA (Open Connect Appliances) server and ISPs caches, the distribution shows that 68% of the connections are slower for Netflix over IPv4, and around 69% of the connections are slower for Netflix over IPv6.

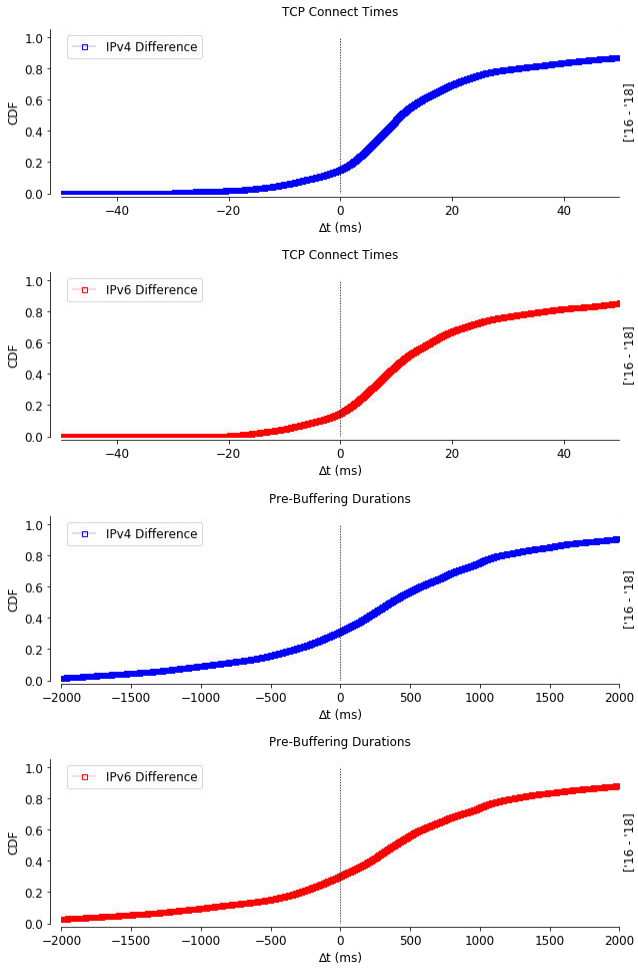

Fig. 5.19.: *CDF of difference of TCP connect times and Prebuffering Durations for Netflix and ISPs over IPv4 and IPv6. The distribution here shows that around 84% of the connections are slower for Netflix OCA servers over IPv4 , and around 85% of the connections are slower for Netflix over IPv6. For prebuffering duration which is the time to fetch 2 seconds of video at the specified bitrate from the content server, here Netflix OCA (Open Connect Appliances) servers and ISPs caches, the distribution shows that 68% of the connections are slower for Netflix over IPv4, and around 69% of the connections are slower for Netflix over IPv6.*

Throughput

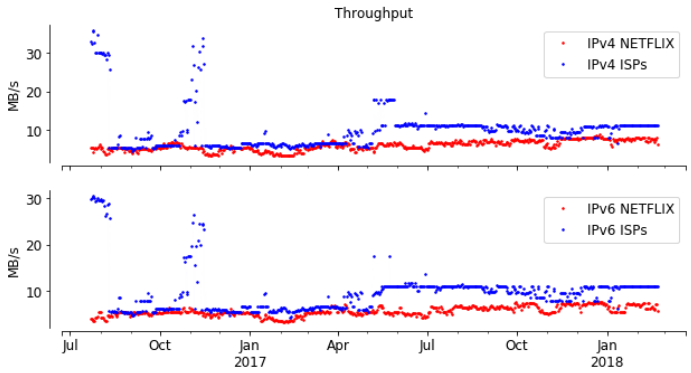

Fig. 5.20.: *Time series of Throughput for individual address family's i.e. IPv4 and IPv6 for Netflix and ISPs. We are considering the bytes_sec field described in table 3.1. As can be seen, the achieved throughput for ISP caches is somwehat comparable or more than Netflix OCA server.*

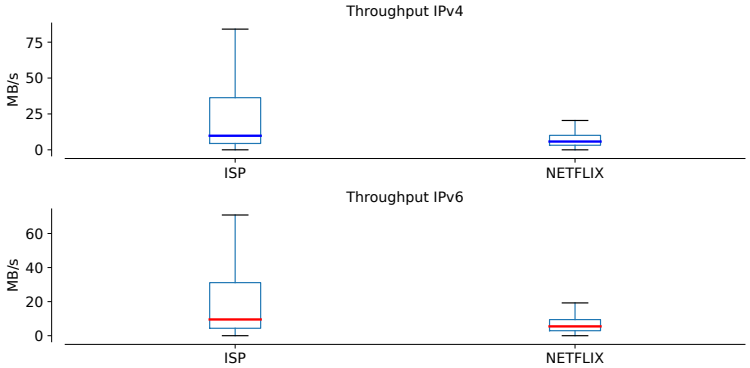

Fig. 5.21.: *Boxplot of Throughput for individual address family's i.e. IPv4 and Ipv6 for Netflix and ISPs. The median line here shows that the achieved throguhput is comparable but the thrid quartile of ISP caches shows that the achieved throughput is higher for ISPs over both address family's.*

After comparing the TCP connect times and Pre-Buffering duration for Netflix and ISPs, we now know that clients take higher TCP connect times and prebuffering duration for Netflix OCA servers as compared to the ISPs caches over both the address family's. We will now be comparing the achieved throughput over IPv4 and IPv6 for Netflix OCA server and ISP caches. We will first look into the individual performance of IPv4 and IPv6 for Netflix and ISPs and then will look into their deltas. fig. 5.20 gives the time series for both the address families for ISPs and Netflix, and as can be seen, the achieved throughput for ISPs caches is somwehat comparable or more than Netflix

OCA server. We are considering the *bytes_sec* field from the Netflix table 3.1, and we are taking the median aggregate across all probes on each day. Also, the throughput for IPv4 and IPv6 lies between 0-40 MB/s for ISP caches. To get more deeper insights we also did the boxplot for IPv4 and IPv6 throughput for ISPs and Netflix, and as can be seen in fig. 5.21 the monthly throughput variation over IPv4 and IPv6. The median line here shows that the achieved throguhput is comparable but the third quartile of ISPs shows that the achieved throughput is higher for ISP caches over both address family's. We have converted the *bytes_sec* field into MB/s to get a more realistic view. fig. 5.9 shows the CDF of throughput over IPv4 and IPv6. We followed the same steps here, and plotted the CDF of *bytes_sec* field. The CDF here shows that around 80% of the probes achieved a throughput of 40 MB/s for ISPs over IPv4, whereas for Netflix this is only 10 MB/s for similar number of probes. For IPv6, 80% of the probes achieved the throughput of 40 MB/s, while for Netflix this was only 10 MB/s for similar number of probes. We will now look into the deltas to get better comparison between ISP caches and Netflix OCA servers.

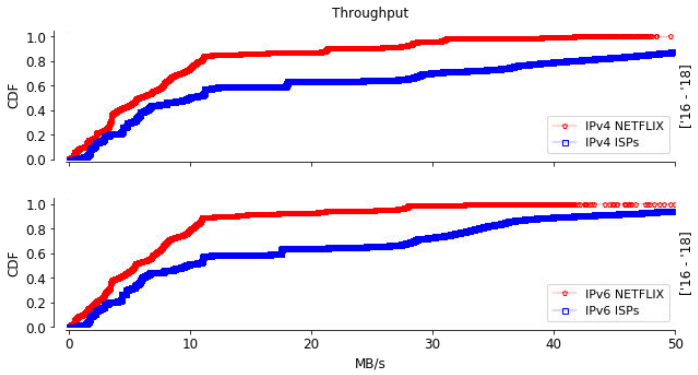

Fig. 5.22.: *CDF of Throughput over IPv4 and IPv6. We followed the same steps here, and plotted the CDF of bytes_sec field. The CDF here shows that around 80% of the probes achieved a throughput of 40 MB/s for ISPs over IPv4, whereas for Netflix this is only 10 MB/s for similar number of probes. For IPv6, 80% of the probes achieved the throughput of 40 MB/s, while for Netflix this was only 10 MB/s for similar number of probes.*

Deltas

Before starting with the deltas, we would live to define the terminologies we used to get the results. We are considering the *bytes_sec* field only and using the same terminilogy Bajpai et al. used in [14]. We denote the throughput over IPv4 for ISPs as *tp(y)* and throughput over IPv4 for Netflix as *tp(x)*, then the *delta* will be Δtp = tp(y) - tp(x). To plot the deltas, we first start with the time series, and fig. 5.23 shows us the time series

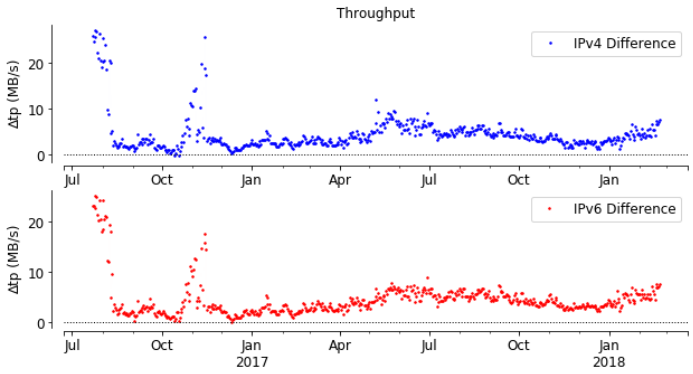

Fig. 5.23.: *Time series of median aggregate of throughput across all probes over each day. We can see that the difference is positive, indicating a higher throughput for ISP caches, also the difference lies between 0-30 MB/s.*

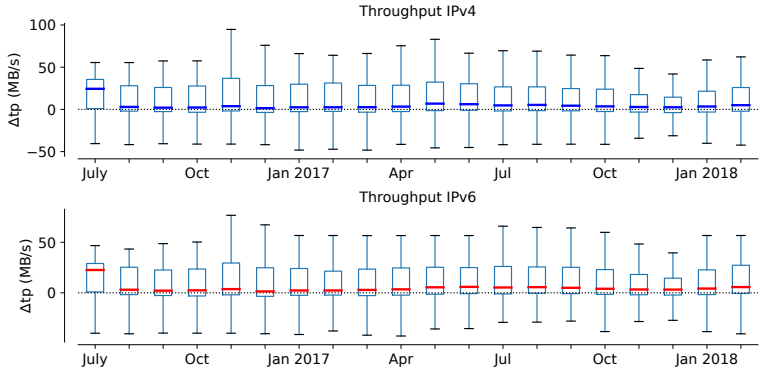

Fig. 5.24.: *Boxplot of difference of throughput over IPv4 and IPv6 between ISP caches and Netflix OCA server. The median line shows similar curves as the time series fig. 5.23, depicting higher throughput for ISPs over both address families.*

of median aggregate of throughput across all probes over each day. We can see that the difference is positive, indicating a higher throughput for ISP caches, also the difference lies between 0-30 MB/s. fig. 5.24 shows the boxplot of difference of throughput over IPv4 and IPv6 between ISP caches and Netflix OCA server. The median line shows similar curves as the time series fig. 5.23, depicting higher throughput for ISPs over both address families. To get a more clear picture about the performance, we plot the CDF of *bytes_sec* field. fig. 5.25 shows the CDF of difference of throughput over IPv4 and IPv6 for ISPs and Netflix. Around 66% of the times, ISP caches achieved higher throughput than Netflix over IPv4 and IPv6.

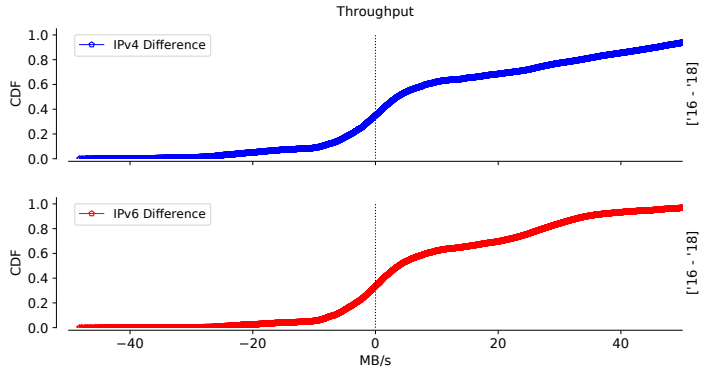

Fig. 5.25.: *CDF of difference of throughput over IPv4 and IPv6 for ISPs and Netflix. Around 66% of the times, ISP caches achieved higher throughput than Netflix over IPv4 and IPv6.*

CHAPTER 6

WHO CONNECTS BETTER - M-LAB OR NETFLIX?

We wanted to compare the Speedtest results to M-Lab media servers and to the Netflix OCA servers. As already discussed in etflix i.e. *Fast.com* measures the the speedtest (bytes_sec) it takes to download 25 MB of video file [41], whereas M-Lab tries to send as much data possible within 10 seconds (both upload and download) to a M-lab server [56]. Here, we will be comparing the M-lab and Netflix achieved speedtest to get a clear picture about who gives better performance. We will first compare the absolute values for both Netflix and M-Lab, individually over IPv4 and IPv6. Then, we will compare the Deltas for example, difference between M-Lab IPv4 measurement and Netflix IPv4 measurement. Lets say, speedtest over IPv4 for M-Lab is tp(y) and speedtest over IPv4 for Netflix is tp(x), then the *delta* will be Δtp = tp(y) - tp(x). Also, to note here that the M-lab IPv6 test ran only till July 2017.

Absolute

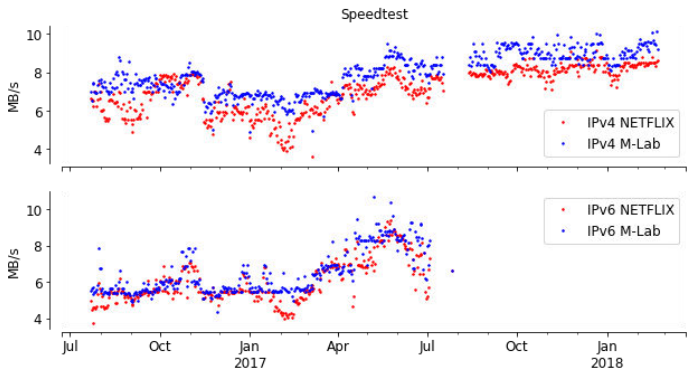

Fig. 6.1.: *Time series of Speedtest for individual address family's i.e. IPv4 and IPv6, for M-Lab and Netflix. We are considering the bytes_sec field ,and speedtest for M-Lab is greater over both address family's and median speedtest for M-Lab and Netflix lies between 4-10 MB/s for both the families.*

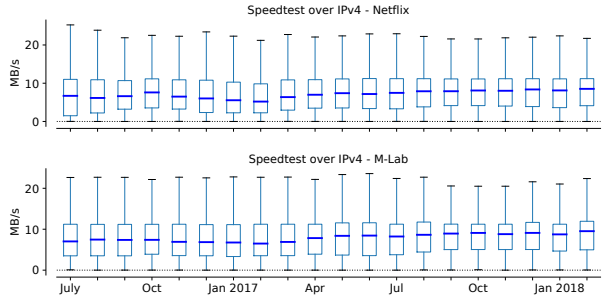

Fig. 6.2.: *Boxplot of speedtest over IPv4 for M-Lab and Netflix We can see the monthly variation here, also the median line resembles the time series in fig. 6.1*

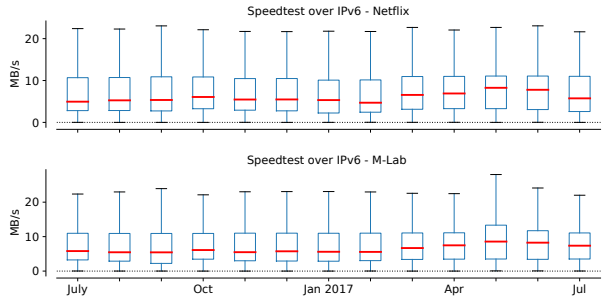

Fig. 6.3.: *Boxplot of speedtest over IPv6 for M-Lab and Netflix. We can see the monthly variation here, also the median line resembles the time series in fig. 6.1*

We will first compare the timeseries plot for both M-Lab and Netflix, as can be seen in fig. 6.1 that the speedtest for M-Lab is greater over both address family's and median speedtest for M-Lab and Netflix lies between 4-10 MB/s for both IPv4 and IPv6. Also, to note here that the speedtest is increasing over time for both M-lab and Netflix over boht IPv4 and IPv6. We are considering the *bytes_sec* field, and we are taking the median aggregate across all probes on each day. To get more deeper insights we also did the boxplot for IPv4 and IPv6 speedtest to compare M-Lab and Netflix performance, and as can be seen in fig. 6.2 and fig. 6.3 the monthly speedtest variation over IPv4 and IPv6. The median line resembles the time series in fig. 6.1. We have converted the *bytes_sec* field into MB/s to get a more realistic view. fig. 6.4 shows the CDF of speedtest over IPv4 and IPv6 for M-Lab and Netflix. We followed the same steps here, and plotted the CDF of *bytes_sec* field. The CDF for individual address family's shows similar curves and does not give a clear difference in the performance of M-Lab and Netflix over IPv4 and IPv6. We need to look into the deltas to get a more detailed performance comparison between M-Lab and Netflix.

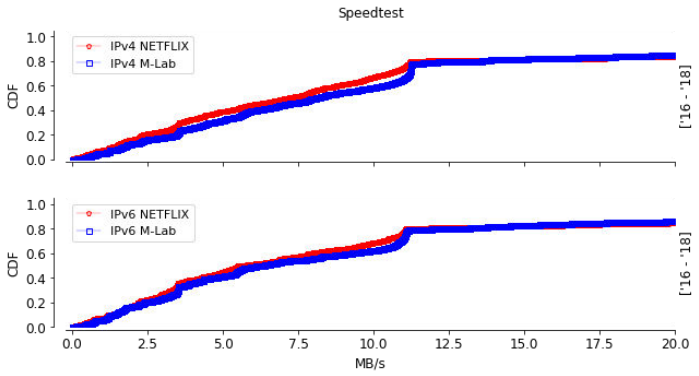

Fig. 6.4.: *CDF of speedtest over IPv4 and IPv6 for M-Lab and Netflix, shows the similar curves over both the family's. We cannot see much difference in the performance of M-Lab and Netflix here.*

Deltas

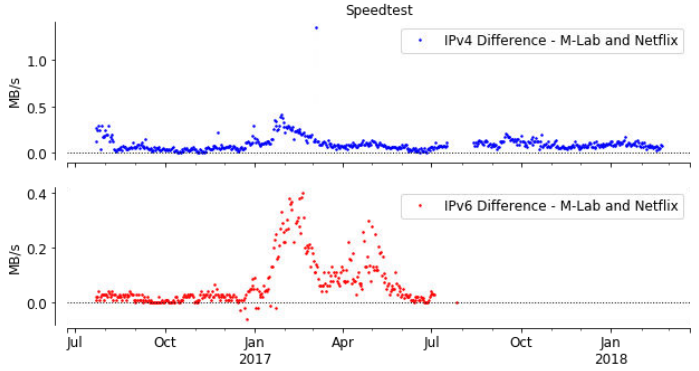

Fig. 6.5.: *Time series of difference of speedtest over IPv4 and IPv6 for M-Lab and Netflix. Here, The Netflix speedtest is subtracted from M-Lab speedtest. We plot the median aggregate across all probes on each day. The positive value indicates that the achieved speedtest is consistently lower for Netflix over both the Address family's.*

We have already defined the terminology we are using here in the beginning of this chapter. We will use the equation that we defined above where Δtp is the difference in achieved speedtest over the individual address family for M-Lab and Netflix. Also to note here, we are doing M-Lab minus Netflix here, so if the speedtest is positive then it means that the speedtest for M-Lab is better. To plot the deltas, we first start with the time series, and fig. 6.5 shows us the time series of median aggregate of speedtest across all probes over each day. We can see that the difference is consistent over time and is lower for Netflix for both IPv4 and IPv6 over the whole duration. The

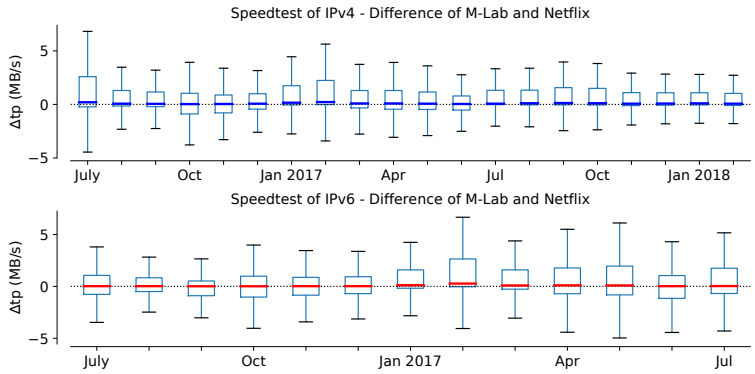

Fig. 6.6.: *Boxplot of difference of Speedtest for M-Lab and Netflix over IPv4 and IPv6. The figure also shows similar results as the time series fig. 6.5, depicting lower speedtest for Netflix over both IPv4 and IPv6.*

positive value indicates a higher speedtest for M-Lab, also the difference lies between 0-0.5 MB/s for both the address families. fig. 6.6 shows the boxplot of difference of speedtest over IPv4 and IPv6. The median line shows similar curves as the time series fig. 6.5, depicting lower speedtest for Netflix. To get a more clear picture about the performance, we plot the CDF of deltas of *bytes_sec* field. For CDF, we first compute the difference of speedtest for M-Lab and Netflix over IPv4 and IPv6 and then take the CDF of that difference. Point to note here is that we are computing the difference between M-Lab and Netflix for a specific address family, i.e. difference between IPv4 measurement of M-Lab and Netflix and difference of IPv6 measurement of M-Lab and Netflix. fig. 6.7 shows the CDF of difference of M-Lab and Netflix speedtest over IPv4 and IPv6. Around 59% of the times, M-Lab achieved higher speedtest over Netflix for both the address families i.e. IPv4 and IPv6. We can see that according to these results M-Lab achieves higher speedtest over Netflix, Now, one possible reason could be the Path length here, therefore in the next section we will consider the path length (TTL) for both M-Lab and Netflix to see how path length impact speedtest for these companies.

Is shorter path length impacting the Speedtest for M-Lab?

As already discussed in the previous section that speedtest to M-Lab servers are better as compared to speedtest to Netflix OCA servers. We wanted to find the reason for this and therefore, we would like to compare the path length i.e. TTL to M-Lab and Netflix servers. We therefore, here are comparing the TTL over IPv4 and IPv6 for M-Lab and Netflix. Also to note here, we are analysing the TTL for each probe. The reason ffor doing is that TTL as a factor is dependent on location and network of the probe. So for

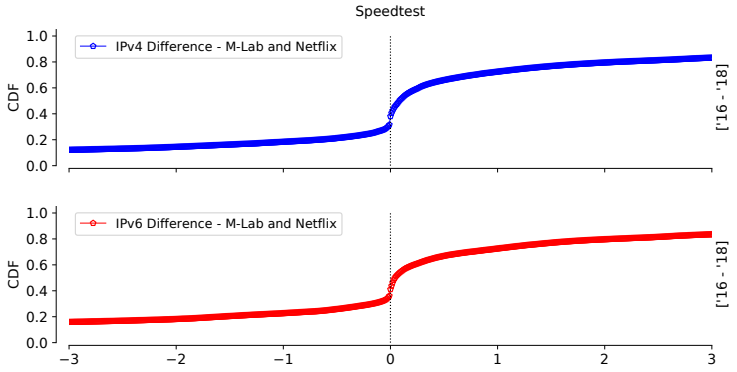

Fig. 6.7.: *CDF of Speedtest deltas for M-Lab and Netflix over Ipv4 and IPv6. Around 59% of the times M-Lab achieved higher speed over Netflix for both the address family's.*

some probe the TTL to M-Lab could be less and for some it could be more. We will thus see the distribution of Speedtest and TTL for each probe. As there are around 100 probes and we can't display everything in this study, we are just showing the results from one probe (Number 33). The results from this probe is trated as an example and mimic the results from most of the probes. The rest fo the graphs from each probe can be found at /data/akapoor/.

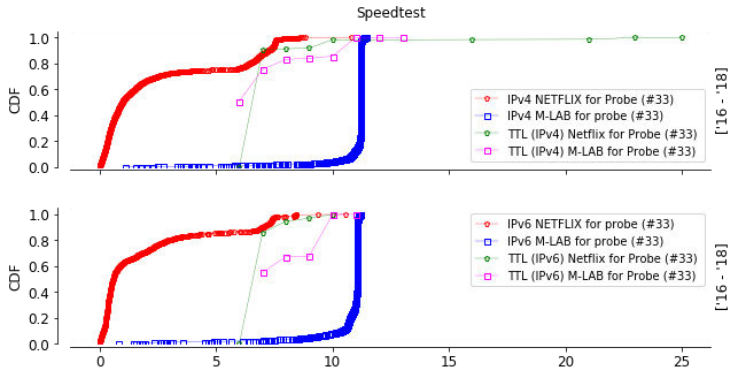

Fig. 6.8.: *CDF of speedtest and TTL over IPv4 and IPv6 for M-Lab and Netflix for Probe number 33. As can be seen, the speedtest for M-Lab is more but the TTL is also less as compared to Netflix.*

Here, fig. 6.8 shows us the CDF of speedtest and TTL over IPv4 and IPv6 for M-Lab and Netflix for probe number 33. As can be seen, the speedtest for M-Lab is more but the TTL is also less as compared to Netflix. This shows us that path length impact speedtest. To be more sure, we will now consider the deltas here, and as can be seen in fig. 6.9 the speedtest for M-Lab is better than speedtest for Netflix but the TTL for

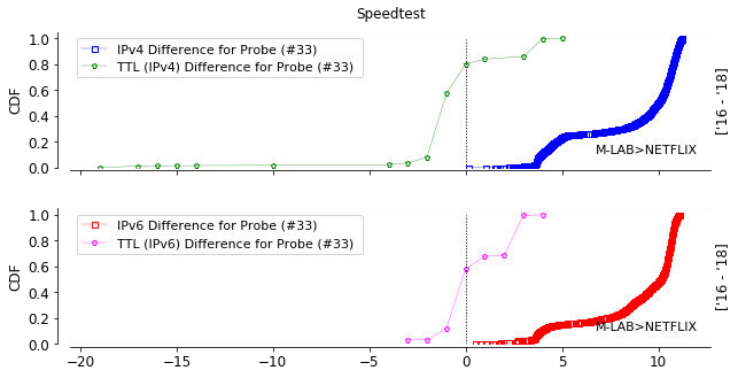

Fig. 6.9.: *CDF of Speedtest and TTL deltas for M-Lab and Netflix over IPv4 and IPv6 for probe number 33. As can be seen from the graph, the speedtest for M-Lab is better than speedtest for Netflix but the TTL for M-Lab is shorter i.e. less path length as compared to TTL for Netflix.*

M-Lab is shorter i.e. less path length as compared to TTL for Netflix. The TTL curve on the left side of Zero axis suggest that the TTL is shorter for M-Lab as we are taking the difference of M-lab minus Netflix here. Therefore, we can conclude that, path length impacts speedtest/throughput for M-Lab and Netflix.

CHAPTER 7

DOES PATH LENGTH IMPACT LATENCY FOR NETFLIX?

In this chapter we will analyse the content delivery on the Internet, we will analyse the traceroute measurements that we have in the dataset. The traces are towards Netflix OCA servers and towards ISPs content caches which hold the Netflix audio and video content. As mentioned in [36], We used scamper [64] on around 100 dual-stacked SamKnows probes [90]. The test runs *paris-traceroute* [11] for IPv4 and IPv6 every hour, towards Netflix OCA servers. The attribute that we are going to analyse is *TTL (Time-To-Live)* [82] between the client and the Netflix OCA server. Other attributes that we will analyse here are the *TCP Connect Times and the Pre-buffering Durations*, and we will see the impact of path lengths (TTL) over these latency attributes.

7.1. Temporal Analysis

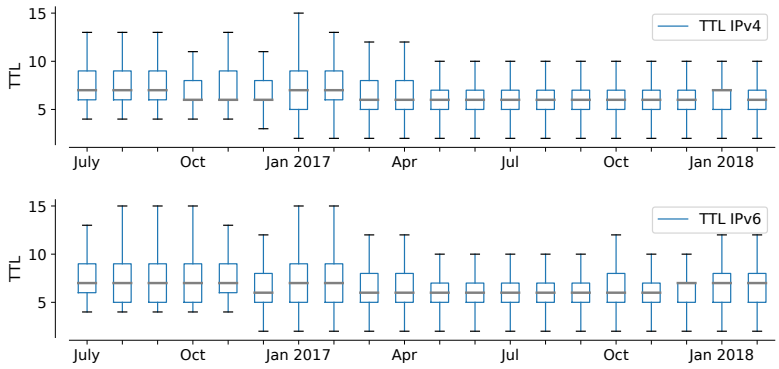

Fig. 7.1.: *Boxplot of TTL absolute values over IPv4 and IPv6, by months and across all the probes. For both the address family's, no significant changes could be observed, and the median along every month is around 5-8 over IPv4 and IPv6.*

We wanted to see the path lengths changes over the entire duration of the dataset and the impact on the latency. We are looking at the maximum TTL values i.e. the

number of hops traversed to reach the destination host (Netflix OCA here). Also, we are considering the TCP Connect Times and Pre-Buffering Duration along this timeline, to see the impact on the latency with respect to path length. As the *scamper* test ran every hour for both IPv4 and IPv6, therefore, the traceroute measurements can be compared for both the address family's. We already computed the difference between the two address family's, refer table *deltas* in table 3.14. As Viet [35] did in his study, we are using the "COMPLETED" measurements here and are considering the maximum TTL values here for computing the deltas, reason for using maximum TTL value is that most of the clients would prefer faster connection time irrespective of the path length and the intermediate hops between the source and the destination hosts.

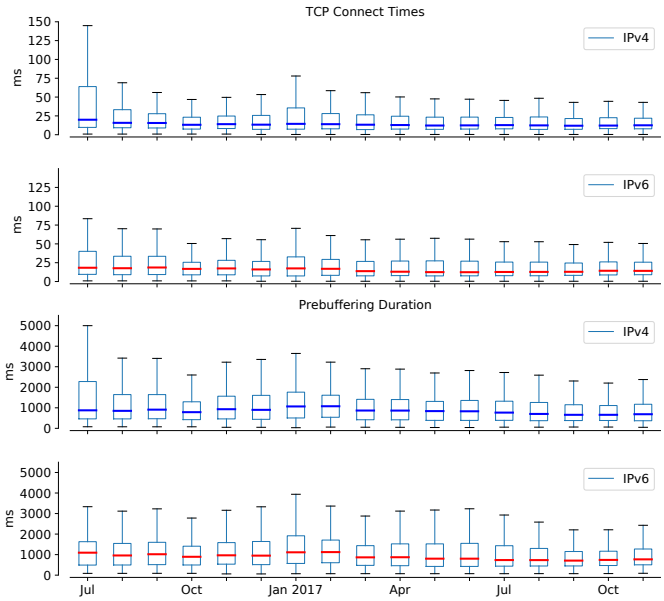

Fig. 7.2.: *Boxplot of TCP connect times and Pre-buffering duration over the entire duration, across all probes. We see a similar pattern over both IPv4 and IPv6 address family's.*

We will first consider the absolute values for the attributes over IPv4 and IPv6. As can be seen in fig. 7.1 the first and third quartiles for maximum TTL values over IPv4 and IPv6 lies between 0-15 hops for the entire duration, and the median is between 5-8 hops for individual months. Also, we see a steady graph i.e. no significant changes during the entire duration. Similarly, we can see the absolute values over IPv4 and IPv6 for the TCP Connect Times and Pre-buffering Durations for the whole timeline, in fig. 7.2. Here, we have converted the TCP connect times and Pre-buffering duration to 'ms' to get a better understanding. As can be seen, there are no major changes along the whole time-line and, we cannot see much difference between IPv4 and IPv6

performance, so it would be good to compare the deltas here.

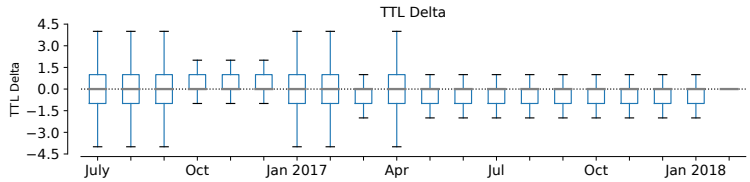

Fig. 7.3.: *Boxplot of TTL delta, along the entire duration and across all probes. No significant changes can be seen here, and the median stays around 0 indicating IPv6 performs at par with IPv4.*

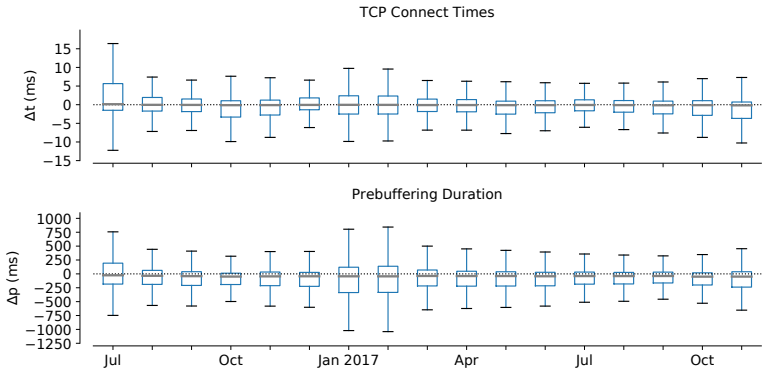

Fig. 7.4.: *Boxplots of TCP connect times and Pre-buffering duration deltas, across all probes. The median remains around 0 indicating IPv6 performance is comparable to IPv4.*

As *deltas* here are computed as the difference between IPv4 and IPv6, the positive values indicate that the IPv6 need less hops to reach the destination. Ideally, as IPv6 adoption has increased, with increased deployment of IPv6 infrastructure and support, the deltas should be around 0, indicating that IPv6 is on par with IPv4. We plotted the deltas as a Bocplot, and as we can see from fig. 7.3, there are no significant changes that could be seen along the entire duration. The median is along 0, which indicates that the IPv6 performed at par with IPv4 in terms of TTL. The reason for varying first and third quartile among certain months is unclear. Similarly, for TCP connect times and Pre-buffering duration, the median stays steady around 0, indicating similar performance over IPv4 and IPv6.

The above analysis provide a holistic view of the traceroute measurements towards Netflix OCA servers, because the SamKnows probes are located all over the world. Also, it can be seen that no drastic changes could be seen over the last two years.

7.2. Path Length and Latency Comparison

Here, we are considering the path length across all probes for the entire duration without considering the time factor. fig. 7.5 describes the CDF of median TTL over the entire duration. Here, around 93% of the IPv4 paths have a TTL of 13 or lower, while 93% of the paths over IPv6 have a TTL of 12 or lower. As the number of paths are similar over both address family's. the data indicates that more paths are shorter over IPv6 as compared to IPv4. This can also be seen from the figure, as IPv6 datapoints (Red) are above the IPv4 datapoints (Blue) on the graph. The difference is not that prominent and thus, both the address family's showing comporable performance.

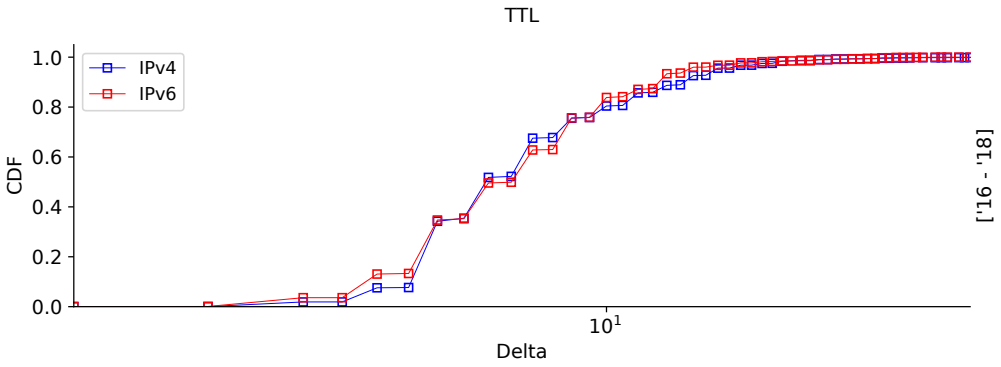

Fig. 7.5.: *CDF of TTL over IPv4 and IPv6. Both the curves, IPv4 and IPv6, are largely overlapping indicating similar performance over both the address family's.*

For TCP Connect times and Pre-Buffering duration, we have already discussed the fig. 7.6 in e cannot see much difference in the performance over IPv4 and IPv6 and therefore, both the address family's show similar latency.

We now can conclude that path lengths over IPv6 and IPv4 are comparable and the latency i.e. TCP connect times and Pre-Buffering duration are also comparable over both the address families. We therefore need to see the deltas for better comparison.

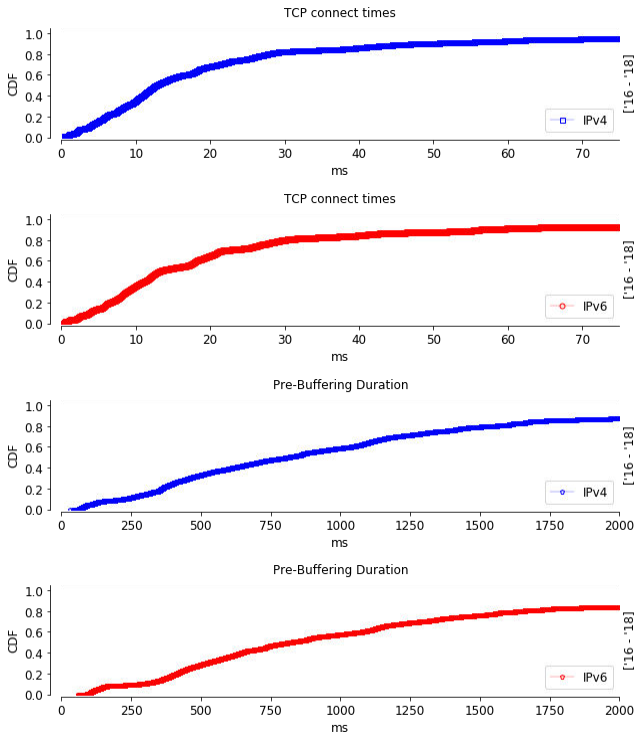

Fig. 7.6.: *CDF of TCP Connect Times and Prebuffering Duration for IPv4 and IPv6. We cannot see much difference between the address family's.*

7.3. Path Latency and Latency Deltas

To get a more clear picture, we plotted the distribution of TTL, TCP connect times, and Pre-buffering duration. We can see the TTL and TCP connect times deltas in fig. 7.7. The plot shows that around 31% of the pairs had a median TTL of less than 0, whereas around 33% of the pairs had a positive delta which indicates less number of hops over IPv6. The remaining 36% of the median TTL deltas are located at 0, which illustrates that fraction of median TTL over IPv6 being shorter, equal or faster to IPv4 are somewhat same. If we compare the TCP connect times, then the distribution shows that 57% of the connections are slower over IPv6, with 13% of them at least 10 ms slower. Also, if we check the Pre-Buffering duration deltas in fig. 7.8, we can see from the distribution that around 70% of the connections are slower over IPv6.

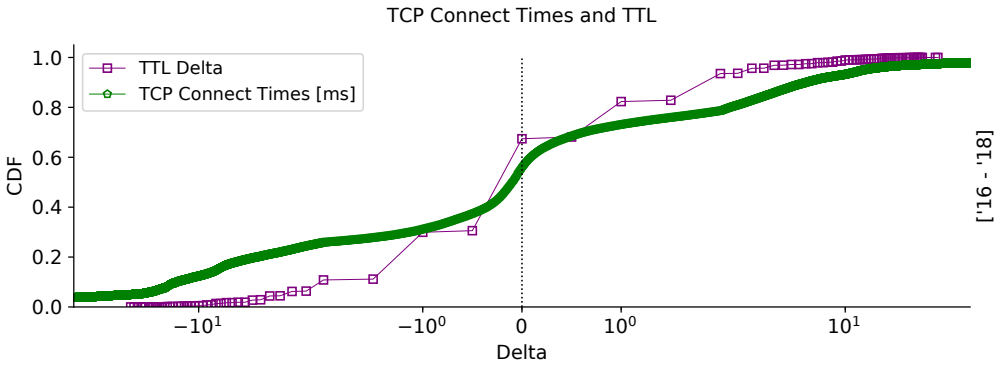

Fig. 7.7.: *CDF of TCP Connect Times and TTL over the entire duration and across all probes. Here, TTL over IPv4 is shorter, equal or longer in about one/third of the measurements compared to IPv6. Regarding TCP Connect times, the distribution here shows that 57% of the connections are slower over IPv6, with 13% of them at least 10 ms slower.*

Going forward, we will analyse the destinations based on CAIDA's [26] classifications, identifying the content caches and the OCA hosts.

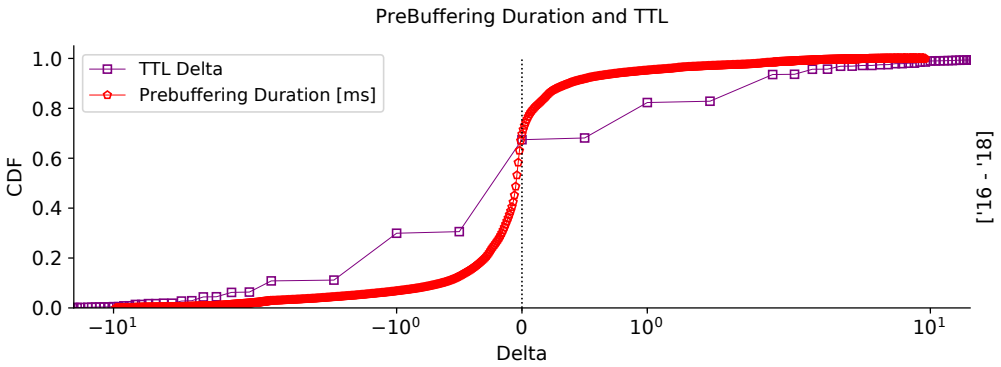

Fig. 7.8.: *CDF of TTL and Pre-buffering Duration over the entire duration and across all probes.Here, TTL over IPv4 is shorter, equal or longer in about one/third of the measurements compared to IPv6. Regarding the Pre-Buffering Duration, distribution shows that around 70% of the connections are slower over IPv6.*

7.4. Content Caches

Content Caches helps reduce the access time to frequently accessed data, as they mirror the content temporarily so as to reduce the load on the original hosts. It is expected and believed that the TTL and Latency should be lower incase a cache is hit as when the origin server (Netflix OCA here) is hit. The reason for this is that the caches are situated closer to the client which make the requests, generally at the ISP networks (refer We can identify the the content caches by comparing the AS Number of the source IP (table 3.8 with the AS Number of the destination IP (table 3.7). If there is a match then it is a cache located near to the probe and the client request never left the ISP network.

Viet in [35] mentioned four different scenarios to identify a cache. These are,

1. IPv4 only cache, i.e. no cache for IPv6.
2. IPv6 only cache, i.e. no cache for IPv4.
3. Both Caches, i.e. ISP caches identified for both IPv4 and IPv6.
4. No ISP cache hit for niether IPv4 or IPv6.

The above four scenarios could be visualized if we split up the graphs in fig. 7.7 and fig. 7.8, based on the above four conditions. Considering only the TTL, hypothetically, an IPv4 only cache should be on the negative side of the x-axis as the paths will be shorter for IPv4 cache. Similarly, for IPv6 only cache, the shift should be on the positive

side of the x-axis. For both the caches, the values should be around 0. If this isn't the case then the possible reason can be regarding the network infrastructure and configuration of the ISP caches. For case 4, where there isn't any cache, the plot should be a less steep curve.

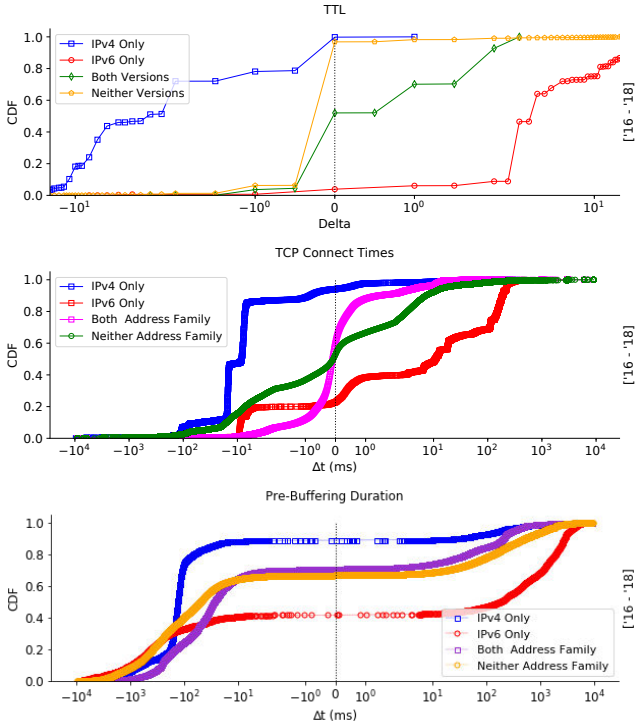

Fig. 7.9.: *CDF over the TTL, TCP and Pre-Buffering duration deltas of destination pairs, identified as cache hits as: 1. IPv4 only, 2. IPv6 Only, 3. Both versions, and 4. Neither Versions. Hitting an ISP cache seems to have improvement over TTL, TCP and Prebuffering Durations.*

Now lets us discuss the plots in **??????**.

1. Here, we can see that the for the IPv4 only case, the curve is shifted towards the left side. For IPv4. TTL measurements of around 47% are shorter by exactly 4 hops. For TCP connect times around 94% of the measurements are less than 0 ms and for Pre-buffering duration around 89% of the measurements are less than 0ms.

2. For IPv6 only case, the right shift on the curves suggest that the IPv6 caches were hit, and mainly all measurements to the IPv6 cache shows shorter path length. Around 23% of the measurements for TCP connect times are less than 0 ms, and around 42% of the measurements are less than 0ms for Prebuffering duration.

3. For both caches i.e. IPv4 and IPv6 requires the same amount of hops, i.e. around 50% of the measurements are resulting in delta of 0. Concerning the TCP connect times, around 60% of the measurements reached IPv4 faster, and rest 40% reasched IPv6 faster. In case of Prebuffering durations, around 71% of the measurements reached IPv4 faster, while the rest 29% reached IPv6 faster.

4. For the case where neither cache was hit, i.e. no match between source AS number and Destination AS number, the curve resembles a flat curve.

Performance Improvements by Caches

After checking the cache hits, we would like to see the performance impact of these caches. As can be seen from the previous section, a cache hit leads to shorter TTL, connect times and pre-buffering duration for that IP address family. Now, to study the impact of this on the performance, we plotted the absolute values of TTL, TCP connect times and pre-buffering duration. We considered two cases here, first when the cache was hit and the second when it was missed. The criteria of selecting the cache is same as previous, that is, source IP ASN is equal to destination IP ASN.

For **IPv4**, 96% of the traces had a TTL of 6, whenever a cache was hit, whereas only 78% of the cache misses met a TTL of 6. Coming to TCP connect times, cache hit had a connect time of 21 ms for 90% of the measurements, while for cache misses this is 62 ms. Similarly, for pre-buffering duration, 90% of the measurements took 1801 ms whenever there was a cache hit, whereas for cache misses the pre-buffering duration was around 2428 ms.

For **IPv6**, 99% of the traces had a TTL of 6, whenever a cache was hit, whereas only 79% of the cache misses met a TTL of 6. For TCP connect times, cache hit had a connect time of 21 ms, whereas for cache misses this is 69 ms for same number of measurements. Similarly, for pre-buffering durations, 90% of the measurements took 1992 ms whenever there was cache hit, whereas for cache misses this is 2752 ms for same number of measurements.

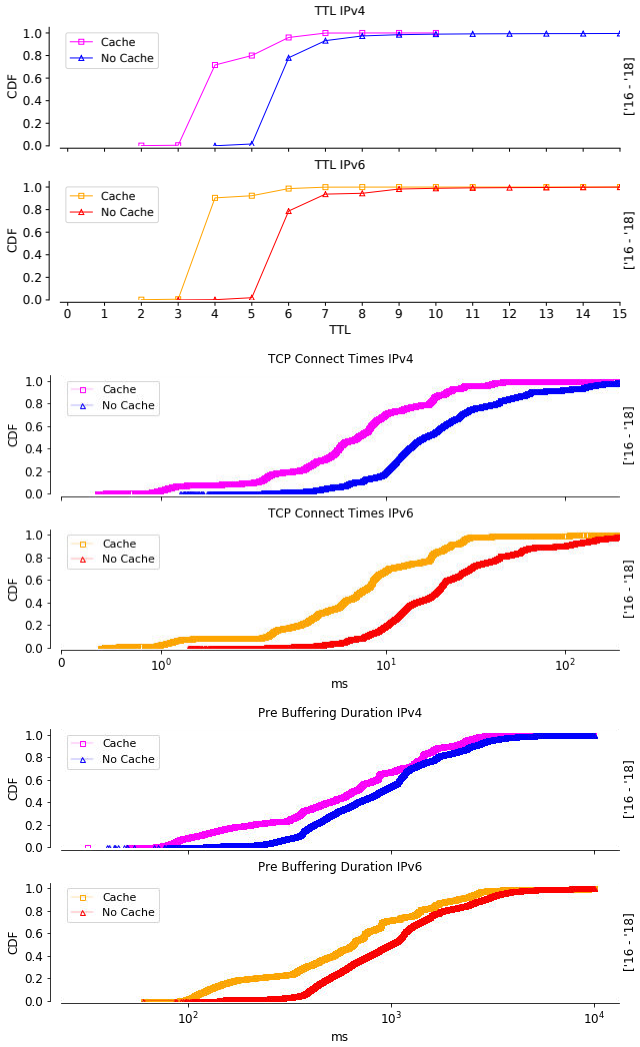

Fig. 7.10.: *CDF over the TTL, TCP and Pre-Buffering duration over IPv4 and IPv6. Cache hits curves are above the curves for cache misses, which depicts effectiveness of caches with reduced path length (TTL) and latency.*

7.5. Path Analysis

Till now, we were considering only the last rows of the traceroute measurements for a particular measurement. An interesting aspect will be to look into the intermediate hops along the path. This could reveal some additional information regarding the state of IPv6 or ISP content caches. We therefore, will consider the original *traceroute-filtered*

table, with "COMPLETED" results but will contain the total path information for a specific source and destination pair.

Classification of Ases

We split the data based on the AS type of the intermediate hops i.e. the endpoints. We mapped the table 3.9 to the **??**, so that every endpoint belongs to one of the three AS types defined by CAIDA [26]. These three types are: *Content, Enterprise, or Transit/Access*.

For filtering we followed the same strategy as Viet [35] did in his study. As per Viet study, we also observed that Content and Enterprise ASes were traversed on the first hop, which shouldn't be the case given that first hop is generally the local network router. One possible reason for this could be that we were considering non-residential probes as well. Therefore, We further limit the observation to residential probes only. We used the metadata to get the probes to filter the probes which are located in a residential setting [12].

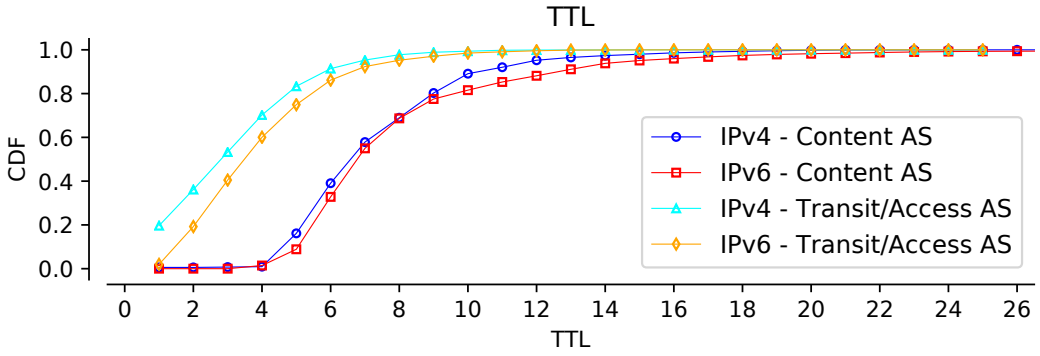

Fig. 7.11.: *CDFs over the TTL values, based on AS type. Both IPv4 and IPv6 showed similar results. TTL value of 5 marks the highest difference between the AS types, indicating that most of the endpoints outside this range belong to Content ASes, and within this range belong to Transit ASes.*

We ignored the "Enterprise" measurements as they were very less compared to Content or Transit type. We could only see 40 "Enterprise" labels and thus didn't consider them in our analysis. We can see the distribution of TTL values based on AS types: *Content and Transit*. As per the graph, we can see that TTL value of 5 marks the highest difference between the AS types, indicating that most of the endpoints outside this range belong to Content ASes, and within this range belong to Transit ASes. Around 83.3% of the endpoints assigned to Transit/Access ASes had a TTL value of less than or equal to 5 for IPv4, and 74.9% for IPv6. Similarly, around 83.9% of the

IPv4 endpoints classified as Content ASes had TTL values of 5 or greater than 5, for IPv6 this percentage was around 91.2%.

This shows that TTL value of 5 can be treated as a kind of separator between Transit/Access and Content ASes. Therefore, it signifies that, a Netflix traceroute measurement is likely to be an ISP cache if it ends in less than 5 hops. Also, the majority of ASes within the range of 5 were seen to be located in Internet Service Provider networks. Similarly, TTL value of 5 and above suggests that the AS is most likely a Content AS.

Conclusion

CHAPTER 8

CONCLUSION AND DISCUSSION

Conclusion

The study focuses on the present state of IPv6 and its performance for Netflix. We focus on the performance metrics such as *success rate, TCP connect times, throughput, stall rate etc* and the benefits of ISP content caches for Netflix using data collected by longitudinal dual-stack SamKnows probes towards the Netflix OCA servers. We also analyzed the speedtest dataset, which includes measurements towards M-Lab servers.

RQ 1. How does IPv6 performance compare with IPv4 for Netflix? We started by analyzing the *error occurrence rate* and showed that occurrence rate for error mostly lies between 15%-45% for a single day, also the errors are evenly distributed across each probe and it's not that only some probes are generating errors. Coming to the success rate, the median success rate was around 100% for the whole duration over IPv4 and IPv6. Also, around 79% of the probes achieve a success rate of more than 90% for IPv4, while for IPv6, only around 63% of the probes were able to achieve this success rate. We also got to know that most of the results are influenced by *residential* probes, as out of 100 probes 79 are residential. The results show that a Happy Eyeballs (HE) race during initial TCP connection establishment leads to a strong (around 93%) preference over IPv6. However, even though clients prefer streaming videos over IPv6, worse performance over IPv6 than over IPv4 was observed, whereby consistent higher TCP connection establishment times and pre-buffering duration ($\sim$40 ms or more) were witnessed over IPv6. Similarly, consistent lower achieved throughput over IPv6 was also observed. Less than 10% stall rates over both address families were observed. All these findings are discussed in chapter **??**.

RQ 2. How beneficial are the ISP content caches? How do the performance over IPv4 and IPv6 compare in accessing these caches? Chapter 5 discusses the benefits of ISP content caches for Netflix. As content delivery is becoming very important, and chapter 2 also discusses the trend about Internet network topology and that it is

becoming "flat", content caches are being deployed within the ISPs to improve user experience by improving content delivery performance. We observed that ISP content caches do have an impact on the latency and throughput, and as per the results it was observed that content caches lead to reduced TCP connect times and Pre-buffering duration. A latency of around 10-20 ms and higher pre-buffering durations (around 0-750ms or more) are observed for Netflix over IPv4 and IPv6 as compared to ISP Content caches. Also, Content caches achieved higher throughput (around 66% of the times) as compared to Netflix over both IPv4 and IPv6.

RQ 3. How do IPv4 and IPv6 Speedtest results compare for Netflix and Measurement Lab? As Speedtest is becoming a commercial way of checking QoS of broadbands, we analyzed the speedtest measurements towards M-Lab servers and Netflix OCA servers in chapter 6. The comparison between M-Lab and Netflix speedtest reveals that the speedtest is better towards M-Lab servers than towards Netflix OCA servers. Around 59% of the times, M-Lab achieved higher speedtest over Netflix for both the address families i.e. IPv4 and IPv6. We did find out that the path length (TTL) to M-Lab servers is comparatively less than the TTL to Netflix OCA servers, indicating that shorter path lengths correlate with a higher speed for M-Lab.

RQ 4. How do path length, latency, and delay compare over IPv4 and IPv6 for Netflix? From the traceroute measurements towards Netflix, it was observed that content caches had reduced path lengths, TCP connect times and Pre-buffering durations, and can be reached within 5 hops and 21ms. Also, the fraction of median TTL over IPv6 being shorter, equal or faster to IPv4 are somewhat same. We observed that the pre-buffering duration was around 1801 ms over IPv4 for a cache hit, and 2428 ms when there was a cache miss for 90% of the measurements. For IPv6, the pre-buffering duration was 1992 ms for a cache hit, and 2752 ms when there was a cache miss for 90% of the measurements. An interesting aspect we looked into is the intermediate hops along the path. Considering AS types, the results showed that *Transit/Access* and *Content* Ases can be separated around a TTL value of 5, which signifies that, a Netflix traceroute measurement is likely to be an ISP cache if it ends in less than 5 hops.

Limitations

The study faces few limitations as most of the probes are situated in Europe (60) and America (32), the results are biased towards these regions. One thing to note here is that the state of present IPv6 deployment is centered around these regions, but it may change in the future. Also, splitting the observations by Region or Network type goes down significantly and therefore, we cannot conclude the performance of IPv6 based

on regions or Network type. Thus, a more general view of IPv6 connectivity cannot be studied.

Our analysis is also limited by *paris-traceroute* which has a drawback such as non-cooperative routers or slow processing of ICMP packets [35], which may not give the actual path length and latency aspects. Also, some routers assign lower priority to ICMP packets, which may impact the actual TCP connect times and pre-buffering duration. Also, few residential probes changed their ISP providers (around 10%) and thus this may have impacted our content cache analysis, but we did check this and the overall picture remain the same.

The identification of caches is limited to the criteria of matching of source ASN and destination ASN. There could be caches deployed in different ASN of an ISP or could be placed in peers of the ISP. Also, as [24] suggests, Netflix tends to place their OCAs at Internet Exchange Points which could also be a cache. CDN providers also place caches outside the ISPs as well. However, the goal of this study was not to identify such caches but to measure the latency and delay when a cache was hit.

Future Work

Overcoming the limitations described in the last section could be one of the reasons to carry out future studies related to Netflix performance over IPv6. More probes could be set up all over the world especially to regions where there are fewer probes like in Africa or in Western Pacific. This will reduce the geographical bias that we faced in our study. Another aspect could be to diversify the probes based on network type.

We faced some question while analyzing, like why is the pre-buffering duration higher over IPv6, given be it a Netflix CDN or an ISP cache. This could be looked into more detailed, by analyzing the pre-buffering duration by different geographical regions, or by a timeline. It could be that a certain Geographical region is causing this higher pre-buffering duration for IPv6. Also to further study here is, that TCP connect times are congruent over both address family's whenever a cache is hit, it would be good to explore why this doesn't happen for the Pre-buffering duration.

The data was collected via fixed-line connections [35], collecting data from Mobile or IoT devices could provide great insights into the performance of IPv6. The selected probes Internet Service Provider can be made static so that the research results don't get alter if the user changes her ISP. Furthermore, better aggregation techniques could be looked up to filter and select the important data points. Cache identification is one such important factor where better techniques to look up the caches can be used. In the end, increase in IPv6 adoption by the Internet Service Providers and the content providers will provide immense knowledge about its performance as compared to IPv4. As an increase in IPv6 traffic will provide more insights about its functioning under

heavy load.

Reproducibility

CHAPTER 9

REPRODUCIBILITY CONSIDERATIONS

The SQLite databases that we used for analysis are hosted at the Leibniz-Rechenzentrum (LRZ) virtual machine. The virtual machine is under the provision of Chair of Connected Mobility, Technische Universität München [68]. The hostname of the VM is cherry and is running *ubuntu 16.04* and has a RAM of 300G, plus we used SQLite3 (version 3.11.0) and python3 (version 3.5.2). The list of python packages used is listed in table 9.1.

To allow reproduction of the results, all the jupyter notebooks that we used for this study are committed to LRZ gitlab [54]. The database paths are hardcoded, please adjust them as per your dataset path.

Dataset and Analysis Directories

The following diagram gives the overview of the dataset directory hosted on *cherry*. Here, *netflix.db* and *speedtest.db* are the two main datasets that we used for this research.

```
Cherry
└── data
    └── akapoor
        ├── databases
        │   ├── netflix.db
        │   └── speedtest.db
        └── 2018-akapoor-masters-analysis
```

Table 9.1.: *Python Packages used for this Study*

Package Name	Version	Package Name	Version
beautifulsoup4	4.4.1	bleach	2.1.1
certifi	14.5.14	chardet	2.3.0
command-not-found	0.3	csvkit	0.9.1
cycler	0.10.0	decorator	4.1.2
entrypoints	0.2.3	gnureadline	6.3.3
html5lib	1.0b10	ipykernel	4.6.1
ipython	5.4.1	ipython-genutils	0.2.0
ipywidgets	7.0.3	jdcal	1.0
jedi	0.11.0	Jinja2	2.9.6
jsonschema	2.6.0	jupyter	1.0.0
jupyter-client	5.1.0	jupyter-console	5.2.0
jupyter-core	4.3.0	language-selector	0.1
lxml	3.5.0	MarkupSafe	0.23
matplotlib	2.1.0	mistune	0.8
nbconvert	5.3.1	nbformat	4.4.0
netaddr	0.7.19	nose	1.3.6
notebook	5.2.0	numpy	1.14.0
openpyxl	2.3.0	pandas	0.22.0
pandocfilters	1.4.2	parso	0.1.0
pbr	3.1.1	pexpect	4.2.1
pickleshare	0.7.4	Pillow	3.1.2
prompt-toolkit	1.0.15	ptyprocess	0.5.2
Pygments	2.2.0	pygobject	3.20.0
pyparsing	2.0.3	pytest	2.8.7
python-apt	1.1.0	python-dateutil	2.4.2
python-debian	0.1.27	python-systemd	231
pytz	2015.2	pyzmq	16.0.2
qtconsole	4.3.1	requests	2.9.1
scipy	1.0.1	seaborn	0.8.1
simplegeneric	0.8.1	six	1.11.0
SQLAlchemy	1.0.11	ssh-import-id	5.5
testpath	0.3.1	tornado	4.5.2
traitlets	4.3.2	ufw	0.35
unattended-upgrades	0.1	urllib3	1.13.1
virtualenv	15.1.0	virtualenv-clone	0.2.6
virtualenvwrapper	4.8.2	wcwidth	0.1.7
webencodings	0.5.1	widgetsnbextension	3.0.6

Analysis Mappings

We looked up the ASN and Holder names using RIPEstat [88], and used the scripts that were generated during the Python3 toolset for measurement study by Bajpai and et al. [40]. The Google IPv6 adoption plot was generated using the scripts written by Bajpai in [13]. Also, the *Pmf.py and Cdf.py* modules that we used are published by Downey [39]. The jupyter notebooks used for generating the plots are mapped to the specific section in table 9.2. We have structured the notebook directories based on the chapters of this study. So, for reproducing the specific section, just refer to the directory for that section and all the jupyter notebooks for that section are in that directory. The name of the notebooks is self-explanatory to the specific figure that you may want to reproduce.

Table 9.2.: *Python Packages used for this Study*

Notebook Directory	Figures
/data/akapoor/2018-akapoor-masters-analysis/success/	4.1 - 4.22, B.1 - B.14
/data/akapoor/2018-akapoor-masters-analysis/preference/	4.23 - 4.26, B.15
/data/akapoor/2018-akapoor-masters-analysis/connect/	4.27 - 4.33, B.16
/data/akapoor/2018-akapoor-masters-analysis/throughut/	4.33 - 4.40
/data/akapoor/2018-akapoor-masters-analysis/stall/	4.41 - 4.46
/data/akapoor/2018-akapoor-masters-analysis/cache-analysis/	5.1 - .5.25, B.17 - B.28
/data/akapoor/2018-akapoor-masters-analysis/speedtest/	6.1 - 6.9
/data/akapoor/2018-akapoor-masters-analysis/traceroute/	7.1 - 7.11

Reproduction Steps

For the reproduction of the results, most of the notebooks are self-explanatory except for the last chapter that is the traceroute analysis. For traceroute analysis, we followed the same steps as Viet [35] did in his study. Chapter wise steps that need to be done are described in the following sub-sections. If the already aggregated datasets are being used then the following section can be skipped.

Chapter 4: IPv4 versus IPv6

There not specific steps that you need to perform before proceeding for the analysis. Just run the desired notebooks to get the results.

Chapter 5: Benefits of Caches

Here, before proceeding with the analysis, we need some metadata information to get the ASNs for each probe. For this, we used the [40] scripts to get the *asn_from_probe_name*. Next, we need the *target ASN* i.e. the ASN of the *targets* present in the dataset. We need this information as for identifying the caches we are using the criteria of matching source ASN and destination ASN. After this, just follow the notebooks.

Chapter 6: M-Lab versus Netflix

For this, we don't need to do any aggregation for reproducing the results. The notebooks are self-explanatory, and the results can be achieved by just running them.

Chapter 7: Traceroute Analysis

Here, as we followed the same strategy Viet [35] followed, the aggregation steps are based on his work. First, we will need to get all the ASN for the *source, destination and endpoint*, for this we will need to run the (src_ip_to_asn.py, dst_ip_to_asn.py, dst_ip_to_hostname.py, endpoint_asn_lookup.py) scripts. After this, filter_data.py, is used to filter Hurricane electric measurements and to remove duplicate rows. With the newly filtered table (*traceroute-filtered*), we can run the *create_tables.ipynb* notebook, which will create the remaining tables that we are using in the analysis.

Appendices

APPENDIX A

DATASET

Netflix Dataset

Table A.1.: *AS Names for IPv4 Addresses*

AS Name	#IPv4 Addresses
AS-SSI - Netflix Streaming Services Inc.	2827
BSKYB-BROADBAND-AS - Sky UK Limited	48
BT-UK-AS - British Telecommunications PLC	38
TELENOR-NEXTEL - Telenor Norge AS	30
TEKSAVVY - TekSavvy Solutions	23
BELGACOM-SKYNET-AS - Proximus NV	22
EIRCOM - Eircom Limited	8
INTERNODE-AS Internode Pty Ltd	7
GET-NO - Get AS	7
PLUSNET - British Telecommunications PLC	6
NEXTGENTEL - NextGenTel AS	6
COMHEM-SWEDEN - Com Hem AB	6
ALTIBOX_AS - Altibox AS	6
AS-VOBIZ - vanoppen.biz LLC	6
ASN-CATCHCOM - Broadnet AS	6
ALLST-15290 - Allstream Corp.	6
XS4ALL-NL - Xs4all Internet BV	6
SNAP-NZ-AS Snap Internet Limited	5

Table A.2.: *Continuing table A.1*

AS Name	#IPv4 Addresses
BEANFIELD - Beanfield Technologies Inc.	5
RCN-AS - RCN	5
AUREON-5056 - Aureon Network Services	5
ASBRUTELE - Brutele SC	4
SUNRISE - Sunrise Communications AG	4
CALLPLUS-NZ-AP CallPlus Services Limited	4
CENTURYLINK-US-LEGACY-QWEST - Qwest Comm.	3
BREDBAND2 - Bredband2 AB	3
WAIA-TRANSIT-AP Western Australian Internet Association	2
NORDUNET - NORDUnet	2
MNET-AS - M-net Telekommunikations GmbH	2
PENNREN - KINBER	2
EWETEL - EWE TEL GmbH	2
MOBILEONELTD-AS-AP MobileOne Ltd.	2
INIT7 - Init7 (Switzerland) Ltd.	2
G8 NETWORKS LTDA	1
T-MOBILE-AS21928 - T-Mobile USA	1
NMAX - Nmax	1
PREMIER-COMMUNICATIONS - Premier Communications	1
WEBFOCO TELECOMUNICACOES LTDA	1
TBNet Informatica LTDA	1
ASSOCIAÇÃO NACIONAL PARA INCLUSÃO DIGITAL	1
PSC-EXT - Pittsburgh Supercomputing Center	1
TEKSAVVY-WEST - TekSavvy Solutions	1

Table A.3.: *AS Name for IPv6 Addresses*

AS Name	#IPv6 Addresses
AS-SSI - Netflix Streaming Services Inc.	2785
BSKYB-BROADBAND-AS - Sky UK Limited	45
BT-UK-AS - British Telecommunications PLC	39
TELENOR-NEXTEL - Telenor Norge AS	30
TEKSAVVY - TekSavvy Solutions	23
BELGACOM-SKYNET-AS - Proximus NV	21
ALGAR TELECOM S/A	8
GET-NO - Get AS	7
COMHEM-SWEDEN - Com Hem AB	7
PLUSNET - British Telecommunications PLC	6
ALTIBOX_AS - Altibox AS	6
SNAP-NZ-AS Snap Internet Limited	6
XS4ALL-NL - Xs4all Internet BV	6
EIRCOM - Eircom Limited	6
AS-VOBIZ - vanoppen.biz LLC	6
AUREON-5056 - Aureon Network Services	5
NEXTGENTEL - NextGenTel AS	5
BEANFIELD - Beanfield Technologies Inc.	5
BREDBAND2 - Bredband2 AB	4
ASBRUTELE - Brutele SC	4
SUNRISE - Sunrise Communications AG	4
INTERNODE-AS Internode Pty Ltd	4
UUNET - MCI Communications Services	3
ASN-IINET iiNet Limited	2
NORDUNET - NORDUnet	2
TPG-INTERNET-AP TPG Telecom Limited	2
INIT7 - Init7 (Switzerland) Ltd.	2
MNET-AS - M-net Telekommunikations GmbH	2
PENNREN - KINBER	2
MOBILEONELTD-AS-AP MobileOne Ltd.	2
WAIA-TRANSIT-AP Western Australian Internet Association	1
PREMIER-COMMUNICATIONS - Premier Communications	1
Rede Connect Telecom	1
CALLPLUS-NZ-AP CallPlus Services Limited	1
TEKSAVVY-WEST - TekSavvy Solutions	1

APPENDIX B

DATA ANALYSIS

B.0.1. Error and Success Rate

Error Occurrence Rate

We showed the error occurrence rate for all errors in **??**, error occurrence rate for each error is depicted in fig. B.9. The error occurrence rate for "STALLED_WILL_STEP_DOWN" is shown in fig. B.3, and it is between 5%-25% for a single day. Error occurrence rate for other errors, "NETWORK_CONTENT_ERROR" in fig. B.4 lies between 0%-22%, "Network_API_ERROR" in fig. B.6 lies between 0%-14%, "STALLED_ON_FINAL" in fig. B.8 lies between 0%-0.5%, "DNS_RESOLUTION_CONTENT_ERROR" in fig. B.1 lies between 0%-2%, "CONNECTION_CONTENT_ERROR" in fig. B.5 lies between 0%-2%, "DNS_RESOLUTION_API_ERROR" in fig. B.2 lies between 0% - 2%, "CONNECTION_API_ERROR" in fig. B.7 lies between 0%-8%.

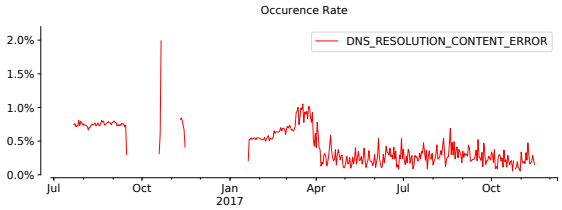

Fig. B.1.: *(a)*

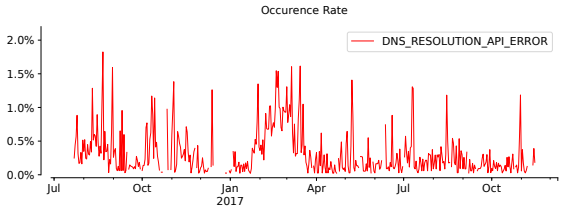

Fig. B.2.: *(b)*

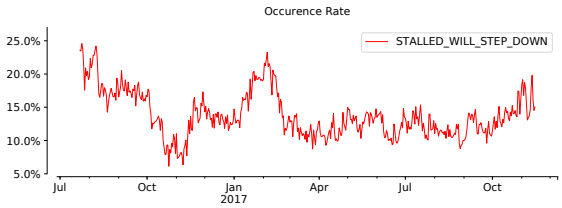

Fig. B.3.: *(c)*

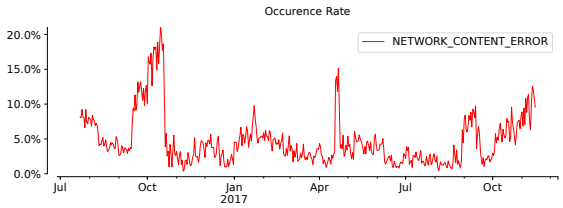

Fig. B.4.: *(d)*

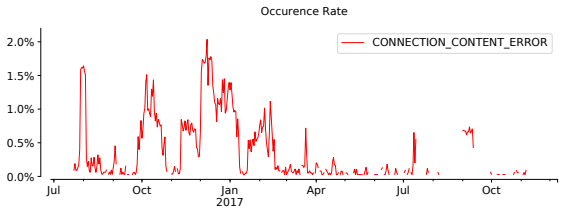

Fig. B.5.: *(e)*

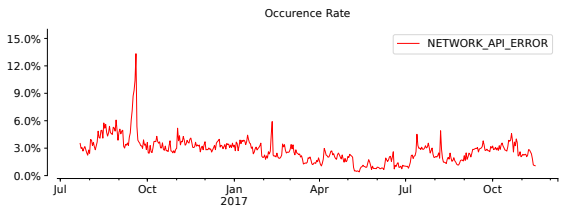

Fig. B.6.: *(f)*

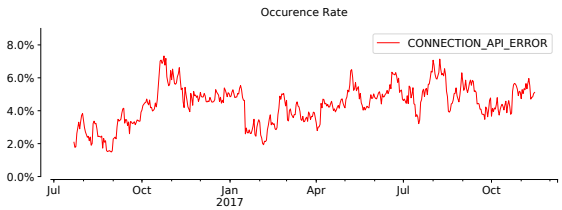

Fig. B.7.: *(g)*

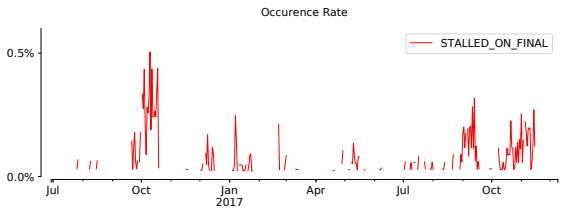

Fig. B.8.: *(h)*

Fig. B.9.: *(a) Error Occurrence Rate for DNS Resolution Content Error, (b) Error Occurrence Rate for DNS Resolution API Error, (c) Error Occurrence Rate for Stalled Will Step Down Error, (d) Error Occurrence Rate for Network Content Error, (e) Error Occurrence Rate for Connection Content Error, (f) Error Occurrence Rate for Network API Error, (g) Error Occurrence Rate for Connection API Error, (h) Error Occurrence Rate for Stalled On Final Error*

Success Rate

As already discussed in section 4.1, we are comparing the success rate over IPv4 and IPv6. fig. B.10 depicts the time series of success rate when we are plotting all the data

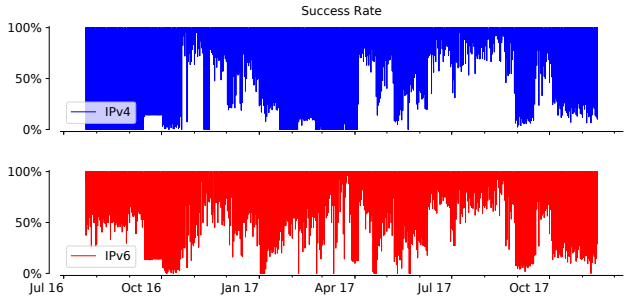

Fig. B.10.: *Time series of success rate over IPv4 and IPv6. We are plotting all the datapoints and not just the median aggregate as in fig. 4.5. IPv4 shows lower success rate of many days over the timeline.*

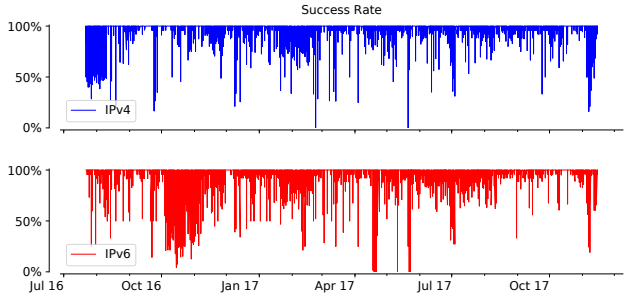

Fig. B.11.: *Time series of success rate over IPv4 and IPv6, removing the 30 probes based on fig. 4.4(b).*

points and not just the median aggregate across all probes on each day as in fig. 4.5. We can see from the density that the success rate over IPv4 is lower for many days along the timeline. We now remove 30 probes based on fig. 4.4(b) which have a lower success rate over both the families. fig. B.11 shows us the time series after removing these outliers. As can be seen both the families show similar spikes, depicting lower success rates on some days. fig. B.12 and fig. B.13 depicts the CCDF of success rate over IPv4 and IPv6 for the month of August 2016 and October 2017 respectively. For August 2016, around 71% of the probes achieve a success rate of more than 97% over IPv4, whereas for IPv6 80% of the probes were able to achieve the success rate. For October 2017, around 88% of the probes achieve a success rate of more than 97% over IPv4, whereas for IPv6 90% of the probes were able to achieve the same success rate. fig. B.14 shows the boxplot over each day of the week. The success rate is 100% over each day of the week when there isn't a stall happening.

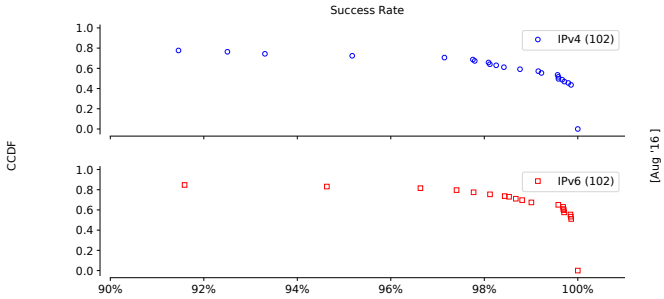

Fig. B.12.: *CCDF of success rate over IPv4 and IPv6 for August 2016. Around 71% of the probes achieve a success rate of more than 97% over IPv4, whereas for IPv6 80% of the probes were able to achieve the success rate.*

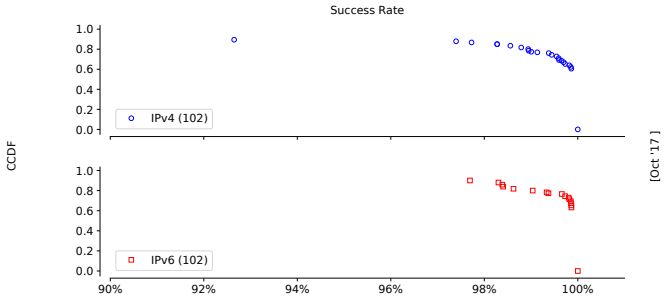

Fig. B.13.: *CCDF of success rate over IPv4 and IPv6 for October 2017. Around 88% of the probes achieve a success rate of more than 97% over IPv4, whereas for IPv6 90% of the probes were able to achieve the same success rate.*

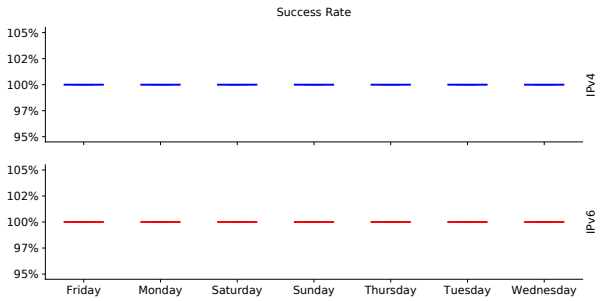

Fig. B.14.: *Boxplot of success rate over IPv4 and IPv6 by each day of the week. Success rate is 100% over each day of the week.*

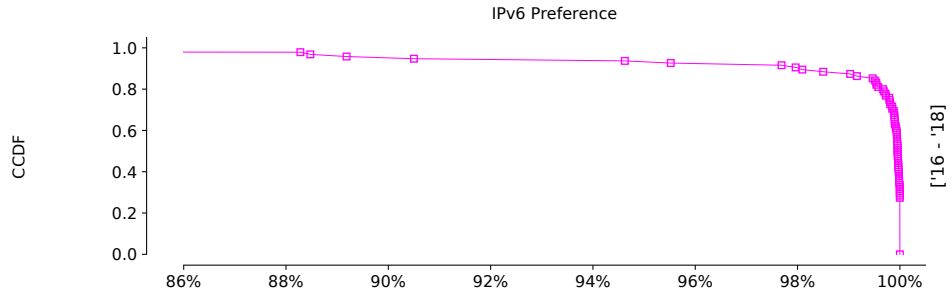

Fig. B.15.: *CCDF of TCP connect time over IPv6 group by probes. Here, TCP connections over IPv6 were preferred for at-least 88% of the time.*

B.0.2. IPv6 Preference

This is in continuation to what we discussed in section 4.2. The effects of HE algorithm [83] can be seen in fig. B.15, and as per that when we cluster the data by probes after calculating the preference, the TCP connect times over IPv6 are preferred for at-least 88% of the time.

B.0.3. Latency and Delays

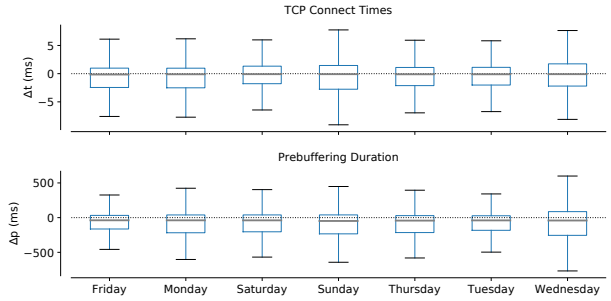

Fig. B.16.

We continue our discussion regarding latency and delays, fig. B.16 shows us the TCP connect times and prebuffering duration to Netflix OCAs for each day of the week. We can see that the latency and delays are consistent during over all the days of the week.

B.0.4. Benifits of ISP Content Caches

British Telecommunications (BT)

TCP Latency and Delay

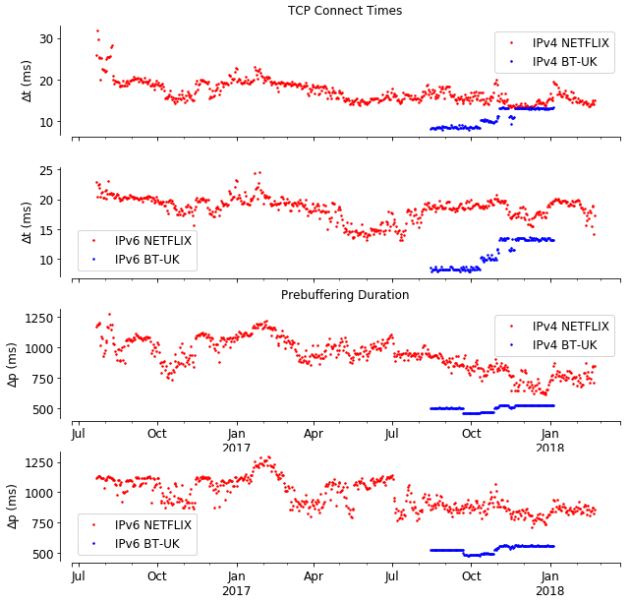

Fig. B.17.: *Timeseries depicting the TCP connect times (connect_time field) and Prebuffering Duration (prebuffering_duration field) for BT against Netflix OCA servers. We are applying a median aggregate here over the absolute values and as can be seen, the TCP connect times are better for BT for both the address families. Prebuffering duration also shows the similar results.*

British Telecommunications (BT) has the AS Number 2856, so for filtering out the rows which are using the criteria was, both AS number of source IP address and AS number of destination IP address belongs to the same AS number which is 2856 here. Here, we are considering the *connect_time* and *prebuffering_duration* fields that are defined in table 3.1. We first wanted to see how TCP connect times and pre-buffering durations perform for both ISP caches and for Netflix (AS number 2906). fig. B.17 shows the timeseries for BT content caches and Netflix OCA servers over both the address families and we have a group by *dtime* field and are considering the median aggregate here so that a single vantage point doesn't bias the results. As we can see, the TCP connect times is better for BT for both the address families, i.e. (0-15 ms), whereas for Netflix its around (0-30 ms). Pre-buffering duration also shows the similar results. We have converted the connect time and pre-buffering duration to 'ms' to get a

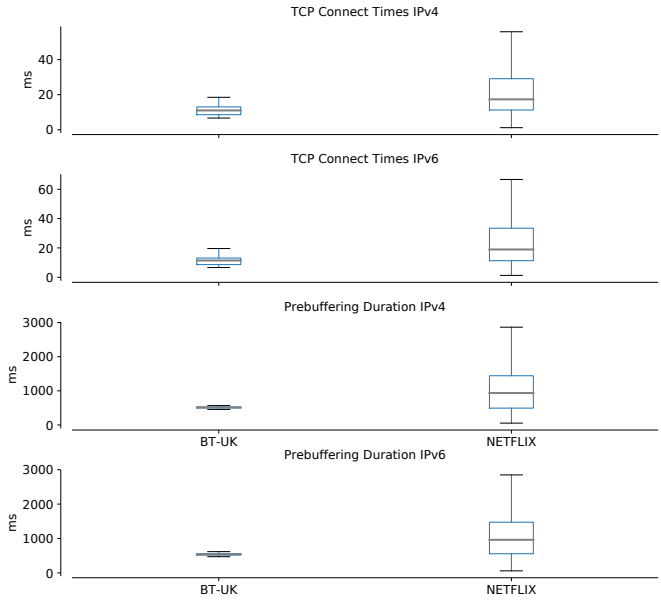

Fig. B.18.: *Boxplot depicting TCP connect times and Pre-Bufferring Duration for BT and Netflix. The median resembles the Time series graph in fig. B.17.*

better understanding. To get a more clear view of the delays, we plot the boxplots for both BT and Netflix over IPv4 and IPv6. The median line in fig. B.18 graph resembles the timeseries in fig. B.17. Furthermore, the third quartile is pretty low for *British Telecommunications* content caches. We further investigate the distribution of latency and delays for BT and Netflix, here also, we filter the data along IPv4 and IPv6 and then take the CDF of the desired attributes. fig. B.19 shows the CDF of TCP connect times and pre-buffering durations for BT and Netflix. Around 80% of the probes require TCP connect times of 13ms for BT over IPv4, whereas it's 38 ms for the same number of probes for Netflix. Similarly, for IPv6, 80% of the probes require 13 ms for BT, whereas it is 40ms for Netflix. For the pre-buffering duration, BT caches require around 527ms for 80% of the probes over IPv4, and for Netflix the number is 1661 ms for the same number of probes. For IPv6, BT requires 559 ms for 80% of the probes and for Netflix the number is 1730 ms. It would be also good to compare the deltas i.e. the difference between the Latencies over IPv4 and IPv6 for BT and Netflix.

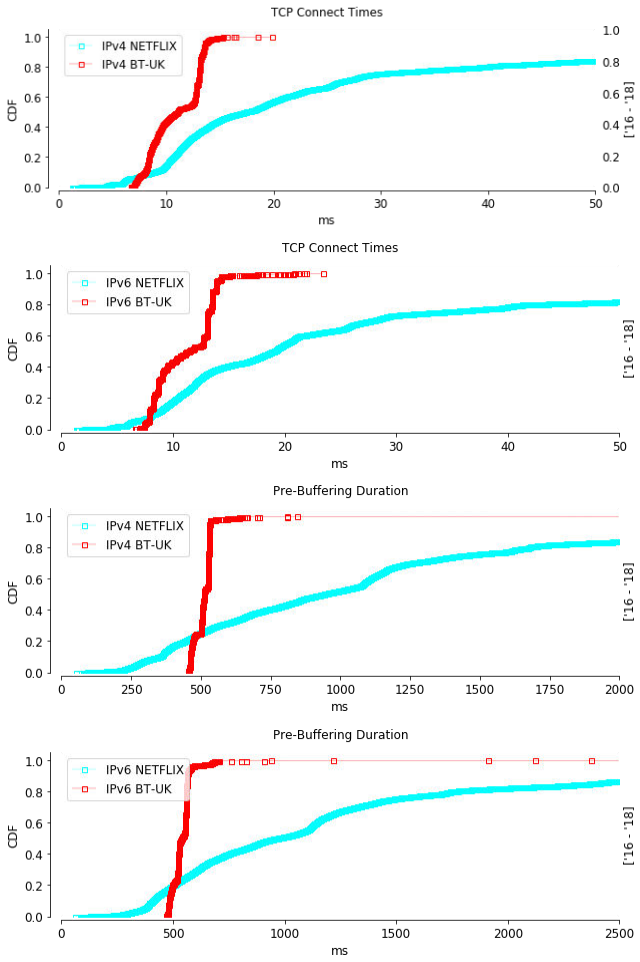

Fig. B.19.: *CDF of TCP Connect times and Pre-Buffering duration for BT and Netflix over IPv4 and IPv6. As we can see, TCP Connect times and Prebuffering duration for BT content caches is better than Netflix OCA servers.*

We will now compare the Deltas that is, the difference between Netflix OCA server TCP connect times and BT caches TCP connect times over IPv4 and IPv6. We will now define the terminology that we are using here, let us say, TCP connect time over IPv4 for BT is tp(y) and TCP connect time over IPv4 for Netflix is tp(x), then the *delta* will be Δtp = tp(x) - tp(y). We did calculate these deltas for TCP connect times and Pre-buffering duration for BT and Netflix over IPv4 and IPv6. Also, as already discussed in chapter 3, pre-buffering duration takes into account DNS resolution times and TCP connect times. We will now analyse the deltas to check the benefits of ISP caches.

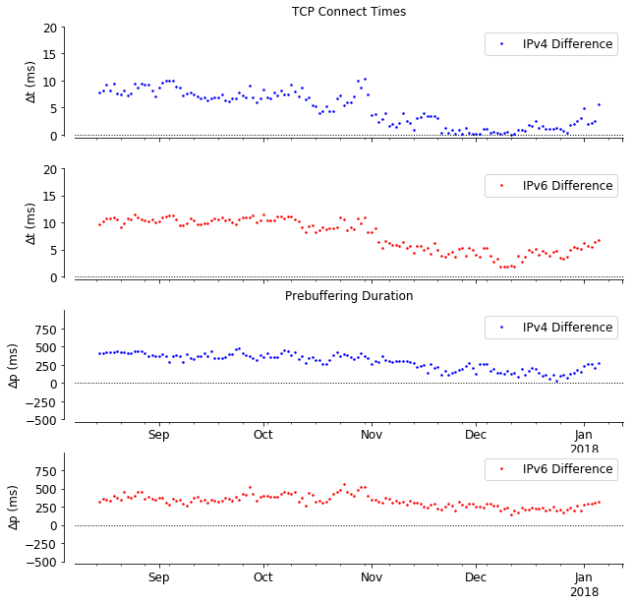

Fig. B.20.: *Time series of difference for TCP connect times and prebuffering durations over IPv4 and IPv6 between Netflix and BT. Here, positive values indicate Netflix OCA server requires more connect time and pre-buffering duration as compared to BT caches. A latency of around 10-15 ms and higher pre-buffering durations (around 0-500 ms or more) are observed for Netflix over both IPv4 and IPv6.*

To plot the deltas, we followed the same analysis [14] did for the Youtube dataset. To plot the timeseries for these deltas, we calculate the median aggregate on the TCP connect times and the prebuffering duration across all probes on each day for the difference between Netflix and BT. fig. B.20 shows the timeseries of median TCP connect times and pre-buffering duration over IPv4 and IPv6 across all probes. Here, positive values indicate Netflix OCA server requires more connect time and pre-buffering duration as compared to BT caches. A latency of around 0-15 ms and higher pre-buffering durations (around 0-500 ms or more) are observed for Netflix over both IPv4

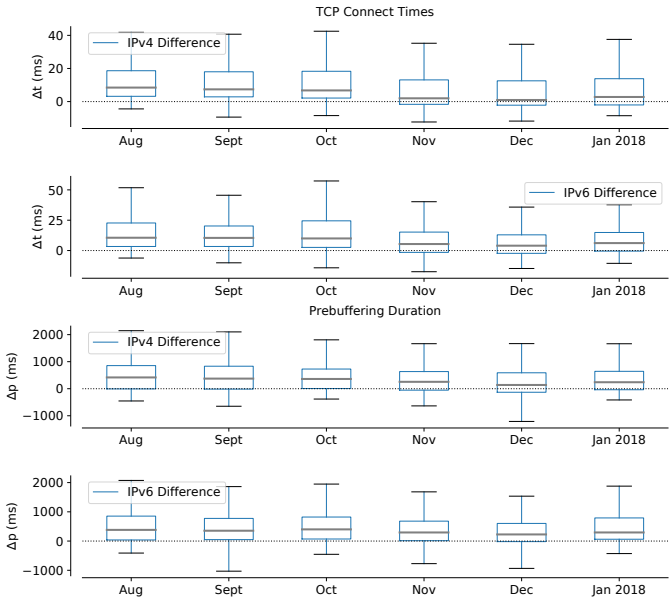

Fig. B.21.: *Boxplot of difference for TCP connect times and prebuffering durations over IPv4 and IPv6 between Netflix and SKY UK Limited. The median line here represents the time series graph in fig. B.20.*

and IPv6. For boxplots in fig. B.21, we have rounded the time to the nearest month. Here, we can see that the median resembles the time series fig. B.20. To get more vivid analysis, fig. B.22 shows us the distribution of the difference in TCP connect times and prebuffering duration over the whole duration. For calculating the CDF, we calculated the deltas and then plotted their CDF. To compare the performance, we are measuring the TCP connect times to the Netflix OCA (Open Connect Appliances) server and to the BT content caches, the distribution here shows that around 78% of the connections are slower for Netflix OCA servers over IPv4, and around 79% of the connections are slower for Netflix over IPv6. For pre-buffering duration which is the time to fetch 2 seconds of video at the specified bitrate from the content server, here Netflix OCA (Open Connect Appliances) server and BT caches, the distribution shows that 71% of the connections are slower for Netflix over IPv4, and around 78% of the connections are slower for Netflix over IPv6.

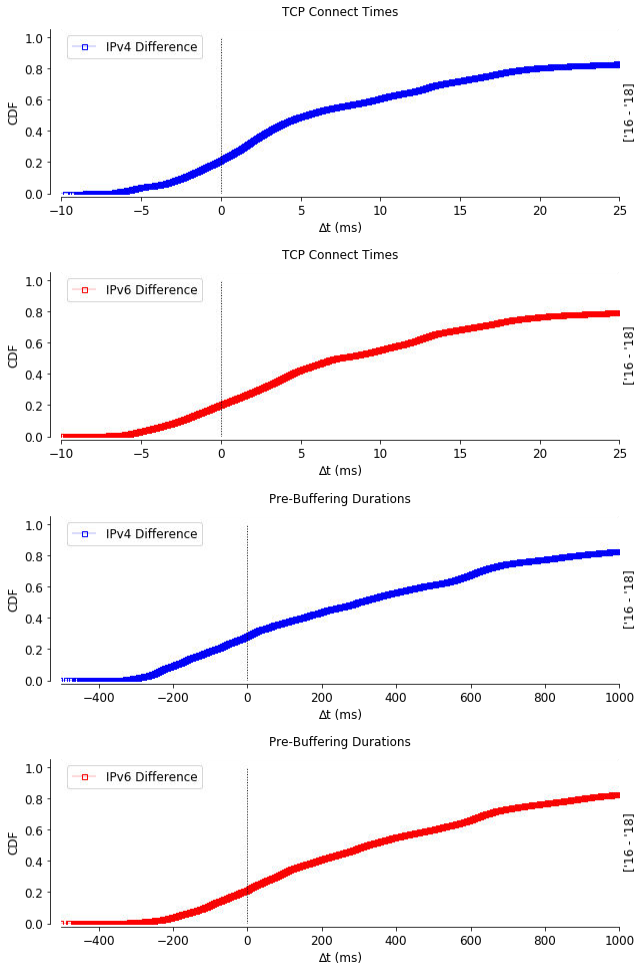

Fig. B.22.: *CDF of the difference of TCP connect times and Prebuffering Durations for Netflix and BT over IPv4 and IPv6. The distribution here shows that around 78% of the connections are slower for Netflix OCA servers over IPv4, and around 79% of the connections are slower for Netflix over IPv6. For pre-buffering duration which is the time to fetch 2 seconds of video at the specified bitrate from the content server, here Netflix OCA (Open Connect Appliances) server and BT caches, the distribution shows that 71% of the connections are slower for Netflix over IPv4, and around 78% of the connections are slower for Netflix over IPv6.*

Throughput

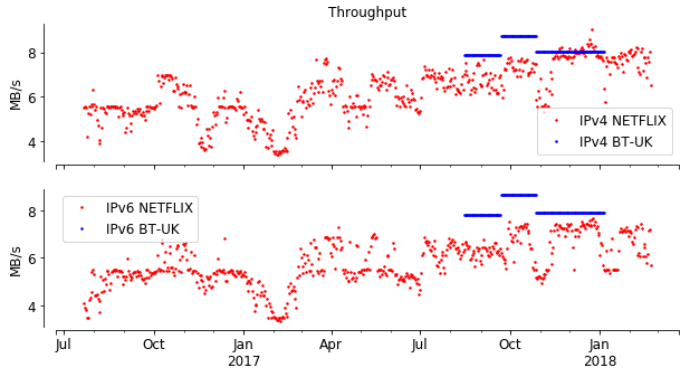

Fig. B.23.: *Time series of Throughput for individual address family's i.e. IPv4 and IPv6 for Netflix and BT. We are considering the bytes_sec field described in table 3.1. As can be seen, the achieved throughput for BT caches is somwehat comparable or more than Netflix OCA server.*

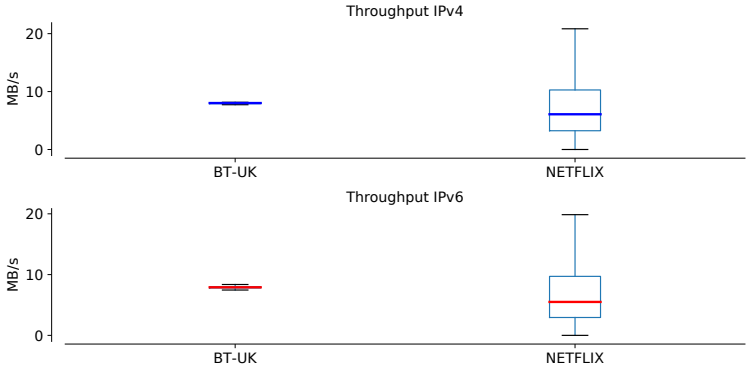

Fig. B.24.: *Boxplot of Throughput for individual address family's i.e. IPv4 and Ipv6 for Netflix and BT. The median line here shows that the achieved throughput is is higher for BT over both address family's.*

After comparing the TCP connect times and Pre-Buffering duration for Netflix and BT, we now know that clients take higher TCP connect times and prebuffering duration for Netflix as compared to the BT caches over both the address families. We will now be comparing the achieved throughput over IPv4 and IPv6 for Netflix OCA server and BT caches. We will first look into the individual performance of IPv4 and IPv6 for Netflix and BT and we will look into their deltas. fig. B.23 gives the time series for both the address families for BT and Netflix, and as can be seen, the achieved throughput for BT caches is more than Netflix OCA server. We are considering the *bytes_sec* field

from the Netflix table 3.1, and we are taking the median aggregate across all probes on each day. Also, the throughput for IPv4 and IPv6 lies between 7-10 MB/s for BT. To get deeper insights we also did the boxplot for IPv4 and IPv6 throughput for BT and Netflix, and as can be seen in fig. B.24 the monthly throughput variation over IPv4 and IPv6. The median line here shows that the achieved throughput is higher for BT over both address families. We have converted the *bytes_sec* field into MB/s to get a more realistic view. fig. B.25 shows the CDF of throughput over IPv4 and IPv6. We followed the same steps here and plotted the CDF of *bytes_sec* field. The CDF here shows that around 80% of the probes achieved a throughput of 8 MB/s for BT over IPv4, whereas for Netflix this is 11 MB/s for a similar number of probes. For IPv6, 80% of the probes achieved the throughput of 8 MB/s, while for Netflix this was 10 MB/s for a similar number of probes. We will now look into the deltas to get a better comparison between BT and Netflix.

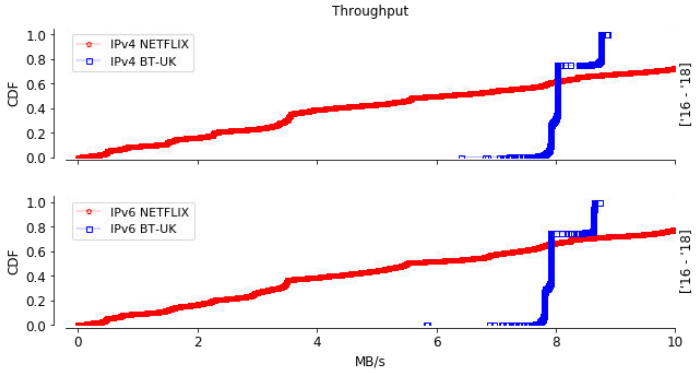

Fig. B.25.: *CDF of Throughput over IPv4 and IPv6 for BT and Netflix. The CDF here shows that around 80% of the probes achieved a throughput of 8 MB/s for BT over IPv4, whereas for Netflix this is 11 MB/s for a similar number of probes. For IPv6, 80% of the probes achieved the throughput of 8 MB/s, while for Netflix this was 10 MB/s for a similar number of probes.*

Before starting with the deltas, we would like to define the terminologies we used to get the results. We are considering the *bytes_sec* field only and using the same terminilogy Bajpai et al. used in [14]. We denote the throughput over IPv4 for BT as *tp(y)* and throughput over IPv4 for Netflix as *tp(x)*, then the *delta* will be Δtp = tp(y) - tp(x). To plot the deltas, we first start with the time series, and fig. B.26 shows us the time series of a median aggregate of throughput across all probes over each day. We can see that the difference is around 0 or more for the whole duration. The positive value indicates a higher throughput for BT, also the difference lies between 0-3 MB/s. fig. B.27 shows the boxplot of the difference of throughput over IPv4 and IPv6 between BT and Netflix. The median line shows similar curves as the time series fig. B.26,

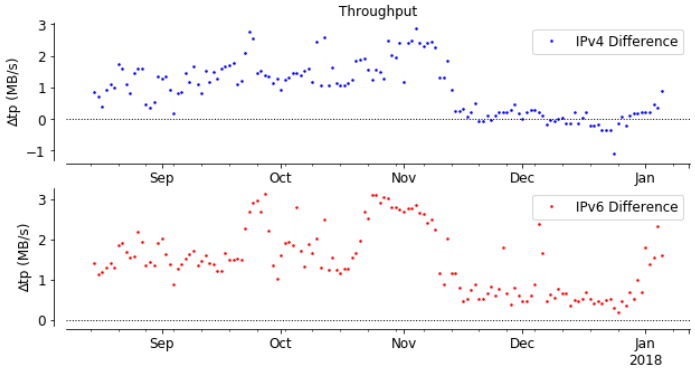

Fig. B.26.: *Time series of deltas of Throughput over IPv4 and IPv6 between BT and Netflix. We can see that the difference is around 0 or more for the whole duration. The positive value indicates a higher throughput for BT, also the difference lies between 0-3 MB/s.*

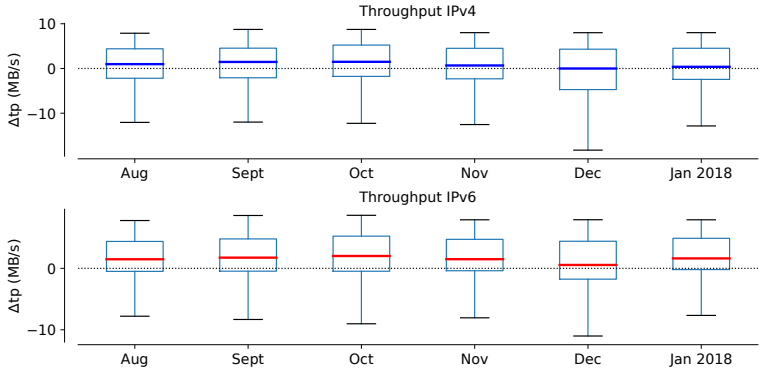

Fig. B.27.: *Boxplot of difference of throughput over IPv4 and IPv6 between BT and Netflix. The median line shows similar curves as the time series fig. B.26, depicting higher throughput for BT over both address families.*

depicting higher throughput for BT over both address families. To get a more clear picture of the performance, we plot the CDF of *bytes_sec* field. fig. B.28 shows the CDF of the difference of throughput over IPv4 and IPv6 for BT and Netflix. Around 59% of the times, BT caches achieved higher throughput than Netflix over IPv4, and for IPv6 around 69% of the times, BT caches achieved higher throughput.

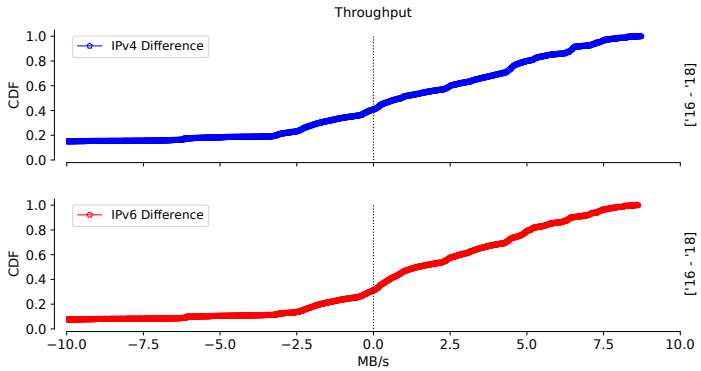

Fig. B.28.: *CDF of difference of throughput over IPv4 and IPv6 for BT and Netflix. Around 66% of the times, BT caches achieved higher throughput than Netflix over IPv4 and IPv6.*

Bibliography

BIBLIOGRAPHY

[1] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, Z. Zhang, M. Varvello, and M. Steiner. "Measurement Study of Netflix, Hulu, and a Tale of Three CDNs." In: *IEEE/ACM Trans. Netw.* 23.6 (2015), pp. 1984–1997. DOI: 10.1109/TNET.2014.2354262.

[2] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z. Zhang. "Unreeling netflix: Understanding and improving multi-CDN movie delivery." In: *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*. 2012, pp. 1620–1628. DOI: 10.1109/INFCOM.2012.6195531.

[3] R. Aggarwal and D. Temkin. *Enabling Support for IPv6*. Nov. 15, 2017. URL: https://medium.com/netflix-techblog/enabling-support-for-ipv6-48a495d5196f.

[4] S. Ahsan, V. Bajpai, J. Ott, and J. Schönwälder. "Measuring YouTube from Dual-Stacked Hosts." In: *Passive and Active Measurement - 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings*. 2015, pp. 249–261. DOI: 10.1007/978-3-319-15509-8_19.

[5] S. Akhshabi and C. Dovrolis. "The evolution of layered protocol stacks leads to an hourglass-shaped architecture." In: *Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Toronto, ON, Canada, August 15-19, 2011*. 2011, pp. 206–217. DOI: 10.1145/2018436.2018460.

[6] R. Almeida, O. L. H. M. Fonseca, E. C. Fazzion, D. O. Guedes, W. M. Jr., and Í. S. Cunha. "A Characterization of Load Balancing on the IPv6 Internet." In: *Passive and Active Measurement - 18th International Conference, PAM 2017, Sydney, NSW, Australia, March 30-31, 2017, Proceedings*. 2017, pp. 242–254. DOI: 10.1007/978-3-319-54328-4_18.

[7] M. M. Amble, P. Parag, S. Shakkottai, and L. Ying. "Content-aware caching and traffic management in content distribution networks." In: *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China*. 2011, pp. 2858–2866. DOI: 10.1109/INFCOM.2011.5935123.

[8] AMSIX. *IPv6 Traffic*. May 30, 2018. URL: https://ams-ix.net/technical/statistics/sflow-stats/ipv6-traffic.

[9] AMSIX. *Statistics*. May 30, 2018. URL: https://ams-ix.net/technical/statistics.

[10] B. Augustin, T. Friedman, and R. Teixeira. "Measuring load-balanced paths in the internet." In: *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC 2007, San Diego, California, USA, October 24-26, 2007*. 2007, pp. 149–160. DOI: 10.1145/1298306.1298329.

[11] B. Augustin, T. Friedman, and R. Teixeira. "Measuring load-balanced paths in the internet." In: *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC 2007, San Diego, California, USA, October 24-26, 2007*. 2007, pp. 149–160. DOI: 10.1145/1298306.1298329.

[12] V. Bajpai. *SamKnows Probes Metadata*. Nov. 15, 2017. URL: http://cnds.eecs.jacobs-university.de/users/vbajpai/yt-ccr-2017/metadata.txt.

[13] V. Bajpai. *Youtube CCR 2017*. Dec. 30, 2017. URL: https://github.com/vbajpai/2017-ccr-youtube-analysis.

[14] V. Bajpai, S. Ahsan, J. Schönwälder, and J. Ott. "Measuring YouTube Content Delivery over IPv6." In: *Computer Communication Review* 47.5 (2017), pp. 2–11. DOI: 10.1145/3155055.3155057.

[15] V. Bajpai and J. Schönwälder. "A Survey on Internet Performance Measurement Platforms and Related Standardization Efforts." In: *IEEE Communications Surveys and Tutorials* 17.3 (2015), pp. 1313–1341. DOI: 10.1109/COMST.2015.2418435.

[16] V. Bajpai and J. Schönwälder. "IPv4 versus IPv6 - who connects faster?" In: *Proceedings of the 14th IFIP Networking Conference, Networking 2015, Toulouse, France, 20-22 May, 2015*. 2015, pp. 1–9. DOI: 10.1109/IFIPNetworking.2015.7145323.

[17] V. Bajpai and J. Schönwälder. "Measuring the Effects of Happy Eyeballs." In: *Proceedings of the 2016 Applied Networking Research Workshop, ANRW 2016, Berlin, Germany, July 16, 2016*. 2016, pp. 38–44.

[18] N. Bargisen. *How Netflix Works*. Apr. 30, 2018. URL: https://www.ripe.net/participate/meetings/regional-meetings/ipv6-day-denmark/presentations/3-nina-ipv6-day-copenhagen.pdf.

[19] S. Bauer, D. Clark, and W. Lehr. *Understanding broadband speed measurements*. Apr. 15, 2018. URL: https://groups.csail.mit.edu/ana/Publications/Understanding_broadband_speed_measurements_bauer_clark_lehr_TPRC_2010.pdf.

[20] A. Berger. *Comparison of Performance over IPv6 versus IPv4*. June 4, 2018. URL: http://ftp.caida.org/workshops/isma/1202/slides/aims1202_acox_supplement.pdf.

[21] A. Berger. *Comparison of Performance over IPv6 versus IPv4*. Apr. 15, 2018. URL: http://ftp.caida.org/workshops/isma/1202/slides/aims1202_acox_supplement.pdf.

[22] R. Beverly and A. W. Berger. "Server Siblings: Identifying Shared IPv4/IPv6 Infrastructure Via Active Fingerprinting." In: *Passive and Active Measurement - 16th International Conference, PAM2015, New York, NY, USA, March 19-20, 2015, Proceedings*. 2015, pp. 149–161. DOI: 10.1007/978-3-319-15509-8_12.

[23] R. Beverly, M. J. Luckie, L. Mosley, and kc claffy. "Measuring and Characterizing IPv6 Router Availability." In: *Passive and Active Measurement - 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings*. 2015, pp. 123–135. DOI: 10.1007/978-3-319-15509-8_10.

[24] T. Böttger, F. Cuadrado, G. Tyson, I. Castro, and S. Uhlig. "Open Connect Everywhere: A Glimpse at the Internet Ecosystem through the Lens of the Netflix CDN." In: *CoRR* abs/1606.05519 (2016). arXiv: 1606.05519.

[25] CAIDA. *Archipelago (Ark) Measurement Infrastructure*. Apr. 30, 2018. URL: https://www.caida.org/projects/ark/.

[26] CAIDA. *AS Classification*. Mar. 15, 2018. URL: https://www.caida.org/data/as-classification/.

[27] K. Cho, M. Luckie, and B. Huffaker. "Identifying IPv6 network problems in the dual-stack world." In: *NetT '04 Proceedings of the ACM SIGCOMM workshop on Network troubleshooting: research, theory and operations practice meet malfunctioning reality* (2004).

[28] L. Colitti, S. H. Gunderson, E. Kline, and T. Refice. "Evaluating IPv6 Adoption in the Internet." In: *Passive and Active Measurement, 11th International Conference, PAM 2010, Zurich, Switzerland, April 7-9, 2010. Proceedings*. 2010, pp. 141–150. DOI: 10.1007/978-3-642-12334-4_15.

[29] J. A. Cordero and O. Bonaventure. "Understanding the topological properties of Internet traffic: A view from the edge." In: *2014 IFIP Networking Conference, Trondheim, Norway, June 2-4, 2014*. 2014, pp. 1–9. DOI: 10.1109/IFIPNetworking.2014.6857090.

[30] J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil, and M. Bailey. "Measuring IPv6 adoption." In: *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014*. 2014, pp. 87–98. DOI: 10.1145/2619239.2626295.

[31] W. I. Day. *World IPv6 Launch Official Website*. Jan. 15, 2018. URL: http://www.worldipv6launch.org/.

[32] A. Dhamdhere and C. Dovrolis. "The Internet is flat: modeling the transition from a transit hierarchy to a peering mesh." In: *Proceedings of the 2010 ACM Conference on Emerging Networking Experiments and Technology, CoNEXT 2010, Philadelphia, PA, USA, November 30 - December 03, 2010*. 2010, p. 21. DOI: 10.1145/1921168.1921196.

[33] A. Dhamdhere, M. J. Luckie, B. Huffaker, kc claffy, A. Elmokashfi, and E. Aben. "Measuring the deployment of IPv6: topology, routing and performance." In: *Proceedings of the 12th ACM SIGCOMM Internet Measurement Conference, IMC '12, Boston, MA, USA, November 14-16, 2012*. 2012, pp. 537–550. DOI: 10.1145/2398776.2398832.

[34] A. Dhamdhere, M. J. Luckie, B. Huffaker, kc claffy, A. Elmokashfi, and E. Aben. "Measuring the deployment of IPv6: topology, routing and performance." In: *Proceedings of the 12th ACM SIGCOMM Internet Measurement Conference, IMC '12, Boston, MA, USA, November 14-16, 2012*. 2012, pp. 537–550. DOI: 10.1145/2398776.2398832.

[35] T. V. Doan. "Analyzing longitudinal datasets from active measurements to identify and locate IPv6 caches in ISP networks." Master's Thesis. Munich, Germany: Technische Universität München, 2017.

[36] T. V. Doan, L. Pajevic, V. Bajpai, and J. Ott. "Quantifying the Latency and Path Lengths towards Content Caches in Dual-stacked ISP networks." In: *Passive And Active Measurement Conference 2018 (In Review)* (2018).

[37] S. Dong, J. Li, L. Zhang, and J. Deng. "A novel algorithm of IPv6 network topology discovery for Campus Network." In: *IEEE Computer Science and Service System (CSSS), 2011 International Conference on* (2011).

[38] B. Donnet. "Internet Topology Discovery." In: *Data Traffic Monitoring and Analysis*. 2013, pp. 44–81. DOI: 10.1007/978-3-642-36784-7_3.

[39] A. B. Downey. *Think Stats*. Dec. 30, 2017. URL: http://greenteapress.com/thinkstats/.

[40] S. J. Eravuchira, V. Bajpai, and J. Schönwälder. *Measurement Research within the Python3 Ecosystem*. Feb. 15, 2018. URL: http://nbviewer.jupyter.org/github/vbajpai/ripe69-python3-toolset/blob/master/ripe69-python3-toolset.ipynb.

[41] S. Fedorov and E. Livengood. *Building Fast.com*. Mar. 15, 2018. URL: https://medium.com/netflix-techblog/building-fast-com-4857fe0f8adb.

[42] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. M. Maggs, J. Rake, S. Uhlig, and R. Weber. "Pushing CDN-ISP collaboration to the limit." In: *Computer Communication Review* 43.3 (2013), pp. 34–44. DOI: 10.1145/2500098.2500103.

[43] B. Fung. *Netflix now accounts for almost 37 percent of our Internet traffic*. May 15, 2018. URL: https://www.washingtonpost.com/news/the-switch/wp/2015/05/28/netflix-now-accounts-for-almost-37-percent-of-our-internet-traffic/?noredirect=on&utm_term=.36fdba12499b.

[44] P. Gill, M. F. Arlitt, Z. Li, and A. Mahanti. "The Flattening Internet Topology: Natural Evolution, Unsightly Barnacles or Contrived Collapse?" In: *Passive and Active Network Measurement, 9th International Conference, PAM 2008, Cleveland, OH, USA, April 29-30, 2008. Proceedings*. 2008, pp. 1–10. DOI: 10.1007/978-3-540-79232-1_1.

[45] V. Giotsas, M. J. Luckie, B. Huffaker, and kc claffy. "IPv6 AS Relationships, Cliques, and Congruence." In: *Passive and Active Measurement - 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings*. 2015, pp. 111–122. DOI: 10.1007/978-3-319-15509-8_9.

[46] V. Giotsas, M. J. Luckie, B. Huffaker, and kc claffy. "IPv6 AS Relationships, Cliques, and Congruence." In: *Passive and Active Measurement - 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings*. 2015, pp. 111–122. DOI: 10.1007/978-3-319-15509-8_9.

[47] V. Giotsas, M. J. Luckie, B. Huffaker, and kc claffy. "IPv6 AS Relationships, Cliques, and Congruence." In: *Passive and Active Measurement - 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings*. 2015, pp. 111–122. DOI: 10.1007/978-3-319-15509-8_9.

[48] Google. *IPv6 Adoption Statistics*. Nov. 15, 2017. URL: https://www.google.com/intl/en/ipv6/statistics.html.

[49] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. "Internet of Things (IoT): A vision, architectural elements, and future directions." In: *Future Generation Comp. Syst.* 29.7 (2013), pp. 1645–1660. DOI: 10.1016/j.future.2013.01.010.

[50] S. Hasan, S. Gorinsky, C. Dovrolis, and R. K. Sitaraman. "Trade-offs in optimizing the cache deployments of CDNs." In: *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*. 2014, pp. 460–468. DOI: 10.1109/INFOCOM.2014.6847969.

[51] B. Holbert, S. Tati, S. Silvestri, T. F. L. Porta, and A. Swami. "Network Topology Inference With Partial Information." In: *IEEE Trans. Network and Service Management* 12.3 (2015), pp. 406–419. DOI: 10.1109/TNSM.2015.2451032.

[52] J. Hyun, J. Li, H. Kim, J. Yoo, and J. W. Hong. "IPv4 and IPv6 performance comparison in IPv6 LTE network." In: *17th Asia-Pacific Network Operations and Management Symposium, APNOMS 2015, Busan, South Korea, August 19-21, 2015*. 2015, pp. 145–150. DOI: 10.1109/APNOMS.2015.7275417.

[53] A. J. Jara, L. Ladid, and A. F. Gómez-Skarmeta. "The Internet of Everything through IPv6: An Analysis of Challenges, Solutions and Opportunities." In: *JoWUA* 4.3 (2013), pp. 97–118.

[54] A. Kapoor. *Chair of Connected Mobility*. May 30, 2018. URL: https://gitlab.lrz.de/cm/2018-akapoor-masters-analysis.

[55] S. Knight, H. X. Nguyen, N. Falkner, R. A. Bowden, and M. Roughan. "The Internet Topology Zoo." In: *IEEE Journal on Selected Areas in Communications* 29.9 (2011), pp. 1765–1775. DOI: 10.1109/JSAC.2011.111002.

[56] M. Lab. *Why are my M-Lab results different from other speed tests?* Apr. 15, 2018. URL: https://www.measurementlab.net/faq/#why-are-my-m-lab-results-different-from-other-speed-tests.

[57] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. "Internet inter-domain traffic." In: *Proceedings of the ACM SIGCOMM 2010 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New Delhi, India, August 30 -September 3, 2010*. 2010, pp. 75–86. DOI: 10.1145/1851182.1851194.

[58] F. Li, J. Yang, X. Wang, T. Pan, C. An, and J. Wu. "Characteristics analysis at prefix granularity: A case study in an IPv6 network." In: *J. Network and Computer Applications* 70 (2016), pp. 156–170. DOI: 10.1016/j.jnca.2016.02.022.

[59] Q. Li, T. Qin, X. Guan, and Q. Zheng. "Empirical analysis and comparison of IPv4-IPv6 traffic: A case study on the campus network." In: *Proceedings of the 18th IEEE International Conference on Networks, ICON 2012, Singapore, December 12-14, 2012*. 2012, pp. 395–399. DOI: 10.1109/ICON.2012.6506590.

[60] S. U. Limited. *SKY UK Official Website*. Feb. 15, 2018. URL: https://www.sky.com/.

[61] I. Livadariu, A. Elmokashfi, and A. Dhamdhere. "Characterizing IPv6 control and data plane stability." In: *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*. 2016, pp. 1–9. DOI: 10.1109/INFOCOM.2016.7524465.

[62] I. Livadariu, A. Elmokashfi, and A. Dhamdhere. "Characterizing IPv6 control and data plane stability." In: *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*. 2016, pp. 1–9. DOI: 10.1109/INFOCOM.2016.7524465.

[63] M. Luckie. *Scamper data collection*. Apr. 15, 2018. URL: https://mailman.caida.org/pipermail/scamper-dev/2004-July/000076.html.

[64] M. J. Luckie. "Scamper: a scalable and extensible packet prober for active measurement of the internet." In: *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia - November 1-3, 2010*. 2010, pp. 239–245. DOI: 10.1145/1879141.1879171.

[65] M. J. Luckie, Y. Hyun, and B. Huffaker. "Traceroute probe method and forward IP path inference." In: *Proceedings of the 8th ACM SIGCOMM Internet Measurement Conference, IMC 2008, Vouliagmeni, Greece, October 20-22, 2008*. 2008, pp. 311–324. DOI: 10.1145/1452520.1452557.

[66] A. Lutu, M. Bagnulo, C. Pelsser, and O. Maennel. "Understanding the Reachability of IPv6 Limited Visibility Prefixes." In: *Passive and Active Measurement - 15th International Conference, PAM 2014, Los Angeles, CA, USA, March 10-11, 2014, Proceedings*. 2014, pp. 163–172. DOI: 10.1007/978-3-319-04918-2_16.

[67] J. Martin, Y. Fu, N. Wourms, and T. Shaw. "Characterizing Netflix bandwidth consumption." In: *10th IEEE Consumer Communications and Networking Conference, CCNC 2013, Las Vegas, NV, USA, January 11-14, 2013*. 2013, pp. 230–235. DOI: 10.1109/CCNC.2013.6488451.

[68] T. U. München. *Chair of Connected Mobility*. May 15, 2018. URL: http://www.cm.in.tum.de/en/home/.

[69] Netflix. *Fast.com*. Mar. 15, 2018. URL: https://fast.com/.

[70] Netflix. *Netflix Official Website*. Dec. 15, 2018. URL: https://www.netflix.com/.

[71] Netflix. *Open Connect Overview*. Mar. 15, 2018. URL: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

[72] M. Nikkhah. "Maintaining the progress of IPv6 adoption." In: *Computer Networks* 102 (2016), pp. 50–69. DOI: 10.1016/j.comnet.2016.02.027.

[73] M. Nikkhah and R. Guérin. "Migrating to Ipv6 - The role of basic coordination." In: *2014 IFIP Networking Conference, Trondheim, Norway, June 2-4, 2014*. 2014, pp. 1–9. DOI: 10.1109/IFIPNetworking.2014.6857119.

[74] M. Nikkhah, R. Guérin, Y. Lee, and R. Woundy. "Assessing IPv6 through web access a measurement study and its findings." In: *Proceedings of the 2011 Conference on Emerging Networking Experiments and Technologies, Co-NEXT '11, Tokyo, Japan, December 6-9, 2011*. 2011, p. 26. DOI: 10.1145/2079296.2079322.

[75] W. H. Organization. *WHO Regions*. Nov. 15, 2017. URL: http://www.who.int/about/regions/en/.

[76] PeeringDB. *AS Classification*. Mar. 15, 2018. URL: https://www.peeringdb.com/.

[77] I. Poese, B. Frank, B. Ager, G. Smaragdakis, and A. Feldmann. "Improving content delivery using provider-aided distance information." In: *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia - November 1-3, 2010*. 2010, pp. 22–34. DOI: 10.1145/1879141.1879145.

[78] E. Pujol, P. Richter, and A. Feldmann. "Understanding the Share of IPv6 Traffic in a Dual-Stack ISP." In: *Passive and Active Measurement - 18th International Conference, PAM 2017, Sydney, NSW, Australia, March 30-31, 2017, Proceedings*. 2017, pp. 3–16. DOI: 10.1007/978-3-319-54328-4_1.

[79] E. Pujol, P. Richter, and A. Feldmann. "Understanding the Share of IPv6 Traffic in a Dual-Stack ISP." In: *Passive and Active Measurement - 18th International Conference, PAM 2017, Sydney, NSW, Australia, March 30-31, 2017, Proceedings*. 2017, pp. 3–16. DOI: 10.1007/978-3-319-54328-4_1.

[80] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 2460 (Draft Standard). RFC. Obsoleted by RFC 8200, updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112. Fremont, CA, USA: RFC Editor, Dec. 1998. DOI: 10.17487/RFC2460.

[81] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard). RFC. Obsoleted by RFCs 7230, 7231, 7232, 7233, 7234, 7235, updated by RFCs 2817, 5785, 6266, 6585. Fremont, CA, USA: RFC Editor, June 1999. DOI: 10.17487/RFC2616.

[82] P. Agarwal and B. Akyol. *Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks*. RFC 3443 (Proposed Standard). RFC. Updated by RFC 5462. Fremont, CA, USA: RFC Editor, Jan. 2003. DOI: 10.17487/RFC3443.

[83] D. Wing and A. Yourtchenko. *Happy Eyeballs: Success with Dual-Stack Hosts*. RFC 6555 (Proposed Standard). RFC. Obsoleted by RFC 8305. Fremont, CA, USA: RFC Editor, Apr. 2012. DOI: 10.17487/RFC6555.

[84] D. Thaler (Ed.), R. Draves, A. Matsumoto, and T. Chown. *Default Address Selection for Internet Protocol Version 6 (IPv6)*. RFC 6724 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor, Sept. 2012. DOI: 10.17487/RFC6724.

[85] J. Postel. *Internet Protocol*. RFC 791 (Internet Standard). RFC. Updated by RFCs 1349, 2474, 6864. Fremont, CA, USA: RFC Editor, Sept. 1981. DOI: 10.17487/RFC0791.

[86] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 8200 (Internet Standard). RFC. Fremont, CA, USA: RFC Editor, July 2017. DOI: 10.17487/RFC8200.

[87] RFC2616. *HTTP Status Codes*. Dec. 15, 2017. URL: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

[88] RIPE. *RIPE NCC - Stats*. Dec. 20, 2017. URL: https://stat.ripe.net/.

[89] RIPE. *RIPE NCC - Stats*. Jan. 20, 2018. URL: https://atlas.ripe.net/.

[90] SamKnows. *SamKnows*. May 15, 2018. URL: https://www.samknows.com/.

[91] SamKnows. *SAMKNOWS TEST METHODOLOGY Methodology and technical information relating to the SamKnows testing platform.* Dec. 15, 2018. URL: https://files.samknows.com/~whitepapers/SQ301-005-EN-Test-Suite-Whitepaper-4.pdf.

[92] Sandvine. *2016 Global Internet Phenomena*. Apr. 30, 2018. URL: https://www.sandvine.com/hubfs/downloads/archive/2016-global-internet-phenomena-report-latin-america-and-north-america.pdf.

[93] N. Sarrar, G. Maier, B. Ager, R. Sommer, and S. Uhlig. "Investigating IPv6 Traffic - What Happened at the World IPv6 Day?" In: *Passive and Active Measurement - 13th International Conference, PAM 2012, Vienna, Austria, March 12-14th, 2012. Proceedings*. 2012, pp. 11–20. DOI: 10.1007/978-3-642-28537-0_2.

[94] H. E. I. Services. *IPv6 Tunnel Broker*. Jan. 30, 2018. URL: https://www.tunnelbroker.net/.

[95] B. Telecommunications. *British TelecommunicationsOfficial Website*. Feb. 15, 2018. URL: http://home.bt.com/.

[96] M. L. N. D. Tool. *Network Diagnostic Tool*. Apr. 15, 2018. URL: https://www.measurementlab.net/tests/ndt/.

[97] N. Trenaman, A. Kiessling, and R. Wilhelm. *"Lost Stars" Why Operators Switch Off IPv6*. Jan. 30, 2018. URL: https://www.nanog.org/sites/default/files/Aben_Lost_Stars_-_v1.pdf.

[98] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie. "Optimal cache allocation for Content-Centric Networking." In: *2013 21st IEEE International Conference on Network Protocols, ICNP 2013, Göttingen, Germany, October 7-10, 2013*. 2013, pp. 1–10. DOI: 10.1109/ICNP.2013.6733577.

[99] P. Wu, Y. Cui, J. Wu, J. Liu, and C. Metz. "Transition from IPv4 to IPv6: A State-of-the-Art Survey." In: *IEEE Communications Surveys and Tutorials* 15.3 (2013), pp. 1407–1424. DOI: 10.1109/SURV.2012.110112.00200.

[100] Y. Wu and X. Zhou. "Research on the IPv6 performance analysis based on dual-protocol stack and tunnel transition." In: *2011 6th International Conference on Computer Science Education (ICCSE)* (2011).

[101] Y. Wu and X. Zhou. "Research on the IPv6 performance analysis based on dual-protocol stack and tunnel transition." In: *6th International Conference on Computer Science and Education ICCSE*. 2011, pp. 1091–1093.

[102] X. Zhou and P. V. Mieghem. "Hopcount and E2E Delay: IPv6 Versus IPv4." In: *Passive and Active Network Measurement, 6th International Workshop, PAM 2005, Boston, MA, USA, March 31 - April 1, 2005, Proceedings*. 2005, pp. 345–348. DOI: 10.1007/978-3-540-31966-5_31.