Technische Universität München
TUM School of Computation, Information and Technology

Examining Throughput of TCP Connections on the Internet: Insights from Active and Passive Measurements

Simon Bauer

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Stefan Wagner

Prüfende der Dissertation:

1. Prof. Dr.-Ing. Georg Carle
2. Prof. Dr. Jürgen Schönwälder

Die Dissertation wurde am 09.09.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 01.05.2025 angenommen.

Abstract

The performance of network connections impacts user experience and, accordingly, user satisfaction. Therefore, service providers are interested in detecting, analyzing, and troubleshooting connections with poor performance. Further, researchers are interested in assessing the impacts of the evolution of user behavior, provided services, and technologies on connection performance, particularly regarding the Internet. Thus, analyzing connection performance, particularly throughput, is an ongoing research field, especially regarding the Transmission Control Protocol (TCP) responsible for the transportation of the vast majority of Internet traffic nowadays. Researchers introduced different perspectives on the analysis of the throughput of TCP connections, like the analysis of the relation of flow characteristics to a flow's transmission rate, the analysis of throughput limitations, or the analysis of performance-related connection characteristics, like TCP option usage. However, recent insights from the mentioned perspectives on the throughput of TCP connections on the Internet are rare. Reviewing the significant evolution of user behavior and technologies involved on the Internet and the rising importance of content delivery networks and cloud services during the past decade motivates surveying described traffic characteristics on today's Internet. Therefore, this thesis extensively examines the throughput of TCP connections on today's Internet by conducting passive and active measurements.

First, this thesis presents passive measurement results assessing the evolution of TCP flow characteristics and the significance of heavy hitter connections on the Internet. Further, this thesis surveys the evolution of throughput limitations of TCP connections and the usage of performance-related TCP options on the Internet during the past few years, considering data sets consisting of several billion TCP connections. One approach to analyze throughput is TCP root cause analysis, which is purposed to determine which entity prevents a TCP connection from further increasing its transmission rate. While related work presents different approaches to determine throughput limitations based on passively observed TCP traffic, approaches to the online analysis of TCP throughput limitations on commodity server hardware are rare. A shortcoming addressed by this thesis, resulting in an approach to the online analysis of TCP throughput limitations on commodity hardware considering monitored bandwidths of several Gbit/s.

Next to measurement results derived from passive measurements, active performance measurements are expected to provide further insights into TCP throughput on the Internet. In order to enable active measurements of the performance of TCP connections through the Internet, this thesis contributes an active measurement approach for measurements with uncontrolled measurement targets, like public Internet servers. The approach covers the identification of suitable measurement targets to initiate TCP-based data transmission, the execution of download-based measurement runs, and the capturing of corresponding traffic for analysis. Based on the introduced approach to conducting active Internet measurements, this thesis examines how different performance-related TCP options impact the throughput of Internet connections. Further, this thesis surveys the accuracy of passive capacity estimation based on connections through today's Internet and applies the introduced measurement approach to survey end-to-end capacities in the Internet, which are an upper bound of a connection's transmission rate.

Zusammenfassung

Die Leistung von Netzwerkverbindungen wirkt sich auf das Nutzererlebnis und damit auf die Nutzerzufriedenheit aus. Daher sind Dienstanbieter daran interessiert, Verbindungen mit schlechter Leistung zu erkennen, zu analysieren und die Ursachen zu beheben. Darüber hinaus sind Wissenschaftler daran interessiert, die Auswirkungen der Entwicklung des Nutzerverhaltens, der angebotenen Dienste und Technologien auf die Verbindungsleistung, insbesondere im Internet, zu bewerten. Daher ist die Analyse der Verbindungsleistung, insbesondere des Durchsatzes, ein fortlaufendes Forschungsgebiet, insbesondere im Hinblick auf das Transmission Control Protocol (TCP), das heutzutage für den Transport des größten Teils des Internetverkehrs verwendet wird. Wissenschaftler haben verschiedene Blickwinkel zur Analyse des Durchsatzes von TCP-Verbindungen vorgestellt, z. B. die Analyse der Relation von Verbindungseigenschaften zur Übertragungsrate, die Analyse von Durchsatzbeschränkungen oder die Analyse von Verbindungsmerkmalen wie der Nutzung von leistungsrelevanten TCP Optionen. Aktuelle Einblicke aus den genannten Perspektiven auf den Durchsatz von TCP-Verbindungen im Internet sind jedoch rar. Ein Blick auf die signifikante Entwicklung des Nutzerverhaltens und der Technologien im Internet sowie die wachsende Bedeutung von Content Delivery Networks und von Cloud-Diensten während des letzten Jahrzehnts motiviert dazu, die beschriebenen Eigenschaften des Datenverkehrs im heutigen Internet zu untersuchen. Daher wird in dieser Arbeit der Durchsatz von TCP-Verbindungen im heutigen Internet mittels passiver und aktiver Messungen umfassend analysiert.

Zunächst werden in dieser Arbeit Ergebnisse passiver Messungen vorgestellt, die die Entwicklung der Charakteristika von TCP Verbindungen und die Bedeutung von Heavy-Hitter-Verbindungen im Internet untersuchen. Des Weiteren wird in dieser Arbeit die Entwicklung der Durchsatzbeschränkungen von TCP-Verbindungen und die Nutzung leistungsrelevanter TCP-Optionen im Internet während der letzten Jahre untersucht, wobei Datensätze mit mehreren Milliarden TCP-Verbindungen analysiert werden. Ein Ansatz zur Analyse von des Durchsatzes von TCP-Verbindungen ist die TCP Root Cause Analysis, die darauf abzielt festzustellen, welche Entität eine TCP-Verbindung daran hindert, ihre Übertragungsrate weiter zu erhöhen. Während in verwandten Arbeiten verschiedene Ansätze zur Bestimmung von Durchsatzbeschränkungen auf der Grundlage von passiv beobachtetem TCP-Verkehr vorgestellt werden, gibt es nur wenige Ansätze zur Echtzeitanalyse von TCP-Durchsatzbeschränkungen auf handelsüblicher Serverhardware. Dieser Lücke im Stand der Technik begegnet diese Arbeit, indem sie einen Ansatz für die Echtzeitanalyse von TCP-Durchsatzbeschränkungen auf Standard-Serverhardware entwickelt, der überwachte Bandbreiten von mehreren Gbit/s berücksichtigt.

Neben den aus passiven Messungen abgeleiteten Messergebnissen sollen aktive Messungen weitere Erkenntnisse über den Durchsatz von TCP Verbindungen im Internet liefern. Um aktive Messungen des Durchsatzes von TCP-Verbindungen durch das Internet zu ermöglichen, wird in dieser Arbeit ein aktiver Messansatz für Messungen mit unkontrollierten Messzielen, wie z.B. öffentlichen Internetservern, vorgestellt. Der Ansatz umfasst die Identifikation geeigneter Messziele zur Initiierung einer TCP-basierten Datenübertragung, die Durchführung von downloadbasierten Messläufen und die Aufzeichnung des entsprechenden Datenverkehrs zur Analyse. Basierend auf dem vorgestellten Ansatz zur Durchführung aktiver Internetmessungen wird in dieser Arbeit

untersucht, wie sich verschiedene leistungsrelevante TCP-Optionen auf den Durchsatz von Internetverbindungen auswirken. Des Weiteren wird in dieser Arbeit die Genauigkeit der passiven Kapazitätsabschätzung basierend auf Verbindungen durch das heutige Internet untersucht und der vorgestellte aktive Messansatz angewandt, um Ende-zu-Ende-Kapazitäten im Internet zu untersuchen, die eine obere Schranke für die Übertragungsrate einer Verbindung darstellen.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

Computer networks are an omnipresent backbone of today's society. Institutions and enterprises build networks within and between sites, and private customers attach to infrastructure maintained by service providers to participate in the globally distributed Internet. While services entail different purposes and requirements, network and service providers have common goals, such as providing reliable and efficient network infrastructure and enabling performant data transmission. Such goals are motivated by *(i)* satisfying rising user expectations and *(ii)* efficiently exploiting available resources. Considering the complexity of today's networks with increasing numbers of participants, the evolution of involved technology, and evolving user behavior, the analysis of connection performance is an ongoing topic of interest, especially regarding the Internet as a composition of networks and services maintained by thousands of providers.

The Transmission Control Protocol (TCP) was introduced in 1981 [1], enabling the detection and retransmission of lost or corrupted data across networks. Further, TCP features flow and congestion control mechanisms purposed to sense the load on the network and involved entities, targeting to avoid degradation of concurrent connections by determining a fair share of available resources, respectively, overload on the receiver. TCP is purposed to ensure maximal utilization of available network resources and, accordingly, to optimize a connection's transmission rate, also referred to as flow rate or throughput. These protocol mechanisms imply varying transmission rates of a connection depending on various network parameters and load conditions. Nowadays, TCP is used by various essential applications and services of network communication, such as web services, file transfer, or media streaming. Thereby, throughput is an essential performance indicator as it, for instance, determines the time required to download files or potentially limits the quality of streamed media sources.

Analyzing the throughput of TCP connections on the Internet is of interest from different perspectives. First, the performance of network connections, and accordingly a connection's transmission rate, is assumed to impact user experience and user satisfaction [2], [3]. Accordingly, service providers are interested in detecting and troubleshooting connections with poor perfor-

mance and understanding transmission rates of their services and corresponding applications to optimize their network and service infrastructure. This interest is motivated by ensuring high utilization of available resources to operate cost-effectively and providing good performance [4]. Further, infrastructure providers are interested in identifying the location of performance limitations to assess whether performance issues are caused within their infrastructure or whether limited performance can be traced back to the endpoints of a connection or another provider's network infrastructure. However, interest in assessing and understanding TCP performance on the Internet is also of interest from a research perspective.

Accordingly, researchers introduced different approaches to analyzing TCP throughput and throughput-related traffic characteristics. Starting from a macroscopic view, as introduced by Zhang et al. [2], researchers survey characteristics of TCP connections, like duration, size, or rate, their correlation, and the significance of so-called heavy hitter connections [2], [5]. Other researchers survey the deployment of different TCP options purposed to improve TCP throughput on the Internet [6]–[9]. Further, researchers introduce different approaches to determine the throughput limitations of an observed TCP connection, also referred to as TCP root cause analysis (RCA) [2], [4], [10]–[13]. In particular, TCP RCA is purposed to determine which entity hinders a TCP connection from further increasing its transmission rate. Potential throughput limitations cover the network infrastructure, defective network entities, misconfigured servers or clients, or even performance bottlenecks due to inefficient applications or limited server and client resources. In addition, researchers survey bandwidth related metrics like end-to-end capacity, in sense of maximum physical bandwidth, of network paths implying potential limitations on a flow's maximum throughput [14]–[21].

As elaborated above, different approaches exist to survey TCP throughput-related connection characteristics. However, insights into such characteristics on today's Internet are rare. For instance, the significance of heavy hitters on Internet traffic and the correlation of TCP flow characteristics were most recently surveyed in 2002 and 2003 [2], [5]. Most recent studies on throughput limitations of Internet traffic were conducted on traffic captured in 2011 [12], [22], [23] and 2013 [4]. This motivates to survey TCP connection characteristics and throughput limitations on today's Internet, considering ongoing change regarding applications and corresponding web traffic, e.g., media streaming and social networks, new user devices and operating systems, or evolving network access technologies. At the same time, requirements for the use of traffic monitoring and analysis tools in productive environments have evolved significantly during the past decades regarding increasing traffic volumes and deployed physical bandwidths of several Gbit/s implying increased load for traffic monitoring and analysis tools.

1.1 Research Objectives

This thesis is purposed to conduct passive and active network measurements to survey TCP throughput and throughput-related traffic characteristics of Internet connections. Thereby, this thesis considers the analysis of

- TCP flow characteristics and heavy hitter significance,
- throughput limitations of TCP connections,
- TCP option usage and corresponding impacts on performance, and,
- the deployment of end-to-end path capacities

on today's Internet.

Accordingly, the core of this thesis is developing and applying measurement methods for the analysis of named traffic characteristics. Thereby, this thesis also aims to meet today's requirements for traffic monitoring and analysis tools regarding network bandwidths of several Gbit/s and corresponding traffic volumes.

This thesis pursues the research questions (RQ) addressed in more detail in the following, corresponding to the following research objectives (RO):

RO 1 Large-scale passive measurements on throughput characteristics of TCP connections on the Internet

RO 2 Conducting active measurements of throughput and throughput-related metrics on the Internet

RO 1: Large-scale passive measurements on throughput characteristics of TCP connections on the Internet

As elaborated above, understanding the TCP throughput of connections and corresponding throughput limitations is of interest from a research and provider perspective. **RO 1** is purposed to assess correlations of different flow characteristics, the significance of heavy hitters, and the occurrence and relevance of different throughput limitations on today's Internet. Therefore, this thesis aims to analyze publicly available large-scale data sets consisting of Internet traffic captured during the past years consisting of billions of TCP connections. Based on passive measurements with these data sets, this thesis surveys how TCP flow characteristics evolved during the past years and are correlating on today's Internet and how the performance of TCP connections and corresponding throughput limitations evolved during the past years. Further, **RO 1** covers surveying the evolution of the usage of throughput-related TCP options, like TCP window scaling, selective acknowledgments, and explicit congestion notifications. Next to the described analysis of large-scale data sets of captured Internet traffic, this thesis pursues the question of how to monitor throughput limitations in productive environments considering monitored links with several Gbit/s of bandwidths.

Accordingly, **RO1** covers pursuing the following research questions:

RQ 1.1 How did TCP flow characteristics and their correlation evolve on the Internet?

RQ 1.2 What limits the throughput of TCP connections on today's Internet?

RQ 1.3 How frequently used are throughput-related TCP options on the Internet?

RQ 1.4 How to monitor TCP throughput limitations in real-time?

RQ 1.5 How to classify TCP throughput changes with TCP throughput limitations?

RO 2: Conducting active measurements of throughput and throughput-related metrics on the Internet

Next to passive measurements on throughput and related connection characteristics, this thesis considers conducting active Internet measurements to survey the throughput of TCP connections on the Internet. In comparison to passively conducted measurements, for instance, based on data sets of captured network traffic, active measurements imply that different factors influencing a connection can be controlled, partly also those influencing the throughput of a connection. For instance, active measurements enable the control of endpoint configurations and the type of observed traffic, provide information regarding measurement targets and the corresponding service provider, and enable repeating experiments to survey impacts on throughput in a more targeted and isolated manner. Thereby, the challenge arises to actively initiate connections to selected measurement targets, for instance, from a client, respectively, user perspective.

In the scope of **RO 2**, this thesis is purposed to introduce an approach to conducting active Internet measurements to survey the throughput of TCP connections. Afterward, such an approach is considered to analyze the impacts of TCP option usage on TCP throughput. Next, active Internet measurements can also be used to survey characteristics of Internet paths, like end-to-end path capacity, which represents an upper bound of achievable connection throughput.

RO2 is purposed to pursue the following research questions by means of active Internet measurements:

RQ 2.1 How to conduct active Internet measurements to survey TCP throughput?

RQ 2.2 How does TCP option usage impact connection performance?

RQ 2.3 How accurate is the estimation of end-to-end path capacity in the Internet?

RQ 2.4 What do active Internet measurements on end-to-end capacity reveal?

1.2 Key Contributions of this Thesis

This section highlights key contributions to the research questions examined in this thesis. An overview of chapters and their key contributions is presented in Table 1.1.

In the scope of RQ 1.1, RQ 1.2, and RQ 1.3, this thesis contributes an analysis of flow characteristics, TCP option usage, and throughput limitations of TCP connections based on large-scale, passive data sets consisting of billions of Internet connections. Thereby, this thesis surveys the evolution of Internet flow characteristics, like flow size, duration, and transmission rate, the evolution of TCP option deployment, and the evolution of TCP throughput limitations during the past years. By applying different methodologies to the named large-scale data sets, this thesis shows the ongoing importance of heavy hitter connections on the Internet, implying relatively small shares of connections responsible for the transmission of the majority of observed traffic volume, and examines the relevance of different throughput limitations based on bytes transmitted, respectively, connection time spent, during limitations by different root causes. Regarding RQ 1.4, this thesis introduces a measurement method for the online monitoring of TCP root cause analysis, providing timely insights into throughput limitations and the detection of changes in the throughput limitation of a connection. A corresponding proof of concept implementation is evaluated regarding performance and shows the feasibility of online throughput limitation monitoring on links with several Gbit/s of bandwidths.

In the scope of RQ 2.1, this thesis contributes an approach for active Internet measurements surveying the throughput of TCP connections. The corresponding measurement pipeline consists of three components to first crawl files hosted by public Internet servers, actively generate traffic based on crawled files afterward, and capture corresponding Internet traffic for further analysis. In the scope of RQ 2.2, this thesis evaluates the impacts of different client configurations regarding TCP option usage on throughput. Such evaluation is based on active measurements from different vantage points and surveys the characteristics of observed throughput and the speed-ups by different client configurations.

Further, in the scope of RQ 2.3, this thesis assesses the accuracy of capacity estimates derived from captured Internet traffic based on measurement conducted in hybrid Internet setups, which provide ground truth data regarding end-to-end capacity due to controlled first and last hops of an Internet path. Regarding RQ 2.4, this thesis presents the results of active measurements on the end-to-end capacity of Internet paths targeting several thousand public web servers. Presented measurements confirm the widespread deployment of Gbit/s links in the Internet.

1.3 Structure

In Chapter 2, this thesis elaborates on related work regarding the analysis of TCP flow characteristics, approaches to TCP root cause analysis and corresponding measurement results, end-to-end path capacity estimation, and measurements on TCP option deployment on the Internet.

Objective	Contribution	Chapter
Part I: Introduction		
Part II: Passive Measurements		
RO 1	Long-term analysis of TCP flow characteristics	Chapter 3
	Large-scale TCP root cause analysis	Chapter 3
	Scalable throughput limitation monitoring	Chapter 4
	Method to classify TCP throughput changes	Chapter 5
Part III: Active Measurements		
RO 2	Analysis of large-scale crawling of web servers	Chapter 6
	Active measurements of TCP option impact on throughput	Chapter 6
	Active measurements of capacity deployment in the Internet	Chapter 7
Part IV: Conclusion		

Table 1.1: Key contributions and structure of this thesis.

Part II is purposed to present passive measurement approaches and measurement results surveying throughput characteristics of TCP connections on the Internet. In particular, Chapter 3 focuses on analyzing introduced connection characteristics based on large-scale passive Internet traffic data sets. Chapter 4 introduces an approach to the online monitoring of TCP throughput limitations. Chapter 5 discusses an approach to classify changes in the throughput of a TCP connection based on the correlation of throughput changes with the results of TCP root cause analysis.

Part III focuses on measurement results derived from active Internet measurements. In particular, Part III presents results of active measurements surveying the impact of TCP options on the throughput of a connection in Chapter 6 and measurements on end-to-end capacity deployment in the Internet in Chapter 7.

In Part IV, this thesis summarizes and discusses major findings regarding examined research questions in Chapter 8 and proposes research topics for future work in Chapter 9.

1.4 Publications in the Context of this Thesis

This thesis builds on results presented in several peer-reviewed publications listed in the following:

Online Monitoring of TCP Throughput Limitations
***Simon Bauer**, Kilian Holzinger, Benedikt Jaeger, Paul Emmerich, Georg Carle*
IEEE/IFIP Network Operations and Management Symposium 2020 (NOMS 2020)

Scalable TCP Throughput Limitation Monitoring
***Simon Bauer**, Florian Wiedner, Benedikt Jaeger, Paul Emmerich, Georg Carle*
IEEE/IFIP Symposium on Integrated Network and Service Management 2021 (IM 2021)

On the Evolution of Internet Flow Characteristics
***Simon Bauer**, Benedikt Jaeger, Fabian Helfert, Philippe Barias, Georg Carle*
ACM/IRTF Applied Networking Research Workshop 2021 (ANRW 2021)

Towards the Classification of TCP Throughput Changes
***Simon Bauer**, Benedikt Jaeger, Max Reimann, Jonas Fromm, Georg Carle*
IEEE/IFIP Network Operations and Management Symposium 2022 (NOMS 2022)

On the Accuracy of Active Capacity Estimation in the Internet
***Simon Bauer**, Janluka Janelidze, Benedikt Jaeger, Patrick Sattler, Patryk Brzoza, Georg Carle*
IEEE/IFIP Network Operations and Management Symposium 2023 (NOMS 2023)

Evaluating the Benefits: Quantifying the Effects of TCP Options, QUIC, and CDNs on Throughput
***Simon Bauer**, Patrick Sattler, Johannes Zirngibl, Christoph Schwarzenberg, Georg Carle*
ACM/IRTF Applied Networking Research Workshop 2023 (ANRW 2023)

Chapter 2

Related Work

This chapter elaborates on related work regarding research topics addressed in this thesis. In particular, this chapter introduces related work regarding

- measurements on heavy hitters and TCP flow characteristics on the Internet,
- approaches to TCP root cause analysis,
- measurement studies on throughput limitations on the Internet,
- approaches to capacity estimation and their accuracy,
- and studies on TCP option deployment on the Internet

in the following sections.

The remainder of this chapter is structured as follows: First, this section summarizes central contributions by the most important related work regarding the contents of this thesis and describes how this thesis extends the corresponding state of the art. Section 2.1 presents measurement studies on flow characteristics and heavy hitter significance. Section 2.2 introduces existing approaches to TCP root cause analysis in more detail. Section 2.3 focuses on measurement studies on throughput limitations of TCP connections on the Internet. Section 2.4 provides an overview of existing work in the field of network path capacity estimation. Section 2.5 summarizes measurement results on the deployment of TCP options on the Internet.

Studies on TCP flow characteristics, like flow size, flow duration, and flow rate, their correlation, and the significance of heavy hitters on the Internet were conducted by Zhang et al. [2] and Lan et al. [5] based on data captured between 2001 and 2002, respectively 2002 and 2003.

Further, different publications [2], [4], [10]–[13], [23] introduce approaches to TCP root cause analysis. Such approaches are applied to data sets passively collected at ADSL access links, campus networks, or data centers [2], [4], [10], [12], [13], [23], [24]. Further, Zhang et al. [22] apply the RCA approach introduced by Siekkinnen et al. [11] to survey throughput limitations in cellular networks. However, most recent studies on throughput limitations in passive data

sets were conducted based on data captured in 2013 [4] and 2011 [12], [22], implying a lack of insights on throughput limitations on today's Internet.

The online analysis of throughput limitations is considered by Ghasemi et al. [4], who survey the integration of the analysis of TCP throughput and corresponding limitations to the data plane. Further, Sun et al. [23] present an approach to survey throughput limitations based on periodic polling of sender network stack statistics, which we also consider to enable analysis of throughput limitations during a connection's lifetime.

Several publications surveyed the estimation of network path capacity throughout the years, especially in the early 2000s. Corresponding publications introduce new approaches to capacity estimation [14]–[21] or compare estimates of different tools [20], [25]. However, there are no recent studies on the accuracy of capacity estimation and on capacities deployed in today's Internet.

Regarding TCP option deployment, different studies surveying the deployment of TCP performance-related options like TCP window scaling (WS), selective acknowledgments (SACK), or explicit congestion notifications (ECN) exist. Thereby, researchers conduct active scans [6], [7] or passive traffic measurements [7]–[9] to survey option deployment and usage, while there are no insights to impacts on connection throughput by such options on the Internet.

This thesis:

Considering the state of the art described above and in more detail in the upcoming sections, this thesis is purposed to contribute to the analysis of TCP transmission rates regarding several aspects:

First, this thesis presents the results of the analysis of large-scale Internet traffic data sets, consisting of several billion TCP connections captured during the past decade, regarding

- flow characteristics, their correlation, and the significance of heavy hitters,
- throughput and throughput limitations of TCP connections,
- and the usage of TCP window scaling, SACK, and ECN.

Second, this thesis introduces an approach to enable large-scale TCP throughput limitation monitoring, i.e., online TCP root cause analysis, considering monitored bandwidths of several Gbit/s running on commodity hardware based on the approach to conduct TCP RCA offline introduced by Siekkinen et al. [11]. Further, this thesis addresses the classification of single changes in the throughput of a TCP connection with results of TCP root cause analysis, which has not been considered by related work so far.

In addition, this thesis conducts active Internet measurements to evaluate the impacts of different endpoint configurations, like the use of performance-related TCP options, on the throughput of TCP connections and to survey the accuracy of end-to-end capacity estimation and deployed capacities in today's Internet.

2.1 Analysis of TCP Flow Characteristics

The field of flow and traffic characterization is a frequently addressed topic. However, a detailed analysis of TCP flow characteristics on today's Internet and their evolution during the past years still needs to be provided.

Our work on TCP flow characteristics, like size, duration, and rate, is closely related to Zhang et al. [2], who analyzed Internet flow rates and their root causes in 2002. Our study confirms the significantly dominating transmission of bytes by big-fast flows and approves findings regarding the correlation of TCP flow characteristics. Lan et al. [5] additionally consider the burstiness of flows as a flow characteristic and classify flows as heavy hitters for duration, size, and rate based on threshold values, respectively, based on the 99th percentiles of flow characteristics, which we also survey in this thesis. As Zhang et al. [2], Lan et al. find a strong positive correlation between flow size and flow rate.

Further, heavy hitters are the object of study regarding their identification and detection [26]–[28]. The burstiness of CAIDA traces is studied by Lazarou et al. [29]. Flow rates are studied in detail for cellular networks by Zhang et al. [22], considering duration, size, and rate of flows and the limiting factor behind such rates. Further studies focus on the characterization of networks [30] and traffic patterns regarding daytimes [31], [32]. Maier et al. [33] present an extensive study of residential broadband Internet traffic characteristics.

2.2 Approaches to TCP Root Cause Analysis

This section introduces and compares existing approaches to TCP root cause analysis (TCP RCA), also referred to as TCP throughput limitation analysis, regarding considered throughput limitations and characteristics of the corresponding measurement methods. A general overview of related work addressing TCP root cause analysis is shown in Table 2.1. Studies on TCP throughput limitations of Internet connections are shown in Table 2.2.

Zhang et al. [2] pioneered TCP root cause analysis in 2002, resulting in the TCP root cause analysis tool *T-RAT*. As the successor of T-RAT, Siekkinen et al. [11], [34], [35] proposed their approach to TCP RCA purposed to overcome shortcomings of *T-RAT* in 2008. Other approaches to TCP RCA were introduced by Timmer et al. [10] in 2006, Sun et al. [23] in 2011, Araújo et al. [12] in 2014, Bąk et al. [13] in 2015, and Ghasemi et al. [4] in 2016.

An overview of considered throughput limitations by the different approaches is presented in Table 2.3. Table 2.4 shows characteristics of the corresponding measurement methods regarding flow segmentation, measurement position, and online analysis capabilities. Both aspects of related work are elaborated in more detail in the following.

Zhang et al. [2]

Zhang et al. pioneered the analysis of throughput limitations of TCP connections in 2002, resulting in the TCP root cause analysis tool *T-RAT*. *T-RAT* provides passive root cause analysis independent of the measurement position based on identifying flights of TCP packets, i.e.,

	Year	RCA approach	Online RCA	Internet study
Zhang et al. [2]	2002	✓	✗	✓
Timmer et al. [10]	2006	✓	✗	✓
Siekkinen et al. [11], [24]	2008	✓	✗	✓
Sun et al. [23]	2011	✓	✓	✓
Zhang et al. [22]	2014	✗	✗	✓
Araújo et al. [12]	2014	✓	✗	✓
Bak et al. [13]	2015	✓	✗	✗
Ghasemi et al. [4]	2016	✓	✓	✓

Table 2.1: Overview of related work regarding TCP root cause analysis.

	Capture year	Data set size	Capture point
Zhang et al. [2]	2001-2002	112 M packets	access network, Tier 1 peering link, backbone link
Timmer et al. [10]	2002-2003	182 GB	access network
Siekkinen et al. [11], [24]	2006	290 GB	access network
Sun et al. [23]	2011	137 K flows	cache server of CoralCDN
Zhang et al. [22]	2011	22.9 M flows	GPRS core network, access network
Araújo et al. [12]	2007-2011	577 M flows	MAWI transit link
Ghasemi et al. [4]	2013	244 K flows	backbone link (CAIDA Equinix-Chicago)

Table 2.2: Overview of Internet studies on TCP throughput limitations.

groups of packets sent during the same RTT, which are then analyzed for different throughput limitations. Based on analyzed flights, *T-RAT* determines throughput limitations after 256 observed packets or after an idle time of 15 seconds.

In addition to short TCP connections that never leave the slow start phase, Zhang et al. consider network bottleneck limitations, transport layer limitations, receiver and sender window limitations, and limitations by the sending application. Thereby, network bottlenecks are defined as limitations due to congestion, i.e., due to a bottleneck link shared by several concurrent connections, or limited bandwidth, i.e., a bottleneck link whose bandwidth is fully utilized by a single connection. To determine the current limitations of a connection, Zhang et al. rely on indicators like RTT, MSS, retransmissions, estimated congestion control phases, and flight sizes.

Siekinnen et al. [11], [35]

During experiments with simulated TCP traffic, Siekkinen et al. [34] find that determining flights of TCP packets and keeping track of congestion control phases does not work reliably in productive environments due to delayed acknowledgments and cross traffic. Further, Shakkottai et al. [36] survey the occurrence of flights in captured ISP backbone traffic and state that flights are not common in observed TCP flows and typically do not consist of more than seven packets. Shakkottai et al. describe flights as a small window phenomenon.

	Sender			Receiver			TCP	Network		
	App. limited	Send buffer	L7 protocol	Receive wnd.	Slow receiver	Recv. shaping	Cong. Avoidance	General	Shared BN	Unshared BN
Zhang et al. [2]	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓
Siekinnen et al. [11]	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓
Timmer et al. [10]	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗
Sun et al. [23]	✓	✗	✗	✓	✗	✗	✓	✓	✗	✗
Bąk et al. [13]	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗
Araújo et al. [12]	✓	❍	✗	❍	✗	✓	✗	✗	✗	✗
Ghasemi et al. [4]	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗

Table 2.3: Throughput limitations considered by related work.

	Passive	Segmentation	Position	Online
Zhang et al. [2]	✓	256 p. or idle time	any	✗
Siekinnen et al. [11], [35]	✓	transfer periods	any	✗
Timmer et al. [10]	✓	-	near sender	✗
Sun et al. [23]	✓	time intervals	sender	✓
Bąk et al. [13]	✗	-	sender	✗
Araújo et al. [12]	✓	-	any	✗
Ghasemi et al. [4]	✓	-	near sender	✓

Table 2.4: Characteristics of different TCP RCA approaches.

To overcome corresponding shortcomings of *T-RAT*, Siekkinen et al. introduce a passive approach to determine throughput limitations based on previously determined transfer periods and so-called limitation scores. Thereby, authors differ between unshared and shared network bottlenecks, limitations by the receiver window, and limitations by the transport layer. The approach by Siekkinen et al. provides properties like passive analysis, independence of the measurement position, and segmentation of analyzed connections.

To differ periods in which the sending application limits a connection from periods in which the sending application is capable of sending bulks of data, Siekkinen et al. [35] introduce the *Isolate & Merge* (*I&M*) algorithm that relies on the MSS, packet sizes, inter-arrival times between subsequent packets, and RTT estimates to segment a flow into three different kinds of periods:

- **Application Limited Periods (ALP)**: the sending application does not provide enough data to utilize available network and endpoint resources fully.
- **Bulk Transfer Periods (BTP)**: the sender produces enough data to continuously send data and fully utilize available network or receiver resources.

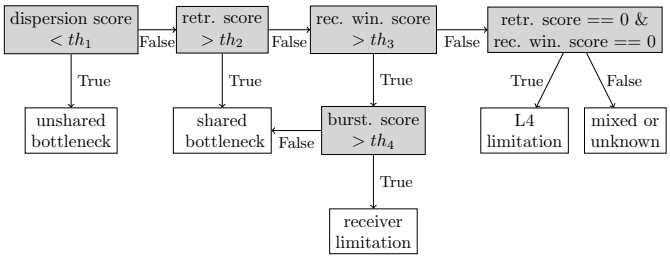


Figure 2.1: Decision tree for TCP RCA by Siekkinen et al. [11].

- **Short Transfer Periods (STP)**: short bulk transfers that contain less than 130 packets and are mainly dominated by congestion control.

To determine the limitations of bulk transfer periods, Siekkinen et al. define so-called limitation scores, which are then compared to threshold values based on a decision tree [11], as illustrated in Figure 2.1. Such limitation scores are calculated based on throughput, estimated path capacity, retransmission rate, RTT, and time series data of the advertised receiver window and outstanding bytes. In addition to extracted packet features and calculated metrics, Siekkinen et al.'s approach requires determining the measurement position at which a connection is captured. The measurement position is then used to shift the observed packet timestamps to approximate the sender's perspective on received and sent packets required to correctly calculate the number of outstanding bytes at a given point in time.

Timmer et al. [10]

Timmer et al. [10] introduce an approach to identify throughput limitations like the network, slow sender and slow receiver limitations, the send and receive buffer sizes, and application layer protocol mechanisms. While considering a wide range of throughput limitations, the introduced approach is limited to determining one single root cause for a connection and is limited to analysis near the sender. To determine root causes, Timmer et al. define criteria based on the number of outstanding bytes, the time series of the advertised receiver window, payload size of acknowledgment packets, estimated send buffer size, RTT, and packet loss. If several limitations are detected, the throughput limiting factor is chosen based on a priority list. As the send buffer size can not be extracted from a passively observed connection, it has to be estimated. Timmer et al. propose estimating the send buffer size as the highest number of outstanding bytes observed throughout the connection when there is currently no receive buffer limitation.

Sun et al. [23]

Sun et al. [23] introduce an approach to determine performance bottlenecks of TCP connections based on access to the sender. The approach aggregates different metrics like the current congestion and receiver window, bytes in flight, bytes in the send buffer, RTT, or retransmissions, polled from the sender's network stack. Accordingly, the approach requires access to the sender of a connection. Based on aggregated metrics, throughput limitations like the server's send buffer size, the congestion window, network congestion, or the receiver advertised window

can be identified. Due to periodic polling of the mentioned metrics, i.e., a poll each 50 ms, the approach is considered to be suitable for online analysis.

Araújo et al. [12]

Araújo et al. [12] study throughput limitations of TCP connections beyond congestion control. Thereby, authors consider application-paced throughput, host limitations, i.e., limitations by the endpoints' buffers, without further differentiating send and receive buffer limitations, and receiver-shaped throughput, i.e., an artificial rate constraint using flow control. The analysis approach relies on metrics like MSS, RTT, retransmissions, estimated congestion window size, and flight size. Further limitations like network bottlenecks are not considered.

Bąk et al. [13]

Bąk et al. [13] introduce an approach to determine the performance limitations of TCP connections, considering slow sender limitations, the receiver window, and the network. The approach requires active probing of path and endpoint characteristics, like path capacity, and the emulation of the TCP connection state, i.e., the congestion window size and congestion control phases. Bąk et al. argue that their approach should be applied to traffic captured near the sender to ensure precise RTT measurements. In addition to capacity and RTT, metrics considered to determine limitations are throughput, retransmissions, emulated congestion window size, buffer delay, and outstanding bytes. Root causes are determined based on a decision tree. Bąk et al. validate their approach based on downloads conducted in a commercial mobile network considering limitations by path capacity and socket buffer size at the server.

Ghasemi et al. [4]

Ghasemi et al. introduce the TCP performance diagnosis tool *Dapper*, prototyped in P4 and able to determine the root cause behind low TCP performance considering the sender, the receiver, and the network, without further differentiating root causes within the limited component. *Dapper* is designed to run near the sender, for instance, in a top-of-rack switch or at the hypervisor, to closely observe application behavior without server access. Further, *Dapper* provides results in real time and uses two different analysis phases. First, the approach detects connections suffering from low performance and, second, determines the throughput limitation of such connections. Thereby, low performance is determined based on a threshold for a connection's average throughput rate. To determine root causes, *Dapper* considers metrics and indicators like application reaction time, sending rate, MSS, estimated congestion window size, RTT, *awnd*, and the occurrence of delayed ACKs. To evaluate their implementation, Ghasemi et al. generate TCP connections with intended throughput limitations considering sender limitations, receiver limitations, network limitations, and connections with mixed limitations.

2.3 Measurement Studies on Throughput Limitations

This section summarizes findings from measurement studies on the throughput limitations of TCP connections on the Internet. An overview of such studies is presented in Table 2.1.

Zhang et al. [2]

Zhang et al. apply root cause analysis based on the tool *T-RAT* on packet traces captured at eight different capture points, i.e., four bidirectional traces captured at a high-speed access link and four unidirectional traces captured at backbone links, respectively, on a peering link between two Tier 1 providers. Traffic was captured between January 2001 and January 2002. In total, traces include over 112 million captured TCP packets.

Zhang et al. survey the prevalence of different throughput limiting factors based on the share of limited bytes and find that network congestion is the most common throughput limitation by limiting transmission of 22% up to 43% of bytes captured in considered traces. Next to congestion, window limitations, including receiver and sender window limitations, are the most significant throughput limiting factor with shares of bytes between 8% and 43%. The sum of bytes limited by opportunity limitations, application limitations, and transport limitations is smaller than 20%. Bandwidth limitation affects less than 2% of analyzed bytes. Further, Zhang et al. study the most dominant limitation per flow, i.e., the limitation affecting most analyzed flights. Thereby, Zhang et al. find that 90% of flows are mainly limited by opportunity or the sending application.

Siekinnen et al. [24]

Siekkinen et al. [24] survey throughput limitations of traffic captured at an access network, i.e., ADSL user traffic, captured in March 2006. The data set consists of 290 GB of traffic generated by approximately 3000 ADSL users, while 15 % of observed clients cause more than 85 % of observed traffic. To study throughput limitations affecting a client, Siekkinen et al. determine a *main limitation* and a set of *experienced limitations* for each client. A *main limitation* refers to the throughput limitation limiting the majority of observed data for a client, while *experienced limitations* cover all limitations with a share of limited bytes larger than 0. Siekkinen et al. observe that 86 %-95 % (depending on the daytime, 4–4:30 a.m. and 3–3:30 p.m.) of clients are mainly limited by the application. 0 %-6 % of clients are mainly limited by the access link, 3 %-4 % by another, distant link. Regarding *experienced limitations*, 90 %-100 % of analyzed clients experience any application limitation, 0 %-7 % any limitation by the access link, and 39 %-66 % of clients any limitation by another link. Further, Siekkinen et al. report an increase of application-limited bytes observed in data sets captured in 2002 and 2004 at an ADSL network in the Netherlands from 40 % to 65 %, which authors explain by an increase of P2P application traffic.

Timmer et al. [10]

Timmer et al. [10] apply their approach to identify throughput limitations on traffic repositories provided by the University of Twente captured at three vantage points located in access networks in the Netherlands in 2002 and 2003. Analyzed traffic is restricted to connections captured at the sender side due to characteristics of the analysis approach, and connections larger than 100 kB to avoid domination by connections' start-up phase. Timmer et al. [10] determine one throughput limitation per connection and aggregate per-connection data to the total share of flows and bytes per connection for each vantage point. In the analyzed traffic captures, 20.9 %-

27 % of connections are limited by the network, transmitting 33.3 %-38.7 % of bytes. Further, the sending application limits throughput for 24.2 %-28.5 % of connections (22 %-34.8 % of bytes), and the receive buffer limits throughput for 10.8 %-12.4 % of connections (9.6 %-16.3 % of bytes). Other limitations like slow receivers, the application protocol, and send buffer sizes are limiting connection shares smaller than 5 %, with exceptions for individual vantage points. For 21.5 %-39 % of connections no limitation could be determined, affecting 9.3 %-21.3 % of bytes.

Sun et al. [23]

Sun et al. [23] apply their approach to identify performance bottlenecks in data collected in February 2011 within the CoralCDN. In total, statistics for 209K connections were available, while only 137K connections were analyzed due to other connections suffering from CDN cache misses, which degrade performance. To study the significance of considered performance limitations, Sun et al. calculate the share of connection duration per limitation. Afterward, Sun et al. determined the share of connections by showing a certain percentage of the time spent during a limitation. 19.7 % of connections show a limitation by the server application, 52.5 % of connections show a limitation by the server's network stack, 6.20 % of connections show a limitation by the network path, and 5.77 % of connections suffer from a client-side performance limitation. At the same time, less than 1 % of connections show a limitation by the application, the network path, or the client for a duration share larger than 99 %. 3.99$\dot{\%}$ of connections are limited by the server's network stack for over 99 % of connection duration.

Araújo et al. [12]

Araújo et al. [12] apply their approach to detect limitations by application-pacing, the sending or receiving host, and receiver-shaped throughput on un-anonymized traffic captures provided by MAWI, captured on a transit link between 2007 and 2011. In total, their study considers 577 million TCP connections. To assess the significance of different flow rate limitations, Araújo et al. calculate the share of bytes per limitation for each connection. In data captured in 2007, 49.47 % of observed bytes are transmitted during a limitation by the application, 18.58 % of bytes are limited during a host limitation, and only 0.5 % of bytes are limited by receiver shaping. Accordingly, 68.6 % of bytes are not limited by the network respectively congestion control. For 2011, Araújo et al. report decreased shares of application-limited bytes (46.1 %) and bytes limited by the end hosts (13.5 %).

Zhang et al. [22]

In 2014, Zhang et al. [22] applied the root cause analysis approach introduced by Siekkinen et al. [11] to data sets collected in a GPRS core network in 2011, consisting of 3.9 million connections, and in a wired access network, consisting of 19 million connections. Authors find that unshared bottleneck limitations dominate wireless client connections in the cellular networks by limiting the transmission of 58% of data, while such limitation only affects 1% of bytes in the wired network data set. Further, transmitted bytes in the wireless network are more frequently limited by shared bottleneck limitations (wireless: 16 %, wired: 12 %) and less frequently limited by the transport layer (wireless: 3 %, wired: 38 %), while no receiver limitations are observed for

both data sets. In addition, the authors report that the majority of connections does not experience limitations by the sending application and that the majority of bytes is not transmitted during application limitation.

Ghasemi et al. [4]

Ghasemi et al. [4] apply their tool for data plane performance diagnosis on captured Internet traffic provided by CAIDA. After removing connections consisting of less than ten packets, the analyzed data set consists of 244 thousand TCP connections. To assess observed throughput limitations, Ghasemi et al. calculate the share of duration a connection spends during a specific limitation. Key observations are that flows show significant shares of being limited by considered limitations, that many connections are limited by two factors during their lifetime, and that 90% of connections spend some time as network limited [4].

2.4 Estimating Capacity of Network Paths

Different capacity estimation tools were introduced and compared throughout the years. Thereby, tools differ regarding their approaches, considering packet pair dispersion (PPD)-based or variable packet size (VPS) probing-based estimation, required deployments, like single- and double-ended, and traffic generation, i.e., conducting measurements actively or passively.

Dovrolis et al. [14], [37] introduce the active and double-ended tool *pathrate*, relying on packet pair dispersion, packet train dispersion, and probing with different packet sizes. Thereby, *pathrate* is focused on accuracy instead of scalability or estimation duration [38]. Other active PPD-based tools like *sprobe* [15], *asymprobe* [17], *capprobe* [16], or *pbprobe* [18] only require single-ended deployments enabling measurements in uncooperative environments while setting focus on estimates on asymmetric network paths, high-speed networks, or estimation duration. Further tools like *clink* [39], *traceprobe* [40], or *pathchar* [41] conduct active measurements with VPS probing.

Regarding passive estimation based on PPD, the tools *nettimer* [19] and *PPrate* [20] were introduced in 2001, respectively 2006. *Nettimer* is purposed to overcome limitations regarding the duration of estimation, the analyzed direction of a connection, and the requirement to actively generate analyzed traffic. Lai and Baker [19] found that *nettimer* results in a relative error of less than 10 % in most considered test cases, including wired and wireless links. *PPrate* aims to implement accurate PPD-based capacity estimation by filtering packet pairs with outlying dispersion to reduce impacts by cross traffic interference. En-Najjary and Urvoy-Keller [20] evaluate *PPrate* with synthetically generated traces and a data set collected on the PlanetLab network. Next to accuracy, authors survey the number of packets required for reliable estimates based on FTP file transfers. A comparison of *PPrate* to *pathrate* shows comparable accuracy [20].

Another passive capacity estimation tool is *MultiQ* [21], based on so-called equally-spaced mode gaps. A comparison of *MultiQ* to *pathrate* and *nettimer* reveals comparable results between *MultiQ* and *pathrate*, while *MultiQ* is shown to be more accurate than *nettimer* [21]. Another study conducts a comparison of the three passive tools *MultiQ*, *nettimer*, and *PPrate* to the

active tool *pathrate* based on PlanetLab paths and traces captured at an ADSL access network in 2008 [25]. The study reveals *PPrate* to be the most accurate passive tool in most cases.

Further, researchers introduce capacity estimation with a focus on wireless networks [42] and integration to the data plane to provide timely estimates for an active connection [43]. Other publications survey measurements of bandwidth-related characteristics of Internet paths and connections, like available bandwidth [44]–[50].

2.5 Measurements on TCP Option Deployment on the Internet

TCP options and their deployment are frequently addressed topics. Studies of TCP option deployments conducted in the early 2000s observed the evolution of option deployment, starting from only small adoption of servers to TCP options like TCP window scaling, selective acknowledgments, explicit congestion notifications, and TCP timestamps [6], [8], [9]. An active and passive measurement study conducted in 2013 by Kühlewind et al. [7] reports widespread deployment of WS and SACK and observes a slower spreading of ECN usage. Such observation is confirmed by Murray et al. [51] in 2017, who only observed small usage of ECN in traffic captured at a university network. A more recent study observes that ECN is supported by the majority of domains listed in the Alexa Top 1 M list [52]. Lim et al. report small shares of connections supporting ECN in passively captured university network traffic [53] in 2022. Edeline and Donnet [54] survey the impact of TCP option usage in controlled test environments.

Part II

Passive Measurements

Chapter 3

Large-Scale Analysis of TCP Connection Characteristics

The evolution of technologies and services involved on the Internet, considering better network expansion and improved customer access or emerging applications and services like social media or audio and video streaming, indicate that characteristics of Internet flows, like duration, size, and rate, are also changing. While patterns, distributions, and correlations of TCP flow characteristics have been studied in previous research [2], [5], [31] in 2002, respectively 2006, there is only little insights into the evolution of flow characteristics over time and TCP flow characteristics on today's Internet.

Described evolutions on the Internet also imply changing throughput limitations of TCP connections. As elaborated in Chapter 2, different approaches to analyzing throughput limitations of TCP connections and corresponding measurement results were presented by related work. However, most recent studies evaluating throughput limitations of Internet traffic are based on traffic captured in 2013 [4] and 2011 [12], [22], [23], motivating a revisit of throughput limitations on the Internet 10 years later.

In this chapter, we examine how the characteristics of TCP flows have changed during the last few years, analyze whether there are significant trends regarding the relevance of heavy hitter flows, and survey the implications of such trends on the correlation of flow characteristics. In particular, this chapter contributes a large-scale analysis of flow characteristics of two data sets composed of Internet traffic captures taken between 2008 and 2016, respectively, in 2018 and 2019, published by CAIDA [55]. In total, we analyze over 2.5 billion TCP flows used to transmit over 65 TB of payload data. We apply different taxonomies to assess the relevance of heavy hitters, the relevance of so-called big-fast flows, and the correlation of flow characteristics as surveyed by related work presented in 2002, respectively 2006 [2], [5]. Further, this chapter surveys the evolution of TCP throughput and throughput limitations of TCP traffic captured on the Internet during the past years. In particular, we analyze throughput and throughput limitations based on a data set consisting of several billion TCP connections captured by MAWI [56] on a transit link between 2016 and 2023. Next to throughput and throughput limitations, this chapter surveys

the usage of throughput-related TCP options like window scaling, SACK, or ECN based on the mentioned MAWI data set purposed to provide a recent view on TCP option usage on the Internet and to assess potential impacts on observed throughput.

This chapter is partly based on the publication:

> **On the Evolution of Internet Flow Characteristics**,
> *Simon Bauer, Benedikt Jaeger, Fabian Helfert, Philippe Barias, Georg Carle*,
> in Applied Networking Research Workshop (Virtual) 2021, Juli 2021

The case study on the evolution of Internet flow characteristics, heavy hitter relevance, and correlation of flow characteristics presented in Section 3.2 correspond to the named publication. The same applies to Section 3.1.1, introducing the applied methodology to evaluate and compare flow characteristics.
In addition to the named publication's contents, this chapter presents a large-scale study on TCP option usage and throughput limitations of TCP connections in Section 3.3 based on a second large-scale data set. The methodology used for the study on throughput limitations and option usage is introduced in Section 3.1.2.
The study on throughput limitations and TCP option usage significantly extends the named publication's contribution by providing a more microscopic view of connection throughput.

3.1 Methodology

This section describes the methodology used to extract flow characteristics like duration, size, and rate from traffic captures and the methods and taxonomies applied to analyze extracted flow characteristics.

3.1.1 Analyzing TCP Flow Characteristics

We refer to a flow as a sequence of packets bidirectionally sent and received by the same IP 5-tuple. To identify flows, we rely on different methods for TCP and UDP traffic. Regarding TCP, we require a flow to start with the TCP three-way handshake. As UDP is a stateless protocol, we define a UDP flow to start with the first packet we observe for an unseen IP 5-tuple. Observing the TCP handshake allows to detect a connection's initiator, which we refer to as the client, while servers respond to the initially sent packet. For UDP, we assume the server often uses a system port. If this heuristic is not feasible, we assume the client sends the first packet. TCP flows end if we observe the TCP connection tear-down, observe a TCP handshake of an IP 5-tuple already tracked, or if we do not observe a packet with the corresponding IP 5-tuple for a specific time interval. For the termination of UDP flows, we only rely on time intervals without observed packets. By default, the analyzer uses a timeout interval of 5 minutes. Note that UDP flows are only considered to determine the share of TCP flows in captured traffic.

We define the duration of a flow as the time interval between the first seen packet and the last packet before the termination of a flow. We refer to flow size as the sum of the Layer-4 payload

sizes of all packets of a flow. Flow rate refers to the fraction of flow size and flow duration. The tool used to process traffic captures and to extract flow characteristics was implemented in the scope of the Master's by Fabian Helfert [57] and Philippe Barias [58], as elaborated in more detail in Section 3.6. An overview of the tool's architecture and implementation is presented in the publication "On the Evolution of Internet Flow Characteristics" [59]. Details on the used tool can be found in the corresponding Master's theses [57], [58].

We apply different methods and taxonomies to describe properties of analyzed flows and differences between single traces to analyze the evolution of flow characteristics. First, we calculate the cumulative distribution function (CDF) for each characteristic and packet trace to describe changes regarding the distribution of measured characteristics. The cumulative distribution of characteristics is purposed to compare the share of flows for a particular interval in the spectrum of measured characteristics.

Second, we apply the threshold-based taxonomy introduced by Lan et al. [5] to analyze the relevance of so-called heavy hitter flows. Lan et al. define Elephant, Tortoise, and Cheetah flows as heavy hitters in terms of size, duration, and rate. Thresholds are determined by taking the average of measured values and adding the standard deviation three times or by calculating the 99th percentile of measured values.

Third, we apply a two-two taxonomy introduced by Zhang et al. [2] to classify flows regarding their size and rate. Based on threshold values for flow size and rate, the taxonomy classifies flows according to four groups: small-slow, small-fast, big-slow, and big-fast flows. We are particularly interested in the share of big-fast flows and the corresponding share of bytes, as Zhang et al. found that a tiny share of big-fast flows transmits most bytes of Internet traffic.

Next to assessing the relevance of heavy hitters, we are interested in analyzing the correlation between flow characteristics as done by earlier studies [2], [5]. Therefore, we calculate correlation coefficients according to Pearson, describing the linear relation between two characteristics. We logarithmically transform input data before calculating correlation due to the extensive spectrum and long-tailed distributions of observed values.

3.1.2 Analyzing TCP Throughput Limitations and TCP Option Usage

This section describes the extraction of performance indicators, the use of TCP options from captured connections, and the approach used to determine throughput limitations based on extracted performance indicators. To analyze traffic captures for option usage, performance indicators, and throughput limitations, a modified version of the tool developed by Fabian Helfert [57] and Philippe Barias [58] was used, implemented in the scope of the Master's thesis by S. Hülkenberg [60]. Adaptions to the implementation by S. Hülkenberg [60] by the author of this thesis are described in more detail in Section 3.6. The tool extracts and aggregates packet and flow features from captured traffic, like TCP header flags, sequence and acknowledgment numbers, IP addresses, ports, and TCP option fields. Such features then determine different TCP performance indicators and throughput limitations.

TCP Performance Indicators

We determine throughput as the fraction of the sum of packet sizes as specified by the total length field in IP packet headers and the duration between a connection's first and last packet. To calculate RTT samples, we rely on the TCP timestamp option. As traffic is captured at an arbitrary position between both endpoints of a connection, RTT calculation requires matching triplets of packets to get an RTT estimate based on *TSval* and *TSecr* values as introduced by Vael et al. [61]. Detection of retransmitted packets is implemented according to the Wireshark documentation [62]. The retransmission rate refers to the fraction of retransmitted and total observed data.

We estimate end-to-end path capacity with an adapted version of the PPrate algorithm introduced by En-Najjary and Urvoy-Keller [20]. In particular, we implement processing of IATs and corresponding bandwidth samples as introduced by *PPrate*, but rely on a simplified method to choose the mode implying the capacity estimate, i.e., we choose the bucket of the bandwidth histogram containing most samples. Note that we rely on ACK-based estimates as also applied by Siekkinen et al. [11].

Determining Throughput Limitations

We implement the analysis of TCP throughput limitations based on the TCP root cause analysis approach introduced by Siekkinen et al. [11], [35]. In particular, we first determine application limited periods based on the Isolate algorithm [35] and then calculate the dispersion score, retransmission score, and receiver window score according to Siekkinen et al. [11]. Accordingly, our implementation is capable of determining limitations by the sending application, unshared bottlenecks, shared bottlenecks detected due to packet loss, limitations due to a fully utilized receiver window, and transport limitations. Based on the decision tree introduced by Siekkinen et al. [11], calculated scores are then compared to predefined threshold values. For our measurements, we use the threshold value 0.1 for dispersion score comparisons, the threshold value 0.01 for retransmission score comparisons, and the threshold value 0.5 for receiver window score comparisons.

We only consider flows with more than 300 data packets to analyze throughput limitations to ensure sufficient samples for capacity estimation, as suggested by En-Najjary and Urvoy-Keller [20]. Assuming an MSS of 1500 B, such a connection transmits more than 400 kB, a size observed to be close to the 99th percentile of Internet flow sizes, as described in Section 3.2. Further, due to performance considerations, the tool splits flows into chunks of 50000 packets. Accordingly, the approach to determine throughput limitation is applied to chunks of 50000 packets, reducing computational costs of required time series analysis. Afterward, results per chunk are concatenated in the appropriate order.

Extracting TCP Option Usage

For the introduced study of TCP option usage on the Internet, we consider the TCP timestamp option (TS) [63], as used to calculate RTT, and performance-related options, like TCP window scaling (WS) [63], selective acknowledgments (SACK) [64], and explicit congestion notifications (ECN) [65].

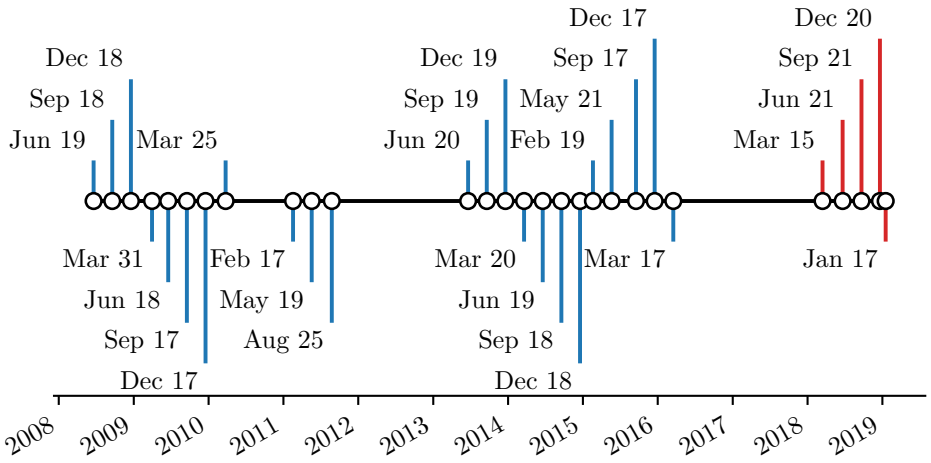


Figure 3.1: Used data sets: traces taken in Chicago [67] in blue, traces taken in New York [68] in red.

We refer to a connection as using TCP timestamps when the sender and the receiver of a connection send the TCP timestamp option during the TCP handshake, confirming that both endpoints support TCP timestamps. Regarding TCP WS, both the sender and receiver of a connection must signal their support of TCP window scaling. We refer to a connection as using TCP WS when both endpoints indicate their support of WS by sending the `Window Scale` option during the TCP handshake. To assess the use of TCP SACK, we analyze SYN packets for the `SACK permitted` option, indicating that the sender of the SYN packet is ready to process the `SACK` option [64]. RFC2018 [64] does not require the sender of the `SACK` option to send the `SACK permitted` option during the TCP handshake. Accordingly, we refer to a connection using SACK if the sender of a connection sends the `SACK permitted` option, as the receiver of the data is then allowed to use the `SACK` option for its acknowledgments. To extract according options from TCP headers, the gopacket library [66] is used. We determine ECN support by analyzing ECE and CWR flags of the TCP header during the initial handshake. In particular, we require the first SYN packet to have the `ECE` and `CWR` flag set, indicating that the initiator of the connection is ECN capable as described in RFC3168 [65]. Further, we require the `ECE` flag to be set in the corresponding SYN-ACK packet. If both requirements are satisfied, we refer to a connection as supporting, respectively, using ECN.

3.2 Long-term Analysis of TCP Flow Characteristics

This section first introduces the data sets used for our study on TCP flow characteristics in Section 3.2.1. After describing applied flow filtering, we present and compare measurement results for traffic captured between 2008 and 2016 in Section 3.2.2 according to the methodology described in Section 3.1.1. Afterward, we compare such findings to observations based on the data set captured in 2018 and 2019 in Section 3.2.3.

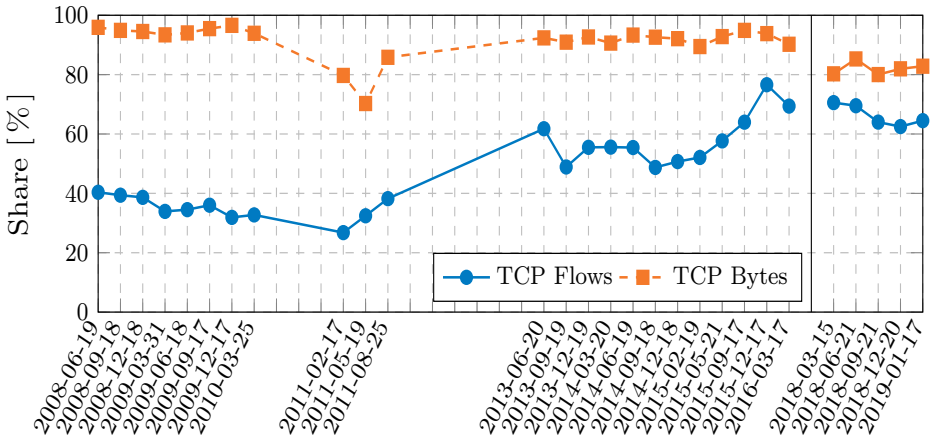


Figure 3.2: Share of TCP flows and transmitted bytes before flow filtering.

3.2.1 Data Set

For our long-term analysis of Internet traffic characteristics, we select traces from two different capture points provided by the Center of Applied Internet Data Analysis (CAIDA) [55]. All traces provide one hour of network traffic captured on Internet backbone links with a bandwidth of 10 Gbit/s. To respect entities' privacy, CAIDA anonymizes IP addresses of captured traffic in a prefix-preserving manner. We select 23 traces captured between June 2008 and March 2016 on a Tier-1 ISP backbone link in Chicago [67]. Considering distributing traces equally over time, we choose 23 traces, mainly at three-month intervals. The selected traces captured in Chicago consist of 1.16 billion TCP flows (2.63 billion flows in total). The long-term data set captured in Chicago includes two significant periods without traffic, as no captures are available in such periods: First, 11 months without traces between March 2010 and February 2011, and second, 18 months without traces between September 2011 and March 2013. To compare our findings of the introduced long-term data set to more recent traffic, we select five traces captured on a Tier-1 ISP backbone link in New York between March 2018 and January 2019 [68]. We find significantly larger amounts of traffic in the New York traces. I.e., the five selected traces carry 1.48 billion TCP flows (2.25 billion flows in total). All selected traces are listed in Figure 3.1.

TCP Traffic: As our analysis focuses on the characteristics of TCP flows, we survey how significant TCP is in the sense of observed flows and transmitted data. Throughout our data set, we find significant dominance of bytes transmitted by TCP. TCP carries over 90 % of bytes in all traces captured in Chicago, except the three traces taken in 2011. The share of TCP flows increases from around 40 % up to shares larger than 70 % in December 2015 and March 2016. Correspondingly, the share of UDP flows shows a significant decrease, while UDP transmits only a small share of bytes. Regarding traces captured in New York, we find shares of TCP flows similar to traces captured in Chicago. However, we find smaller shares, i.e., around 80 %, of bytes transmitted by TCP. Figure 3.2 shows the share of TCP flows and corresponding bytes over time.

Pre-filtering: Previous studies of flow rates [2], [22] filter flows for a minimum duration of 100 ms. Such filtering aims to remove single packet flows whose duration is zero, making rate calculation unfeasible. Further, all packets of very short flows might be sent back-to-back, which falsifies the measured rate, too. Therefore, we filter flows shorter than 100 ms. We observe that such filtering excludes nearly 20 % of TCP flows, while the remaining flows still carry 99.94 % of total bytes transmitted by TCP in the long-term Chicago data set. For traces collected in New York, nearly 35 % of TCP flows get filtered out, while the remaining flows still carry 99.96 % of bytes. We conclude that such filtering of short flows still allows representative analysis of flow characteristics since almost all transferred bytes are considered for further analysis.

3.2.2 Evolution of Flow Characteristics

Distributions: To survey the distribution of measured flow characteristics, we analyze the corresponding cumulative distribution function, as shown in Figure 3.3. Note that the x-axes of CDFs are cut for readability. Regarding flow rates and sizes, we observe long-tailed distributions and significant clusters covering most analyzed flows. We find that around 85 % of TCP flows carry between 100 B and 10 kB, with maximum flow sizes of several GB. Regarding flow rates, over 50 % of flows transmit payload with an average rate between 1 kbit/s and 100 kbit/s, while we observe flow rates up to several Gbit/s.

Heavy hitters: Lan et al. [5] define heavy hitters based on the average and standard deviation of measured flow characteristics or based on the 99th percentile of a characteristic. Just as Lan et al., we find similar results for both definitions. Therefore, we rely on the 99th percentile for further analysis.

Before assessing the relevance of heavy hitter flows, we are interested in trends regarding the longest, largest, and fastest flows. Therefore, we calculate the 99th percentiles of duration, size, and rate, hereafter notated as D_{P99}, S_{P99}, R_{P99}, for each trace and compare such values over time. Figure 3.4 shows measured 99th percentiles over time. D_{P99} shows little change between 2008 and 2010. Between June 2013 and March 2016, D_{P99} increases for a factor near 1.5, i.e., from around 400 s for early traces to around 600 s for more recent traces. Flow size shows larger variances between measured percentiles of different traces. We observe that S_{P99} for traces between 2008 and 2009 tendentiously cluster around 250 kB, while S_{P99} measured for traces between 2013 and 2016 cluster around 400 kB. Regarding flow rates, we find increasing R_{P99} over time. For traces before 2013, R_{P99} is between 300 kbit/s and 400 kbit/s, except for the trace captured in Dec 2009. After 2013, R_{P99} varies between 400 kbit/s and 800 kbit/s. For traces taken in 2015 and 2016, R_{P99} regularly exceeds 600 kbit/s.

To assess how relevant flows within the 99th percentiles are, we analyze the share of bytes transmitted by such flows. Further, we are interested in the share of flows and bytes by intersections of such flow sets to survey relations between different percentiles. As we do not find trend-like evolutions within the share of bytes for the 99th percentiles over time, we calculate the average of measured byte shares, as shown in Table 3.1. We find that flows within the S_{P99}, on average, carry 89 % of all TCP bytes with a fairly small standard deviation of 2 %. On average, nearly 20 % of bytes are transmitted by the intersection of all three 99th percentiles $D_{P99} \cap S_{P99} \cap R_{P99}$.

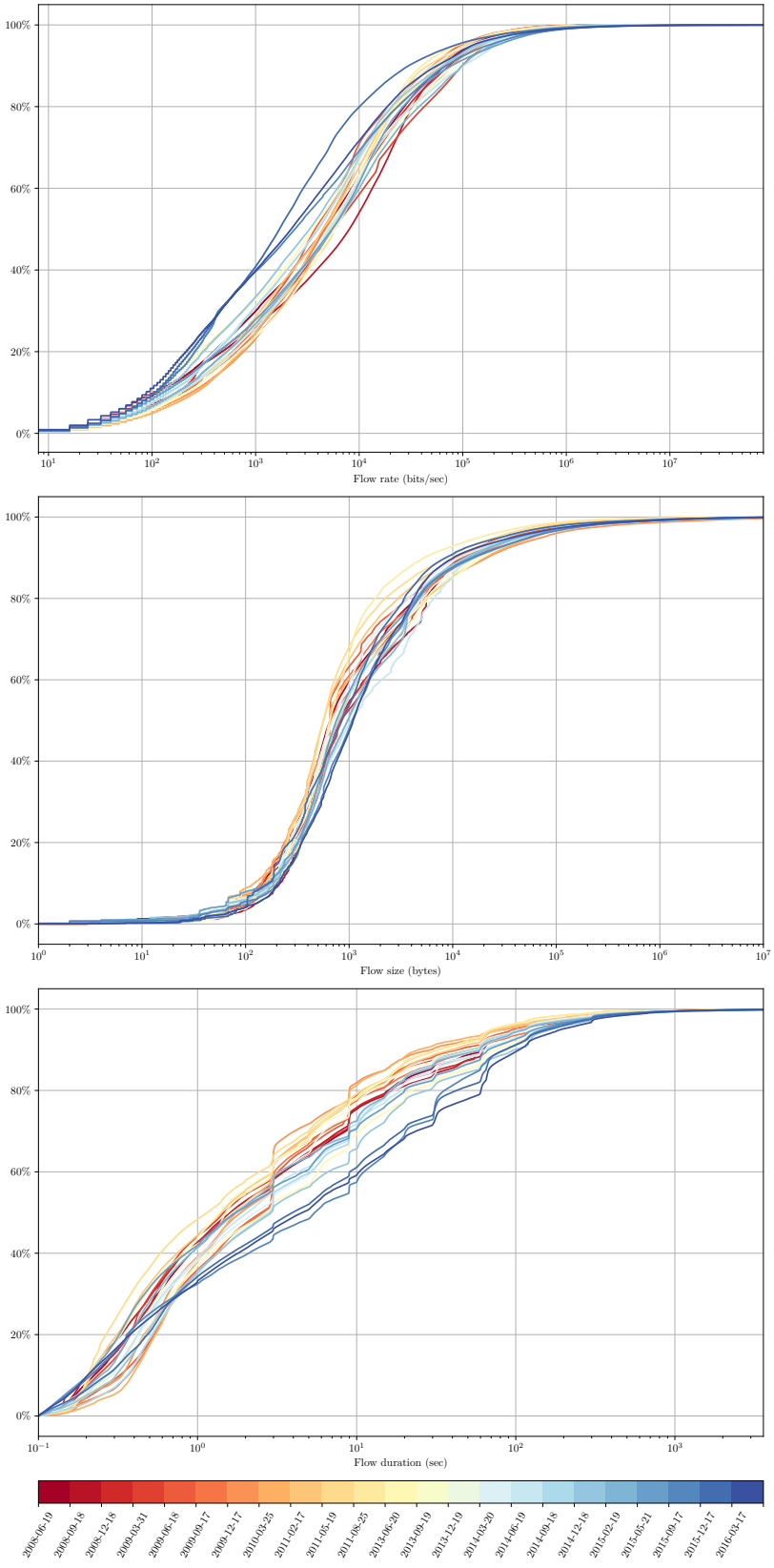


Figure 3.3: Cumulative distributions of flow characteristics for the Chicago data set.

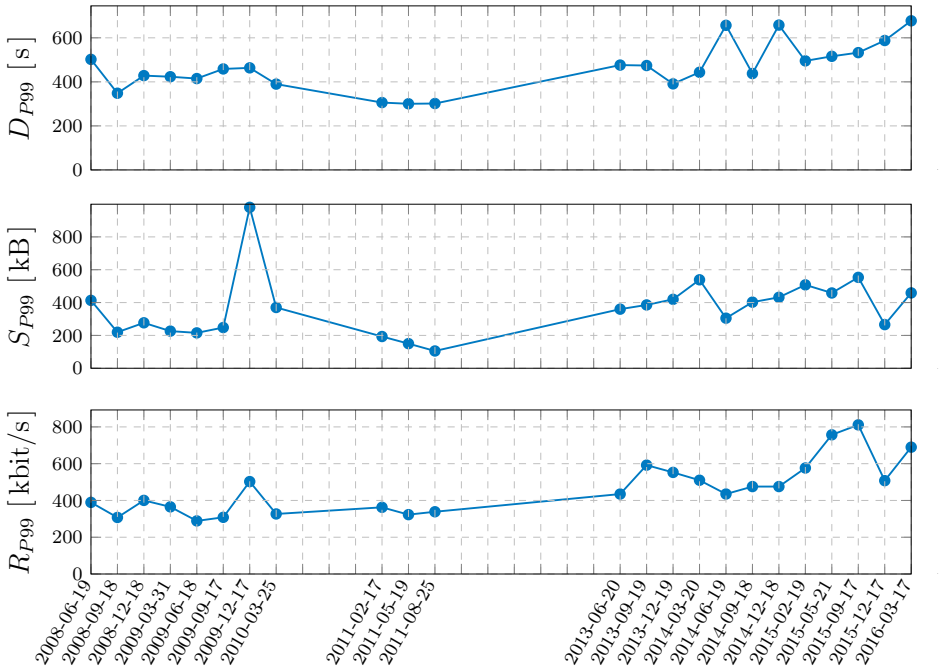


Figure 3.4: 99th percentiles of flow characteristics.

Note that these 20 % of TCP bytes are transmitted by only 0.009 % of TCP flows. For the intersection of D_{P99} and R_{P99}, we measure the same values as for the intersection of all three percentiles. This observation indicates that the longest and fastest flows are also part of the 1 % of the biggest flows. The difference between bytes transmitted by R_{P99} and $S_{P99} \cap R_{P99}$ is relatively small. This observation implies that flows in R_{P99} are mostly also part of S_{P99}. At the same time, the share of flows in $S_{P99} \cap R_{P99}$ is only a third of the share by R_{P99} while both sets nearly transmit the same amount of data. A similar pattern applies for D_{P99} and $D_{P99} \cap S_{P99}$. Considering such observations, our analysis points out the relevance of small subsets of analyzed flows regarding the total traffic volume.

	Chicago		New York	
Flow set	Share	Bytes	Share	Bytes
D_{P99}	1.000 %	40.5 %	1.000 %	43.1 %
S_{P99}	1.000 %	89.2 %	1.000 %	88.4 %
R_{P99}	1.000 %	55.9 %	1.000 %	68.0 %
$D_{P99} \cap S_{P99}$	0.185 %	39.9 %	0.142 %	42.6 %
$D_{P99} \cap R_{P99}$	0.009 %	19.9 %	0.005 %	31.4 %
$S_{P99} \cap R_{P99}$	0.337 %	54.8 %	0.332 %	67.2 %
$D_{P99} \cap S_{P99} \cap R_{99}$	0.009 %	19.9 %	0.005 %	31.4 %

Table 3.1: Relevance of intersections of 99th percentiles.

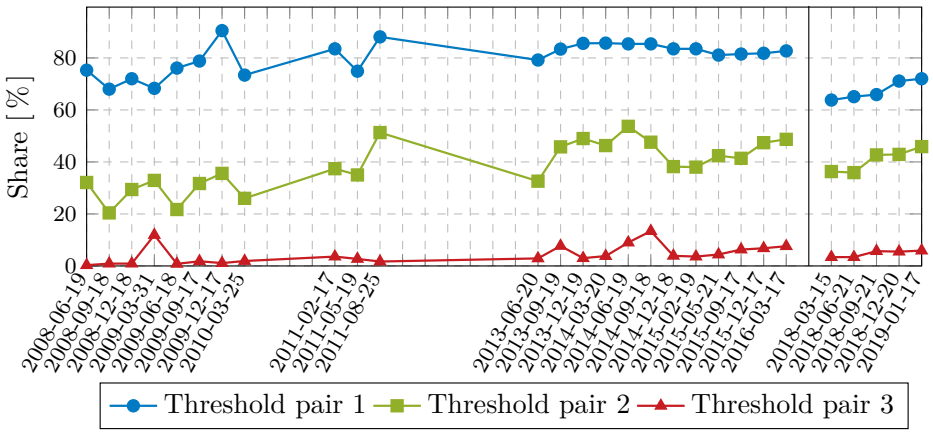


Figure 3.5: Share of bytes transmitted by big-fast flows.

Relevance of Big-Fast Flows: Next, we apply a two-two taxonomy according to Zhang et al. [2] to assess the relevance of so-called big-fast flows and corresponding transmitted bytes. The taxonomy is based on two thresholds regarding flow size and flow rate and classifies flows into four groups, considering small or big flow sizes and slow or fast flow rates. We select three pairs of thresholds. First, we rely on the thresholds chosen by Zhang et al. [2], who use 100 kB as a threshold between small and big flows and 10 kB/s as a threshold between slow and fast flows. We find that the share of big-fast flows is below 2 % for all traces. The most common flow type is small-slow which represent 90 % to 95 % of flows. As shown in Figure 3.5, big-fast flows carry between 70 % up to over 80 % of bytes transmitted by TCP. Analysis over time shows a slight increase in such byte shares between 2008 and 2013, as shown in Figure 3.5. The increase of bytes by big-fast flows correlates to a decrease of bytes by big-slow flows. For early traces, around 20 % of TCP bytes are transmitted by big-slow flows, decreasing to around 10 % for more recent traces. This finding indicates that the rates of big flows increase with time, which we also observe for the 99th percentile of TCP flow rates. As the second pair of thresholds, we select significantly larger values to survey the relevance of very big and very fast flows. We set the threshold regarding size to 1 MB and the threshold for flow rates to 100 kB/s. The shares of bytes transferred by very big and very fast TCP flows are significantly smaller than bytes by big-fast flows. However, such shares show a significant increase across all traces of the data set from around 30 % up to over 50 %. Last, we apply even larger thresholds, i.e., 10 MB for flow size and 1 MB/s for flow rates. We only find small shares of bytes transmitted by these extremely big and fast flows, while we also observe a slight increase of bytes transmitted by such flows. Note that the shares of big-fast flows for the second and third threshold sets are by far smaller than 0.5 %.

Correlations: Zhang et al. [2] restrict correlation analysis to longer flows, i.e., flows longer than 1 s, 5 s, and 30 s, and observe slight differences between different flow lengths. We apply the same filters to our data set to compare results while we describe results for flows longer than 5 s in the following. We observe that correlation coefficients do not indicate specific trends over time. Therefore, we calculate the average of correlation coefficients and the standard

Traces	Corr.	5 seconds Avg.	5 seconds Std.	30 seconds Avg.	30 seconds Std.
Chicago	D & R	-0.0943	0.116	-0.1058	0.075
	D & S	0.2947	0.117	0.1798	0.092
	S & R	0.8847	0.0169	0.8783	0.014
New York	D & R	-0.2143	0.035	-0.2501	0.0502
	D & S	0.2878	0.059	0.1979	0.0557
	S & R	0.8311	0.044	0.8044	0.0509

Table 3.2: Correlations of flow characteristics.

deviation as a measure of dispersion. On average, we find a weak negative correlation between duration and rate (avg.: -0.09) and a weak positive correlation between duration and size (avg.: 0.29). We measure a relatively large standard deviation for both combinations. We find a very strong correlation between size and rate, with an average coefficient of 0.88 and a slight standard deviation. Such a strong correlation of size and rate was observed by earlier studies of correlations between flow characteristics [2], [5]. We find a slightly less significant correlation between duration and size, respectively, size and rate, for flows longer than 30 s, as shown in Table 3.2.

3.2.3 Comparison between CAIDA Data Sets

As our data set for long-term analysis of flow characteristics ends in 2016, we compare our findings to more recent traces taken in 2018 and 2019 as described in Section 3.2.1. In the following, we focus on the differences and similarities between the findings for both data sets. CDFs of measured flow characteristics show similar distribution patterns between both data sets. CDFs for New York traces indicate larger shares of slower TCP flows and larger shares of small TCP flows. This observation is confirmed by the 99th percentiles of size and rate, which are nearly a magnitude smaller than the 99th percentiles measured for the Chicago data set. We observe more bytes transmitted by R_{P99}. For example, flows in R_{P99} for the more recent data set, on average, transmit 68.0% of bytes (55.9% for Chicago traces). Flows in the intersection of the 99th percentiles of all characteristics averagely transmit 31.4% of TCP bytes, while the intersection only includes 0.005% of TCP flows. Further relations between intersections observed for the Chicago data set also apply to results for the more recent New York data set, as shown in Table 3.1. Regarding the share of data transmitted by big-fast flows, we find slightly smaller shares for traces taken in New York than for the more recent traces within the Chicago data set. Average correlation coefficients only show minor differences between both data sets. The correlation between the size and rate of TCP flows decreases from 0.88 to 0.83 for flows longer than 5 s and from 0.87 to 0.80 for flows longer than 30 s. Further correlation coefficients of the New York data set are almost within the range of one standard deviation measured for Chicago traces.

		All TCP		D ≥ 100 ms				RCA applied			
Year	Traces	Flows	Bytes	Flows	Bytes	Flows	Bytes	Flows	Bytes	Flows	Bytes
		[$\times 10^9$]	[TB]	[$\times 10^6$]	[TB]	[%]	[%]	[$\times 10^6$]	[TB]	[%]	[%]
2016	360	1.12	9.45	60.86	9.44	6.40	99.89	0.38	3.48	0.04	35.50
2017	360	1.35	11.73	56.55	11.72	4.27	99.90	0.40	5.25	0.03	43.93
2018	361	2.05	6.78	58.85	6.77	2.87	99.90	0.20	2.74	0.01	37.29
2019	364	3.07	7.49	88.43	7.49	2.89	99.90	0.38	2.67	0.01	35.33
2020	365	3.92	5.99	103.24	5.98	2.82	99.89	0.11	1.48	0.00	25.72
2021	350	3.68	6.12	148.83	6.08	4.42	99.29	0.18	2.14	0.01	30.49
2022	322	4.70	6.23	138.92	6.21	4.12	99.55	0.16	1.59	0.00	24.62
2023	159	1.79	2.97	68.98	2.96	4.27	99.63	0.06	0.61	0.00	20.07
Total	2641	21.66	56.75	724.66	56.64	3.98	99.76	1.87	19.97	0.01	32.60

Table 3.3: Summary of analyzed TCP traffic and impacts of flow filtering.

3.3 Large-Scale Analysis of TCP Throughput

This section surveys the evolution of throughput and throughput limitations of TCP traffic based on several billion TCP connections captured by MAWI between 2016 and 2023 on the Internet, i.e., on a transit link in Japan. Further, this section studies the significance of heavy hitters and the usage of different TCP options in the analyzed data set consisting of over 2600 packet captures.

3.3.1 Data Set

For our study, we rely on traffic captures provided by the MAWI Working Group Traffic Archive [56], in particular, traces captured at MAWI's Samplepoint-F. Samplepoint-F monitors a 1 Gbit/s transit link in Tokyo and provides 15 minutes of captured traffic each day. To ensure privacy, MAWI anonymizes IP addresses of captured traffic in a prefix-preserving manner. In total, we consider 2641 traces containing 21.66 billion TCP connections carrying over 56 TB of payload data, captured from the beginning of 2016 until 9 June 2023. Table 3.3 lists the number of traces per year, the number of TCP flows, and the sum of bytes transmitted by TCP per year.

Traffic Filtering: We consider two filters for captured connections for our study. First, we filter traffic for connections with a minimum duration of 100 ms, referred to as $D \geq 100ms$. Filtering for a minimum duration, as also applied by related work studying TCP traffic characteristics [2], [22], is purposed to remove connections that only consist of a few packets, which potentially screw up metric calculations like throughput. This first filter set is used to study flow characteristics, option usage, and throughput of TCP connections. The second filter set only considers connections successfully analyzed by TCP RCA, i.e., connections providing a complete TCP handshake, consisting of at least 300 packets, and supporting TCP timestamps.

Filtering considered MAWI traces for a minimum duration removes over 95% of connections, while the remaining connections transmit over 99% of observed payload data. As shown in Table 3.3, the remaining 724.66 million flows still carry 56.64 TB of payload data. This observation

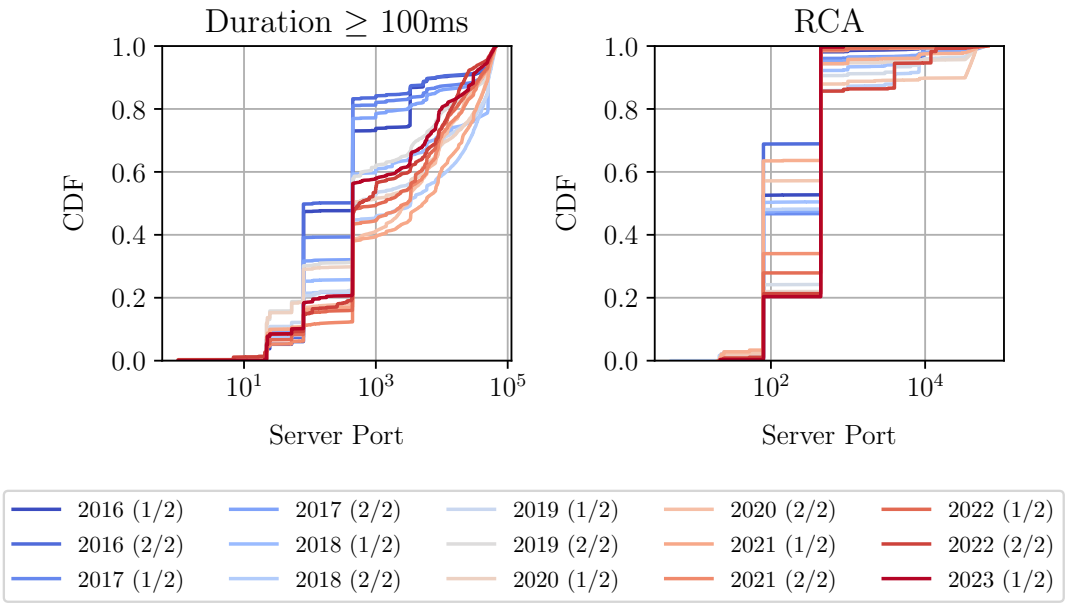


Figure 3.6: Distribution of observed server ports for connections filtered for minimum duration and flows considered for throughput limitation analysis (RCA).

implies that such filtering mainly affects very small flows and that a small share of connections transmits most payload data. Filtering the data set for connections successfully considered by root cause analysis results in 1.87 million (0.01%) flows transmitting 19.97 TB (32.60%) of payload data.

Data per Connection Direction: For the analysis of throughput limitations, we analyze both directions of a connection separately, as data might be transferred in both directions. We refer to node A as the sender of the first SYN packet during the TCP handshake, while node B refers to the receiver of such SYN packet. However, we are interested in the significance of bytes transmitted per direction to assess the significance of limitations observed for each direction. Figure 3.7 shows the share of bytes per direction for each trace. Data is uniformed with a sliding window of 14 traces. We find that bytes transmitted from B to A significantly dominate bytes transmitted from A to B despite a period around the end of 2017 and the beginning of 2018. The significance of data transmitted from A to B decreases after 2021 compared to earlier traces. We conclude that data transmitted from B to A is of major importance for assessing the throughput limitations based on byte shares in the analyzed data set. Note that due to the dominance of the B to A direction, we refer to node B as the server and node A as the client for the remainder of this chapter.

Traffic Mix: We survey the distribution of observed server ports to determine the protocols behind analyzed TCP traffic. Figure 3.6 shows the cumulative distribution of observed server ports in both filtered sets of connections aggregated in 6-month intervals. We find that over 50 % of connections filtered for $D \geq 100ms$ are associated with port 80, respectively 443.

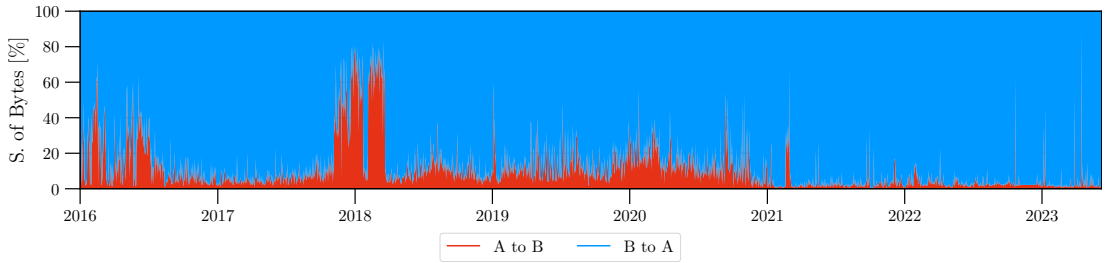


Figure 3.7: Share of bytes per trace observed in both directions of connections. Node A refers to the sender of the first SYN packet during the TCP handshake, while node B is the receiver of the first packet. Data is uniformed with a sliding window of 14 traces.

Year	S_{99}	D_{99}	R_{99}
2016	89.41 %	47.78 %	65.43 %
2017	92.51 %	44.43 %	63.49 %
2018	94.42 %	43.97 %	83.91 %
2019	93.33 %	44.31 %	82.16 %
2020	95.12 %	53.54 %	84.16 %
2021	93.84 %	41.71 %	84.75 %
2022	92.31 %	30.54 %	84.58 %
2023	92.26 %	37.90 %	85.61 %
Total	92.96 %	43.61 %	78.70 %

Table 3.4: Average share of bytes in analyzed MAWI traces transmitted yearly by heavy hitters.

Another frequently observed server port is port 22/SSH. Further, there are significant shares of connections on registered and dynamically used ports. Early traces show a significantly increased share of HTTP traffic, while more recent traces show increasing shares of HTTPS traffic, as also observed by Tsareva et al. [69]. Analyzing the server ports of connections with successfully applied RCA results in over 90 % of traffic on port 80/443 for nearly all aggregated intervals of traces.

3.3.2 Heavy Hitters in the MAWI Data Set

This section is purposed to study the significance of heavy hitter connections, as also conducted on CAIDA traces, as described in Section 3.2. We refer to heavy hitter connections as flows within the 99th percentile of flow size, duration, or rate, as proposed by Lan et al. [5]. Note that percentiles are calculated after filtering for a minimum duration of 100 ms.

We find that in total, over 90 % of transmitted bytes are carried by connections with a flow size larger than the 99th percentile of all flow sizes (S_{P99}), as summarized in Table 3.5. The share of bytes transmitted by heavy hitters regarding flow size, plotted as a timeline in Figure 3.8, is relatively stable. Note that timeline values were uniformly filtered for readability, i.e., we apply a sliding window of 14 traces and calculate the average of the according values. Such filtering weakens the impact of temporal fluctuations and outliers while the curve's general trends remain. The share of bytes carried by connections within D_{P99} shows decreasing relevance of D_{P99}

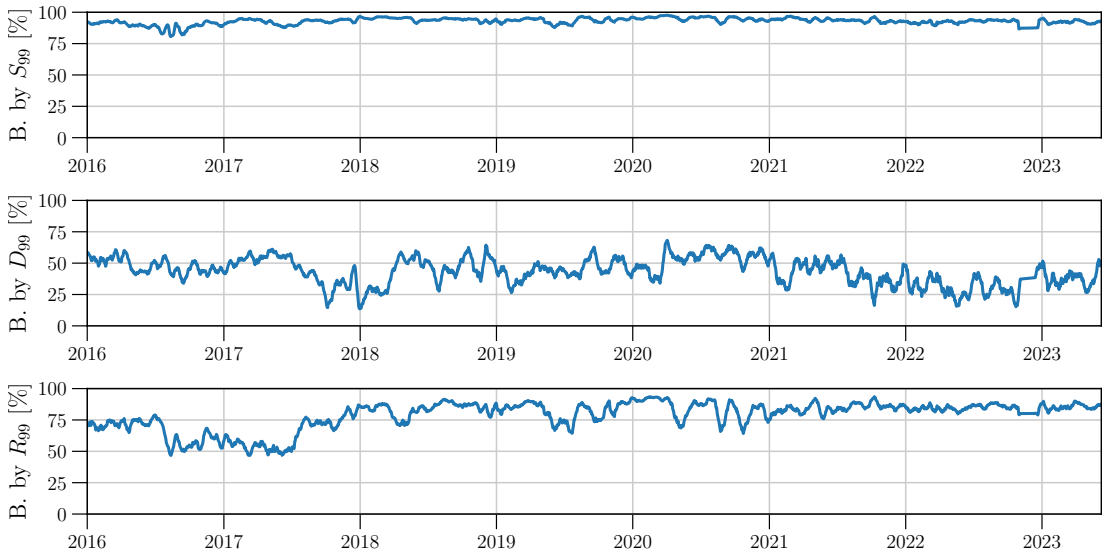


Figure 3.8: Share of bytes in analyzed MAWI traces transmitted by heavy hitter connections over time. Data is uniformed with a sliding window of 14 traces.

connections over time. Since 2018, R_{P99} connections carry between 82% and 86% of observed TCP bytes, implying that most bytes are carried by connections within both S_{P99} and R_{P99}. Such correlation between the biggest and fastest flows was also observed by related work [2], [5] and our study on CAIDA traces, presented in Section 3.2.

3.3.3 Evolution of TCP Option Usage

To assess the usage of TCP options in the analyzed data set, we consider both the share of connections and the share of bytes transmitted by such connections. All four options' shares are summarized in Table 3.5 per year. Note that we only consider connections filtered for $D \geq 100ms$ and providing a complete handshake, which is required to determine supported options.

Throughout the data set, we observe increased shares of flows not supporting TCP timestamps and window scaling between 2020 and 2023 compared to the period between 2016 and 2019. This is not intuitive, assuming increasing adoption of operating systems and, accordingly, clients to TCP options. One explanation for such increasing shares of flows not supporting the options could be scans or other active measurement experiments that rely on fully established connections. However, this reasoning requires further analysis. Bytes transmitted by connections supporting TS show a similar pattern, i.e., besides a peak between 2017 and 2019, the share of bytes is around 80 %. Regarding bytes transmitted by connections using TCP WS, we find an increase from 93.55 % in 2016 to 96.90 % in 2023. This implies that connections not supporting window scaling do not carry significant amounts of data. For instance, 59.96 % of connections that do not use TCP WS only transmit 3.1 % of data in 2023.

Across the data set, only 26.13 % of connections support selective acknowledgments, which carry 37.91 % of totally transmitted bytes. Again, the share of connections using the option is larger between 2016 and 2019 than between 2020 and 2023, as also observed for TS and WS. Regarding

	Timestamps		Windows Scaling		SACK		ECN	
Year	Conn.	Bytes	Conn.	Bytes	Conn.	Bytes	Conn.	Bytes
2016	81.49 %	82.49 %	79.86 %	93.55 %	27.44 %	31.99 %	0.89 %	0.92 %
2017	87.57 %	90.40 %	81.99 %	93.86 %	25.78 %	27.46 %	2.93 %	4.15 %
2018	87.50 %	90.76 %	85.00 %	95.06 %	39.16 %	41.83 %	4.55 %	7.31 %
2019	80.82 %	90.92 %	78.76 %	95.64 %	25.46 %	36.34 %	3.75 %	10.56 %
2020	68.14 %	79.97 %	61.31 %	97.72 %	20.58 %	45.88 %	4.28 %	5.70 %
2021	74.06 %	83.78 %	49.24 %	97.81 %	23.90 %	43.18 %	2.45 %	6.42 %
2022	71.38 %	83.51 %	52.17 %	97.92 %	21.63 %	39.45 %	2.45 %	5.40 %
2023	68.35 %	81.57 %	40.04 %	96.90 %	22.61 %	36.73 %	2.42 %	6.69 %
Total	78.19 %	85.75 %	68.31 %	95.96 %	26.13 %	37.91 %	3.02 %	5.85 %

Table 3.5: Share of connections using TCP options aggregated per year.

ECN, we observe even smaller shares of connections supporting the option. In total, only 3.02 % of connections use ECN, responsible for 5.85 % of transmitted bytes.

Summarizing the state of usage of TCP performance-related TCP options, we observe widespread usage of TCP WS regarding transmitted bytes, while SACK and ECN are not used during the transmission of most bytes.

3.3.4 Evolution of Throughput Rates

To study the evolution of throughput rates across the analyzed data set, we determine the mean, the maximum, and different percentiles of connections' average throughput for each trace. Figure 3.9 shows the evolution of throughput statistics over time with a uniform filter of 14 traces to improve readability. Note that Figure 3.9 only shows maximum throughput for RCA-filtered connections for readability.

Regarding connections filtered for $D \geq 100ms$ we observe uniformed mean throughput between 0.01 Mbit/s and 0.1 Mbit/s. Despite a period with lower values between 2020 and 2021, mean throughput has remained mainly stable over the years. The median of observed throughput rates per trace is significantly smaller than the corresponding means. The timeline shows a significant decrease in median throughput since the middle of 2018. The same observation applies to further considered percentiles. Connections filtered for successfully applied RCA show significantly increased throughput statistics compared to $D \geq 100ms$ filtered connections. The timeline of mean throughput shows mainly values between 1 Mbit/s and 10 Mbit/s, implying an increase between one and two magnitudes compared to mean throughput observed for $D \geq 100ms$. Further, the timeline of mean throughput shows a trend towards larger values over time. Such a trend is also observed for the timelines of analyzed percentiles. The maximum observed throughput rate per trace is around 100 Mbit/s for traces captured between 2016 and 2021. After 2021, maximum throughput per trace regularly exceeds 100 Mbit/s.

To further understand the characteristics of throughput rates in the analyzed data set, we survey the cumulative distribution of throughput for traces aggregated to intervals of 6 months as shown in Figure 3.10. Regarding connections filtered for $D \geq 100ms$, we find significant

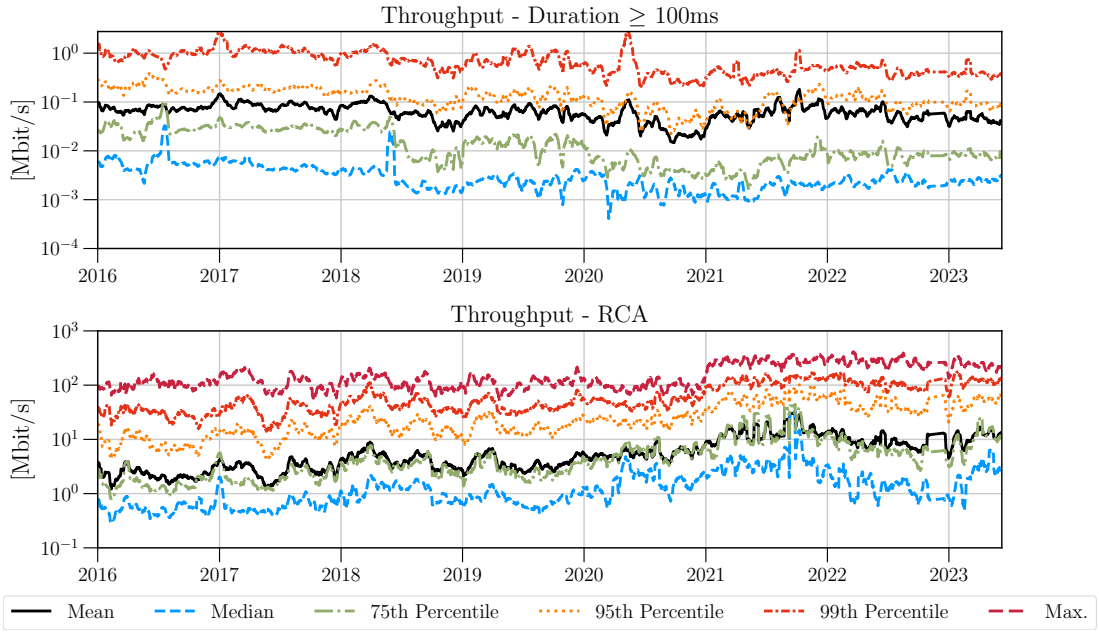


Figure 3.9: Mean, maximum, and different percentiles of mean throughput per connection observed per trace. Data is uniformed with a sliding window of 14 traces.

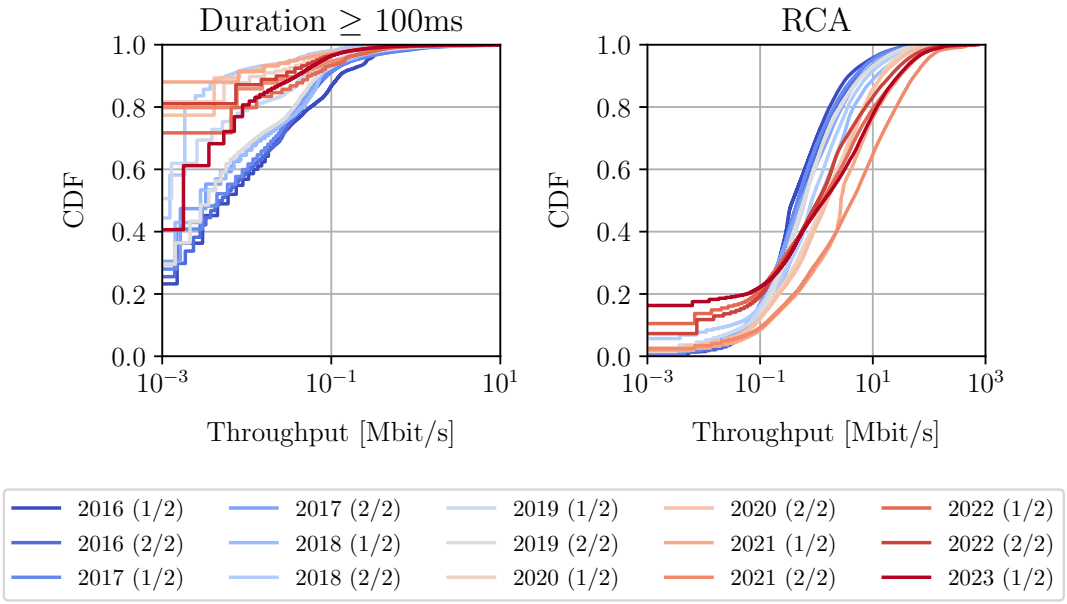


Figure 3.10: Cumulative distribution of mean throughput rates. Traces are aggregated to six-month intervals.

Year	ALP	UB	SB	RL	TL	Unknown
A to B						
2016	98.5 %	0.3 %	0.2 %	0.2 %	0.3 %	0.6 %
2017	98.6 %	0.1 %	0.1 %	0.0 %	0.2 %	0.9 %
2018	96.8 %	0.3 %	0.5 %	0.0 %	0.6 %	1.8 %
2019	98.3 %	0.3 %	0.3 %	0.0 %	0.5 %	0.6 %
2020	98.1 %	0.3 %	0.3 %	0.1 %	0.4 %	0.7 %
2021	98.9 %	0.1 %	0.1 %	0.1 %	0.5 %	0.4 %
2022	99.0 %	0.2 %	0.2 %	0.0 %	0.4 %	0.2 %
2023	96.7 %	0.1 %	0.3 %	0.0 %	2.0 %	0.9 %
Total	98.2 %	0.2 %	0.2 %	0.1 %	0.5 %	0.8 %
B to A						
2016	33.2 %	1.8 %	47.7 %	0.1 %	10.3 %	7.0 %
2017	21.1 %	1.8 %	65.4 %	0.1 %	6.1 %	5.6 %
2018	44.1 %	4.7 %	27.4 %	0.1 %	14.9 %	8.7 %
2019	50.8 %	4.9 %	21.2 %	0.1 %	16.5 %	6.5 %
2020	41.0 %	8.0 %	22.3 %	0.1 %	17.6 %	11.0 %
2021	56.6 %	5.0 %	8.2 %	0.1 %	22.4 %	7.7 %
2022	59.1 %	3.7 %	16.9 %	0.1 %	11.5 %	8.7 %
2023	59.0 %	7.9 %	12.0 %	0.0 %	14.6 %	6.4 %
Total	44.4 %	4.5 %	29.0 %	0.1 %	14.2 %	7.8 %

Table 3.6: Share of connection duration per throughput limitation per year.

shares of connections with a mean throughput smaller than 1 kbit/s, as also observed for the timeline of different percentiles of throughput rates per traced discussed above and shown in Figure 3.9. As indicated by the timeline, there has been a significant shift towards larger shares of connections with lower throughput for traces captured since the middle of 2018. In contrast, the cumulative distribution of mean throughput of connections successfully analyzed with RCA shows a tendency towards increased shares of larger throughput for more recent traces, despite traces captured since begin of 2022, which show significantly larger shares of connections with throughput smaller than 1 kbit/s.

3.3.5 Evolution of TCP Throughput Limitations

Significance of throughput limitations: To survey the significance of throughput limitations, we calculate the share of transmitted bytes and the share of connection duration limited by each limitation per trace. Table 3.6 shows the mean of trace duration shares per year, and Table 3.7 shows the mean of trace byte shares per year for both directions of connections. Further, Figure 3.11 and Figure 3.12 shows stacked timelines of shares of connection duration and transmitted bytes per connection per trace, uniformed with a sliding window of 14 traces for readability. Note that the remainder of this section focuses on results observed for data transmitted from B to A due to the dominance of data transmitted in such direction, as described in Section 3.3.1

Year	ALP	UB	SB	RL	TL	Unknown
A to B						
2016	43.1 %	11.1 %	1.9 %	5.4 %	11.9 %	26.6 %
2017	50.4 %	7.5 %	2.3 %	0.8 %	9.5 %	29.4 %
2018	45.3 %	11.6 %	4.7 %	0.5 %	6.9 %	31.0 %
2019	50.8 %	13.3 %	3.9 %	0.3 %	11.7 %	20.0 %
2020	44.5 %	19.6 %	3.3 %	4.4 %	7.5 %	20.7 %
2021	55.0 %	5.4 %	1.7 %	4.3 %	10.9 %	22.8 %
2022	60.7 %	8.0 %	4.0 %	1.9 %	8.2 %	17.2 %
2023	63.3 %	6.2 %	3.7 %	0.5 %	11.1 %	15.2 %
Total	50.6 %	10.7 %	3.2 %	2.4 %	9.6 %	23.5 %
B to A						
2016	12.0 %	5.6 %	31.5 %	0.1 %	25.3 %	25.5 %
2017	9.9 %	6.2 %	42.3 %	0.1 %	16.5 %	24.9 %
2018	4.9 %	20.3 %	18.8 %	0.1 %	22.5 %	33.4 %
2019	16.4 %	20.4 %	12.9 %	0.1 %	21.0 %	29.3 %
2020	2.5 %	19.6 %	11.4 %	0.1 %	23.2 %	43.2 %
2021	16.0 %	20.3 %	5.9 %	0.1 %	28.6 %	29.1 %
2022	7.2 %	19.0 %	15.3 %	0.1 %	20.9 %	37.5 %
2023	1.4 %	39.2 %	10.8 %	0.1 %	19.9 %	28.7 %
Total	9.3 %	17.3 %	19.3 %	0.1 %	22.4 %	31.6 %
Both directions						
2016	13.7 %	6.6 %	28.2 %	1.4 %	23.1 %	27.0 %
2017	12.2 %	5.7 %	37.9 %	0.1 %	15.8 %	28.2 %
2018	8.8 %	19.4 %	15.7 %	0.1 %	19.1 %	37.0 %
2019	20.7 %	19.6 %	11.7 %	0.1 %	20.2 %	27.8 %
2020	8.2 %	20.1 %	10.3 %	0.4 %	20.9 %	40.1 %
2021	16.3 %	19.8 %	5.7 %	0.3 %	27.9 %	29.9 %
2022	8.5 %	18.8 %	15.0 %	0.2 %	20.5 %	37.0 %
2023	2.3 %	38.7 %	10.5 %	0.1 %	19.7 %	28.7 %
Total	12.1 %	17.1 %	17.4 %	0.4 %	21.0 %	32.1 %

Table 3.7: Share of bytes per throughput limitation per year.

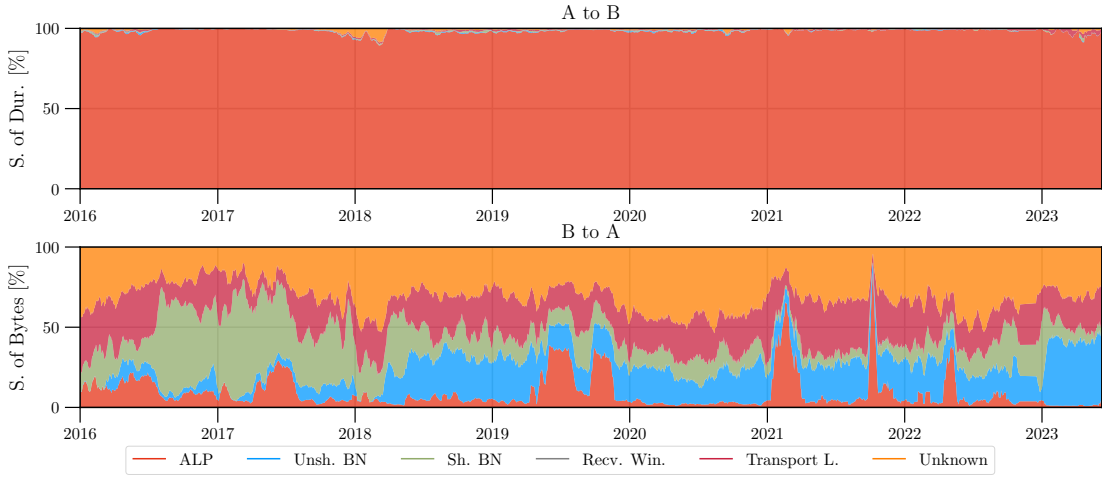


Figure 3.11: Shares of connection duration per trace. Data is uniformed with a sliding window of 14 traces.

Regarding the share of connection duration, application limitation (ALP) is the most significant limitation. In total, 45.4 % of aggregated connection duration is limited by the sending application, i.e., periods with decreased packet sizes or long idle times, e.g., by persistent HTTP(S) connections. We find increasing shares of connection duration limited by ALPs over time, starting from 33.2 % in 2016 up to 59.0 % in 2023. Across the data set, 29.0 % of aggregated connection duration is spent during shared bottleneck (SB) limited periods, followed by 14.2 % limited by the transport layer (TL). Unshared bottlenecks (UB) and unknown limitations are detected for 4.5 %, respectively 7.8 %, of total connection duration. Receiver window limitations (RL) are limiting 0.1 % of connection duration across the data set.

Regarding bytes transmitted during periods of a specific limitation, we do not find such significant shares of application limitation as detected for connection duration. In total, 9.3 % of bytes are transmitted during periods limited by the sending application. This can be traced back to decreased throughput during such periods as the sending application not providing an ongoing stream of data to send. Instead, we observe most bytes limited by unknown limitations responsible for transmitting 31.6 % of all transmitted bytes. Comparing the significance of unknown limitations regarding connection duration (7.8 %) and transmitted bytes (31.6 %) indicates that such periods transport a significant amount of data in short duration shares, implying increased transmission rates compared to other limitations. Further, we find 22.4 % of bytes limited by transport layer limitations and 19.3 % of bytes limited by shared bottleneck limitations across the data set. Unshared bottleneck limitations are limiting 17.3 % of transmitted bytes. As shown in Table 3.7, 2023 shows a significantly increased share of unshared bottleneck limited bytes (39.2 %) compared to previous years. The receiver window only limits 0.1 % of bytes across the data set.

Share of bytes per connection: Further, we are interested in the share of connections showing a particular limitation and how many bytes are affected by such limitation per individual connection. Figure 3.13 shows the cumulative distribution of the shares of bytes limited by each

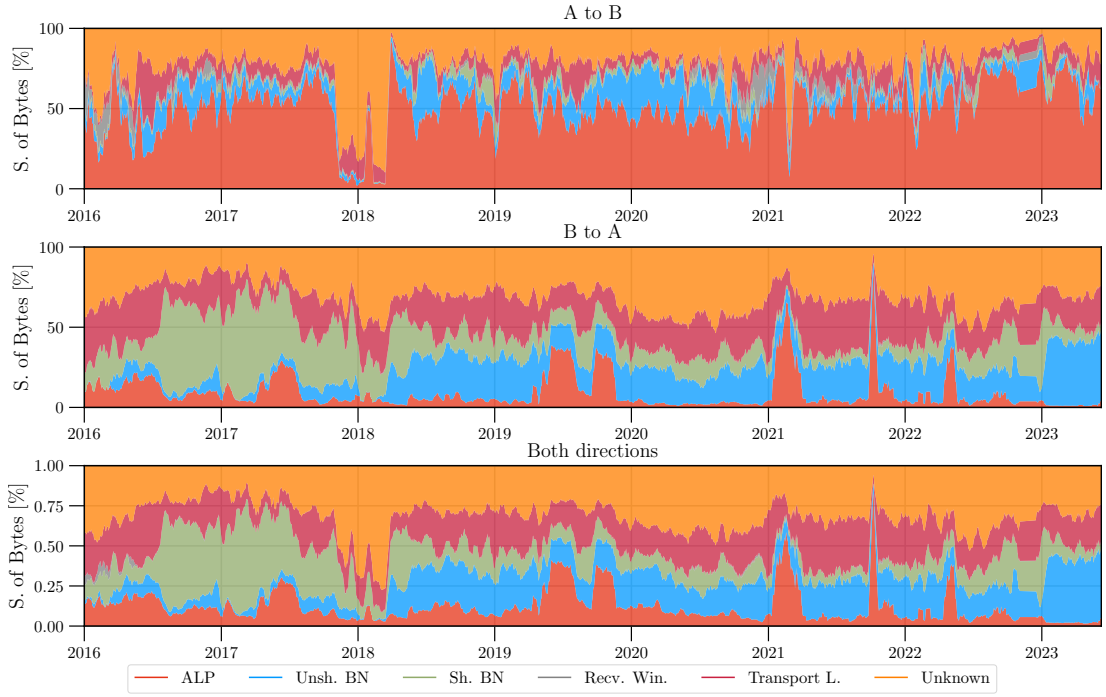


Figure 3.12: Shares of transmitted bytes per trace. Data is uniformed with a sliding window of 14 traces.

limitation per connection observed in the B to A direction aggregated in 6-month intervals. Note that the lower bound of the y-axes of CDFs is cut at different shares to increase readability.

All connections considered by RCA at least include one application limited period, as the Isolate algorithm [35] begins a connection within an ALP. Accordingly, all connections show some bytes transmitted during ALPs. Further, we observe significant clusters of connections with less than 5 % of data transmitted during ALPs. Observed shares vary between aggregated trace intervals. A second cluster is observed for connections that transmit nearly all data within ALPs, corresponding to a share of bytes limited by ALPs near 1.

We find increasing shares of connections showing unshared bottleneck limited bytes over time, i.e., the share of connections with zero bytes during UBs decreases. For early trace intervals, we find around 90 % of connections without UB limited bytes, between 90 % and 70 % for intervals between 2019 and 2022, and 60 % for traces captured in 2023. At the same time, traces captured in 2023 show the maximum share (>20 %) of connections with more than 90 % of transmitted bytes limited by unshared bottlenecks.

Regarding shared bottleneck limited bytes, we find that most connections do not suffer from any shared bottleneck limitation and observe clusters near bytes shares equal to 1. The significance of SB-limited byte shares varies over time without a clear trend. However, we observe the increased significance of shared bottlenecks between the middle of 2016 and the end of 2017. Receiver window limitations only occur for a very small share of connections. This observation corresponds to the small shares of RW limitation regarding total duration and total bytes, as

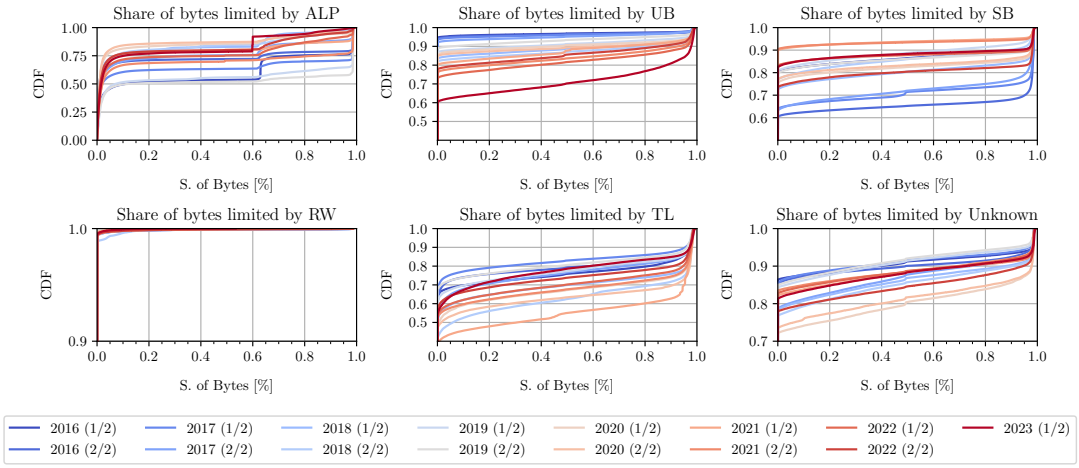


Figure 3.13: Cumulative distributions of the shares of bytes transmitted form node B to node A during a certain throughput limitation per connection. Traces are aggregated to six-month intervals.

described above. The share of bytes limited by the transport layer shows significant clusters near zero and for shares larger than 95 % of bytes. In particular, between 40 % and 70 % of connections do not show any TL limitation.

Regarding bytes limited by mixed periods, respectively by periods with unknown throughput limitation, we observe byte shares equal to zero for more than 70 % of connections. However, significant shares of connections transmit nearly all bytes during unknown periods, motivating further analysis of characteristics of such connections regarding end-to-end capacity, receiver window utilization, and packet loss.

3.4 Comparison with Measurement Results by Related Work

This section is purposed to compare presented measurement results to related studies on TCP flow characteristics and throughput limitations. Note that details on analysis approaches and applied methodologies for result presentation by related work differ and are presented in detail in Chapter 2.

Flow rates were analyzed by Zhang et al. based on traffic captured between 2001 and 2002 [2] and by Lan et al. based on traffic captured in 2003 [5]. Zhang et al. observe 99th percentiles of flow rates between 10 kbit/s and 100 kbit/s depending on the analyzed vantage point, while Lan et al. report flow rates smaller than 50 kbit/s. For comparison, we observe 99th percentiles of several hundreds kbit/s for the analyzed CAIDA and MAWI traces. When only considering MAWI traffic successfully analyzed by RCA, our measurements result in 99th percentiles of flow rate larger than 100 Mbit/s for most traces captured since 2021. Regarding the correlation of flow characteristics, Zhang et al. and Lan et al. report mostly similar correlation coefficients with deviations for different traffic captures. Our analysis of correlations between flow characteristics

conducted on CAIDA traces confirms a strong positive correlation between flow size and flow rate as observed by related work [2], [5], while we observe a slightly less significant negative correlation of flow duration and flow rate for CAIDA traces captured in Chicago, which is also observed for one of two data sets considered by Lan et al. [5].

Regarding the results of TCP root cause analysis, we find similarities and differences between related work and our measurement results, which are elaborated in the following. However, considering that most studies were conducted considering different root causes and according to heuristics to detect them and the heterogeneity of used data sets, further studies on TCP throughput limitations comparing results received with different approaches applied on the same data set as well as the analysis of different data sets with the same tool are of interest for future work. Note that we reference measurement results observed in our study's B to A direction in the following.

Zhang et al. [2] report that between 22 % and 43 % of data is limited by network congestion similarly defined as shared bottlenecks in our measurement study which we detect to limit 17.4 % of data across the MAWI data set. Comparable rates for shared bottleneck limitations were reported by Zhang et al. [22] applying the RCA approach introduced by Siekkinen et al. [11] as also used for our study. Further, Zhang et al. [2] observe between 8 % and 43 % of bytes to be limited by either the receiver or the sender window, i.e., the TCP send buffer. While we do not consider TCP send buffer limitations, we only observe negligible shares of data limited by the receiver window in the MAWI data set. This observation resembles the observations by Zhang et al. [22] that do not observe receiver window limitations in their data sets. Larger shares of receiver window limited bytes are reported by Timmer et al. [10], i.e., between 9.6 % and 16.3 % for different considered traces. Zhang et al. [2] also consider opportunity limitation, i.e., short connections that are assumed to never leave the slow start phase, which is detected for 90 % of analyzed TCP connections. While we require a significantly higher minimum count of packets for conducting RCA, the impact by filtering flows for a minimum duration of 100 ms shows that the vast majority of TCP flows on the Internet tend to be very short.

Timmer et al. [10] report that between 33.3 % and 38.7 % of bytes in their data sets are limited by the network. We observe a similar share (36.6 %) of network limitations, i.e., unshared and shared bottleneck limitations, across the MAWI data set. Further Timmer er al. [10] determine the sending application to limit between 22 % and 34.8 % of bytes. Araújo et al. [12] report between 36.78 % and 49.55 % of bytes to be limited by by the application. In contrast, we observe fewer bytes limited by ALPs (9.3 %). Further, Araújo et al. [12] observe that in 2007 68.6 % of bytes are not limited by the network, respectively congestion control, a share that is significantly larger compared to our measurements as we find more than 50 % of data to be limited either by unshared bottlenecks, shared bottlenecks, or the transport layer.

3.5 Summary

This chapter extensively studied TCP flow characteristics, throughput, and throughput limitations on the Internet.

We conduct passive measurements based on Internet traffic captured between 2008 and 2019 by CAIDA consisting of over 2.5 billion TCP flows [55]. Our measurements show that the 99th percentiles of TCP flow duration and TCP flow rate show significant change between 2008 and 2016, while bytes transmitted by different intersections between 99th percentiles show significant relevance of such heavy hitters. The share of bytes transmitted by big-fast flows and very big-very fast flows increases over time, while the share of such flows remains small. The observed correlation between flow characteristics stays constant over time and confirms findings from related work regarding a strong correlation between flow size and flow rate [2], [5].

Further, we conduct a large-scale study of TCP throughput, throughput limitations of TCP connections, and TCP option usage based on over 2600 traces of captured Internet traffic consisting of 21.66 billion TCP connections provided by MAWI [56]. Filtering connections for a minimum duration of 100 ms removes the vast majority of connections, while the remaining connections still carry over 99 % of totally observed bytes. Considering requirements for passively applied root cause analysis (TCP timestamps, captured handshake, minimum packet count) amplifies this observation to a remainder of 0.01 % of connections carrying 32.60 % of data.

We observe that TCP timestamps and TCP window scaling are used for the transmission of the majority of bytes, while selective acknowledgments and ECN are not commonly used. While different statistics like mean, maximum, and different percentiles of throughput do not show increasing rates for connections filtered for a minimum duration of 100 ms, connections successfully analyzed with RCA indicate increasing transmission rates over time.

We observe application limitations and shared bottleneck limitations as the most significant limitations regarding connection duration, while unknown, respectively mixed, throughput limitations and the transport layer are responsible for limiting the largest share of bytes. Surveying the share of each limitation for individual connections shows large shares of connections not suffering from a particular limitation, while we observe connections limited by a particular root cause for nearly all transmitted bytes.

3.6 Author's Contribution

This chapter builds on the publication "On the Evolution of Internet Flow Characteristics" [59], which is a joint work by the author of this thesis, Benedikt Jaeger, Fabian Helfert, Philippe Barias, and Georg Carle. The author contributed significantly to the publication regarding research objectives and applied methodologies. Further, the author conducted the final evaluation of measurement results and composed contents for the named publication.

To analyze the large-scale data sets, a traffic analysis tool implemented in the scope of the Master's thesis by Fabian Helfert [57] was used. The first version of Helfert's traffic analysis tool was extended in the scope of the Master's thesis by Philippe Barias [58]. The data set of extracted flow characteristics based on considered CAIDA traces was generated by Philippe Barias. The author of this thesis applied filtering of flows and, as described above, evaluated such data according to the introduced methodology for publication.

Packet capture analysis for the study on throughput, throughput limitations, and option usage was conducted with an adapted version of the mentioned flow analysis tool, implemented in the scope of the Master's thesis by Simon Hülkenberg [60]. The author supervised all named theses, while the latter two theses closely follow research objectives specified by the author.

The mentioned traffic analysis tool for the study on throughput, throughput limitations, and option usage was adapted by the author of this thesis, among others, regarding the identification of application limited periods, the calculation of time series of advertised receiver windows and outstanding bytes, the correct use of configured threshold parameters, and regarding the extraction of ECN and SACK usage. Listed adaptations change measurement results significantly.

Based on the adapted traffic analysis tool, the author conducted all measurements based on MAWI traces and implemented the final querying and analysis of extracted data for this thesis.

Chapter 4

Online Monitoring of TCP Throughput Limitations

Providing reliable and fast data transmission is of primary interest to service providers and data center operators. As the throughput of a connection influences user experience, service providers need to understand the causes behind observed transmission rates. Understanding and analyzing such causes allows network infrastructure, operating systems, and application optimizations and helps detect network incidents or anomalies.

As elaborated in Chapter 2 different approaches to the root cause analysis of TCP throughput are presented in related work. However, only Ghasemi et al. [4] and Sun et al. [23] consider online root cause analysis, respectively, introduce online analysis capable RCA approaches. Thereby, Ghasemi et al. [4] focus on integrating TCP performance analysis into switches' data plane, while the approach by Sun et al. [23] requires access to the sender's network stack. Accordingly, no approach to passive online TCP throughput limitation monitoring on commodity server hardware independent of the measurement position was introduced by related work.

Considering the use of online TCP root cause analysis in productive environments, one encounters challenges regarding bandwidths beyond 1 Gbit/s commonly deployed in data centers nowadays. Such bandwidths imply high requirements for the performance of monitoring tools. At the same time, modern commodity hardware provides increasing computational power and high parallelization capabilities, while traffic, respectively, packet processing on commodity hardware was massively accelerated by means of user space packet processing, as implemented by the data plane development kit (DPDK) [70].

These circumstances motivate surveying the capabilities of efficient and scalable online monitoring of TCP throughput limitations, which is addressed in this chapter. Thereby, the key contributions of this chapter are: First, introducing and evaluating an approach to the online analysis of TCP throughput limitations based on the offline RCA approach introduced by Siekkinen et al. [11]. Second, a performance study of a scalable monitoring tool for TCP throughput

limitations, enabling the online analysis of throughput limitations considering several Gbit/s of workload on commodity hardware.

The contents of this chapter are based on the following publications:

> **Online Monitoring of TCP Throughput Limitations**,
> *Simon Bauer, Kilian Holzinger, Benedikt Jaeger, Paul Emmerich, Georg Carle*,
> IEEE/IFIP NOMS 2020, Budapest, Hungary (Virtual Conference), April 2020
>
> **Scalable TCP Throughput Limitation Monitoring**,
> *Simon Bauer, Florian Wiedner, Benedikt Jaeger, Paul Emmerich, Georg Carle*
> IEEE/IFIP IM 2021, Bordeaux, France (Virtual Conference), May 2021

For this thesis, the contents of the name publications have been recomposed and partly shortened to focus on the introduced measurement method and the performance of a corresponding proof of concept implementation. Thereby, this chapter is intended to demonstrate the usability of online root cause analysis in productive environments with bandwidths of several Gbit/s.
Section 4.1.1 and Section 4.1.2, describing the introduced measurement method for online analysis of TCP throughput limitations and design decisions for the proof of concept implementation, is based on the publication "Online Monitoring of TCP Throughput Limitations" [71]. The same applies to the contents of Section 4.2.1 describing findings of a performance evaluation conducted on an implementation relying on only one analysis pipeline. Section 4.1.3, describing the parallelization of analysis pipelines for scalability, and Section 4.2.2, evaluating performance with multiple analysis pipelines, are based on the publication "Scalable TCP Throughput Limitation Monitoring" [72].

4.1 Online TCP Root Cause Analysis

4.1.1 Measurement Method

To enable online root cause analysis, we rely on the approach introduced by Siekkinen et al. [11] due to its passive RCA capabilities, independence of the measurement position, and considered root causes. As described in Chapter 2, the TCP root cause analysis approach presented by Siekkinnen et al. [11] applies segmentation of a connection into different limitation periods, which are further classified in a second analysis iteration based on so-called limitation scores and a threshold-based decision tree. However, relying on such two analysis iterations is not suitable for online analysis, as the whole connection is required for the used *Merge* algorithm. Therefore, we decide to not merge isolated transfer periods. Further, we decide to conduct root cause analysis on sliding windows of TCP packets, respectively, data extracted from TCP packets. Thereby, a sliding window is maintained for each connection. Root cause analysis is then conducted in periodic intervals based on data currently stored in the sliding window. This approach enables *(i)* insights into throughput limitations during a connection is active and *(ii)* to configure the length of periodic intervals, representing a trade-off between the performance of the analyzer and the timeliness of RCA results.

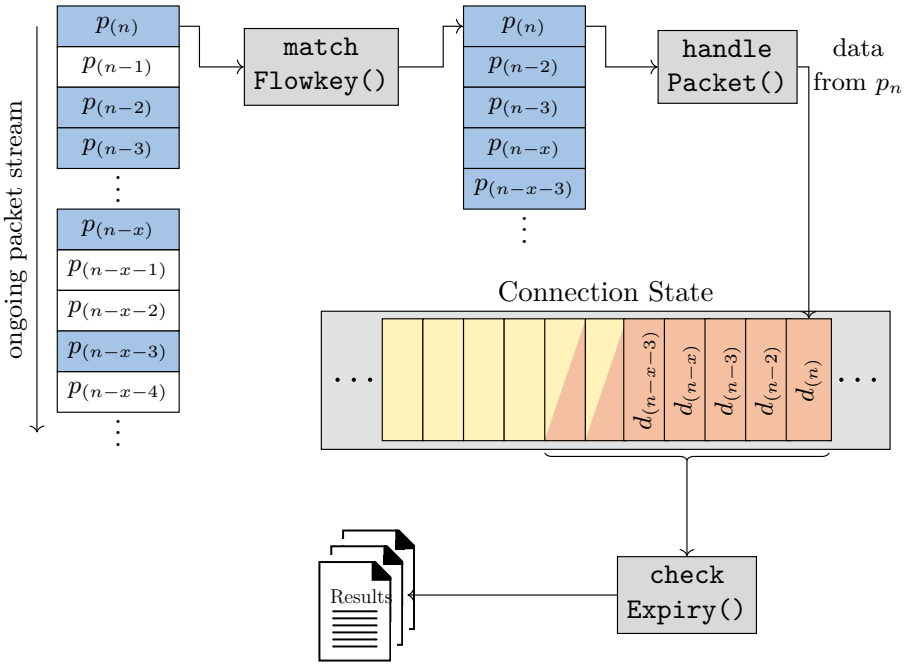


Figure 4.1: Sliding window-based packet processing for monitoring of TCP throughput limitations.

4.1.2 Design and Implementation of Analysis Pipelines

To implement our online throughput limitation analysis tool, we build on the traffic capture tool *FlowScope* [73], [74] due to its efficient packet processing capabilities. *FlowScope* uses a hashtable indexed with `flowkeys` to keep track of flows and to store flow-specific information. We refer to a flow as packets with the same 5-tuple consisting of server and client IP addresses, ports, and the used L4 protocol. From an analysis point of view, we can only consider flows for which we observe the complete TCP three-way handshake. Observing the handshake is required to extract data necessary to run the TCP throughput limitation analysis, such as the MSS, sender and receiver information, and measurement position.

We use static hashtable entry sizes, as dynamically resizing hashtable entries is assumed to be too expensive. While this avoids expensive resizing, the state possibly stored for each flow gets limited. As we require information extracted from both directions of a flow for our analysis, we rely on bidirectional flow tracking. We do this based on the `flowkey`, calculated as a hash of a sorted connection identifier based on the source and destination addresses, respectively ports. To keep the association of the sender, respectively, the receiver, and the corresponding IP address, we store such information in the connection state of each flow.

As mentioned above, we use a fixed connection state size. Therefore, we must choose a sufficiently large hashtable entry size to provide enough space for different data collected over time, which is required for the analysis. We choose a connection state size of 156 kB. Tests show this size to be sufficiently large, while memory requirements are not too extensive. For instance, a connection state size of 156 kB implies an approximated memory need of 160 GB for 1 million

concurrent flows. This relation seems reasonable considering modern commodity-off-the-shelf (COTS) hardware.

We split the implementation of result and KPI calculation tasks into so-called modules to provide flexibility and maintainability. In order to achieve better performance, we decide to separate the purpose of each module into two steps: First, we extract features of newly arrived packets and store them in the connection state. Feature extraction is done by a `handlePacket()` function each module provides. Second, a periodically called function calculates the final results of a particular module based on the previously stored data. This step is implemented in the `checkExpiry()` function of each module. In the following, we refer to a call of this function as the periodical calculation step. As a default, we set the interval between calculation steps to one second. We orchestrate and call the different modules in the main module. The main module invokes packet handling by the other modules, runs the calculation steps, and collects return values. Accordingly, the main module enables the configuration of the execution order of the analysis modules and the frequency of their calculations. The different implemented modules are used to calculate metrics and limitation scores as introduced by Siekkinnen et al. [11]. Details on the implementation of these modules and an according evaluation of limitation detection are presented in the publication "Online Monitoring of TCP Throughput Limitations" [71] and the Master's thesis by Kilian Holzinger [75].

A further design decision is made regarding the use of the available connection state. In contrast to the original offline implementation of the TCP RCA toolkit, we have limited capabilities to store data per analyzed flow. We use the connection state as a ring buffer to handle overrunning connection states. The ring buffer enables us to maintain a sliding window of currently considered data that was previously extracted per packet. Figure 4.1 shows the mentioned processing steps and their interplay with the connection state. As illustrated, data in the connection state can be used several times by different calculation steps, depending on the rate of arriving packets.

4.1.3 Parallelization of Analysis Pipelines

As we aim to monitor network links with bandwidths of several Gbit/s, we require efficient use of computational resources and a scalable analysis design. Our prototype implementation already considers efficient packet analysis as described in Section 4.1.2. In particular, efficiency is achieved by reducing computational load per packet by offloading expensive calculations, e.g., calculating limitation scores, to periodic result calculations. However, to scale up monitoring capabilities, we consider the parallelization of the whole analysis pipeline.

Our tool uses the packet capturing and analysis tool *FlowScope* [73], [74] as a base for packet processing and analysis. *FlowScope* natively supports multi-threaded packet analysis based on Receive Side Scaling (RSS). RSS enables NICs to distribute packets to several RSS queues while each is mapped to a specific CPU core. The distribution of packets to RSS queues relies on hardware-based hashing of packet header data. So far, FlowScope only supports the multi-threaded analysis of unidirectional traffic, as RSS natively does not ensure that packets of both directions of a TCP flow are mapped to the same RSS queue. However, our monitoring approach

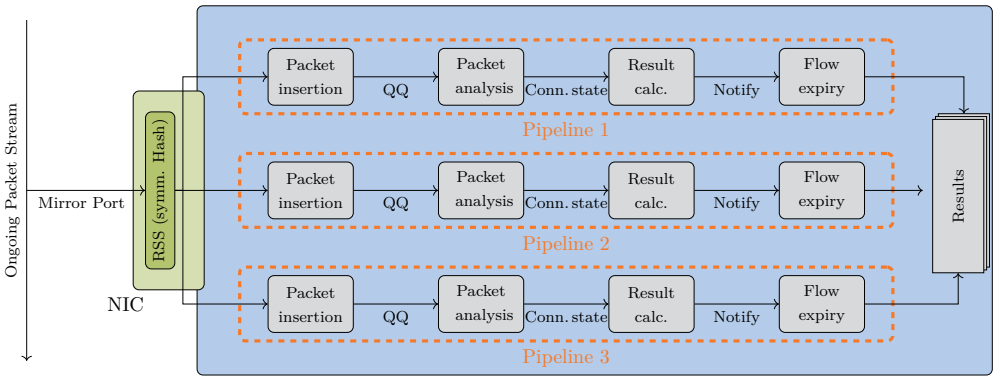


Figure 4.2: Monitoring architecture with several parallel analysis pipelines.

requires the same analysis pipeline to analyze both flow directions. To meet the bidirectional RSS requirement, we extend *FlowScope's* RSS capabilities to support symmetric hashing of packet data. In particular, we use a symmetric hash key introduced by Woo et al. [76].

To fully exploit speed up by RSS, we parallelize all further steps of the analysis pipeline. Each RSS queue forwards received packets to a pipeline consisting of four threads:

- First, a thread responsible for copying packets from the RSS queue to FlowScope's packet buffer.
- A thread that reads packets from the packet buffer to perform per-packet analysis and feature extraction. Extracted packet information is written to a flow's connection state.
- A third thread to run periodic result calculations based on data in the connection states and to detect expired flows.
- A thread to finally expire flows by deleting connection states.

The resulting architecture of the fully parallelized traffic analysis is illustrated in Figure 4.2.

To optimize the memory consumption of our tool, we further take care of proper flow expiration. As soon as a flow expires, its connection state is no longer needed, and there is no need to consider the flow for further periodic result calculation. Therefore, flow expiration contributes to efficient memory and CPU utilization. A flow is expired if we do not observe a packet with the corresponding 5-tuple for a specific time interval. For upcoming measurements, we use a time interval of 180 seconds. This policy indicates that we keep connection states for 180 seconds since the latest packet of the flow.

4.2 Performance Evaluation

After implementing a proof of concept version of the introduced measurement method, we are interested in its performance and the scalability of its analysis capabilities on commodity hardware.

First, we analyze the throughput, in the sense of analyzed traffic rate and run times of different analysis modules, to identify potential performance bottlenecks with a single analysis pipeline. Afterward, this section presents a performance evaluation of the proof of concept implementation extended for the parallelization of analysis pipelines. The performance of such multi-threaded implementation of the introduced measurement method is evaluated with generated data sets and captured Internet traffic.

4.2.1 Performance with a Single Analysis Pipeline

We first evaluate the performance of the proof of concept implementation by relying on a single analysis pipeline to determine performance bottlenecks. For the measurements, a Device under Test (DuT) with two Intel Xeon CPU E5-2620 processors, each with six physical cores at 2.10 GHz clock rate and a total of 128 GB memory was used. We run a single analysis pipeline consisting of four threads handling tasks like receiving packets, extracting packet features, calculating limitation scores, and a thread to expire inactive connections. Note that each thread runs on its own CPU core while all cores were chosen from the same CPU socket. We generate PCAPs with varying numbers of concurrent flows between two physical servers. Several *iperf3* servers are started in parallel and respond to several clients on different ports. Generated captures are replayed at a rate of 2 Gbit/s to overload the DuT.

We measure the run time of the periodical calculations for all different modules, which are responsible for different tasks of the analysis pipeline, as described in Section 4.1.2 , with a single flow. Most modules take less or around 20 ns. However, three modules consume significantly more runtime - especially the capacity estimation module with over 15 ms and the receiver window score calculation with over 2.5 ms. The long run time of the capacity estimation can be traced back to expensive operations like the applied differentiation of histograms and repeated operations on a sliding window of previously observed data. Calculating the receiver window score requires working on time series data for the advertised receiver window and the number of outstanding bytes. Note that the module was configured to work on a sliding window of 1000 packets. Such a long run time of single modules can cause a limitation regarding the number of concurrently processed flows. Assuming a load of 70 flows and a capacity estimation module runtime of up to 15 ms might result in a run time of over one second for a single periodic calculation of limitation scores for each flow. Therefore, the periodical calculations would be too slow for an interval of one second. While this issue might be mitigated by increasing the time between periodic checks, it still represents a trade-off between scalability and analysis effectiveness.

Another potential performance limitation of our analysis tool is the analyzer throughput. To avoid influences from run times that are too extensive, we turned off the capacity estimation module for the benchmarking of the analyzer throughput. We start measuring analyzer throughput for a single flow and achieve a data rate of 620 Mbit/s. In further measurements, we analyze the impact of concurrently analyzed flows and increase the number of current flows up to 4096. For up to 32 concurrent flows, throughput decreases continuously. Afterward, throughput decreases significantly for the following increments of the flow count. We trace this observation back to load on the CPU's Last Level Cache (LLC). Significant performance decrease occurs

when the sum of all connection states approaches the size of the LLC. When the LLC is utilized, connection states must be loaded from the main memory, making packet analysis more expensive. In our measurements, the impact of LLC utilization is observed between 32 and 64 flows. This observation matches our assumption, as 64 flows have an aggregated connection state size of about 11 MB with the used connection state size of 156 kB, while the DuT has an LLC size of 12 MB. After the drop due to the filled LLC, throughput remains constant for increasing flow counts. As expected, the aggregated module run time increases linearly with the number of concurrent flows.

4.2.2 Performance with Multiple Analysis Pipelines

We consider two major performance limitations of our prototype implementation: *(i)* computational resources, respectively analysis throughput, and *(ii)* memory consumption. To survey our optimized and extended monitoring tool's analysis capabilities, we benchmark our tool with a synthetically generated data set and packet traces taken from the MAWI Working Group Traffic Archive [56]. Note that measurements are conducted with an implementation considering further performance optimizations, besides the parallelization of analysis pipelines, described in the Master's thesis by Florian Wiedner [77].

Analyzer throughput

We run measurements in a test setup consisting of two physical hosts to evaluate the performance of our implementation. We replay packet captures from a dedicated host referred to as Load Generator. The Load Generator runs the packet generator *MoonGen*, capable of replaying packet captures at rates of several Gbit/s [78]. Our monitoring tool runs on a second host referred to as Device under Test (DuT). The LG and the DuT are connected with 10 GE cabling.

The LG is equipped with two AMD EPYC 7601 CPUs, each capable of 32 physical cores with 2.2 GHz clock frequency, 1 TB of DDR4 main memory, and an Intel X550T 10 GE NIC connected to the DuT. The DuT is equipped with two Intel Xeon Gold 6130 CPUs with 16 physical cores per socket with 2.2 GHz clock frequency, 395 GB DDR4 main memory, and two Intel X722 10 GE NICs. As hyper-threading is enabled, each socket provides 32 virtual cores. All hosts run Debian stretch.

To benchmark our monitoring tool's analysis throughput, we generate traffic captures consisting of different numbers of concurrent TCP connections. In particular, we use the traffic generator *iperf* to generate TCP flows with 1500 B packet size, a data rate of 1 Mbit/s, and a duration of 10 seconds. *iperf* is limited in the number of concurrently generated flows, as too many concurrent flows result in inaccurate data rates. We observe such inaccuracies for flow counts larger than 128. Therefore, we capture several traces with 128 concurrent flows and merge them to achieve larger numbers of concurrent flows. This merging procedure requires time-shifting of the captures' timestamps to synchronize the flows from different captures. The time-shifting of captured flows is done with *editcap*. The resulting data set provides traces from 1 flow up to 8192 concurrent flows while the number of flows increases according to the powers of two. As we generate flows at 1 Mbit/s data rate, the maximum expected data rate in our data set is around 8 Gbit/s corresponding to about 1.2 million packets per second. However, we found that

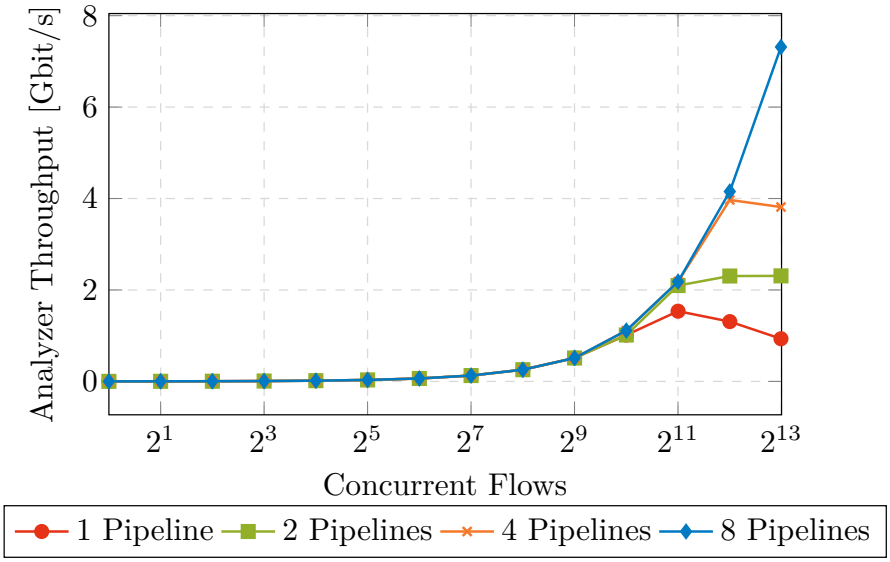


Figure 4.3: Analyzer throughput for varying numbers of concurrent flows.

the achieved average data rate for the trace with 8192 flows is slightly lower than expected, i.e., about 7.5 Gbit/s, due to inaccuracies in the merged PCAPs.

We measure the achievable throughput of the analysis pipelines with the generated data set. We find that a single pipeline can analyze 1024 concurrent flows without dropping packets or ignoring flows for the analysis. With increasing load, i.e., more than 1024 concurrent flows, packets get dropped due to fully exploited computational resources, and the maximum achieved analysis throughput is around 1.5 Gbit/s for 2048 flows. We observe that throughput additionally decreases for measurements with a single pipeline and flow counts larger than 2048 down to 1 Gbit/s. For setups with more than one single pipeline, we find a throughput of approximately 1 Gbit/s per pipeline. This observation holds for all setups, even for eight parallel pipelines and an offered load of 8192 concurrent flows. Figure 4.3 shows the measured analyzer throughput for all pipeline setups. We conclude that these observations confirm the linear scalability of our tool due to independent analysis pipelines.

Memory consumption

Next, we want to assess the memory consumption of the implemented tool. Therefore, we conduct measurements with a trace taken from the MAWI Working Group Traffic Archive [56] containing captured Internet traffic. In particular, we use the trace captured at MAWI's Samplepoint-F on 23 December 2017. Choosing such a trace for evaluating memory consumption is motivated by the expectation that real-world traffic captures carry larger numbers of concurrent flows than our synthetically generated dataset.

Again, we replay traffic from a dedicated load generator (LG) while the monitoring tool runs on a dedicated device under test (DuT). The DuT is a server equipped with two AMD EPYC 7601 CPUs, each capable of 32 physical cores with 2.2 GHz clock frequency and 1 TB memory. As

hyper-threading is enabled, the DuT provides 64 threads, which are used to run 16 concurrent analysis pipelines.

During the measurements with 16 pipelines, we find a maximum memory consumption of 160 GB. In addition to the connection state size allocated in memory for each analyzed flow, our tool consumes significant amounts of memory per pipeline to run the ring buffer QQ that *FlowScope* uses to buffer received packets before forwarding them to further analysis steps. By default, our tool allocates 8 GB per analysis pipeline to maintain the QQ buffer. Correspondingly, 128 GB of memory are allocated for 16 parallel pipelines. These numbers indicate that the setup with 16 pipelines allocates connection states up to 32 GB while analyzing the considered MAWI trace. Our analysis shows that providing the required memory resources for analyzing such traces is feasible considering modern commercial off-the-shelf hardware.

4.3 Summary

This chapter introduced a measurement method and design decisions for a proof of concept implementation to conduct the analysis of TCP throughput limitations in real-time considering high traffic rates of several Gbit/s. The introduced method builds on the offline TCP root cause analysis approach by Siekkinen et al. [11], [35]. Siekkinen's RCA approach is modified to work on sliding windows of observed packet data to enable online analysis, providing insights into throughput limitation in fixed time intervals.

We evaluate the performance of a single-threaded and multi-threaded proof of concept implementation corresponding to the introduced measurement method based on the high-performance traffic capturing tool *FlowScope* [73], [74]. We consider measurements on achievable analyzer throughput, with generated TCP traffic, and measurements to survey required memory consumption with captured Internet traffic provided by MAWI [56].

Our evaluation shows that online throughput limitation analysis is feasible in real-time with reasonable performance on commodity hardware. Further, we find several trade-offs between the accuracy and responsiveness of the tool and its performance. Applying root cause analysis on a sliding window of packet data may imply long run times, while run time is limited due to periodical calculations. We find linear scalability of analyzer throughput with increasing numbers of analysis pipelines. During our measurements, we achieve a maximum analyzer throughput larger than 7 Gbit/s composed of several thousand concurrent flows. Our design implies that memory concerns are not crucial for modern commodity hardware, as our tool requires approximately 160 GB of memory for one million concurrent flows with a connection state size of 156 kB. Measurements on memory consumption based on captured Internet traffic confirm this hypothesis.

4.4 Author's Contribution

This chapter builds on the publications "Online Monitoring of TCP Throughput Limitations" [71], which is a joint work by the author of this thesis, Kilian Holzinger, Benedikt Jaeger, Paul Em-

merich, and Georg Carle, and "Scalable TCP Throughput Limitation Monitoring" [72], which is joint work by the author of this thesis, Florian Wiedner, Benedikt Jaeger, Paul Emmerich, and Georg Carle. The author significantly contributed to both publications regarding research objectives, the design of the online RCA method, and the methodologies to evaluate the implemented proof of concept. Further, the author conducted measurements of achievable analysis throughput with the multi-pipeline implementation and generated the used data set.

The online RCA tool was implemented in the scope of the Master's theses by Kilian Holzinger [75], who implemented and evaluated the first proof of concept version, including discussed module run times, and by Florian Wiedner [77], who extended the tool for parallelization of analysis pipelines with RSS and robust traffic handling. The latter also conducted measurements based on captured Internet traffic provided by the MAWI working group to survey memory consumption. Both theses were supervised by the author and pursued research objectives and methodologies specified by the author. The author has re-evaluated measurement results and composed the final contents of both theses for publication.

Chapter 5

Classifying TCP Throughput Changes

As elaborated in Chapter 2, existing TCP root cause analysis approaches consider different segmentation of connections to determine root causes. Thereby, considered intervals range from one root cause per flow [10], over one root cause per transfer period [11], up to root cause estimates after every 256 packets [2]. However, none of the existing approaches considers the classification of individual changes in observed throughput. As a change in the current throughput limitation is expected to correlate with a change in observed throughput, a change-based segmentation of a connection promises fine-grained analysis of throughput limitations while enabling the classification of individual changes in an observed throughput signal.

This chapter surveys the capabilities of classifying individual throughput changes with throughput limitations, respectively, changes of throughput limitations. Such an approach requires *(i)* , the reliable detection of changes in the throughput signal of a connection and *(ii)* , the matching between detected changes and determined throughput limitations. As a first step towards classifying throughput changes, we compare the results of different change point detection methods to transfer periods resulting from the Isolate and Merge (I&M) algorithm introduced by Siekkinen et al. [35]. We survey the reliability of different change point detection methods with ground truth data, i.e., TCP connections with intended throughput changes, and conduct measurements with captured Internet traffic to survey the occurrence of throughput changes and their matching to transitions between transfer periods.

The contents of this chapter are based on the publication:

> **Towards the Classification of TCP Throughput Changes**,
> *Simon Bauer, Benedikt Jaeger, Max Reimann, Jonas Fromm, Georg Carle*,
> in 2022 IEEE/IFIP Network Operations and Management Symposium (NOMS 2022),
> Budapest, Hungary, Apr. 2022

The classification approach for throughput changes presented in Section 5.1, the implementation of the evaluation tool for the application of change point detection methods on throughput

time series described in Section 5.2, and the evaluation of considered CPD methods presented in Section 5.3 are based on the publication "Towards the Classification of TCP Throughput Changes" [79]. The same applies to the results of the conducted case study on Internet traffic in Section 5.4, which was extended for an analysis of detected transfer periods based on different drop values and an analysis of the distribution of shares of matched period changes and matched change points across connections, providing a more detailed perspective on matching shares of individual connections in the data set.

5.1 Approach

To survey the matching between changes in the throughput of a TCP connection and changes in throughput limitations, we survey the correlation of results by the I&M algorithm, as described in Section 5.1 and different change point detection methods, as described in Section 5.1.2. Considered classes of throughput changes are introduced in Section 5.1.3.

5.1.1 TCP Transfer Periods

To determine time intervals in which the sending application limits the throughput of a TCP connection, Siekkinen et al. [35] introduce the Isolate & Merge (I&M) algorithm that segments a flow in three different kinds of periods:

- Application Limited Periods (ALP): the sending application does not provide enough data to utilize available network resources fully.
- Bulk Transfer Periods (BTP): the sender produces enough data to continuously send data and fully utilize available network or receiver resources.
- Short Transfer Periods (STP): short bulk transfers that contain less than 130 packets and, accordingly, are assumed to be mainly dominated by congestion control.

First, the I&M algorithm isolates ALPs, BTPs, and STPs from each other. Therefore, the Isolate procedure processes all packets of a connection according to their order. Initially, the algorithm starts to assign packets to an ALP. If three consecutive data packets as large as the Maximum Segment Size (MSS) are observed, the Isolate procedure starts a BTP. Further packets are assigned to such BTP as long as two conditions are satisfied: *(i)* , the share of packets smaller than the MSS within the last ten packets is not larger than a predefined threshold value (th), and *(ii)* , the inter-arrival time (IAT) between two subsequent packets is not larger than $\frac{RTT}{2}$. If such conditions are no longer satisfied, the Isolate procedure assigns packets to an ALP and terminates the current BTP. After isolating transfer periods, the Merge procedure is purposed to merge ALPs with adjacent BTPs as far as the impact of the ALP is insignificant regarding the periods' average throughput. An ALP gets merged to the adjacent transfer periods if merging the periods does not result in a significantly lower average throughput of the new period than the average throughput of both BTPs without merging. Merging relies on a predefined threshold value ($drop$) that decides whether a merged period's throughput is significantly lower.

5.1.2 Change Point Detection

Change point detection (CPD) is purposed to detect changes in a signal, respectively, in a sequence of values. Detected changes are referred to as change points (CP). CPD is applied in different contexts, considering different characteristics of the applied method. While some approaches require the number of expected changes within a signal before the estimation, other methods do not require such information beforehand. A further differentiation exists between online and offline detection. Online CPD aims to provide fast detection of changes, while offline CPD allows signal processing without constraints regarding execution time. According to Truong et al. [80], a change point detection method is defined by a cost function, a search method, and different constraints depending on the method. The cost function is purposed to assess the homogeneity of a signal. The search method determines the processing of the signal to detect changes.

The application of change point detection in network traffic analysis has been considered before. For example, change point detection is used to detect anomalies and network intrusion in real-time with a focus on efficient and fast detection of changes in traffic parameters, as, for instance, caused by denial of service attacks [81]–[83]. Besides its application to anomaly detection, change point detection has been considered to optimize TCP congestion control in combination with the application of deep learning by Li et al. [84].

5.1.3 Classifying TCP Throughput Changes

We define four types of change points according to their relation to transfer periods as determined by the I&M algorithm: The first type is referred to as a *matching* change point, defined as CPs that match a border between transfer periods. If a connection changes from application limitation to bulk transfer, the connection's throughput is expected to increase. Conversely, we expect throughput to decrease significantly when a bulk transfer ends and an ALP begins. Next to matching change points, there might be changes that do not coincide with the borders of transfer periods. We refer to such CPs as *unmatched*. Unmatched CPs are subdivided according to the transfer period surrounding the CP. Correspondingly, the four types of CPs are

- matched,
- unmatched in an ALP,
- unmatched in a BTP,
- and unmatched in an STP.

Figure 5.1 shows an example of the throughput of a TCP connection over time. In this example, the flow includes two BTPs separated by an ALP. The changes between the transfer periods imply throughput changes, marked as change points CP_1 and CP_4. During the ALP, throughput changes again. The corresponding unmatched change points CP_2 and CP_3 can be traced back to the resource-limited behavior of the sending application, i.e., a lack of data to be sent continuously. Further, we expect changes in throughput when an application switches between sending small packets and completely idling during an ALP. As illustrated, the throughput during the

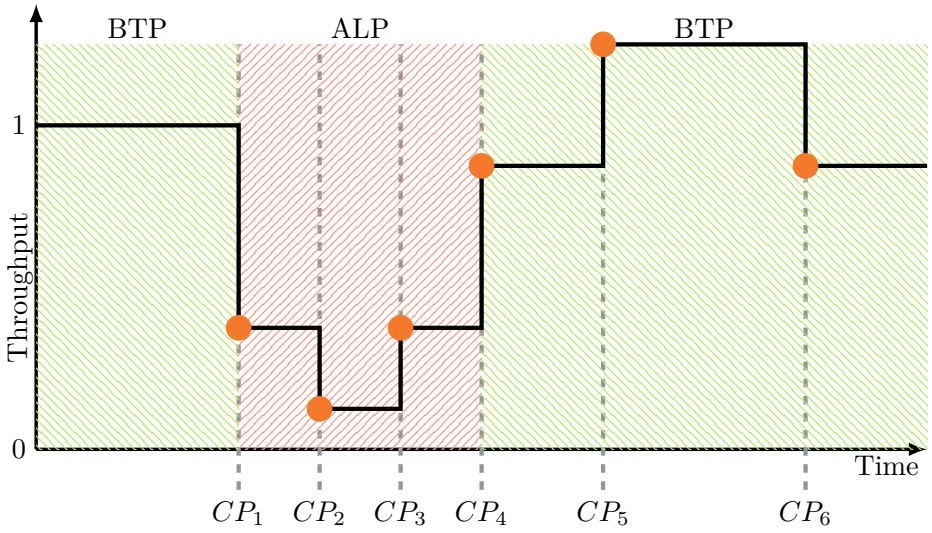


Figure 5.1: Illustration of the matching between transfer periods and different types of change points.

second BTP increases, resulting in CP_5 and CP_6, for example, as there is temporarily more available bandwidth on the bottleneck link.

5.1.4 Evaluating Change Point Detection Methods

To the best of our knowledge, the application of CPD on throughput changes of TCP connections has not been studied so far. Therefore, we evaluate the suitability of different change point detection methods for our problem. The metrics used for evaluating CPD methods are described in this section. In order to evaluate different CPD methods, we require the generation of TCP connections with intended changes in throughput. The implementation of the generation process of labeled TCP connections is described in Section 5.2.

The first metric we consider is the Annotation Error [80]. The Annotation Error is defined as the difference between the number of actual changes (derived from ground truth labeling) and the detected change points. A significant Annotation Error implies a large number of undetected changes (false negatives) or a large number of detected changes without an actual change (false positives).

A further indicator of the accuracy of estimated CPs is the F1-score according to Truong et al. [80]. The F1-score is calculated as the harmonic mean of precision and recall, assessing the impact of false negatives and false positives on the detection results. Precision is calculated by the ratio of detected change points that match an actual signal change. Therefore, strong over-segmentation by the detected change points results in low precision due to many false positives. Recall is the ratio of actual change points matched by a detected change point. The definition of recall indicates that many false negatives, i.e., not detected changes, result in low recall values. Whether an actual change and an estimated CP are referred to as matched depends on the error margin, also referred to as tolerance interval, used for the score calculation. As the harmonic

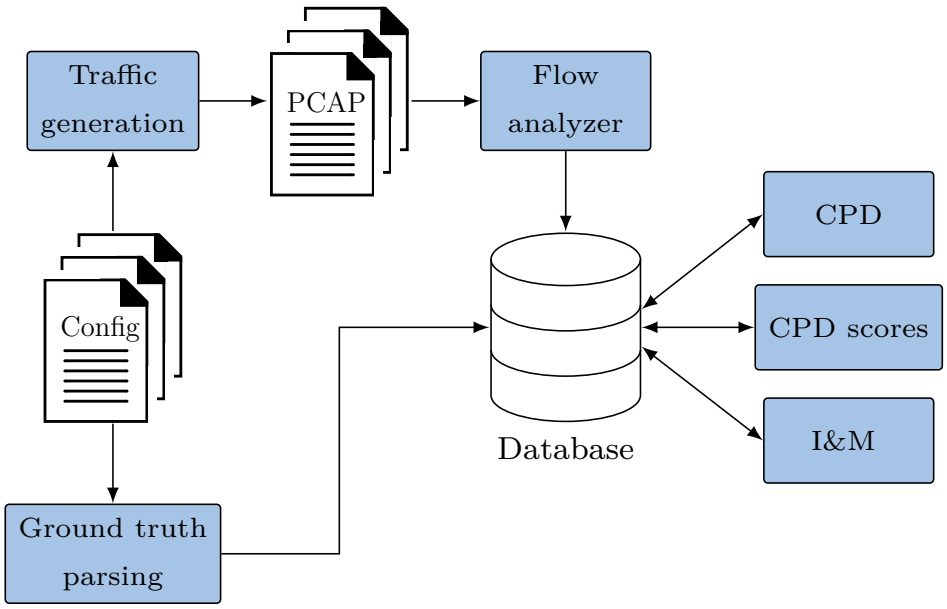


Figure 5.2: Pipeline for CPD evaluation and analysis of the matching between CPs and transfer periods.

mean of precision and recall, the F1-score is normalized between 0 and 1, while larger F1-scores indicate better change point detection results.

5.2 Implementation

We use two tools to implement traffic analysis and traffic generation. First, we implement a traffic analysis tool to parse captured network traffic, conduct CPD, and run the I&M algorithm. The second tool is purposed to generate TCP connections with intended throughput changes while tracking ground truth information for each connection. Both tools result in a pipeline for different kinds of conducted measurements as illustrated in Figure 5.2.

Flow Analysis and Feature Extraction

The introduced traffic analysis tool consists of a pipeline of analysis components, each interacting with a database backend to read and write results. The first analysis step is extracting packet information from a PCAP containing captured traffic and aggregating packets to connections, also referred to as flows in the following. The flow aggregation and feature extraction component is written in Go and relies on the *gopacket* library [66] to read packets. We extract packet timestamps, payload sizes, the MSS, and TCP flags for each packet and compose packets to a flow based on the IP5-tuple, considering packets in both directions. Further, the tool calculates a sequence of throughput for each flow. Throughput is calculated by segmenting the flow duration in fixed-sized intervals and aggregating transmitted bytes for each interval. We use a resolution for throughput calculation of 100 ms for our measurements.

Change Point Detection

To implement change point detection, we rely on the Python library *ruptures* introduced by Truong et al. [85]. *Ruptures* enables users to compose CPD methods from over eight different cost functions and four search methods applicable to our use case, i.e., offline CPD for an unknown number of changes. For our measurements, we choose the cost function that assesses signal homogeneity based on the least absolute deviation. The least absolute deviation is purposed to detect shifts in the median of the analyzed signal while it is expected to provide outlier robust change point detection [86]. The considered search methods are the optimal method *Pelt* [87] and the three approximate search methods *Binary Segmentation* (Bin. Seg.) [88], *Bottom Up Segmentation* (Bott. Up.) [89], and *Window-based CPD* (Win.-based). Further, *ruptures* uses the penalty value to influence the sensitivity of applied change point detection. In the following, we refer to a CPD method as a tuple of cost function and search method. In order to estimate changes, the analysis tool reads a sequence of measured throughput for each connection and conducts change point detection. Afterward, the CPD component writes detected changes to the tool's database backend. A further component, responsible for validating estimated change points, receives ground truth data by processing configuration files of the traffic generation tool introduced in Section 5.2. Such configuration files include the time when an intended throughput change occurred. Based on the set of intended changes and the set of estimated change points, the analysis tool calculates scores to assess CPD results for generated traffic as introduced in Section 5.1.4.

Transfer Period Analysis

A further analysis component identifies transport periods for each flow according to the I&M algorithm introduced by Siekkinen et al. [35], as described in Section 5.1. The implemented I&M component calculates inter-arrival times (IAT) and round trip time (RTT) estimates for each connection. Further, we determine the distance between the capturing point and the sender as required for the I&M algorithm. Our tool relies on two approaches to derive RTT estimates of a flow. First, we calculate the initial RTT during the TCP Three-Way-Handshake. As the sequence of packets during the handshake is well-known, initial RTT can be estimated by the intervals between the SYN packet and the consecutive $SYN + ACK$ packet, referred to as d_1, and between the $SYN + ACK$ packet and the following ACK packet, referred to as d_2. Next to estimating the initial RTT, we calculate RTT estimates based on the TCP timestamp option if a connection uses it. The timestamp option-based approach identifies triples of packets with matching TCP values and calculates the intervals between such packets. As the capturing point might not be close to the sender, Siekkinen et al. [35] propose shifting of packet timestamps in order to calculate IATs between a received ACK and a sent data packet. Shifting is based on the relative distance between the capture point and the sender, i.e., the intervals d_1 and d_2 are used to estimate the relative distance between the sender and the capture point as $\frac{d_1}{d_1+d_2}$. Based on the extracted data, our tool determines the segmentation of each connection into different transfer periods as specified by Siekkinen et al. [35].

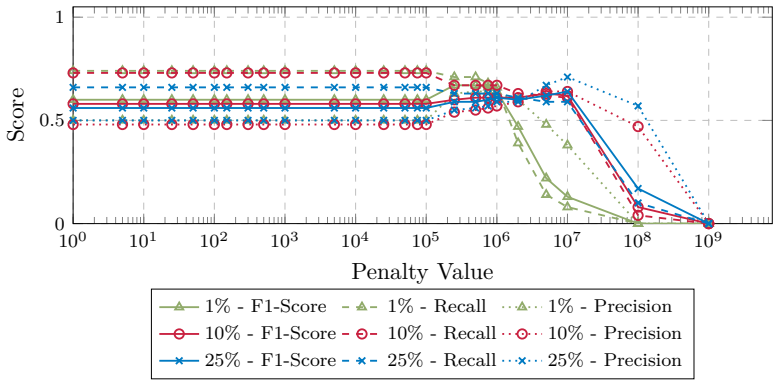
Labelled Traffic Generation

In order to evaluate different CPD methods, we require the generation of TCP traffic with intended throughput changes. We rely on the TCP measurement framework introduced by Jaeger et al. [90], based on the network emulation tool Mininet [91]. The measurement framework enables network and host parameters to be dynamically reconfigured based on user-defined schedules. According to the configured schedule, the framework orchestrates network entities to generate traffic and adapts link parameters like bandwidth, delays, or loss. TCP traffic is generated by transmitting data with the *netcat* utility. A simple topology consisting of two hosts connected by a switch is sufficient for our traffic generation, as we only aim to modify the throughput of a specific connection. Further, the measurement framework already satisfies our requirement to dynamically change a flow's throughput, as the schedule-based configuration can configure the bandwidth of a link, which we use as an upper limit of throughput.

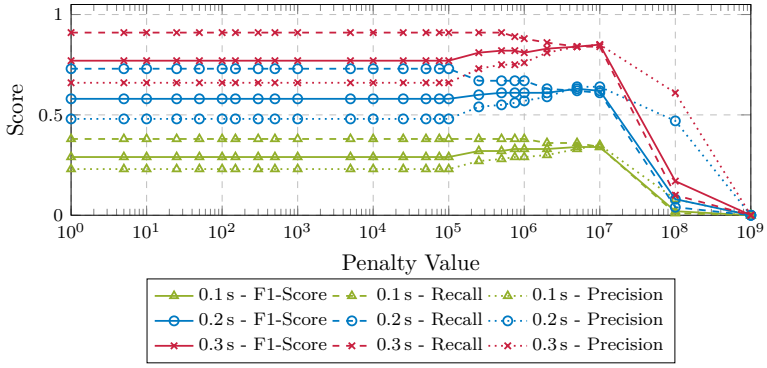
5.3 Reliability of Change Point Detection

To evaluate different CPD methods, we conduct measurements on generated TCP traffic with intended throughput changes. As described in Section 5.2, we implement change point detection with the Python library *ruptures* and use the least absolute deviation as the cost function. Measurements are conducted using the search methods Pelt, Bin. Seg., Bott. Up, and Win.-based CPD. We calculate the Annotation Error, F1-score, Recall, and Precision as defined in Section 5.1.4. Our evaluation is purposed to determine how reliable changes are detected with different search methods, penalty values, and amplitudes of changes. We generate three traces for each traffic configuration to ensure that results are robust to measurement artifacts. The plotted results show the average of the different measurement runs per configuration.

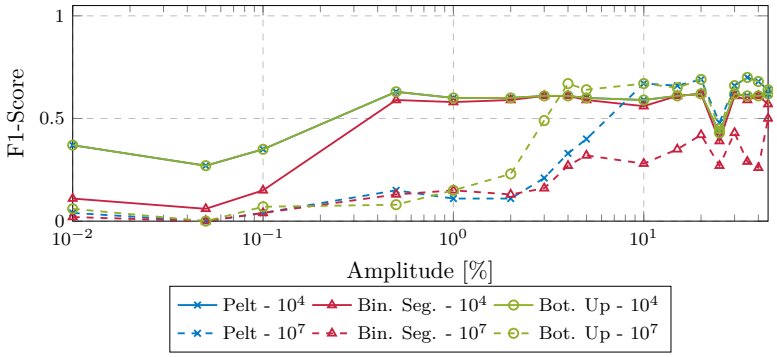
First, we assess the impact of different penalty values on the accuracy of conducted change point detection. We generate connections with a duration of 30 s and 30 equidistantly distributed changes and configure change amplitudes of 1 %, 10 %, and 25 %. We observe that very high penalties imply worse accuracy depending on the amplitude of changes as CPD gets less sensitive. This pattern is shown in Figure 5.3 a) for measurements conducted with the search method Pelt, different amplitudes, and an error margin of 200 ms. We observe constant F1-scores, recall, and precision for smaller penalties. Further, we note that F1-scores slightly increase for specific penalties before penalties become too high and no changes are detected. This observation can be traced back to fewer false positives for larger penalties. We find a significant impact by the error margin used for the F1-score calculation. For example, increasing the error margin from 200 ms to 300 ms increases the F1-score for measurements conducted with Pelt from 0.7 to 0.9 for most penalties as shown in Figure 5.3 b) for an amplitude of 10%. The same patterns are observed for other search methods. With an error margin of 300 ms, we observe large recall values for Pelt, Bin. Seg., and Bottom Up. This observation indicates that nearly all intentionally generated throughput changes are detected. A significantly lower precision value implies that the CPD methods detect more changes than intended. For Win.-based CPD, we observe larger precision values than recall values, indicating that fewer intended changes are detected, while the false



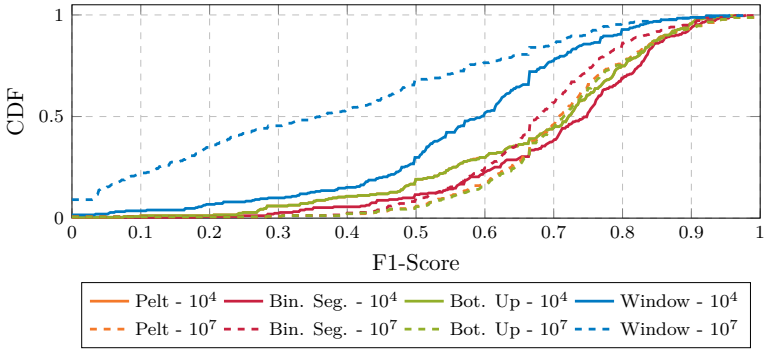
(a) Impact of penalty value for different amplitudes (Pelt).



(b) Impact of error margin (Pelt).



(c) Impact of change amplitude for different penalties.



(d) Traffic with random parameters for different penalties.

Figure 5.3: Measurement results of CPD method evaluation with different generated data sets.

positives share is significantly lower than for other search methods. However, Pelt, Bin. Seg., and Bottom Up result in significantly larger F1-scores.

To assess the impact of the amplitude of changes, we generate connections of 30 s, apply an error margin of 200 ms, and consider different penalty values. Figure 5.3 c) shows F1-scores for considered amplitudes and penalties of 10^4 and 10^7 for Pelt, Bin. Seg., and Bottom Up. We observe small F1-scores for smaller amplitudes for both penalties independent of the search method. As expected, smaller penalty values enable the detection of changes with smaller amplitudes. We find constant accuracy after the amplitude exceeds a certain threshold for both penalties.

To further examine differences between search methods, we generate 250 connections with randomly selected amplitudes for each change, randomly selected numbers of changes per connection, and randomized intervals between two changes. The cumulative distribution of measured F1-scores for an error margin of 200 ms reveals that 50 % of measured F1-scores are larger than 0.7 for Pelt, Bin. Seg., and Bottom Up with a penalty of 10^4, as shown in Figure 5.3 d). Win.-based CPD shows significantly lower accuracy. The accuracy with a penalty of 10^7 is slightly smaller than that of 10^4 for Pelt, Bin. Seg., and Bottom Up, and significantly smaller for Window-based CPD. Pelt and Bottom Up Segmentation result in nearly identical F1-scores for both applied penalties. The same observation applies to other experiments discussed above.

For our analysis of Internet traffic, we conclude that Pelt, Bott. Up, and Bin. Seg. are expected to provide more accurate results than Window-based CPD. The measured recall values for such search methods imply that they are more likely to detect false positives than not to detect an intended change.

5.4 Case Study on Internet Traffic

To evaluate the suitability of our approach to classify changes in TCP throughput, we conduct measurements on captured Internet traffic. We compare measurement results for different CPD methods and penalty values. Note that measurements with Pelt were not feasible due to Pelt's computational complexity. To survey the results of our matching analysis between period changes (PC) and change points (CP), we calculate two shares: We define the *CP share* as the share of detected change points matching a period change and the *PC share* as the share of period changes that match a detected change point. The *CP share* represents the share of matched change points and implies the share of unmatched change points as defined in Section 7.2. The *PC share* indicates how successful period changes were matched to detected change points and whether period changes do not coincide with an estimated throughput change.

For our study, we analyze a traffic capture provided by the Measurement and Analysis on the WIDE Internet (MAWI) Working Group [56]. The trace taken on 31 July 2021 includes 15 minutes of Internet traffic captured at MAWI's Samplepoint-F. We filter out TCP connections consisting of less than 130 packets, which is considered as the minimum packet count for a BTP. Further, we only consider IPv4 traffic. Besides packet count and IPv4, we constraint analyzed connections to those who provide a fully captured TCP Three-Way-Handshake as data extracted

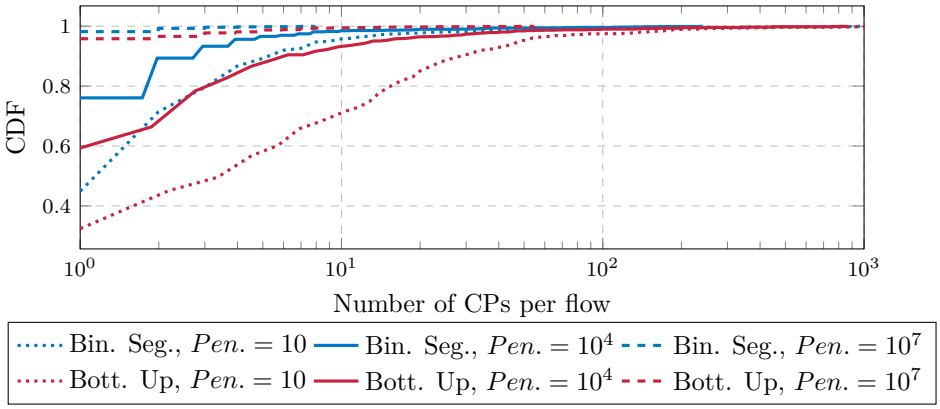


Figure 5.4: Number of CPs per connection.

Drop	TPs	BTPs	STPs	ALPs
0.8	3588	554	617	2417
0.9	3814	590	698	2526
1.0	4550	684	967	2899

Table 5.1: Number of transfer periods for different drop values.

during the handshake is needed for the I&M algorithm. In total, we find 1649 connections that fulfill all analysis requirements in the analyzed traffic capture. We choose a matching margin of 300 ms implying, that we refer to a change point as matched if the difference between the timestamp of the change point and the nearest period change is smaller 300 ms. We calculate throughput in intervals of 100 ms. For measurements with the I&M algorithm, we choose a *drop* value of 0.9 and a *th* value of 3 as proposed by Siekkinen et al. [35].

First, we are interested in the number of change points detected per connection. We conduct measurements with different change point detection configurations and calculate the cumulative distribution function (CDF) of the number of change points per connection as shown in Figure 5.4. We observe that different penalty values result in significantly different numbers of detected changes per flow for both search methods. Further, we find that the number of changes decreases significantly with increasing penalty values. Bottom Up segmentation results in significantly more detected change points than Binary Segmentation. Independent of the search method and the penalty, we find that the vast majority of connections include less than ten estimated change points while observing a very long tail of detected change points per flow.

Second, we are interested in the number of determined transfer periods. Therefore, we survey the impact of period merging on the results of the I&M algorithm by running measurements with different *drop* values, i.e., 0.8, 0.9, and 1.0. Table 5.1 shows the measured numbers of detected periods for considered drop values. Within the 1649 analyzed connections, we find 1100 connections that only consist of one single ALP. We remove such connections from our data set as they are not of interest for further analysis of throughput change classification. In

Method	Penalty	Total CPs	Matched CPs	Unmatched CPs	Matched PCs
Bin. Seg.	10^1	4502	683 (15.17 %)	3819 (84.83 %)	31.41 %
	10^4	2065	446 (21.60 %)	1619 (78.40 %)	20.51 %
	10^7	510	286 (56.08 %)	224 (43.92 %)	12.32 %
Bottom Up	10^1	8991	1195 (13.29 %)	7796 (86.71 %)	54.97 %
	10^4	5578	878 (15.74 %)	4700 (84.26 %)	40.39 %
	10^7	819	301 (36.75 %)	518 (63.25 %)	13.85 %

Table 5.2: Total number and share of matched period changes (PC) and change point (CP) types.

Method	Penalty	Unmatched CPs	in BTP	in ALP	in STP
Bin. Seg.	10^1	3819 (84.83 %)	3491 (77.54 %)	286 (6.35 %)	42 (0.93 %)
	10^4	1619 (78.40 %)	1527 (73.95 %)	69 (3.34 %)	23 (1.11 %)
	10^7	224 (43.92 %)	209 (40.98 %)	0 (0.0 %)	15 (2.94 %)
Bottom Up	10^1	7796 (86.71 %)	6128 (68.16 %)	1562 (17.37 %)	106 (1.18 %)
	10^4	4700 (84.26 %)	4071 (72.98 %)	567 (10.16 %)	62 (1.11 %)
	10^7	518 (63.25 %)	502 (61.29 %)	1 (0.12 %)	15 (1.84 %)

Table 5.3: Types of unmatched change points.

total, the remaining connections consist of 2714 periods for a *drop* value of 0.9 as used for further analysis.

Regarding the results of our matching analysis between CPs and PCs, as shown in Table 5.2 and Table 5.3, we observe that the share of matched change points increases with increasing penalties. At the same time, the absolute number of matched CPs decreases, such as the total number of detected CPs. In contrast, the share of matched period changes increases with decreasing penalty. This pattern can be explained by fewer change points detected for larger penalties and more change points detected for smaller penalties. For changes detected by Bin. Seg. and a penalty of 10^7, we observe a CP share of 56 %, i.e., a period change can explain 56 % of estimated throughput changes. However, at the same time, the mentioned CPD configuration only has a PC share of 12.3 %. This indicates that over 85 % of the period changes are not matched by a change point. With Bottom Up segmentation and a penalty of 10^4, we observe a PC share of over 40 %, respectively 54.9 % for a penalty of 10. Such large PC shares can be explained by a larger number of detected change points, implying large numbers of unmatched changes as shown in Table 5.3. We conclude that the penalty value can be used to either increase the CP share at the cost of a worse PC share or vice versa.

We conduct a more fine-grained analysis of CP and PC matching shares and compute such shares for each flow in our data set. Figure 5.5 shows the cumulative distribution of measured CP and PC shares for measurements conducted with Bottom Up segmentation. For a penalty of 10, we observe that nearly 40% of flows show a PC share equal to 1, implying that all period changes within such connections coincide with a change point. At the same time, the CDF of the CP shares for the same penalty implies large shares of unmatched change points. Contrary to such observation, a penalty value of 10^7 results in significantly larger CP shares than PC

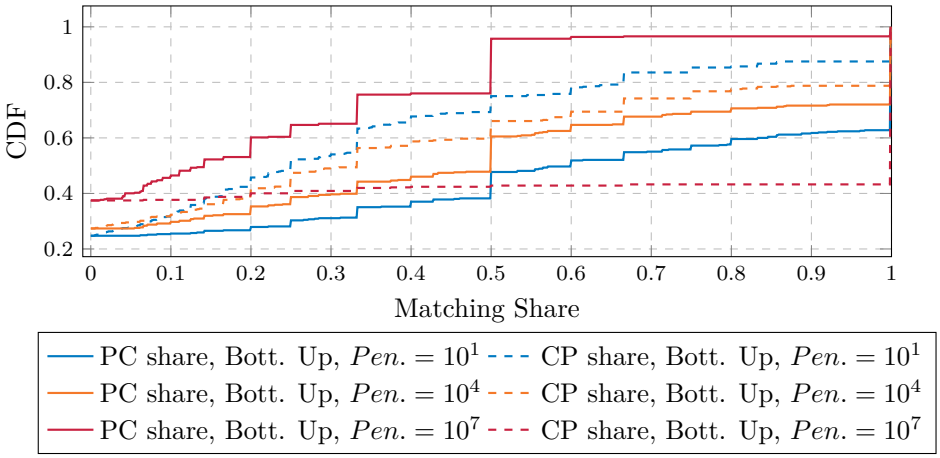


Figure 5.5: Measured share of matched period changes and matched change points.

shares. This implies a trade-off between increasing CP and PC shares depending on the penalty value.

5.5 Summary

In this chapter, we introduce a first approach toward the classification of individual changes in the throughput of a TCP connection using TCP throughput limitation analysis. As a first step, we survey the matching between the results of change point detection methods and the Isolate and Merge algorithm introduced by Siekkinen et al. [35].

We evaluate different change point detection methods based on generated TCP traffic providing ground truth data and conduct measurements on captured Internet traffic to survey the matching between transfer periods and detected throughput changes. Regarding analyzed Internet traffic, we find that, depending on the change point detection configuration, up to over 50 % of detected changes can be matched to a change between transfer periods. High rates of matching change points coincide with low shares of matched period changes, i.e., most period changes can not be matched to a detected throughput change. Our measurements also reveal large numbers of unmatched change points that mainly occur during bulk transfers.

Our observations motivate further studies on the coincidence between throughput changes and transfer periods, like optimizing CPD configuration. Throughput changes within bulk transfers are considered to be of interest for further analysis regarding their occurrence with changes in the throughput limitation of a connection. Such refinement requires extending our approach with other concepts from throughput limitation analysis, i.e., more detailed root cause analysis of bulk transfer periods.

5.6 Author's Contribution

This chapter builds on the publication "Towards the Classification of TCP Throughput Changes" [79] which is a joint work by the author of this thesis, Benedikt Jaeger, Max Reimann, Jonas Fromm, and Georg Carle. The author significantly contributed to the publication regarding addressed research objectives, the classification approach for throughput changes, and measurement methodologies for ground truth-based CPD evaluation. Further, the author conducted all measurements to evaluate change point detection, conducted measurements on captured Internet traffic, and evaluated corresponding measurement results.

The initial traffic analysis tool used for parsing captured traffic, connection mapping, and measurements with the I&M algorithm and different CPD methods was implemented as part of the Bachelor's thesis by Jonas Fromm [92]. The pipeline to evaluate change point detection with ground truth data and to survey matching was implemented as part of the Bachelor's thesis by Max Reimann [93]. Both theses were supervised by the author and pursued research objectives, implementation requirements, and methodologies defined by the author.

Part III

Active Internet Measurements

Chapter 6

Active Measurements of TCP Throughput

While large-scale passive measurements on captured Internet traffic ensure considering a broad spectrum of involved endpoint population, observed traffic types, and connection characteristics, like the amount of transmitted data, they only provide limited insights into conditions and configurations of endpoints, the actual traffic type, and applications behind traffic. Further, passive data sets make it difficult to compare the performance of connections, as a comparison requires reference points, for instance, two connections between the same client and server. Accordingly, passive measurements are suitable for investigating a representative population of Internet connections but do not provide the possibility to survey impacts by specific factors, like client configurations, on the throughput of Internet connections.

Therefore, we consider conducting active measurements on throughput, which enables the control of at least one endpoint and its configuration, allows the repeat of measurements several times, and implies increased knowledge regarding initiated traffic. However, conducting active performance measurements on the Internet requires identifying suitable measurement targets to initiate traffic, for instance, public servers that host files to be downloaded during measurements. Therefore, it is necessary to carry out dedicated measurements to identify targets.

As observed in Chapter 3, the transmission control protocol is responsible for transmitting the majority of Internet traffic [59]. Further, Chapter 3 shows the widespread use of TCP window scaling but not a common use of ECN and SACK on the Internet. Accordingly, the question arises of how using such TCP performance-related options impacts the throughput of Internet connections. While different publications survey the performance and the evaluation of web applications [94]–[98] and publications examining the deployment of TCP options on the Internet, as described in Section 2.5, there are only limited insights into the impact of TCP option usage on the performance of Internet connections.

This chapter is purposed *(i)* to introduce an approach to active TCP performance measurements on the Internet, relying on downloads of files hosted by public and uncontrolled web servers, and *(ii)* to evaluate measurement results from correspondingly conducted active Internet measurements. We examine differences between considered vantage points used to conduct

measurements, potential impacts by edge caching by CDNs, and impacts by performance-related TCP options on throughput. Further, this chapter presents an analysis of results by the applied crawling procedure, i.e., the procedure to identify measurement targets, regarding the intrusiveness of crawls and the success rate of crawling public web servers.

In particular, this chapter contributes:

- An active measurement approach and a corresponding pipeline for measurements targetting public Internet web servers.
- An analysis of characteristics of large-scale crawling of suitable measurement targets.
- An analysis of performance indicators observed during active Internet measurements conducted from different vantage points hosted on the Internet.

The contents of this chapter are partly based on the publication:

> **Evaluating the Benefits: Quantifying the Effects of TCP Options, QUIC, and CDNs on Throughput**
> *Simon Bauer, Patrick Sattler, Johannes Zirngibl, Christoph Schwarzenberg, Georg Carle*
> ACM/IRTF Applied Networking Research Workshop (ANRW 2023), San Francisco, USA, July 2023

The measurement approach for active Internet measurements on TCP performance, introduced in Section 6.1, and the implementation of the according measurement pipeline, described in Section 6.2, are based on the publication "Evaluating the Benefits: Quantifying the Effects of TCP Options, QUIC, and CDNs on Throughput" [99].
The analysis of a large-scale target crawl, presented in Section 6.3, and measurements on TCP option impact on throughput, discussed in Section 6.4 were conducted, respectively, repeated for this thesis. Further, as an extension of the named publication, this thesis surveys impacts by a so-called warm-up run, i.e., a download at the beginning of each measurement run per target, to ensure that downloaded files are cached by involved points-of-presence responding for domains hosted by CDNs.

6.1 Conducting Measurements with Public Web Servers

This section introduces an approach for active measurements with uncontrolled measurement targets. For our study, we download files provided by public web servers taken from Internet top lists with varying TCP options. This section describes the different steps of our active measurement approach, as illustrated in Figure 6.1, covering the crawling of measurement targets, conducting downloads of successfully crawled files, and extracting different performance indicators from initiated traffic.

Further, this section describes considered TCP option configurations, introduces the vantage points used, and elaborates on ethical considerations.

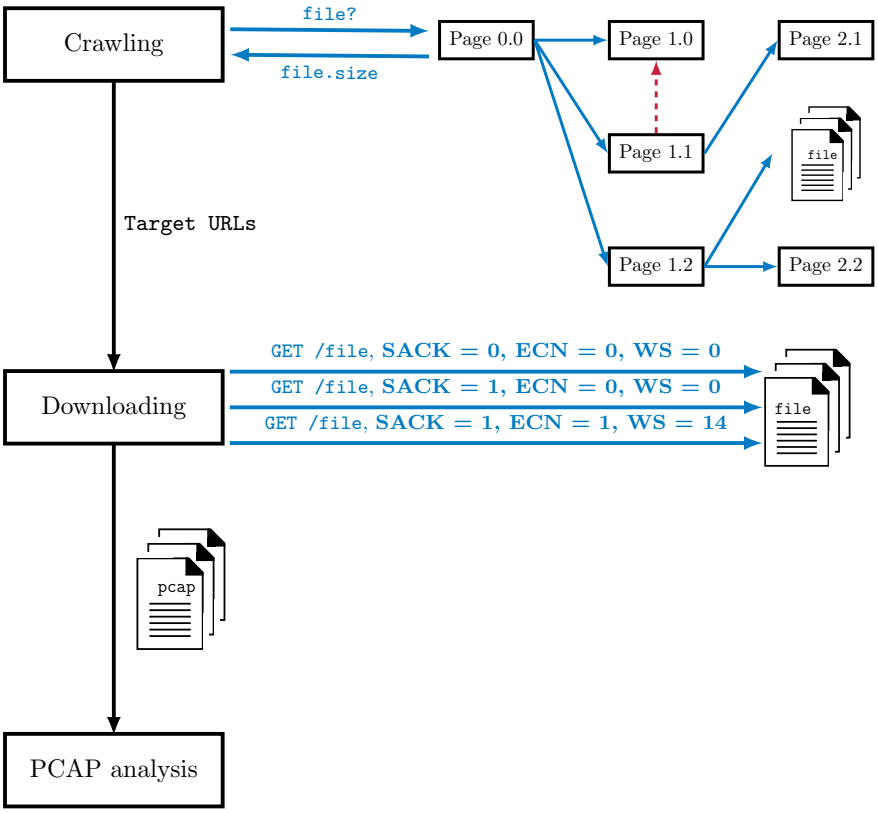


Figure 6.1: Illustration of the measurement pipeline used for Internet measurements with public web servers.

6.1.1 Crawling Measurement Targets

In order to conduct active measurements on TCP throughput on the Internet, we require suitable measurement targets. Thereby, publicly available, uncontrolled measurement targets providing productive web services are of interest in order to cover a varied mix that is representative for publicly used web services on the Internet regarding properties like operating system configurations, web server implementations, or deployed service infrastructures, Active measurements require such targets to be responsive to other hosts to initiate measurements. Accordingly, we decide to use web servers hosted on the Internet as they are *(i)* publicly available and *(ii)* responsive to HTTP(S) requests by clients, such as the used measurement host.

Internet top lists like the, meanwhile retired, Alexa Top 1M list [100] or Google's Chrome User Experience Report (CrUX) dataset [101] provide public domains chosen based on their estimated popularity. Accordingly, such lists are suitable for identifying publicly reachable measurement targets.

For our study, we refer to a suitable target as a server hosting a file satisfying a specific minimum file size, purposed to provide comparable results between downloads from different domains and to ensure a sufficient amount of generated traffic, for instance, to conduct capacity estimation, as surveyed in Chapter 7. We do not focus on certain kinds of files, like images, PDF files,

web resources, or content pages, but only require a specific minimum file size. In order to determine files for downloads, the crawler recursively follows links found on a domain's index page referencing the same domain and reached by the same IP as the initially crawled URL. In the following, we refer to the process of identifying files to be downloaded as crawling.

For crawling the target set for our measurements, we choose a minimum file size of 1 MB. While this is a relatively large file size considering the distribution of flow sizes observed in passive data sets as discussed in Chapter 3, the same study emphasizes the relevance of connections with such size regarding the total observed bytes.

6.1.2 Conducting Downloads

Afterward, we initiate downloads of the crawled files for each domain in the composed target set. Thereby, we compose measurement runs per target consisting of several subsequent downloads, for instance, considering different client configurations. In the following, one measurement iteration refers to one measurement run per domain in the target set. We consider different permutations of TCP option usage for downloads, introduced in Section 6.1.3. Further, we conduct a warm-up download to avoid bias between downloads of the same measurement run due to edge caching by CDNs. For our study, we run ten measurement iterations on all target domains. Before each download, we freshly resolve the target domain to ensure adaption to DNS-based load balancing. All downloads are conducted using HTTPS.

By design, the introduced measurement approach is limited to control configuration and conditions at the client side of a connection, i.e., the used vantage points. This implies that the target server and its characteristics, like operating systems, server implementations, or used congestion control algorithms, are not known. The same applies to load conditions on the Internet paths and at the target server.

Running downloads with different configurations back to back for one domain ensures that conditions on the network path and the server side are as similar as possible for all downloads of one run, e.g., regarding daytime patterns of service usage and corresponding load. Further, measured speed-ups by different configurations are compared within one measurement run.

6.1.3 Considered Option Configurations

To study the impacts of TCP options usage, we consider three options addressing the performance of TCP connections: window scaling (WS), selective acknowledgments (SACK), and explicit congestion notifications (ECN).

TCP window scaling [63] is proposed to exceed limitations of bytes in flight implied by the length of the `window` field in the TCP header from 65 kB up to 1 GB. TCP WS enables the exchange of a factor during connection establishment to shift all following values of the `window` field. Accordingly, the client can specify the scaling factor he will apply when advertising the receiver window. Selective acknowledgments [64] are purposed to avoid unnecessary retransmissions by explicitly specifying lost packets. This is achieved by exchanging ranges of sequence numbers of successfully received packets. Today, WS and SACK are enabled on all major operating systems like Linux, MacOS, or Windows [51]. However, using SACK for active measurements

Label	WS	SACK	ECN
BL (Baseline)	0	✗	✗
BL+	0	✓	✓
WS1	1	✗	✗
WS2	2	✗	✗
WS2+	2	✓	✓
WS3	3	✗	✗
WS7	7	✗	✗
WS7+	7	✓	✓
WS14	14	✗	✗
WS14+	14	✓	✓

Table 6.1: Considered TCP option configurations.

requires SACK support on the server and client side, signaled by the `SACK permitted` option. Explicit congestion notifications (ECN) [65] are a measure to avoid overload on the network and, accordingly, lost packets by enabling routers to signal congestion actively. As routers do not access Layer 4 headers, ECN relies on two bits each in a packet's IP and TCP header. ECN support is signaled during connection establishment. Note that we do not consider TCP Fast Open (TFO) [102] for this study. TFO is particularly effective if a client requests several sources from a server, resulting in several TCP handshakes, implying overhead. This use case does not match our approach to explicitly download single files from a target.

For our measurements, we consider different TCP option configurations, listed in Table 6.1. Next to a baseline run that does not support any option, we consider different window scaling factors announced by the vantage points and consider a second run with additionally enabled SACK and ECN for selected scaling factors. Note that the number behind the WS option label specifies the exponent used to calculate the maximum receiver window size by the formula $65.535\,\mathrm{B} \cdot 2^{x}$.

6.1.4 Vantage Points

As connection performance is expected to also depend on the location of the vantage point (VP), considering the distance to target servers and last-mile network conditions, we use three different vantage points for our measurements: First, a physical server located in a campus data center in Munich (MUC), and second, two virtual machines hosted by the cloud provider DigitalOcean in data centers located in San Francisco (SFO) and Singapore (SGP). The physical server hosted in Munich is connected with a 1 Gbit/s up- and downlink to the German science network (DFN) that connects to the Internet via a major Tier 1 provider. The measurement host is equipped with an Intel Xeon E5-2630 CPU providing six physical cores at a clock frequency of 2.6 GHz, 32 GB memory, and a Broadcom NetXtreme BCM5719 Gigabit NIC. The virtual machines hosted in SFO and SGP are equipped with two virtual CPU cores and 4 GB memory.

6.1.5 Ethical Considerations

Active measurements on public infrastructure like the Internet require responsible measurement practices. We followed a set of ethical measures, i.e., informed consent [103] and community

best practices [104] during all our measurements. The physical measurement hosts' IP addresses can be identified via reverse DNS or WHOIS information, while the measurement host operates an explanatory website. We maintain an abuse contact email and react quickly to all requests, including the option to exclude a domain or IP range from further measurements. We use a custom HTTP user agent to be identifiable as a research group and follow crawling instructions in the robots.txt according to the Robots Exclusion protocol [105].

6.2 Implementation

This section describes the implementation of the measurement pipeline used for Internet measurements with public web servers covering crawling files on measurement targets, running downloads of such files with varying TCP option configurations, and extracting performance indicators from captured download traffic.

6.2.1 Crawling

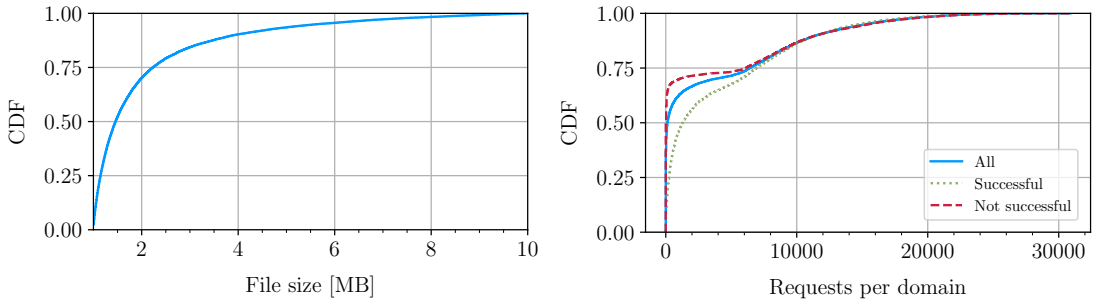
The crawling component sends HTTP HEAD requests in order to extract the optional HTTP Content-Length field [106], based on the Python crawling library *Scrapy* [107] to determine the size of a file. As the first tests showed that many potential targets did not provide Content-Length information, we implemented a fallback by starting downloads of files and stopping them if the minimum file size was successfully downloaded. The crawler allows configuring a maximum file size to avoid time- and resource-intensive downloads of huge files. The crawler keeps track of all already crawled URLs to avoid duplicated crawling.

In order to limit network load induced by crawling, we define several abort criteria. First, crawling of a target domain gets aborted if a certain number of files of sufficient size are found, which is, by default, five files. Second, the crawler enables limiting a crawl to a configurable maximum depth per target domain, where depth corresponds to the number of links followed to reach a certain URL. Further, the total number of requests per depth is limited by a configurable upper bound, which, by default, is 5000 requests.

6.2.2 Downloading

Crawling targets results in a list of domains and corresponding URLS of files suitable for our downloads. The download component iterates over the list of determined files and conducts a measurement run, i.e., a download with each specified TCP configuration per file. Before downloads are established, the downloader resolves the target domain's IP address to ensure an up-to-date resolution. Afterward, an HTTPS GET request is sent to the freshly resolved target IP with *wget2* [108]. We use the *ForcedIPHTTPS adapter* [109] to enable the use of specific IP addresses while establishing a TLS/SSL connection. Download traffic gets captured with *tcpdump* [110].

The downloader relies on the corresponding settings of the Linux kernel to configure TCP option usage. In particular, the downloader configures the Kernel's *net.ipv4.tcp_ecn* and *net.ipv4.tcp_sack* parameters to control the usage of ECN and SACK. Window scaling is enabled



(a) Sizes of detected target files.

(b) Total number of requests per crawled domain.

Figure 6.2: Cumulative distribution of sizes of detected target files and the total number of requests per domain.

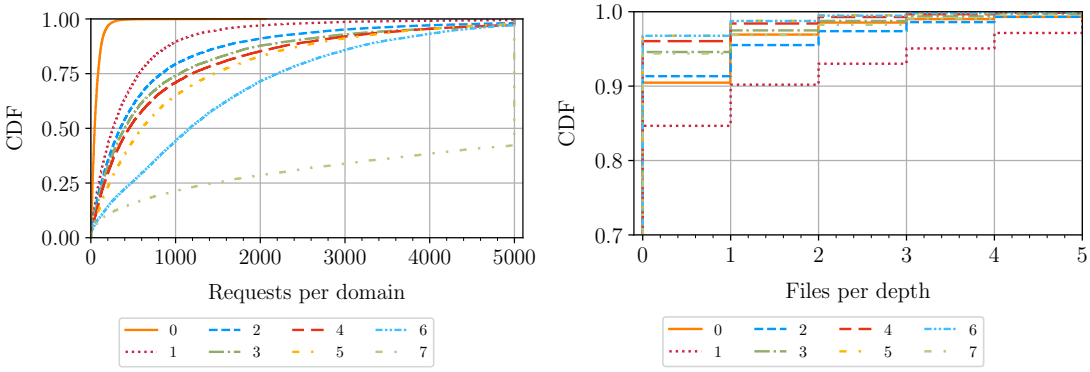
by the *net.ipv4.tcp_window_scaling* flag, while the maximum used scaling factor is limited by configuring the maximum receive memory for TCP connections. This ensures that the advertised receiver window does not exceed a certain size corresponding to the maximum scaled receiver window size according to the intended window scaling factor.

6.2.3 Traffic Analysis

We calculate the mean throughput (TP), mean round trip time (RTT), and retransmission rate (RR) for each connection. We refer to mean throughput as the fraction of transferred data and the duration of a connection. The amount of transferred data is determined by the sum of packet sizes of a connection, as specified by the *total length* field in IP packet headers. We calculate the RTT based on the TCP timestamp option, which enables matching tuples of corresponding packets based on *TSval* and *TSecr* values. The difference of corresponding packet timestamps then determines an RTT sample. We calculate the RR to assess packet loss, implying reduced sending rates due to congestion control. We determine packets as retransmissions with *tshark* [111]. The retransmission rate is then calculated as the fraction of data transferred by retransmitted packets and the total number of observed data.

6.3 On the Results of Large-scale Crawling

To analyze characteristics of the applied crawling approach regarding large-scale target sets, we conduct a crawl on 35 thousand domains taken from the CrUX data set [101], i.e., a CrUX-based top list publicly available on Github [112] accessed in September 2023. In particular, we crawl all domains of the top 10 thousand entries of the data set and 25 thousand domains taken from the rank range 10 thousand to 50 thousand. For crawling, we choose a minimum file size of 1 MB, as used for active Internet measurement studies presented in this chapter and Chapter 7. Further, we configure an upper bound of five files per domain, a maximum of 5000 requests per depth, and a maximum crawling depth of 8 levels. The maximum considered file size is 10 MB. Crawling was conducted in November and December 2023.



(a) Number of requests per depth per domain.

(b) Number of files per depth per domain.

Figure 6.3: Cumulative distributions of sent requests and identified target files per crawling depth per domain.

6.3.1 Crawling Characteristics

In total, 7700 out of the 35 thousand crawled domains result in at least one successfully identified file, corresponding to 22% of crawled domains. By analyzing the distribution of sizes of identified files, as shown in Figure 6.2, we find that around 70 % of files have a size between 1 MB and 2 MB, while less then 10 % of determined files are larger than 5 MB.

In order to assess the intrusiveness of crawling, we analyze the number of sent HTTP(S) requests per domain, as shown as cumulative distribution in Figure 6.2. For over 50 % of domains identified as suitable measurement targets, less than 1000 requests were sent, while over 80 % of such domains were crawled with less than 10000 requests. The distribution of requests per domain shows a long tail up to over 30 thousand requests. Further, we find that domains with the fewest sent requests tend not to result in successfully crawled files. In particular, we observe a large share of domains that did not result in a successfully crawled target file, while the domain was crawled with less than 250 requests. However, the cumulative distributions of requests per domain, shown in Figure 6.2, indicate that there are also domains with a large number of requests not resulting in suitable target files. The long tail distribution of requests per domain suggests implementing an additional total maximum of requests per domain, next to a maximum per depth, to limit the intrusiveness of crawling.

To further optimize future crawl runs and to assess the effectiveness of crawling for different depths, we survey the number of requests per depth and the number of files identified per depth. Figure 6.3 shows the cumulative distribution of requests per depth per domain. We observe a trend towards increased request counts with increased depth. As the crawler follows each link found on depth n, more URLs are likely crawled for depth $n + 1$, resulting in more requests. However, the distribution of requests sent for depth 7 shows a significant increase in domains reaching the maximum number of requests.

Further, we analyze the distribution of the number of files per depth per domain, as shown in Figure 6.3. Crawling on depth 0, i.e., the index website of a domain, depth 1, and depth 2, results in the most significant shares of domains providing at least one file. Depth 1 and depth

2 show the largest shares of domains, resulting in more than one file being found. We do not observe an increased probability of finding suitable files on depth 7 compared to other depths while crawling on depth 7 results in the largest share of domains reaching the maximum request count. This observation indicates that crawling with a decreased maximum depth improves the trade-off between intrusiveness and the effectiveness of the used crawling approach.

6.3.2 Properties of Measurement Targets

For performance measurements presented in Section 6.4, we use measurement targets based on crawling results of the top ten thousand entries of the mentioned CrUX-based top list [112]. We are interested in properties of such measurement targets that potentially affect conducted measurements, like the distance between vantage points and targets, in the sense of network hops, and the CDNs behind targeted web servers.

To survey hops between vantage points and measurement targets, we conduct measurements with tcptraceroutes [113]. We configure a maximum hop count of 30 and a timeout value of 60 seconds. Note that measurements were conducted in June and July 2024, i.e., after the performance measurements presented in Section 6.4 were conducted. For each measurement target, we conduct five traceroutes and determine the median hop count. Figure 6.4 shows the cumulative distribution of median hop counts for each vantage point. We observe a minimum hop count of 6 for the vantage point in Munich (MUC) and 7 for the vantage points in San Francisco (SFO) and Singapore (SGP). Around 50 % of targets are reached by at maximum one more hop compared to the minimum hop count. Afterward we observe a similar distribution for the vantage points in MUC and SFO, while measurements from SGP tend to result in increased median hop counts.

Next to the distance between vantage points and measurement targets, we are interested in CDNs potentially hosting targeted web services. Accordingly, we map domains to CDN providers based on the IP addresses observed for a domain, their mapping to the announcing autonomous system (AS) based on BGP dumps from a Route Views [114] collector, and a mapping of ASes to their respective organizations based on the work from Arturi et al. [115]. Table 6.2 shows the number of domains for the five most frequently observed organizations behind ASes of target IPs. Over 50 % of all considered measurement targets are hosted by Cloudflare or Amazon. Additionally, around 16 % of target domains are associated with Akamai, Fastly, or Google, while we observe a significant decrease of domains per organization within the top five organizations.

6.4 Active Measurements of TCP Throughput on the Internet

This section evaluates the throughput of TCP connections observed during actively initiated downloads of files hosted by public web servers. First, we introduce the used target set, i.e., a set of domains providing a suitable file for downloads in Section 6.4.1. Afterward, we compare performance indicators between used vantage points (VP) in Section 6.4.2. Section 6.4.3 surveys impacts by warm-up runs, which aim to ensure that downloaded files are cached by edge servers

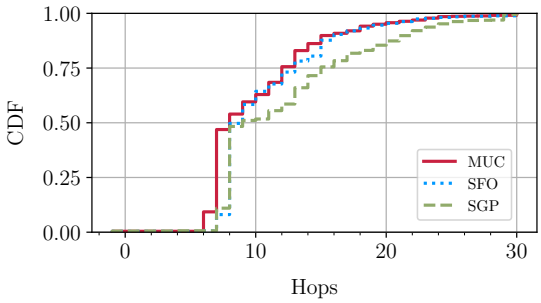


Figure 6.4: Cumulative distribution of median hop counts per measurement target. For each target, five measurements with tcptraceroute were conducted.

Provider	Count of targets
Cloudflare	767
Amazon	305
Akamai	177
Fastly	95
Google	65
Others	659
Total	2068

Table 6.2: Overview of the five organizations with the largest numbers of associated domains.

of CDNs for subsequent downloads. Further, Section 6.4.4 addresses the impacts of TCP options on throughput.

6.4.1 Measurement Targets

The used target set consists of crawling results based on the CrUX dataset [101], i.e., the top ten thousand entries of a CrUX-based top list publicly available on Github [112]. The mentioned top list aggregation is based on CrUX data published in September 2023. In particular, the target list corresponds to the crawling results of the top ten thousand domain crawl examined in Section 6.3.

We conduct two measurement series from each VP, each consisting of 5 measurement iterations per target. We observe that not all downloads from successfully crawled domains succeed,

Run	Successful DLs	BL domains	WS14 domains	WS14+ domains
MUC_1	88,964	1849	1849	1603
MUC_2	88,787	1824	1823	1584
SFO_1	85,760	1767	1767	1509
SFO_2	84,807	1740	1744	1487
SGP_1	82,095	1710	1708	1464
SGP_2	81,330	1685	1681	1442

Table 6.3: Conducted measurement iterations, the total number of successful downloads, and domains resulting in successful downloads.

e.g., because crawled files are no longer available. Table 6.3 lists the total number of successful downloads and the number of domains resulting in successful downloads for different option configurations for each measurement series. While we observe minor deviations of domains resulting in successful downloads between the two measurement series conducted from the same VP, there are more significant differences regarding measurement runs conducted from different VPs. Such observation might be explained by domains blocking download attempts from the VP, for instance, through human verification applied for selected IP ranges as used by Cloudflare [116].

Further, we find that the number of domains resulting in successful downloads is smaller for downloads intended to use ECN and SACK compared to domains with a successful baseline download. This does not imply that downloads were not conducted successfully but that the target server did not support ECN and SACK. Corresponding downloads are not considered for further analysis of the impacts of TCP options.

6.4.2 Comparison of Performance Indicators per Vantage Point

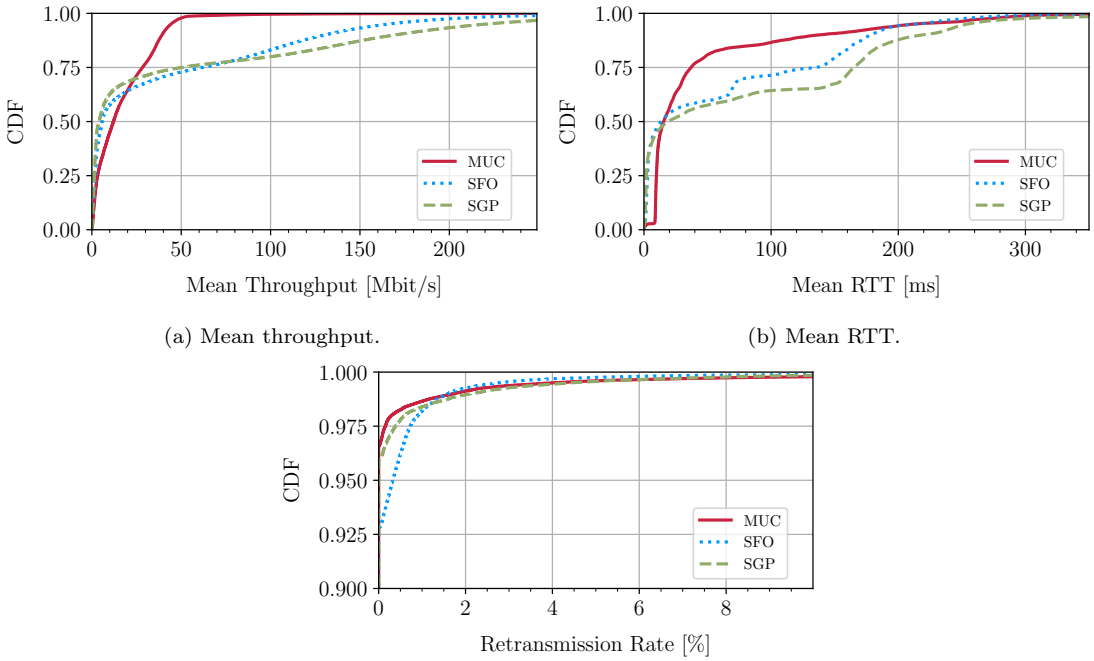
Conducting measurements from different vantage points suggests varying performance and potentially differing impacts by option configurations, for instance, due to different traversed Internet paths and distances between vantage points and measurement targets. Therefore, we compare extracted performance indicators of the used vantage points. Figure 6.5 shows the cumulative distribution function (CDF) of mean throughput, mean RTT, and retransmission rate for the three vantage points for all considered option configurations except warm-up runs. The VP in MUC results in a larger mean throughput up to about the 70th percentile of samples. Afterward, the VPs in SFO and SGP show larger shares of downloads with significantly increased mean throughput compared to the VP in MUC.

This observation correlates to the distribution of mean RTTs. In particular, we find that the VPs in SFO and SGP result in a significant share of mean RTTs smaller than 5 ms, which is not observed for the VP in MUC. We observed such a pattern in previous measurements as well [99], where we traced back such small RTTs to connections to targets hosted by CDNs like Akamai, Cloudflare, and partly Amazon. Such observation indicates that edge serves of such CDNs are located very close to the used vantage points in SFO and SGP, e.g., within the same physical data center. The VP in MUC shows smaller mean RTTs for most of the remaining samples.

Analyzing retransmission rates per VP shows that most downloads do not suffer from any retransmission. For downloads conducted from the VP in Munich over 92.5 % of connections result in a retransmission rate equal to zero. For downloads conducted from SFO and SGP, this observation applies to over 95 %. However, distributions of observed retransmission rates also show connections suffering from retransmission rates of several %.

6.4.3 Impacts by Warm-Up Runs

During previous measurements [99], we observe significant performance differences between the first download of a measurement run per domain and the subsequent downloads of the same measurement run, even when downloads are conducted with the same option configuration. This



(a) Mean throughput.

(b) Mean RTT.

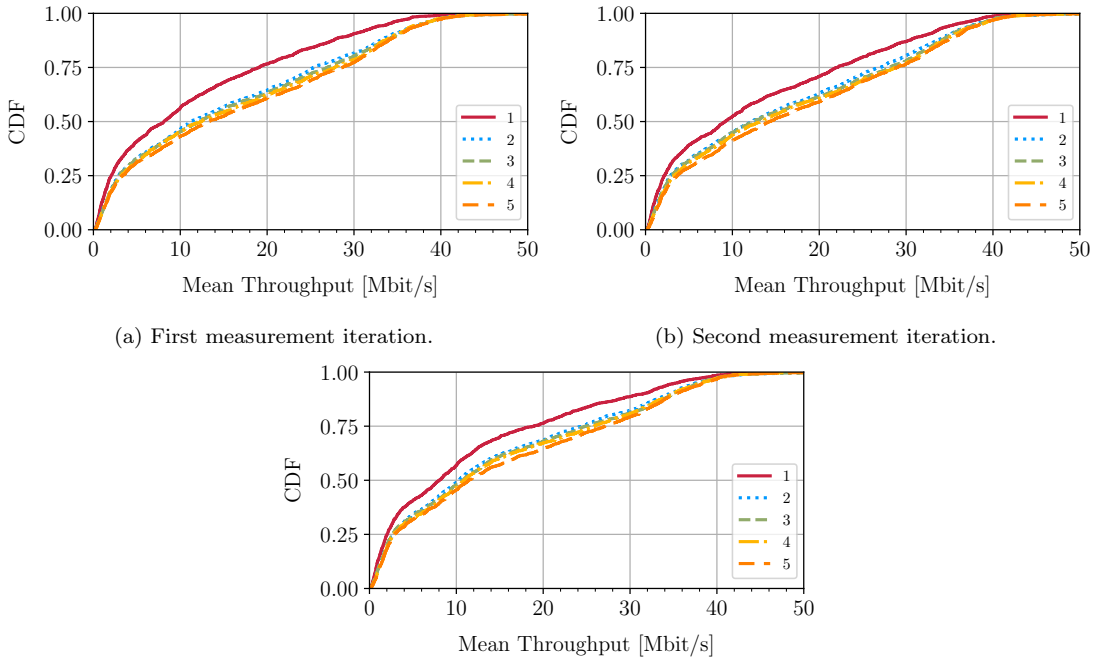
(c) Retransmission rate.

Figure 6.5: Cumulative distribution of measured performance indicators for each vantage point across all measurement iterations.

observation suggests impacts due to edge caching applied by CDNs hosting web services behind a domain to optimize performance, i.e., downloaded files get cached at an edge server after a download to faster provide the file for following requests of the same resource by users served by the same point-of-presence. In order to avoid bias by the order of downloads within one measurement run, we propose so-called warm-up downloads to ensure that all downloads with different option configurations are conducted based on cached files.

However, we are interested in the impacts of applied edge caching on observed performance. Accordingly, we conduct measurements only considering the baseline configuration, i.e., downloads without any supported TCP option. In particular, we conduct three measurement iterations, while each run per domain consists of five downloads with the baseline configuration. Note that such measurements were only conducted with the VP in Munich.

Figure 6.6 shows the cumulative distribution of mean throughput for each conducted download according to their order within the measurement run per domain separated for each conducted measurement iteration. For all measurement iterations, we observe a similar pattern, i.e., a significant right shift of throughput after the first conducted download. Further, examined distributions show further right shifts after the second conducted download correlating to the order of downloads, while such shifts are significantly smaller compared to the observed performance decrease after the first download.



(a) First measurement iteration.

(b) Second measurement iteration.

(c) Third measurement iteration.

Figure 6.6: Distribution of mean throughput measured during experiments only considering the baseline configuration. All measurements were conducted from the vantage point in Munich. Each measurement iteration consists of five subsequent downloads per domain.

As CDFs only represent performance across all conducted downloads, we additionally compare the throughput of downloads conducted subsequently within one measurement run directly. This is done by calculating the so-called speed-up factor describing the performance differences between two downloads of the same measurement run, i.e., we determine the speed-up factor as the fraction of the throughput of two downloads. Accordingly, the speed-up factor solves the formula $downloadB\_throughput = speed\_up\_factor \cdot downloadA\_throughput$. Table 6.5 shows buckets of speed-up factors determined between different positions of downloads of a measurement run. Note that we refer to a speed-up as positive (+) if the speed-up factor is larger than 1. Correspondingly, a negative speed-up (-) refers to a factor smaller than 1.

Analyzing speed-ups confirms significant shares of measurement runs with significantly increased throughput after the first download. Around 70 % of measurement runs show a positive speed-up between the first download (1) and the subsequent downloads (2, 3, 4, 5), implying that the subsequent speed-ups result in increased throughput. Around 20 % of measurement runs result in a negative speed-up after the first download between 0.9 and 1.0, indicating only a slight performance decrease. In contrast, over 30 % of such samples show a positive speed-up larger than 1.3, and over 20 % of samples show a positive speed-up larger than 1.5.

Further, we survey speed-up factors after the second download, i.e., we compare the performance of the second, third, fourth, and fifth downloads. As shown in Table 6.5, such downloads result in significant clusters of speed-up factors between 0.9 and 1.1 consisting of around 60 % of all

B	A	+	-	≤0.5	0.7 - 0.8	0.8 - 0.9	0.9 - 1.0	1.0 - 1.1	1.1 - 1.2	1.2 - 1.3	1.3 - 1.5	≥1.5
2	1	68.2	31.8	2.8	2.7	4.7	19.1	22.5	9.4	6.4	7.3	22.5
3	1	69.9	30.1	2.1	2.7	4.8	17.0	23.6	9.3	6.3	8.0	22.7
4	1	70.8	29.2	2.3	2.4	5.4	16.3	23.4	8.7	6.3	7.1	25.3
5	1	71.2	28.8	2.1	2.5	5.2	16.7	22.9	8.9	6.2	7.5	25.6
3	2	53.1	46.9	3.7	4.7	6.0	28.0	30.1	6.5	4.5	4.0	8.0
4	2	53.6	46.4	3.5	3.7	6.5	29.2	29.8	6.3	4.0	4.3	9.4
5	2	56.2	43.8	3.4	3.6	6.4	27.2	31.7	7.1	3.3	4.6	9.5
4	3	51.4	48.6	3.9	4.2	7.1	30.6	28.4	6.5	4.3	3.6	8.6
5	3	51.7	48.3	3.5	3.7	7.2	31.2	29.5	5.7	3.6	4.9	8.0
5	4	51.1	48.9	4.0	4.5	6.1	30.9	29.7	6.0	3.7	4.4	7.4

Table 6.4: Speed-ups between downloads conducted with the baseline configuration within one measurement run. All values are in %.

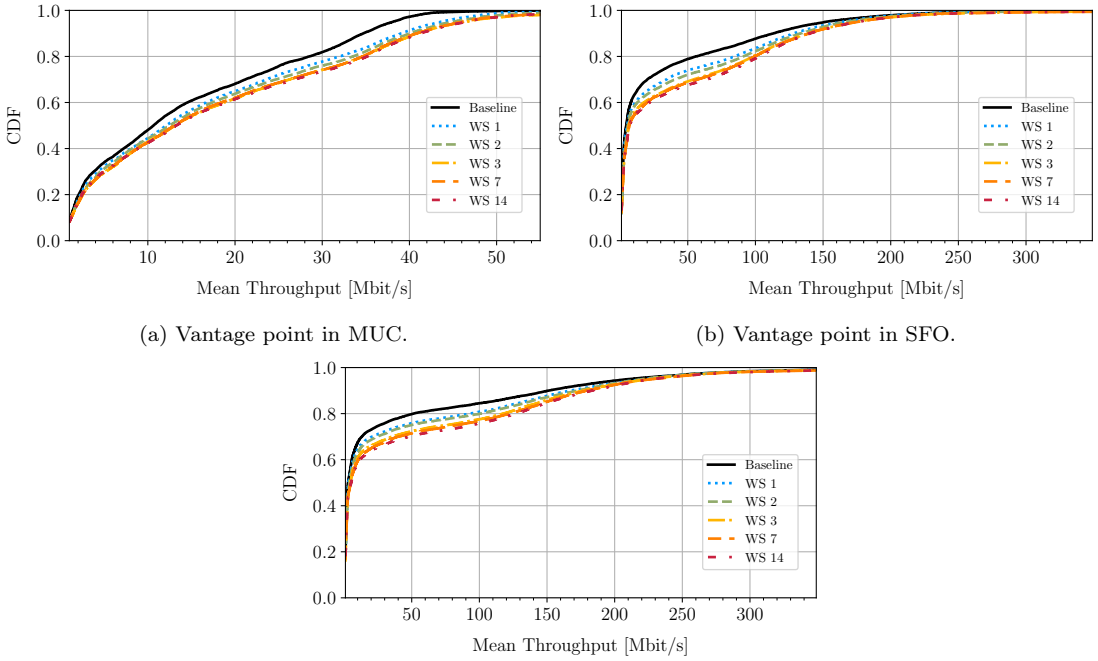
compared samples, while over 70 % of samples result in a speed-up factor between 0.8 and 1.2. Further, we observe nearly balanced shares of positive and negative speed-ups between such a subset of downloads with a slight tendency towards positive speed-ups, indicating that a higher position in the measurement run slightly tends to result in larger throughput. However, we still find shares of large speed-up values, i.e., between 3.4 % and 4 % of measurement runs result in a factor smaller than 0.5 and between 7.4 % and 9.5 % of samples result in a speed-up larger than 1.5.

We conclude that warm-up runs are necessary to avoid bias on subsequent downloads by caching downloaded files. Further, while we observe significant clusters, the throughput of downloads may vary significantly without any client configuration change, which needs to be considered for upcoming analysis.

6.4.4 Impacts by TCP Options

This section evaluates the impacts of the use of performance-related TCP options on mean throughput. We first survey impacts by different TCP window scaling factors and, afterward, the impacts by additionally using ECN and SACK. Measurements were conducted from three vantage points located in Munich, San Francisco, and Singapore. We conduct ten measurement iterations, i.e., ten measurement runs per domain. Note that downloads are conducted with a warm-up run, which is not considered for analysis.

Impacts by window scaling: We survey the cumulative distribution of mean throughput observed for measurements with a particular window scaling configuration, as shown in Figure 6.7 for each vantage point. Corresponding CDFs indicate that conducting downloads with the configuration WS1, as introduced in Table 6.1, significantly increases throughput compared to the baseline configuration, which does not support window scaling. Increasing the window scaling factor to 2 and 3 implies an increasing right shift of the cumulative distributions of through-



(a) Vantage point in MUC.

(b) Vantage point in SFO.

(c) Vantage point in SGP.

Figure 6.7: Distribution of mean throughput measured with different TCP window scaling factors.

put compared to the baseline, respectively, WS1. However, regarding the comparison between window scaling factors 3, 7, and 14, CDFs only indicate marginal performance improvements. This is reasonable, considering the relation between the file sizes used for conducted downloads and the maximum window size implied by such factors. A window scaling factor of 3 implies a maximum advertised receiver window of 524 kB while file sizes are mostly around 1 MB. This relation implies only a small potential for throughput being limited by the receiver window. A window scaling factor of 7, respectively 14, implies even larger maximum receiver window sizes, i.e., the whole file can be sent within one receiver window size. Accordingly, increasing the factor from 7 to 14 is expected to have no impact on measured throughput.

As analyzing cumulative distributions across several measurement runs and all targets does not reveal actual speed-ups by window scaling for individual samples, we calculate the speed-up factor between downloads conducted with different option configurations but within the same measurement run. As described above, we refer to speed-up as the factor describing the relation between the mean throughput of measurement samples observed for two different option configurations, i.e., speed-up described by the formula $second\_config\_throughput = speed\_up\_factor \cdot first\_config\_throughput$.

Table 6.5 shows buckets of speed-up factors for considered window scaling configurations to the baseline configuration. Doubling the maximum receiver advertised window compared to the baseline configuration, i.e., comparing WS1 throughput to BL throughput, implies positive speed-up for 63.9 % of conducted measurement runs. Running downloads with a window scaling

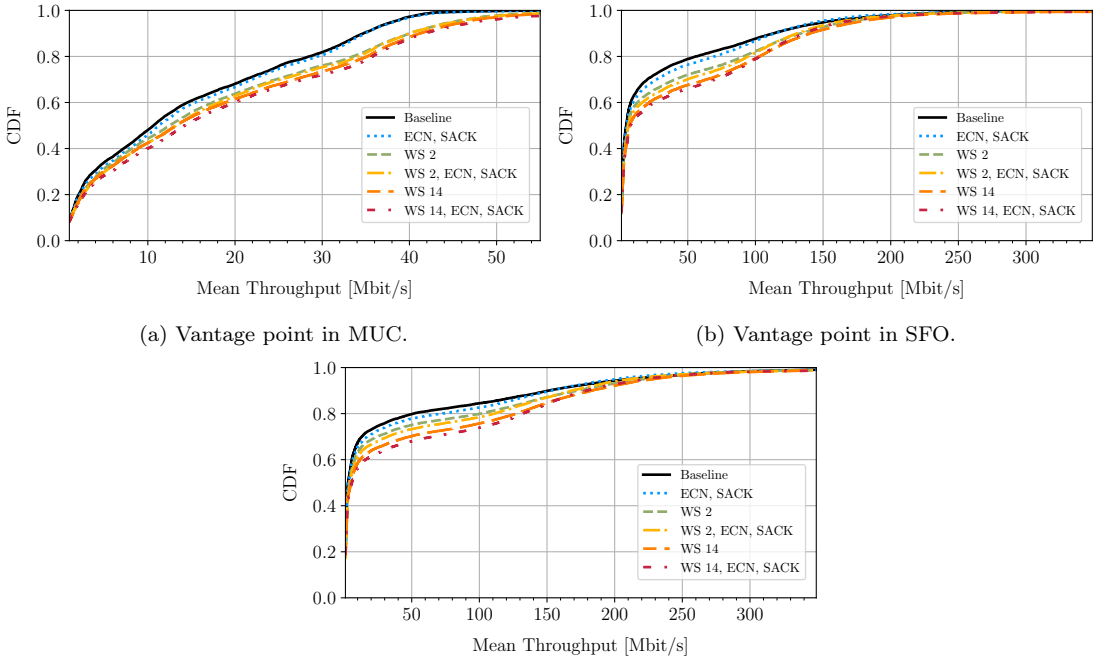
Conf.	vs.	+	-	*0.7 - 0.8*	*0.8 - 0.9*	*0.9 - 1.0*	*1.0 - 1.1*	*1.1 - 1.2*	*1.2 - 1.3*	*1.3 - 1.5*	*1.5 - 2*	*> 2*
WS1	BL	63.9	36.1	3.6	6.0	17.0	23.8	11.5	7.5	5.3	5.1	10.7
WS2	BL	67.0	33.0	3.5	5.6	15.3	18.7	10.4	7.6	7.0	10.8	12.5
WS3	BL	69.1	30.9	3.3	5.2	14.4	18.1	10.3	7.6	6.3	8.2	18.5
WS7	BL	69.3	30.7	3.4	5.2	14.3	18.1	10.2	7.7	6.2	7.4	19.7
WS14	BL	70.2	29.8	3.3	5.1	13.7	17.8	10.3	7.6	6.1	7.5	20.9
WS2+	BL	69.6	30.4	3.1	5.0	14.2	18.0	10.6	7.9	7.0	11.2	14.8
WS7+	BL	70.8	29.2	3.1	4.9	13.6	17.5	10.3	7.7	5.9	7.2	22.1
WS14+	BL	72.3	27.7	2.9	4.9	13.3	17.0	10.4	8.0	6.1	7.7	23.1
BL+	BL	53.9	46.1	3.8	6.2	25.3	26.1	6.1	3.5	4.4	4.0	9.8
WS2+	WS2	51.6	48.4	4.1	6.9	25.4	24.4	6.4	3.5	3.9	4.0	9.4
WS7+	WS7	50.8	49.2	4.2	7.5	25.5	24.7	6.5	3.6	4.0	3.9	8.1
WS14+	WS14	51.4	48.6	4.4	7.6	26.0	25.1	6.8	3.8	4.3	3.8	7.6

Table 6.5: Shares of downloads with a specific configuration (Conf.) resulting in a certain speed-up compared to another configuration (vs.) aggregated for all VPs. All values are in %.

factor of 2 (WS2) results in a positive speed-up for 67.0 % of samples compared to the baseline. Larger considered window scaling factors, i.e., 3, 7, and 14, show comparable positive speed-up factors between 69.1 % and 70.2 %. Thereby, around 40 % of measurement runs result in a speed-up factor larger than 1.2. Further, nearly 20 % of measurement runs result in a speed-up factor larger than 2, implying that mean throughput measured with WS3, WS7, and WS14 at least doubled compared to throughput measured with the baseline configuration. Such share is significantly smaller for throughput measured with WS1 and WS2, i.e., 10.7 %, respectively, 12.5 %.

Impacts by ECN and SACK: Next, we are interested in the impacts of enabling ECN and SACK on throughput. Again, we first analyze the cumulative distributions of mean throughput observed per VP to assess the general impacts of such options, as shown in Figure 6.8. In particular, Figure 6.8 shows the distributions of observed mean throughput for different window scaling configurations with and without ECN and SACK support. CDFs show that the distributions with and without ECN and SACK are quite similar, indicating only little impact by such additional used options, while throughput with ECN and SACK usage tends to slightly increase shares of downloads with larger throughput. As elaborated above, the most significant performance differences can be traced back to enabling TCP window scaling.

Analyzing corresponding speed-up factors confirms slightly increased throughput for downloads with ECN and SACK, as shown in Table 6.5. In particular, Table 6.5 compares the throughput of downloads with ECN and SACK, i.e., BL+, WS2+, WS7+, and WS14+, to downloads without both options, i.e., BL, WS2, WS7, and WS14. Comparing WS2+, WS7+, and WS14+ to the baseline download results in similar speed-ups as observed for the comparison of the baseline to WS2, WS7, and WS14, while all shares of positive speed-ups increase, despite the bucket 1.0 - 1.1. Conducting downloads with ECN and SACK support but without window scaling



(a) Vantage point in MUC.

(b) Vantage point in SFO.

(c) Vantage point in SGP.

Figure 6.8: Distribution of mean throughput measured with and without ECN and SACK.

results in 53.9 % of samples with positive speed-ups, while over 50 % of samples cluster between 0.9 and 1.1. For option configurations with enabled window scaling, we observe even smaller performance impacts by enabling ECN and SACK with positive speed-up shares between 50.8 % and 51.4 %.

In summary, these observations imply a significantly smaller impact by ECN and SACK compared to TCP window scaling. ECN and SACK are expected to improve the throughput of TCP connections in scenarios with network congestion by avoiding packet loss (ECN) or mitigating the impacts of packet loss (SACK) or in scenarios with other causes of packet loss (SACK). As elaborated in Section 6.4.2, analyzed download traffic only shows small shares of connections suffering from retransmissions, indicating that connections do not experience significant network congestion or that congestion is mitigated sufficiently by buffering middleboxes. However, it is likely that the measurement setups used impact retransmission rates, e.g., regarding the closeness of the vantage points used to the core of the Internet or last-mile network path capacities. This circumstance motivates conducting measurements on ECN and SACK effectiveness in more diverse setups, e.g., considering artificially induced packet loss.

6.5 Summary

This chapter introduced a measurement approach to conduct active TCP performance measurements with public targets on the Internet. The approach relies on crawling public web servers

VP	0	1	2	3 - 5	6 - 10	11 - 100	>100
ECN CWR Flags							
MUC	0.00 %	97.78 %	1.42 %	0.53 %	0.12 %	0.09 %	0.05 %
SFO	0.00 %	97.13 %	1.25 %	0.90 %	0.45 %	0.27 %	0.00 %
SGP	0.00 %	97.79 %	1.24 %	0.60 %	0.21 %	0.14 %	0.00 %
SACK Blocks							
MUC	92.32 %	2.90 %	0.96 %	0.94 %	0.54 %	1.29 %	1.05 %
SFO	80.67 %	10.12 %	3.28 %	2.44 %	0.97 %	1.72 %	0.79 %
SGP	90.35 %	4.14 %	1.17 %	0.92 %	0.59 %	1.77 %	1.06 %

Table 6.6: Share of connections with different numbers of observed CWR flags and SACK blocks. Note that only measurements with enabled ECN, respectively, SACK, are considered.

to determine files suitable for downloads to generate traffic for analysis. We conduct a crawl on 35 thousand domains taken from the top 50 thousand domains determined by Google's CrUX dataset [101] for files larger than 1MB. Such large-scale crawl results in a success rate around 20 %. Analyzing meta statistics of crawling shows that increased crawling depth tends to imply larger numbers of sent requests, i.e., increased intrusiveness, without increasing the probability of detecting files compared to less-deep crawling levels.

Based on a subset of successfully crawled public Internet targets, we conduct a measurement study on the impacts of edge caching and TCP option usage, considering window scaling, ECN, and SACK. During experiments without varying client configuration, we observe that around 70 % of measurement runs per domain result in a throughput increase after the first conducted download, while throughput is increased by a factor larger than 1.5 for over 22 % of samples. Such a significant performance increase is likely caused by edge servers of CDNs caching the requested file and providing the file faster for subsequent downloads. Accordingly, conducting warm-up downloads before downloads that are used for the further analysis of configuration impacts on throughput is strongly recommended. We find that window scaling has the most significant impact on throughput compared to other considered options by increasing throughput for nearly 70 % of samples with a scaling factor larger than two compared to the baseline configuration without any supported option. Further, sufficiently large scaling factors result in at least doubled mean throughput for nearly 20 % of samples compared to throughput achieved with the baseline configuration. Enabling ECN and SACK next to window scaling improves mean throughput for only small shares of samples. However, we also observe only small shares of connections suffering from retransmissions, indicating minor impacts by network congestion causing packet loss. To quantify the actual use of ECN and SACK during conducted measurements with enabled ECN, respectively, SACK, we analyze the number of ECN CWR flags and SACK blocks for each conducted download. Such numbers, as shown in Table 6.6, confirm that the majority of connections did not use CWR flags after the TCP handshake, respectively, SACK blocks. This circumstance should be taken into account when assessing the impacts of ECN and SACK.

6.6 Author's Contribution

The contents of this chapter partly build on the publication "Evaluating the Benefits: Quantifying the Effects of TCP Options, QUIC, and CDNs on Throughput" [99], which is a joint work by the author, Patrick Sattler, Johannes Zirngibl, Christoph Schwarzenberg, and Georg Carle. The author contributed significantly to the research objectives pursued, and the measurement approach introduced and conducted presented measurements. Further, the author implemented significant parts of the PCAP analysis and downloader component.

The author repeated all measurements for this thesis using the measurement pipeline introduced in the named publication. For the repetition of measurements, the measurement approach was extended for further TCP option configurations and warm-up runs. Further, the author conducted the surveyed large-scale crawl of potential measurement targets and examined the discussed characteristics of the crawl.

The described crawling approach and the corresponding implementation, as well as the initial version of the downloader component, were developed in the scope of the Master's thesis by Christoph Schwarzenberg [117] which was advised by the author, Benedikt Jaeger, and Patrick Sattler. The extension of the downloader component to support different TCP options was implemented in the scope of the Master's thesis by Mohammad Shaukat [118]. The author of this thesis supervised the named theses, which followed research objectives and methodologies specified by the author. The mapping of target domains to ASes and their organizations discussed in Section 6.3 was conducted by Johannes Zirngibl.

Chapter 7

Active Capacity Measurements

Capacity is a metric of major importance for service and network providers, applied for analysis of infrastructure deployments or measurements of available bandwidth. Further, as elaborated in Chapter 2, the capacity of network paths is used to detect throughput limitations by unshared bottleneck links. Accordingly, capacity estimation is a frequently addressed topic, resulting in different approaches and corresponding implementations, addressed in Section 2.4.

Packet-pair dispersion (PPD) is one of the most common capacity estimation approaches enabling single-ended, double-ended, and passive measurements, as the approach only requires inter-arrival times (IATs) between consecutively observed packets and corresponding packet sizes. Further, PPD-based capacity estimation is independent of the transport layer protocol used, enabling different kinds of traffic for actively conducted measurements. PPD-based estimates might be affected by interference with cross-traffic and buffering by middleboxes, as IATs of an analyzed packet flow may get compressed or expanded by foreign packets, resulting in falsified capacity estimates.

However, there are no recent studies on the accuracy of PPD-based capacity estimation in the Internet. At the same time, we assume a significant evolution of capacities, traffic volumes, and evolved queuing mechanisms since PPD-based capacity estimation was evaluated by related work between 2000 and 2010 [20], [21], [25]. This chapter addresses the accuracy of PPD-based capacity estimation in the Internet nowadays and presents active measurement results surveying end-to-end capacities of Internet paths. In particular, this chapter surveys the application of TCP download traffic and responses to bursts of ICMP Echo requests for PPD-based capacity estimation. To assess accuracy based on ground truth data, we conduct measurements in physical and emulated test environments and with controlled targets on the Internet. In addition to the accuracy of estimates, we survey the responsiveness of Internet routers to ICMP-based measurements. Further, we run measurements with public Internet targets to survey deployed capacities.

The contents of this chapter are partly based on the publication:

> **On the Accuracy of Active Capacity Estimation in the Internet**
> *Simon Bauer, Janluka Janelidze, Benedikt Jaeger, Patrick Sattler, Patryk Brzoza, Georg Carle*
> IEEE/IFIP Network Operations and Management Symposium (NOMS 2023), Miami, USA, May 2023

The approach for measurements in controlled setups, on hybrid Internet paths, and with public Internet targets, as described in Section 7.2, are based on the publication "On the Accuracy of Active Capacity Estimation in the Internet" [119]. The same applies to the description of the used implementation of the PPrate algorithm and traffic generation for ICMP-based measurements, introduced in Section 7.3, and the tool used to conduct Internet measurements, introduced in Section 7.3.2.
Measurements conducted in hardware-based test setups, discussed in Section 7.4.2, on hybrid Internet path, discussed in Section 7.5.1, and measurements on public Internet target, evaluated in Section 7.5.2 are also based on the named publication [119].
The contents of the named publication are extended by a technique proposed to mitigate compressed IATs due to interrupt coalescence, described in Section 7.3. The introduced technique is tested based on measurements with enabled and disabled generic receive offloading (GRO) conducted for this thesis, presented in Section 7.5.2. Further, this thesis presents an evaluation of the used implementation of capacity estimation in emulated test environments in Section 7.4.1.

7.1 Terminology and PPD-based Capacity Estimation

We refer to capacity as the minimum physical bandwidth of a network path, corresponding to the maximum achievable transmission rate. Considering a network path $\mathcal{P}$ consisting of n hops h_i between the sender $\mathcal{S}$ and the receiver $\mathcal{R}$, i.e., $\mathcal{P} = (h_1, ..., h_n)$, we define C_i as the capacity of the link between hop h_i and h_{i-1}. The capacity C of path $\mathcal{P}$ is then determined by the hop providing the smallest capacity C_i, i.e., $C_\mathcal{P} = \min\{C_1, ..., C_n\}$. We refer to the link with the smallest C_i as the narrow link of the path $\mathcal{P}$.

Packet pair dispersion is one of the most common approaches to capacity estimation [120]. PPD-based capacity estimation relies on pairs of packets p_0 and p_1 of packet size L sent consecutively without delay between each other. The inter-arrival time (IAT) between p_0 and p_1 measured at the receiver $\mathcal{R}$ is then referred to as dispersion. With the time interval between p_0 and p_1 at hop i referred to as Δ_i and $\Delta_1 = \frac{L}{C_1}$, $\Delta_i = \max(\Delta_{i-1}, \frac{L}{C_i})$. Accordingly, the narrow link determines the dispersion observed at $\mathcal{R}$ and allows to calculate the capacity of path $\mathcal{P}$ as $C = \frac{L}{\Delta_{narrow\_link}}$.

Researchers introduced different tools exploiting the advantages of PPD-based capacity estimation, like the independence of the used protocol, single-ended measurements, or utterly passive capacity estimation conducted on captured network traffic. However, PPD-based estimation is sensitive to interference with cross-traffic and corresponding queuing effects when packets get buffered. In particular, interference with cross-traffic potentially results in either compression

or expansion of IATs between packet pairs, implying falsified estimates. Next to PPD-based capacity estimation, variable packet size probing (VPS) is a frequently considered approach to capacity estimation [39], [121]. VPS-based approaches enable estimating capacities for single hops of a network path. This is achieved by actively probing capacity to each hop on the path. However, VPS-based estimates require active measurements and are sensitive to delays by store-and-forward devices [122].

As scalable measurements with reduced intrusiveness are of major interest for future research projects, and PPD-based estimation can be applied more flexibly, we focus on PPD-based capacity estimation for this work. However, determining the narrow link position on a path is also of interest to researchers and service providers. Therefore, we consider PPD-based measurements based on ICMP Echo requests described in Section 7.3.

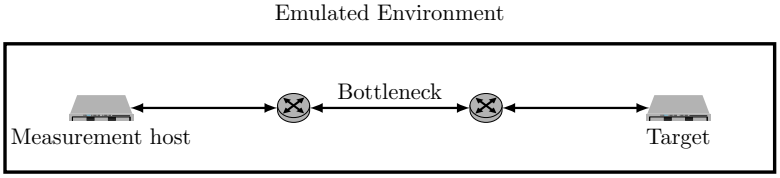
7.2 Measurement Approaches

One purpose of this study is to survey the accuracy of capacity estimates conducted in the Internet. We refer to accuracy as the deviation of an estimate to the actual capacity. Further, we are interested in end-to-end capacities observed during measurements with public Internet targets.

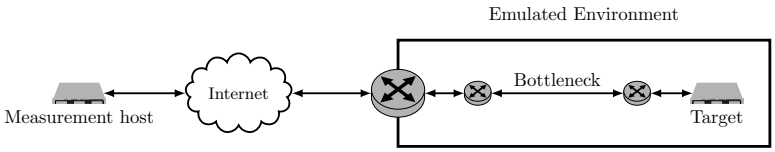
For our study on capacity estimation in the Internet, we surveyed the availability of capacity estimation tools in order to consider them for our measurements. We find that only *pathrate* is available and maintained [38]. While *clink* is also available, first measurements with the tool in controlled test environments did not result in meaningful results. *pbprobe* is also available for download but did not compile on different tested operating systems. Other tools either provide broken references to corresponding sources or were not published. Therefore, we implement capacity estimation based on the *PPRate* algorithm introduced by En-Najjary and Urvoy-Keller [20]. We choose *PPRate* as it is assumed to be robust regarding outlying dispersion pairs and allows flexible traffic generation for active measurements as it is a passive estimation tool.

To estimate capacities, we implement a tool based on the *PPrate* algorithm in Python. As *PPrate* is a passive estimation tool, we need to actively initiate traffic for our measurements. We generate TCP traffic by requesting files via HTTP and ICMP traffic by sending bursts of ICMP Echo requests and capturing corresponding ICMP Echo replies. Details on the implementation of passive capacity estimation and traffic generation are addressed in Section 7.3.

Evaluation in Controlled Environments: First, we conduct measurements in emulated test environments, as shown in Figure 7.1a, to validate our measurement tools and to assess the accuracy and robustness of PPD-based capacity estimation regarding varying network conditions. Running measurements in emulated environments enables the control of network properties like cross-traffic interference, packet loss, delays, or path lengths. Therefore, we implement a measurement framework providing automated testing of different network scenarios based on Mininet [123], as introduced in Section 7.4. As emulated environments imply limited maxi-



(a) Measurement setup with fully controlled network conditions.



(b) Setup for measurements with hybrid Internet paths.

(c) Setup for measurements with public target servers.

Figure 7.1: Considered measurement setups.

mum bandwidths, we additionally conduct measurements in physical setups with capacities of 1 Gbit/s, respectively 10 Gbit/s.

Measurements on Hybrid Internet Paths: Second, we conduct measurements on paths through the Internet while we control both endpoints and the first, respectively, last hop capacity implemented with emulated network segments, as illustrated in Figure 7.1b. These hybrid setups allow measurements with ground truth data to assess accuracy, while analyzed traffic is expected to be affected by varying conditions on the Internet paths. For our measurements, we use four different virtual nodes hosted in data centers in Munich (MUC), Helsinki (HEL), San Francisco (SF), and Singapore (SG), equipped with resources as shown in Table 7.1, resulting in a total of 12 paths considering both directions between all nodes.

Measurements with Public Internet Targets: Third, we conduct measurements with public and uncontrolled Internet servers to survey capacity deployments. Conducting measurements with uncontrolled targets requires traffic generation without access to the server side. Traffic generation with ICMP Echo requests is straightforward, assuming targets reply reliably. We establish downloads of files hosted by the measurement targets for TCP-based measurements. This approach requires crawling potential targets for suitable files, as introduced in Chapter 6.

7.3 Implementation

This section describes the implementation of tools used for conducted measurements, i.e., an implementation of the *PPrate* algorithm, generation of ICMP traffic, a tool to evaluate capacity

Name	Location	vCPU	Mem.
MUC	Munich, Germany	4	18 GB
HEL	Helsinki, Finland	4	8 GB
SF	San Francisco, USA	2	2 GB
SG	Singapore	2	2 GB

Table 7.1: Measurement hosts used for hybrid Internet measurements.

estimation in controlled environments, and a tool to conduct measurements on hybrid Internet paths.

7.3.1 Capacity Estimation and Traffic Generation

Passive capacity estimation based on the PPrate algorithm: We implement a tool based on the *PPrate* algorithm, introduced by En-Najjary and Urvoy-Keller [20], in Python3. *PPrate* is a passive capacity estimation tool applying analysis of packet pair dispersion and filtering of observed outliers. The tool expects either a traffic capture or a list of inter-arrival times of a connection with the corresponding maximum segment size (MSS) of the connection. En-Najjary and Urvoy-Keller propose to filter outlying IATs to prevent impacts by cross-traffic interference, the application layer, or other throughput limitations like flow control. In particular, *PPrate* filters IATs larger than $min(p75 + IQR, p95)$ and smaller than $p25 - IQR$ with IQR referring to the interquartile range of all observed IATs.

Based on filtered IATs, the algorithm determines a histogram of capacity probes calculated based on observed IATs δt and the MSS by the formula $C = \frac{MSS}{\delta t}$. The histogram is generated based on the configured capacity resolution, i.e., the used bin width for the histogram. The capacity resolution represents a trade-off between increasing the tool's numerical accuracy and the probability of getting multi-modal instead of uni-modal probe distributions [20]. As proposed by En-Najjary and Urvoy-Keller, we choose a bin width of 5 % of the IQR of observed IATs. Afterward, the histogram is analyzed for a single mode. If such a mode is detected, it directly determines the capacity estimate. If there are several modes, the *PPrate* algorithm tries to determine the so-called Asymptotic Dispersion Rate (ADR) by eliminating less significant modes. This is done by calculating the dispersion between trains of packets, i.e., between sequences of n packets. Thereby, n increases until the capacity probe histogram shows a single mode corresponding to the ADR. Afterward, capacity is estimated as the first mode larger than the ADR in the capacity probe histogram determined for packet pairs.

Due to our observations during Internet measurements, as presented in Section 7.5, we extend the *PPrate* algorithm by two features. The first feature is an option to skip the second phase of mode determination and instead choose the most significant bin of the capacity probe histogram, i.e., the bin with the most samples, as the final estimate. Accordingly, the adapted version does not determine the ADR as initially proposed for the second phase of *PPrate* but directly chooses the bin with the most samples. Second, our implementation enables the use of actual packet sizes instead of the MSS of a connection. Using observed packet sizes is purposed to counteract the effects of receive offloading, which implies merging TCP packets by the network interface card

(NIC) and, accordingly, larger packet sizes observed by the operating system. We determine packet sizes as the Ethernet frame size to estimate Layer 2 capacities.

Filtering IATs compressed by interrupt coalescence: As observed during measurements in controlled hardware setups, discussed in Section 7.4.2, and hybrid Internet path setups, discussed in Section 7.5.1, capacity estimates based on software timestamps by the operating system might be falsified by significantly compressed inter-arrival times. Such compression is caused by interrupt coalescence (IC) by the NIC. IC is a technique to avoid one hardware interrupt per packet by holding back the corresponding event for a certain time interval or number of received packets. When IC is applied, several packets are processed by the same interrupt, resulting in nearly equal software timestamps, as timestamps correspond to the point in time when the operating system processes the packets and not to the point in time when packets are received by the NIC. As a result, capacity might be highly overestimated.

To prevent such IC-caused overestimation, we consider a technique to aggregate packets that are likely processed by the same interrupt to one large packet after capturing. The technique relies on a fixed threshold value to classify IATs as a result of two packets processed by the same interrupt. If an observed IAT is smaller than the threshold, we sum up the packet sizes of both packets. Afterwards, a tuple of the aggregated packet size and the IAT between the first packet of the IC-batch and the last packet before such batch are used as input for the implemented *PPrate* algorithm.

Traffic generation for active ICMP measurements: While TCP-based capacity estimation is limited to estimating end-to-end capacity, it is also of interest which link on a network path is the narrow link, i.e., the link with the smallest capacity. Therefore, we are interested in the suitability of ICMP traffic, particularly ICMP Echo requests and corresponding Echo replies, analyzed with PPD-based capacity estimation. Relying on ICMP Echo requests enables measurements to be conducted in a traceroute-like manner, i.e., estimating capacity from the measurement host to each hop on the network path. Such a procedure results in an estimate for each hop, while the minimum estimate equals the end-to-end capacity.

One major aspect of this ICMP-based approach is the generation of ICMP Echo requests, as the intervals between two requests imply an upper bound to capacities that can be estimated. For instance, estimating a capacity of 1 Gbit/s requires sending ICMP Echo requests with intervals of at maximum 12 µs assuming packet sizes of 1500 B, as $\delta t = \frac{L}{C} = \frac{1500\,\mathrm{B}}{1\,\mathrm{Gbit/s}} \approx 12\mu s$. Therefore, we implement multi-threaded ICMP traffic generation in C based on preloading Echo requests in memory. After sending ICMP Echo requests, the tool captures corresponding Echo replies and extracts a list of inter-arrival times, which is then processed by our implementation of the *PPrate* algorithm.

7.3.2 Tools for Measurement Setups

We implement two independent measurement tools for our measurements with ground truth data. First, we implement a tool for measurements in emulated environments based on the network emulation tool Mininet [123] purposed to validate capacity estimation tools in various

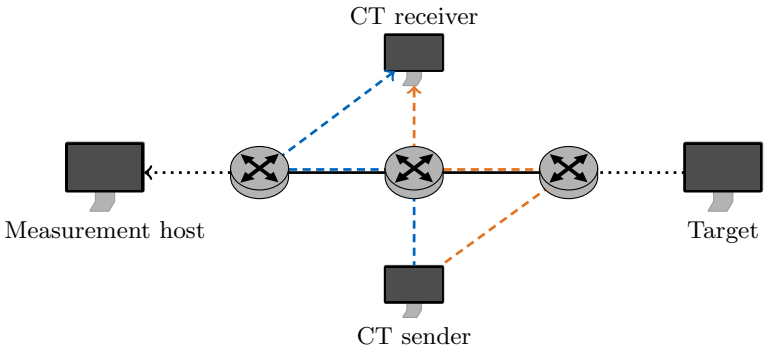


Figure 7.2: Topology for capacity measurements in emulated environments.

measurement scenarios considering network parameters like delays, packet loss, interference with cross-traffic, or narrow link capacity. Second, we implement a distributed measurement tool purposed to control first and last hop capacities in uncooperative environments like the Internet.

Testing in controlled environments: For evaluating capacity estimation tools in controlled environments, we implement an automated measurement orchestration tool based on the network emulation tool *Mininet* [123]. The tool generates network topologies consisting of a sender and a receiver interconnected by a chain of routers of variable length. Each router is connected to two further hosts proposed to generate cross-traffic flows as illustrated in Figure 7.2. *Mininet* implements links between nodes with its TCLink class, enabling configuring capacities, delays, and link loss. Cross-traffic nodes generate UDP and TCP traffic with *iperf3* [124], while the number of flows, used links, and cross-traffic bandwidths can be configured. While we implemented active ICMP traffic generation as described in Section 7.3.1, TCP traffic is generated by a file download conducted by the receiver (measurement host) from a Python3 web server running on the sender (measurement target). Traffic is captured using *tcpdump*.

Tool for hybrid Internet setups: As ground truth data regarding Internet paths' capacities is not generally available, we implement a distributed tool enabling setups with configurable first and last hop capacities on the sender and receiver, respectively. A tool setup consists of two endpoints, one running the client component of the tool and the other running the server component.

The client component is responsible for measurement orchestration and traffic initiation, while the server component runs a publicly reachable web server, which provides a file for downloads and responds to ICMP Echo requests. Both the client and the server component are capable of setting up *Mininet* topologies, which are connected to the Internet. In particular, the client- and server-side topology consists of a router, two switches, and a host, which is used as the measurement target when the framework runs in server mode. The router runs in the root namespace of the used measurement host and acts as a gateway between external networks, like the Internet, and the emulated topology. Address translation between the internal and external

network is implemented with the *iptables* utility. Further, the client and server share a dedicated communication channel implemented with *grpcio* [125], which is used for synchronization and exchange of configuration data, i.e., enabling the client to reconfigure server parameters. When running in the client mode, the framework establishes downloads to a specified server, sends trains of ICMP Echo requests, and conducts a traceroute measurement before each estimation run. For traceroutes, we rely on *dublin-traceroute* [126] providing detection of load balancing and NATing.

The client mode can be used to either interact with a running server component or with public and uncontrolled target servers. Traffic is captured at the receiver with *tcpdump*. Next to its bottleneck capacity, the client component enables the configuration of receiver offloading, the number of measurement runs per configuration, and a series of server configurations when intended to interact with a server component. When running the framework in server mode, a web server, implemented with *webfsd* [127], is set up and provides a downloadable file of configurable size for TCP-based capacity measurements. In the case of ICMP-based measurements, it is sufficient that the server responds to ICMP Echo requests. Further, the server component allows the configuration of the server-side bottleneck capacity between the switches of the emulated topology.

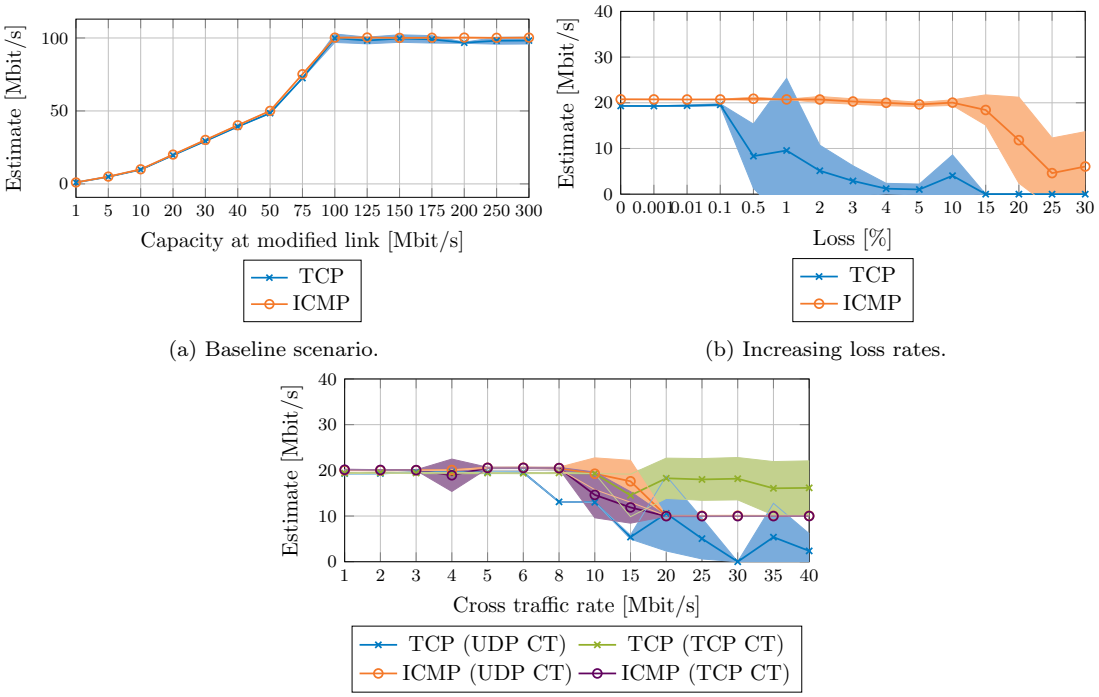
7.4 Measurements in Controlled Test Environments

This section presents the results of measurements conducted in controlled and isolated network environments to assess the accuracy of implemented capacity estimation. Thereby, we consider measurements in emulated environments and hardware-based measurement setups.

7.4.1 Measurements in Emulated Environments

We first conduct measurements in emulated network environments to evaluate implemented capacity estimation. Thereby, estimates' accuracy and robustness, i.e., the relative error and the deviation of estimates, are of interest. To survey the robustness of estimates regarding influences by the network, we conduct measurement series with varying packet loss and cross-traffic load. Results discussed in the following were measured with a path length of 5 links interconnected by routers and a delay of 5 ms per link. Based on conducted measurements, we find that results for larger chain lengths and higher delays do not impact observed accuracy until the device under test gets overloaded. For TCP-based measurements, we initiate TCP-based downloads, which are terminated after 5 s. We generate 1000 ICMP Echo requests with a size of 1500 B for ICMP-based measurements. Measurements were repeated ten times for each setup and configuration.

First, we conduct two baseline experiments: a baseline consisting of equal capacities on the whole path and a baseline consisting of a sequence of 100 Mbit/s links interrupted by a single link with varying capacity from 1 Mbit/s up to 300 Mbit/s in the middle of the path. Estimates of both baseline experiments confirm reliable estimation for both kinds of generated traffic, as shown in Figure 7.3 for the second baseline setup.



(a) Baseline scenario.

(b) Increasing loss rates.

(c) Increasing cross-traffic bandwidth.

Figure 7.3: Mean of estimated capacities and standard deviation in controlled test environments.

Second, we conduct measurements with varying loss rates configured on all links, while link capacity is 20 Mbit/s. We observe that TCP-based measurements show a drop in accuracy for lower loss rates than ICMP-based measurements. As shown in Figure 7.3, the ICMP-based measurements are robust to packet loss up to a loss rate of 10 % on each link of the path. For larger loss rates, accuracy decreases, accompanied by increasing deviation of estimates. We observe increasing deviations for estimates with loss rates between 0.5 % and 2 % for TCP-based measurements. Further, we find an increasing ratio of measurement runs not resulting in an estimate for larger loss rates, which we trace back to a lack of input data due to the heavily degraded TCP connection.

Third, we conduct measurements with TCP and UDP cross-traffic. We configure a path with a capacity of 20 Mbit/s for each link and generate one cross-traffic flow passing the whole path at different rates from 0 up to 40 Mbit/s. For estimates based on ICMP traffic, we find decreased accuracy for cross-traffic rates larger than 8 Mbit/s for UDP and 15 Mbit/s for TCP cross-traffic. For larger cross-traffic rates, ICMP-based estimates remain stable around 10 Mbit/s, indicating that single cross-traffic packets expand IATs between a pair of ICMP packets, thereby doubling IATs. For TCP-based estimates, we observe that the mean of estimates narrows the configured capacity even for larger TCP cross-traffic rates, while we observe increased deviation. Such robustness compared to other test cases can be explained by congestion control determining fair shares of available capacity for both flows, avoiding significant impacts by packet loss. However, UDP cross-traffic dominates the analyzed TCP flow for larger cross-traffic bandwidths, implying

significant underestimation and measurement runs that do not result in estimates due to a lack of input data.

We conclude that our implementation of the *PPrate* algorithm is capable of estimating capacity accurately. As expected, high packet loss rates and cross-traffic result in deviations of estimates from the actual capacity, while we observe protocol-specific patterns of deviations.

7.4.2 Evaluating Measurements of High Capacities

Next to the evaluation in fully emulated environments, we conduct measurements in controlled physical test setups providing capacities of 1 Gbit/s, respectively, 10 Gbit/s. These measurements are motivated by surveying potential performance limitations by our tools and the general accuracy of estimates regarding large capacities. For our measurements, we use two pairs of commodity hardware servers. Each of the hosts of the first pair is equipped with an Intel Xeon D-1518 CPU providing four physical cores, 32 GB of memory, and an Intel I210 NIC. Both hosts are interconnected by a 1 Gbit/s link. The second pair's hosts are equipped with Intel Xeon E31230 CPUs providing four physical cores, 16 GB of memory, and Intel X540-AT2 NICs, interconnected by a 10 Gbit/s link. TCP traffic is generated by the distributed measurement tool, as described in Section 7.3.2, while we disable the Mininet topology. We repeat measurements ten times for each setup and calculate the mean relative error of all estimates.

For TCP-based measurements we observe signed mean relative errors of −3.3 % in the 1 Gbit/s setup, respectively −6.6 % in the 10 Gbit/s setup, indicating underestimation. In the case of ICMP-based measurements, we find significantly larger mean relative errors of 70.3 %, respectively −76.6 %. Next to the significant errors of estimates, the difference between both ICMP-based measurement series is conspicuous as we observe apparent overestimation in the 1 Gbit/s setup and underestimation in the 10 Gbit/s setup. We trace back measurement errors of ICMP-based estimates in the 1 Gbit/s setup to active interrupt coalescence (IC), which is enabled on the used NICs by default. IC is proposed to reduce the computational load in interrupt-based network stacks by only performing interrupts when a certain number of packets is received or after a specific time interval. IC is known to impact network measurements, as surveyed by Salehin et al. [128], and impacts IATs between packets when capturing and time stamping is done in software, i.e., after the operating system receives packets from the NIC. However, TCP-based estimates do not suffer from a comparable significant impact of IC. We find that packet sizes of captured TCP traffic are significantly larger than the configured MSS. This observation can be explained by enabled generic receive offloading (GRO). GRO implies merging received TCP packets by the NIC, resulting in larger segments observed by the operating system. As our *PPRate* implementation takes actual packet sizes instead of the MSS of a connection, estimates based on merged packets are still approximating capacity.

Repeating measurements with disabled interrupt coalescence results in decreased mean relative errors for both kinds of generated traffic in the 1 Gbit/s setup, where ICMP-based relative errors improve significantly down to a mean of 1.8 %. Measurements with ICMP traffic in the 10 Gbit/s setup still show large relative errors with a mean of −80.5 %. This observation can be traced back to the performance limitations of the implemented ICMP load generation, which directly

impacts maximum capacity by the IATs between generated ICMP Echo requests. Estimating a capacity of 10 Gbit/s requires IATs to be at least smaller than 1.2 μs assuming a packet size of 1500 B, as $C = \frac{1500\,\mathrm{B}}{1.2\mu s} \approx 10\,\mathrm{Gbit/s}$. However, we observe that ICMP packets are generated with a mode of IATs around 5.5 μs in the 10 Gbit/s setup, while $C = \frac{1500\,\mathrm{B}}{5.5\mu s} \approx 2.2\,\mathrm{Gbit/s}$ implying significant underestimation.

7.5 Active Internet Measurements

As described in Section 7.2, we conduct Internet measurements with controlled and uncontrolled measurement targets to assess the accuracy and stability of capacity estimates in the Internet. Measurements are conducted with the measurement framework described in Section 7.3.2 in either dual- or single-ended mode.

7.5.1 Measurements on Hybrid Internet Paths

We set up the described measurement framework in client and server mode on the virtual measurement hosts, introduced in Section 7.2, to conduct measurements with two controlled endpoints and ground truth data. The narrow link is configured at the first hop on the path from the server to the client, while data packets, respectively ICMP Echo replies, are sent from the server, pass the emulated narrow link, the Internet paths, and finally, get captured at the client. Accordingly, packet pairs pass the whole Internet path after being dispersed at the narrow link on the server side. We conduct downloads of a 10 MB file for TCP-based measurements and generate bursts of 2000 ICMP Echo requests with a packet size of 1500 B for ICMP-based measurements. For each direction of the six paths between the four measurement hosts, we conduct measurements for 48 hours. We configure nine different capacities from 10 Mbit/s up to 200 Mbit/s, i.e., 10 Mbit/s, 25 Mbit/s, 50 Mbit/s, 75 Mbit/s, 100 Mbit/s, 125 Mbit/s, 150 Mbit/s, 175 Mbit/s, and 200 Mbit/s, while estimates for larger tested capacities suffered from performance limitations on most measurement hosts, respectively, are likely dominated by interrupt coalescence. For each capacity, we conduct seven measurements back-to-back before the following capacity is configured. This procedure is repeated for 48 hours, resulting in over 60 iterations, each comprising seven estimates per capacity for both kinds of traffic.

To survey the accuracy of conducted estimates, we calculate the mean and median of relative errors for each direction of a path. As we observe significantly compressed IATs, likely caused by interrupt coalescence, on setups with larger capacities, resulting in several magnitudes of overestimation, we further differentiate between two sets of estimates. First, a set of estimates for capacities smaller or equal than 150 Mbit/s and a set consisting of all estimates, i.e., capacities smaller or equal than 200 Mbit/s.

For capacities $\leq 150\,\mathrm{Mbit/s}$ we observe a mean relative error of 6.6 % for TCP-based measurements and 521.9 % for ICMP-based measurements across all considered paths. Such larger errors for ICMP-based estimates can be explained by receive offloading applied for TCP but not for ICMP, while ICMP-based estimates are assumed to suffer partly from IC. This explanation is confirmed by a decreasing number of observed TCP packets and packet sizes larger than the MSS with increasing capacity. Mean relative errors for capacities $\leq 200\,\mathrm{Mbit/s}$ are even larger as

TCP

Path	$C \leq 150\,\text{Mbit/s}$ Mean	$C \leq 150\,\text{Mbit/s}$ Median	$C \leq 200\,\text{Mbit/s}$ Mean	$C \leq 200\,\text{Mbit/s}$ Median
All	6.60	5.76	44.80	5.20
All \ SG	7.36	6.51	7.21	6.50
HEL ↔ MUC	6.02	5.43	5.78	5.42
SF ↔ MUC	7.11	7.12	6.66	7.12
SF ↔ HEL	9.07	7.55	9.30	7.55
MUC ↔ SG	5.20	2.75	156.40	2.08
SF ↔ SG	5.20	2.20	85.08	1.26
HEL ↔ SG	6.78	6.11	45.06	3.60

TCP (adapted algo.)

Path	$C \leq 150\,\text{Mbit/s}$ Mean	$C \leq 150\,\text{Mbit/s}$ Median	$C \leq 200\,\text{Mbit/s}$ Mean	$C \leq 200\,\text{Mbit/s}$ Median
All	-1.47	-0.86	35.81	-0.92
All \ SG	-1.15	-0.85	-1.50	-0.92
HEL ↔ MUC	-0.78	-0.76	-1.15	-0.79
SF ↔ MUC	-0.84	-0.80	-1.08	-0.87
SF ↔ HEL	-1.85	-1.34	-2.28	-1.59
MUC ↔ SG	-1.59	-0.95	150.99	-0.97
SF ↔ SG	-1.03	-0.84	75.00	-0.89
HEL ↔ SG	-3.43	-0.87	33.33	-0.93

ICMP

Path	$C \leq 150\,\text{Mbit/s}$ Mean	$C \leq 150\,\text{Mbit/s}$ Median	$C \leq 200\,\text{Mbit/s}$ Mean	$C \leq 200\,\text{Mbit/s}$ Median
All	521.89	-0.34	$> 1K$	0.04
All \ SG	-0.61	-0.70	994.55	-0.58
HEL ↔ MUC	-0.74	-0.79	481.05	-0.75
SF ↔ MUC	0.01	-0.39	$> 1K$	-0.02
SF ↔ HEL	-1.05	-0.77	750.46	-0.69
MUC ↔ SG	438.53	1.08	$> 1K$	2.70
SF ↔ SG	$> 1K$	2.93	$> 1K$	5.66
HEL ↔ SG	788.44	0.12	$> 1K$	0.73

Table 7.2: Mean and median relative errors in % for all considered capacities measured during hybrid measurements.

the impact of compressed IATs gets more significant. Further, we observe that measurements on paths to or from the measurement host in Singapore show more significant errors than the other paths regarding larger capacities. Mean relative errors without such paths decrease drastically for ICMP, resulting in a mean relative error of $-0.61\,\%$ for capacities $\leq 150\,\text{Mbit/s}$ across all paths. In general, medians of relative errors are more robust to outliers and, therefore, do not show such significant error rates as observed for the means of relative errors. However, we still observe median relative errors larger than $5\,\%$ for TCP-based estimates. Table 7.2 summarizes observed mean and median errors for all paths. Note that we merge results for both directions of each path.

As we observe more significant errors for TCP than ICMP-based measurements for capacities $\leq 150\,\text{Mbit/s}$ on paths without the measurement host located in SGP, we analyze histograms of bandwidth probes for both kinds of traffic. We find that our implementation of the *PPrate* algorithm does not detect uni-modal distributions for the vast majority of TCP-based measurement runs. Therefore, the *PPrate* algorithm determines the ADR and estimates the capacity as the first mode larger than the ADR. However, we observe that the ADR already approximates the actual capacity and that the first mode is larger than the ADR, which implies overestimation. Accordingly, we adapt our implementation of capacity estimation to directly estimate capacity based on the bin in the capacity probe histogram containing the most samples. As shown in Table 7.2, the adapted version of capacity estimation decreases means and medians of relative errors for TCP-based measurements, resulting in relative errors between $-1\,\%$ and $-2\,\%$ for most paths. Next to more accurate estimates, the adapted version also results in a more lightweight implementation of capacity estimation, as analysis of dispersion between packet trains of increasing length is not required. Accurate estimates based on the most significant bin were also observed by En-Najjary and Urvoy-Keller [25] for a significant share of measurements during a comparison of passive capacity estimation tools.

7.5.2 Blackbox Internet Measurements

This section evaluates the results of active Internet measurements conducted with public and uncontrolled web servers on the Internet. First, we evaluate capacity estimates of measurements conducted in December 2022, focusing on the median of estimated per measurement target. Second, we discuss measurement results based on a second target set conducted with and without GRO to evaluate the technique to mitigate IC impacts as introduced in Section 7.3.1.

Median Estimates to Survey Capacity Deployments

The first target set was generated by crawling 15 thousand sampled entries of the Alexa Top 1 Million list [100] accessed in September 2022. Domains were sampled from the Alexa Top 1 Million list considering different rank ranges, i.e., all domains within the top 1000 ranks, 5000 domains from rank range 1001-10000, 5000 domains sampled from ranks 10001 to 100000, and 5000 samples taken from the remaining ranks of the Alexa list. The crawl, conducted in November and December 2022, resulted in 3,736 measurement targets. The crawl was conducted with a minimum file size of 1 MB and a maximum depth of 4. For each target domain, we

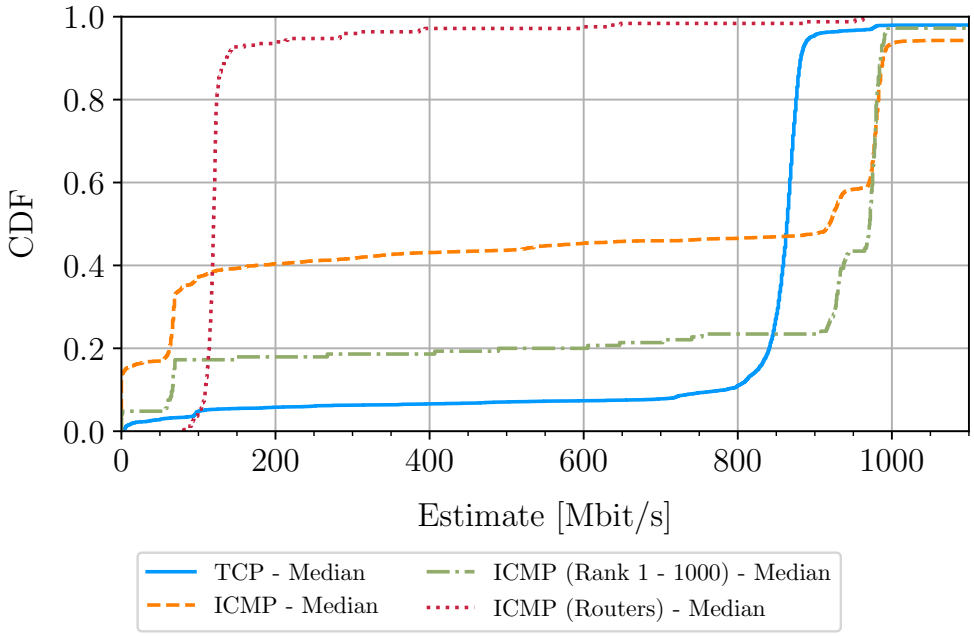


Figure 7.4: Median capacity estimates of measurements conducted with public web servers and routers.

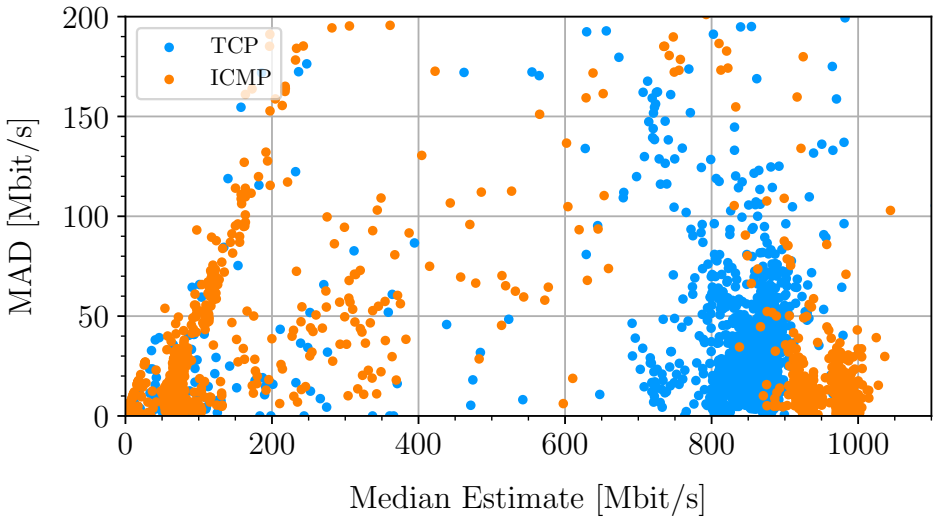


Figure 7.5: Median and MAD of capacity estimates observed during measurements with public web servers.

conduct 10 downloads to initiate traffic for TCP-based estimates and conduct 10 ICMP-based measurement runs by generating 300 ICMP Echo requests with a size of 1500 B for each run.

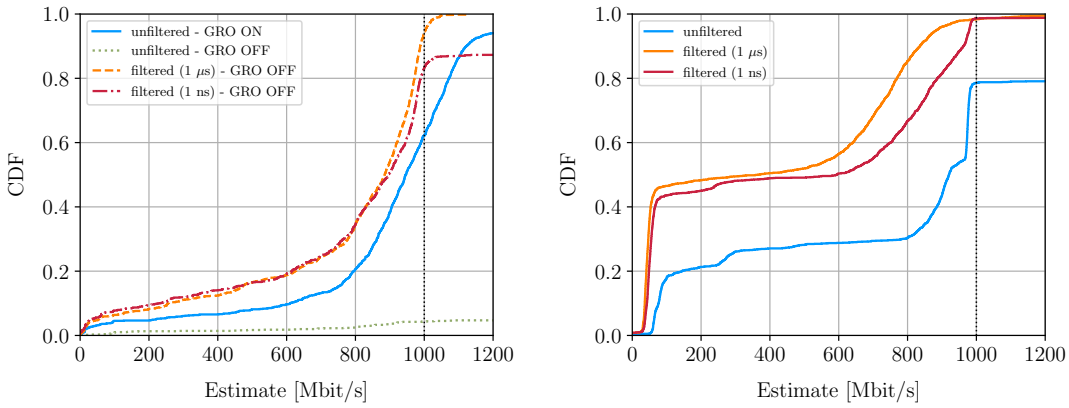
Measurements are conducted from a physical server hosted in a campus data center in central Europe, connected with a 1 Gbit/s up- and downlink to a national science network that connects to the Internet. The measurement host is equipped with an Intel Xeon E5-2630 CPU providing six physical cores at a clock frequency of 2.6 GHz, 32 GB memory, and a Broadcom NetXtreme BCM5719 Gigabit NIC. We configure an upper limit of 1 packet to delay an interrupt after receiving a packet on the measurement host to minimize impacts by IC and enable generic receive offloading.

TCP-based estimates per target show mostly means and medians between 800 Mbit/s and 900 Mbit/s, while less than 10 % of targets show significantly smaller estimated capacity, as shown in Figure 7.4 for observed medians. For ICMP-based measurements, we find significant outliers due to compressed IATs, likely caused by interrupt coalescence, despite the configuration of the measurement host, resulting in means larger than downlink capacity. In addition, we observe larger shares of median capacity estimates smaller than 100 Mbit/s, indicating underestimation due to artifacts like ICMP rate limiting by the measurement targets, as TCP-based measurements on the same path result in larger estimates. Considering only targets chosen from rank one up to 1000 of the Alexa top list, we observe larger shares of targets responding to ICMP Echo requests and larger shares of ICMP-based estimates approximating downlink capacity, implying differences in the handling of ICMP Echo requests by targets. We conclude that TCP-based estimates are more reliable and robust regarding impacts by measurement targets. Regarding the stability of estimates, we observe median absolute deviations (MAD) mostly smaller than 50 Mbit/s for TCP-based estimates, which also applies to ICMP-based estimates approximating downlink capacity, as shown in Figure 7.5.

One intention behind surveying capacity estimates based on ICMP Echo requests is conducting capacity measurements to intermediate hops on the path to a specific target. To survey the suitability of routers as targets for active capacity measurements, we conduct three ICMP-based measurement runs to over 250 routers observed during traceroutes to targets chosen from the first 2000 entries of the Alexa top list. We receive responses from 215 routers, while 52 do not respond reliably to ICMP Echo requests. Median estimates per router indicate a significant underestimation of several hundreds of Mbit/s for most targets, as shown in Figure 7.4, resulting in a cluster of median estimates around 100 Mbit/s. Such observation can be traced back to ICMP rate limiting, which was observed on the Internet before [129], [130].

Analysis of GRO Impact and IAT-filtering

During capacity measurements based on the Alexa-based target set described above, we observe significant impacts on ICMP-based estimates by applied IC, whose impact is mitigated by GRO for TCP-based estimates. Similar impacts on capacity estimates were observed during active QUIC-based Internet measurements conducted in the scope of the Master's thesis by Niklas Beck [131].



(a) Mean of TCP-based estimates per target.

(b) Mean of ICMP-based estimates per target.

Figure 7.6: Cumulative distribution of mean capacity estimates per target for TCP-based estimates and ICMP-based estimates conducted with enabled and disabled GRO.

To further investigate the impacts of GRO on TCP-based capacity estimates and the potential benefits from IAT-filtering, respectively packet merging, as described in Section 7.3, we conduct further Internet measurements with public web server targets. We conduct measurements with the target set also used for active throughput measurements as introduced in Section 6.4 and run measurements with enabled and disabled GRO. Measurements are conducted from the same measurement host as used for capacity measurements with the Alexa-based target set, implying a downlink capacity of 1 Gbit/s. Again, we configure an upper limit of 1 packet to delay an interrupt after receiving a packet on the measurement host, which is introduced above. We run five measurement runs for both protocols, TCP and ICMP, per GRO configuration (enabled, disabled). ICMP-based measurements are conducted with bursts of 1000 ICMP Echo requests. After capturing initiated traffic and extracting IATs and packet sizes, we apply the merging of packets based on IAT thresholds of 1 µs and 1 ns. An IAT of 1 µs and a packet size of 1500 Byte implies a capacity of 12 Gbit/s and an IAT of 1 ns corresponds to a capacity of 12 Tbit/s. Accordingly, filtering does not affect IATs implied by a path's end-to-end capacity, which is limited to 1 Gbit/s by the measurement host's downlink.

Figure 7.6 shows the cumulative distributions of the mean estimates per target. Regarding TCP, we find that measurements with disabled GRO suffer significantly under overestimation with only a small share of means smaller than the setup's bottleneck capacity. In contrast, mean estimates conducted with GRO enabled cluster between 800 Mbit/s and 1200 Mbit/s. Applying merging of packets with IATs smaller than 1 µs on estimates conducted with disabled GRO significantly mitigates observed overestimation of unfiltered estimates. Resulting in over 60% of means between 800 Mbit/s and 1000 Mbit/s. However, compared to estimates with enabled GRO, 1 µs-filtered estimates show increased shares of underestimating mean estimates. Decreasing the threshold to filter IATs to 1 ns results in mean estimates indicating larger shares of overestimation, compared to filtering with 1 µs. This implies that IATs between packets processed by the same batch are partly larger than 1 ns and that compressed IATs are not

filtered reliably. However, filtering still implies significant improvement of estimates compared to results with disabled GRO and unfiltered IATs.

For ICMP-based estimates, the merging of packets based on IAT thresholds results in a significant shift of mean estimates towards smaller capacities for both considered thresholds. A significant share of means suffering from outliers is mitigated by filtering, while we observe significantly increased shares of mean estimates smaller than 100 Mbit/s. Further, unfiltered estimates show about 25% of estimates close to 1 Gbit/s, which is not observed for filtered estimates.

We conclude that filtering of IATs is a promising approach to enable capacity estimates in setups without GRO and with applied IC. However, so far, we have only tested the approach on one measurement server. Assessing the approach on different hardware and surveying the impact of the applied filter threshold in more detail are proposed as future work.

7.6 Summary

This chapter presented measurement results of different experiments to survey the accuracy of PPD-based capacity estimation in the Internet. We implement capacity estimation according to the *PPrate* algorithm and conduct active measurements by initiating TCP downloads and generating bursts of ICMP Echo requests.

Conducting active measurements on hybrid Internet paths providing ground truth data confirms the general accuracy of PPD-based measurements in the Internet, while we observe significant relative errors for ICMP-based measurements affected by interrupt coalescence. In the case of TCP traffic, the impact of interrupt coalescence on estimates is mitigated by receiver offloading.

We conduct Internet measurements to over 3500 public web servers to study capacity estimates and their characteristics. Conducted TCP measurements show large shares of median estimates per target approximating path capacities of at least 1 Gbit/s. For nearly 50 % of measurement targets, we find significant differences between TCP- and ICMP-based median estimates, indicating underestimation of several hundreds of Mbit/s for ICMP-based measurements while remaining ICMP-based estimates are close to 1 Gbit/s. Further, we observe that routers are no suitable measurement targets due to ICMP rate limiting increasing inter-arrival times.

Studying mean estimates of measurements repeated on a second target set considering disabled GRO and merging packets based on an IAT threshold indicates increased accuracy for filtered IATs compared to unfiltered IATs for measurements with disabled GRO. This motivates the application of packet merging in setups without GRO and software timestamps of packets. Regarding ICMP, filtering IATs does not show such an improvement of mean estimates, as overestimation is mitigated, but filtering also results in more significant shares of underestimating means. Evaluating the approach on different hardware and surveying the impact of applied thresholds in more detail are proposed as future work.

7.7 Author's Contribution

This chapter builds on the publication "On the Accuracy of Active Capacity Estimation in the Internet" [119] which is a joint work by the author, Janluka Janelidze, Benedikt Jaeger, Patrick Sattler, Patryk Brzoza, and Georg Carle. The author significantly contributed to the publication regarding research objectives, used methodologies and measurement setups, conducted measurements, and the evaluation of measurement results. The author conducted Internet measurements with public Internet targets and conducted measurements in emulated environments.

The tool to set up hybrid Internet measurements was implemented as part of the Master's thesis by Patryk Brzoza [132] and was extended for the evaluation of capacity estimates in the scope of the Bachelor's thesis by Janluka Janelidze [133]. Results of measurements on hybrid Internet paths and in physical test environments were conducted in the scope of the Bachelor's thesis by Janluka Janelidze [133], based on close coordination with the author of this thesis, who also further evaluated the measurement results for publication.

The Python implementation of the *PPrate* algorithm resulted from the Bachelor's thesis by Patryk Brzoza [134]. Traffic generation for ICMP-based measurements was implemented as part of the Bachelor's thesis of Ferdinand List [135]. The author of this thesis conducted measurements in emulated environments using the measurement framework contributed by the Bachelor's thesis of Eric Rosche [136]. The named theses build on research objectives and approaches specified by the author, who also supervised the named theses.

The tested IAT-filtering technique was developed in the scope of the Master's thesis by Niklas Beck [131], based on ideas by Johannes Zirngibl and the author who advised the thesis. Measurements repeated with different GRO settings and the application of the filtering technique were conducted by the author of this thesis.

Part IV

Conclusion

Chapter 8

Conclusion

This chapter summarizes the key findings of this thesis regarding passive and active measurement results and corresponding measurement approaches according to the research questions introduced in Chapter 1.

First, this chapter presents findings regarding passive measurements, i.e., regarding the research objective

RO 1 Large-scale passive measurements on throughput characteristics of TCP connections on the Internet

Second, this chapter summarizes findings regarding active measurements according to the research objective

RO 2 Conducting active measurements of throughput and throughput-related metrics on the Internet

RQ 1.1 How did TCP flow characteristics and their correlation evolve on the Internet?
To survey the evolution of flow characteristics of TCP traffic captured on the Internet, we conducted passive measurements based on Internet traffic captured between 2008 and 2019 by CAIDA [55], consisting of over 2.5 billion TCP flows.

Our measurements show that heavy hitter connections are responsible for the transmission of significant shares of totally observed data, while there are no trends regarding their significance over time. Across the data set, connections with a flow size larger than the 99th percentile of all flow sizes transmit nearly 90 % of observed data. Heavy hitters regarding flow duration carry over 40 % of observed data. Heavy hitters regarding flow rates are responsible for 55.9 % (Chicago traces), respectively 68.0 % (New York traces). Further, we find that the majority of heavy hitters regarding flow rate are also heavy hitters regarding flow size. This observation is confirmed by a strong positive correlation between flow size and flow rate, which was also observed by related work in 2002, respectively 2006 [2], [5].

The importance of small shares of flows regarding data transmission becomes especially evident when analyzing the intersection of the three kinds of heavy hitters, i.e., 0.009 % of TCP connections transmit 19.9 % of data observed for traces captured in Chicago and 0.005 % of connections transmit 31.4 % of data observed for traffic captured in New York. Analyzing heavy hitters in a second data set consisting of over 2600 packet captures provided by MAWI [56] confirms the apparent dominance of the biggest flows and shows increased shares of bytes transmitted by heavy hitters regarding flow rate.

RQ 1.2 What limits the throughput of TCP connections on today's Internet?

To analyze the throughput limitations of TCP connections on today's Internet, we conducted an analysis on a large-scale data set of Internet traffic captured by MAWI [56]. In particular, the data set consists of over 20.34 billion TCP connections transmitting 54 TB of data captured on over 2600 days between 2016 and 2023. For the analysis of throughput limitations, we implement root cause analysis of TCP throughput based on the RCA approach introduced by Siekkinen et al. [11], [35].

We find that the most significant throughput limitations regarding connection duration are the sending application and shared bottleneck links. Thereby, limitation by the sending application limits 45.4 % of connection duration across the data set. Considering the share of bytes limited by a specific root cause, we find that unknown, respectively mixed, limitations are the most significant limitation by limiting the transmission of 31.6 % of observed bytes. The second most significant share are bytes limited by the transport layer (22.4 %), followed by bytes limited by shared bottleneck links (19.3 %) and unshared bottlenecks (17.3 %). Further, we find that limitations by the receiver window are negligible regarding connection duration and transmitted bytes.

Analyzing the evolution of the shares of duration, respectively bytes, by each root cause indicates network-dependent, respectively capture point-dependent, characteristics. For instance, we observe a period of significantly increased shares of shared bottleneck limitations between mid-2016 and the end of 2017, indicating that analyzed connections suffer significantly from packet loss. Further, we observe an abrupt increase of bytes limited by unshared bottleneck connections since 2023, implying an increase by more than 100 % compared to 2022.

RQ 1.3 How frequently used are throughput-related TCP options on the Internet?

To assess the use of throughput-related TCP options on the Internet, we surveyed connections supporting TCP window scaling (WS), selective acknowledgments (SACK), and explicit congestion notifications (ECN) based on the MAWI data set described above. Across the data set, 68.29 % of TCP connections use TCP WS, while we observe decreasing shares of usage over time. However, such connections carry over 96 % of observed bytes, implying that window scaling is commonly used for data transmission on the Internet. In contrast, our measurements indicate that SACK and ECN are not commonly used on the Internet. The share of connections supporting SACK is significantly smaller compared to the use of window scaling, i.e., 26.13 %

across the data set transmitting 37.99 % of observed bytes. Further, we observe that 3.02 % of connections support ECN usage at both endpoints, which transmit 5.87 % of observed bytes.

Our measurements indicate significant shares of connections not supporting options that likely are part of Internet measurement experiments or scans, relying on a complete TCP handshake and result in connections with a minimum duration of 100 ms, which are applied filter criteria for connections considered for analysis. For instance, we observe decreasing shares of connection supporting TCP WS and the TCP timestamp option over time. Further, the decrease in the share of connections supporting such options does not correlate to a decrease of transmitted bytes with such options over time.

In contrast to actual option usage, the deployment of considered TCP options on Internet servers is widespread, as we observed during an active measurement on all Alexa Top 1M list domains in February 2023. We observe that ECN is supported by 85.8 %, SACK by 91.4 % and WS by 91.1 % of the domains by establishing a TCP handshake to the index page of each domain.

RQ 1.4 How to monitor TCP throughput limitations in real-time?
While related work introduces different approaches to the analysis of throughput limitations of TCP connections, the online monitoring of throughput limitations is only considered by Ghasemi et al. [4] and Sun et al. [23], limited to traffic analysis on the data plane, respectively requiring access to the sender side's network stack.

In this thesis, we introduced an approach for the online monitoring of TCP throughput limitations on commodity hardware considering several Gbit/s of monitored bandwidth. The approach builds on a sliding window of packet data extracted from the last seen packets. Such data is then analyzed based on the RCA approach introduced by Siekkinen et al. [11], [35] in fixed time intervals.

To assess the feasibility of throughput limitation monitoring in productive environments considering several Gbit/s of monitored bandwidth from a performance perspective, we implement a proof of concept version of our approach based on the packet capture and traffic analysis tool FlowScope [73], [74]. We conduct measurements with a generated data set and observe up to 8 Gbit/s of monitored traffic consisting of several thousand concurrent flows with commodity hardware providing 32 virtual cores.

RQ 1.5 How to classify TCP throughput changes with TCP throughput limitations?
While related work applies TCP RCA on whole connections or periods of connections, this thesis surveyed the classification of individual changes in throughput with changes in throughput limitations. As a first step towards such classification, we introduce an approach to classify throughput changes, detected by applying change point detection on time series of throughput, with changes between application limited periods and bulk transfer periods as introduced by Siekkinen et al. [35].

A first case study on a packet capture provided by MAWI shows the significant impact of the changepoint detection method and its configuration on detected throughput changes. This

also implies that varying shares of throughput changes are classifiable with changes between different kinds of transfer periods. However, measurement results indicate throughput changes within bulk transfer periods, which motivate further work on the classification approach.

RQ 2.1 How to conduct active Internet measurements to survey TCP throughput?
This thesis proposed an approach to conducting active measurements on TCP throughput on the Internet covering identifying suitable measurement targets based on public web servers hosting suitable files, referred to as crawling, conducting HTTP(S) downloads of such files, and capturing corresponding traffic for further analysis.

Crawling 35 thousand domains taken from the top 50 thousand entries of a toplist [112] based on Google's CrUX [101] data set for files with a minimum size of 1 MB and a maximum depth of seven followed links results in 7700 (22 %) suitable measurement targets. We observe that crawling the index page and the subsequent two levels of followed links result in larger shares of detected files compared to other depth levels, while increased depth implies larger numbers of sent requests. This observation should be considered for further crawling in order to optimize the trade-off between effectiveness and intrusiveness.

Conducting downloads from different vantage points and target server selection has a significant impact on observed performance indicators of TCP connections, like throughput or latency. For instance, we find significant shares of connections with mean RTT smaller than 5 ms initiated from virtual vantage points hosted by DigitalOcean in Singapore and San Francisco. Such small RTTs are not observed from a vantage point hosted in a campus data center near Munich. This observation implies significantly decreased geographical distance between used virtual vantage points, e.g., due to the co-location of vantage points and CDN edge servers in the same physical data center or hosting in the same geographic area.

Further, we observe several measurement targets hosted by Cloudflare that do not result in successful downloads for the used virtual vantage points, while downloads from the campus vantage point succeeded during several measurement runs. This observation indicates that Cloudflare blocked some of our download attempts, e.g., through human verification for selected IP ranges [116]. Accordingly, measurements of TCP throughput on the Internet should consider different vantage points to detect vantage point-dependent characteristics.

Further, we observe significant impacts of edge caching of files by CDNs on throughput, i.e., increased throughput rates for downloads after an initial download run conducted with the same client configuration. Therefore, throughput and other KPIs measurements should consider a warm-up download to ensure the same caching conditions of target files.

RQ 2.2 How does TCP option usage impact connection performance?
To assess the impacts of TCP options on throughput, we conducted an active measurement study with the measurement approach introduced in the scope of RQ 2.1. Thereby, we consider over 2000 potential measurement targets and conduct downloads from three different vantage points.

We find that window scaling increases throughput for nearly 70 % of samples compared to the baseline configuration. Further, sufficiently large scaling factors, for our study 2^3, 2^7, and 2^{14}, result in at least doubled mean throughput for nearly 20 % of samples compared to the baseline. Enabling ECN and SACK improves mean throughput less significantly while we observe only small shares of connections suffering from retransmissions, indicating minor impacts by network congestion causing packet loss.

RQ 2.3 How accurate is the estimation of end-to-end path capacity in the Internet?

We conducted active measurements on hybrid Internet paths to assess the accuracy of packet pair dispersion-based capacity estimation in the Internet. Thereby, hybrid refers to a setup where the first, respectively last, hop on a path through the Internet is controlled, enabling the control of link capacity to provide ground truth data.

Our measurements show the general accuracy of PPD-based measurements on the Internet with absolute median relative errors smaller than 8 %, with improved results depending on the used protocol and algorithm adaptions. However, we also observe significant relative errors due to highly compressed inter-arrival times, e.g., due to interrupt coalescence, resulting in significantly biased estimates. This observation also applies to Internet measurements with uncontrolled targets, as discussed below.

RQ 2.4 What do active Internet measurements on end-to-end capacity reveal?

We conducted active Internet measurements with uncontrolled public web servers to survey capacity deployments and characteristics of active capacity measurements.

A measurement series targeting over 3500 public web servers shows large shares of median estimates approximating path capacities of at least 1 Gbit/s for ICMP- and TCP-based downloads corresponding to the downlink capacity of the used measurement host. This implies that capacities of at least 1 Gbit/s are commonly deployed on the Internet. Around 40 % of ICMP-based measurements show an underestimation of several hundred Mbit/s compared to TCP measurements targeting the same servers. This observation likely implies impacts by ICMP rate limiting on capacity estimation in the Internet. The same observation applies to the vast majority of measurements on routers on paths to targeted web servers. Accordingly, public routers are no suitable measurement target for PPD-based capacity measurements.

During the described Internet measurements, we observed significant impacts by interrupt coalescence and TCP receive offloading. Interrupt coalescence compresses inter-arrival times between packet pairs, resulting in highly overestimated capacity. Receive offloading, i.e., the NIC generating a larger TCP segment from several TCP packets processed by the same interrupt, mitigates such impacts but also results in an offset towards smaller capacities. As impacts by interrupt coalescence and receiver offloading are artifacts due to software packet timestamping by the operating systems, hardware timestamping, i.e., packet timestamps by the NIC, is expected to result in more reliable and accurate estimates.

Chapter 9

Future Work

This chapter points out follow-up research questions and objectives resulting from the findings described in this thesis.

This thesis extensively surveyed characteristics of TCP connections on the Internet based on large-scale passive data sets provided by CAIDA and MAWI. Further, this thesis examined TCP option usage and TCP throughput limitations of Internet connections captured by MAWI. As MAWI still publishes daily traffic captures, a continuous analysis based on the methodologies applied in this thesis promises further insights into the evolution of TCP connection characteristics on the Internet. For instance, existing tools could be used to implement a fully automated and public service. Further, our approach can be applied to further data sets considering different kinds of vantage points to compare traffic characteristics and throughput limitations of connections captured at various locations.

In this thesis, we used a heuristic-based approach to analyze TCP throughput limitations. To the best of our knowledge, the application of machine learning to the problem of TCP root cause analysis was not considered so far. Supervised learning requires a sufficient amount of labeled data for training, which is complex to generate synthetically considering the diversity of network characteristics, services, or endpoints, for instance, on the Internet. An approach to this problem is generating ground truth data based on network stack statistics collected on productive servers. Such data could be used to train root cause analysis based on captured network traffic without access to the sender's network stack.

This thesis introduced an active measurement approach to survey TCP performance on the Internet. While conducted crawling was sufficient to run measurements in the short term, the responsiveness of identified targets over time is of interest, for instance, to plan long-term active measurement series. Further, a continuous crawling service promises to reduce intrusiveness over time, e.g., by only crawling a target server when a considered file is not available anymore. Such a service could also provide measurement targets to other researchers.

The active measurements of TCP throughput on the Internet can be extended in various manners. First, measurements could be conducted long-term to gain insights into the evolution

of performance for individual targets over time. Further, measurements could be conducted without different option permutations to enable a finer resolution of samples per target, such as surveying impacts by daytime patterns or load balancing. Measurements conducted in this thesis relied on vantage points hosted in a campus data center, respectively, by a global cloud provider. Accordingly, measurements are conducted from vantage points with wired connections providing more bandwidth and access closer to the Internet core compared to common customer Internet access. This motivates the consideration of a broader variety of vantage points. Also, the active measurement approach could be extended for further protocols like QUIC and survey impacts by using different QUIC implementations on the client, respectively, on the server side.

This thesis presented measurement results on end-to-end capacities deployed on the Internet. The used measurement host was limited to a downlink capacity of 1 Gbit/s. Accordingly, extending downlink capacity enables the detection to differ end-to-end capacities of paths equal to or larger than 1 Gbit/s. Conducting such measurements based on hardware timestamping promises to avoid artifacts by interrupt coalescence and receiver offloading.

Bibliography

[1] *Transmission Control Protocol*, RFC 793, Sep. 1981. DOI: 10.17487/RFC0793. [Online]. Available: https://www.rfc-editor.org/info/rfc793.

[2] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the characteristics and origins of Internet flow rates", in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '02, Pittsburgh, Pennsylvania, USA: ACM, 2002, 309–322. DOI: 10.1145/633025.633055. [Online]. Available: https://doi.org/10.1145/633025.633055.

[3] B. Finley, E. Boz, K. Kilkki, J. Manner, A. Oulasvirta, and H. Hämmäinen, "Does network quality matter? A field study of mobile user satisfaction", *Pervasive and Mobile Computing*, vol. 39, pp. 80–99, Sep. 2016. DOI: 10.1016/j.pmcj.2016.08.014.

[4] M. Ghasemi, T. Benson, and J. Rexford, "Dapper: Data Plane Performance Diagnosis of TCP", in *Proceedings of the Symposium on SDN Research, SOSR 2017, Santa Clara, CA, USA, April 3-4, 2017*, ACM, 2017, pp. 61–74. DOI: 10.1145/3050220.3050228. [Online]. Available: https://doi.org/10.1145/3050220.3050228.

[5] K.-c. Lan and J. Heidemann, "A measurement study of correlations of Internet flow characteristics", *Computer Networks*, vol. 50, no. 1, 46–62, 2006.

[6] A. Medina, M. Allman, and S. Floyd, "Measuring interactions between transport protocols and middleboxes", in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04, Taormina, Sicily, Italy: ACM, 2004, 336–341. DOI: 10.1145/1028788.1028835. [Online]. Available: https://doi.org/10.1145/1028788.1028835.

[7] M. Kühlewind, S. Neuner, and B. Trammell, "On the State of ECN and TCP Options on the Internet", in *Passive and Active Measurement: PAM 2013, Hong Kong, China, March 18-19*, Springer Berlin Heidelberg, 2013, pp. 135–144.

[8] M. Allman, "A web server's view of the transport layer", *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 5, pp. 10–20, 2000. DOI: 10.1145/505672.505674.

[9] K. Pentikousis and H. Badr, "Quantifying the deployment of TCP options - a comparative study", *IEEE Communications Letters*, vol. 8, no. 10, pp. 647–649, 2004. DOI: 10.1109/LCOMM.2004.835308.

[10] M. Timmer, P.-T. de Boer, and A. Pras, "How to Identify the Speed Limiting Factor of a TCP Flow", in *4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, 2006, pp. 17–24. DOI: 10.1109/E2EMON.2006.1651275.

[11] M. Siekkinen, G. Urvoy-Keller, E. Biersack, and D. Collange, "A root cause analysis toolkit for TCP", *Computer Networks*, vol. 52, no. 9, pp. 1846–1858, Jun. 2008. DOI: 10.1016/j.comnet.2008.03.005.

[12] J. T. Araújo, R. Landa, R. G. Clegg, G. Pavlou, and K. Fukuda, "A longitudinal analysis of Internet rate limitations", in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, IEEE, Apr. 2014. DOI: 10.1109/INFOCOM.2014.6848078.

[13] A. Bąk, P. Gajowniczek, and M. Zagożdżon, "Measurement methodology of TCP performance bottlenecks", in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, 2015, pp. 1149–1156. DOI: 10.15439/2015F284.

[14] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?", in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, IEEE, vol. 2, 2001, pp. 905–914. DOI: 10.1109/INFCOM.2001.916282.

[15] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Sprobe: A fast technique for measuring bottleneck bandwidth in uncooperative environments", in *IEEE INFOCOM*, 2002, p. 1.

[16] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi, "CapProbe: A Simple and Accurate Capacity Estimation Technique", in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '04, Portland, Oregon, USA: ACM, 2004, pp. 67–78. DOI: 10.1145/1015467.1015476.

[17] L.-J. Chen, T. Sun, G. Yang, M. Y. Sanadidi, and M. Gerla, "End-to-End Asymmetric Link Capacity Estimation", in *NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, Springer Berlin Heidelberg, 2005, pp. 780–791.

[18] L.-J. Chen, T. Sun, B.-C. Wang, M. Sanadidi, and M. Gerla, "PBProbe: A capacity estimation tool for high speed networks", *Computer Communications*, vol. 31, no. 17, pp. 3883–3893, 2008. DOI: 10.1016/j.comcom.2008.05.047.

[19] K. Lai and M. Baker, "Nettimer: a tool for measuring bottleneck link, bandwidth", in *Proceedings of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 3*, ser. USITS'01, San Francisco, California: USENIX Association, 2001, p. 11.

[20] T. En-Najjary and G. Urvoy-Keller, "PPrate: A Passive Capacity Estimation Tool", in *2006 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, Vancouver, BC, Canada, 2006, pp. 82–89. DOI: 10.1109/E2EMON.2006.1651283.

[21] S. Katti, D. Katabi, C. Blake, E. Kohler, and J. Strauss, "MultiQ: automated detection of multiple bottleneck capacities along a path", in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04, Taormina, Sicily, Italy: ACM, 2004, 245–250. DOI: 10.1145/1028788.1028820. [Online]. Available: https://doi.org/10.1145/1028788.1028820.

[22] Y. Zhang, Å. Arvidsson, M. Siekkinen, and G. Urvoy-Keller, "Understanding HTTP flow rates in cellular networks", in *2014 IFIP Networking Conference*, IEEE, Trondheim, Norway, 2014, pp. 1–8.

[23] P. Sun, M. Yu, M. J. Freedman, and J. Rexford, "Identifying performance bottlenecks in CDNs through TCP-level monitoring", in *Proceedings of the First ACM SIGCOMM Workshop on Measurements up the Stack*, ser. W-MUST '11, Toronto, Ontario, Canada: ACM, 2011, 49–54. DOI: 10.1145/2018602.2018615. [Online]. Available: https://doi.org/10.1145/2018602.2018615.

[24] M. Siekkinen, D. Collange, G. Urvoy-Keller, and E. W. Biersack, "Performance limitations of ADSL users: a case study", in *Proceedings of the 8th International Conference on Passive and Active Network Measurement*, ser. PAM'07, Louvain-la-Neuve, Belgium: Springer, 2007, 145–154.

[25] T. En-Najjary and G. Urvoy-Keller, "Passive capacity estimation: comparison of existing tools", in *SPECTS 2008, International Symposium on Performance Evaluation of Computer and Telecommunication Systems, June 16-18*, IEEE, Ed., Edinburgh, UK, 2008, pp. 346–352.

[26] Z. Zhang, B. Wang, and J. Lan, "Identifying elephant flows in internet backbone traffic with bloom filters and LRU", *Computer Communications*, vol. 61, pp. 70–78, Dec. 2014. DOI: 10.1016/j.comcom.2014.12.003.

[27] R. Harrison, S. L. Feibish, A. Gupta, R. Teixeira, S Muthukrishnan, and J. Rexford, "Carpe elephants: Seize the global heavy hitters", in *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, ser. SPIN '20, Virtual Event, USA: ACM, 2020, pp. 15–21. DOI: 10.1145/3405669.3405820.

[28] A. Pekar, A. Duque-Torres, W. K. Seah, and O. Caicedo, "Knowledge Discovery: Can It Shed New Light on Threshold Definition for Heavy-Hitter Detection?", *Journal of Network and Systems Management*, vol. 29, no. 3, pp. 1–30, 2021. DOI: 10.1007/s10922-021-09593-w. [Online]. Available: https://doi.org/10.1007/s10922-021-09593-w.

[29] G. Y. Lazarou, M. S. Alam, and J. Picone, "Measuring the variability of CAIDA internet traffic traces", in *2016 19th International Conference on Computer and Information Technology (ICCIT)*, IEEE, 2016, pp. 1–6. DOI: 10.1109/ICCITECHN.2016.7860158.

[30] P. Velan, J. Medková, T. Jirsík, and P. Čeleda, "Network traffic characterisation using flow-based statistics", in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, IEEE, Istanbul, Turkey: IEEE, 2016, pp. 907–912. DOI: 10.1109/NOMS.2016.7502924. [Online]. Available: https://doi.org/10.1109/NOMS.2016.7502924.

[31] K. Thompson, G. Miller, and R. Wilder, "Wide-area Internet traffic patterns and characteristics", *IEEE Network*, vol. 11, no. 6, pp. 10–23, 1997. DOI: 10.1109/65.642356.

[32] A. Salihu, M. Shefkiu, and A. Maraj, "Characteristics and Temporal Behavior of Internet Backbone Traffic", *International Journal of Business & Technology*, vol. 6, pp. 1–8, May 2018. DOI: 10.33107/ijbte.2018.6.3.03.

[33] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic", in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '09, Chicago, Illinois, USA, 2009, pp. 90–102. DOI: 10.1145/1644893.1644904. [Online]. Available: https://doi.org/10.1145/1644893.1644904.

[34] M. Siekkinen, G. Urvoy-Keller, E. W. Biersack, and T. En-Najjary, "Root Cause Analysis for Long-lived TCP Connections", in *Proceedings of the 2005 ACM Conference on*

Emerging Network Experiment and Technology, 1st ACM/e-NEXT International Conference on Future Networking Technologies, ser. CoNEXT '05, Toulouse, France: ACM, 2005, pp. 200–210.

[35] M. Siekkinen, G. Urvoy-Keller, and E. W. Biersack, "On the Interaction Between Internet Applications and TCP", in *Proceedings of the 20th International Teletraffic Conference on Managing Traffic Performance in Converged Networks*, ser. ITC20'07, Ottawa, Canada: Springer-Verlag, Jan. 2007, pp. 962–973. DOI: 10.1007/978-3-540-72990-7_83.

[36] S. Shakkottai, R Srikant, N. Brownlee, A. Broido, and C. Kc, "The RTT distribution of TCP flows in the Internet and its impact on TCP-based flow control", *IEEE Transactions on Reliability - TR*, Jan. 2004.

[37] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet-dispersion techniques and a capacity-estimation methodology", *IEEE/ACM Transactions On Networking*, vol. 12, no. 6, pp. 963–977, 2004. DOI: 10.1109/TNET.2004.838606.

[38] C. Dovrolis and R. Prasad. "README for pathrate-2.4.1". Last accessed on 2024/02/28. (2017), [Online]. Available: https://github.com/brucespang/pathrate.

[39] A. B. Downey. "Clink: a tool for estimating internet link characteristics". Last accessed on 2024/02/28. (1999), [Online]. Available: https://allendowney.com/research/clink/.

[40] C. Marcondes, C. Palazzi, M. Y. Sanadidi, M. Gerla, M. Martinello, and M. T. Torres, "Regenerating TCP Dynamics From Traces Path Characteristics", in *2007 3rd International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities*, 2007.

[41] V. Jacobson, *Pathchar: A tool to infer characteristics of Internet paths*, 1997. [Online]. Available: https://www.ittc.ku.edu/Projects/DREN/software/bandwidth/pathchar/msri-talk.pdf.

[42] A. Jindal, K. Psounis, and M. Liu, "Capest: A measurement-based approach to estimating link capacity in wireless networks", *IEEE Transactions on Mobile Computing*, vol. 11, no. 12, pp. 2098–2108, Jul. 2010. DOI: 10.1109/TMC.2011.245.

[43] N. S. Kagami, R. I. T. da Costa Filho, and L. P. Gaspary, "Capest: Offloading network capacity and available bandwidth estimation to programmable data planes", *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 175–189, Mar. 2020. DOI: 10.1109/TNSM.2019.2934316. [Online]. Available: https://doi.org/10.1109/TNSM.2019.2934316.

[44] M. Jain and C. Dovrolis, "Pathload: A Measurement Tool for End-to-End Available Bandwidth", Fort Collins, Colorado, USA, Mar. 2002.

[45] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput", *SIGCOMM Computer Communication Review*, vol. 32, no. 4, 295–308, 2002. DOI: 10.1145/964725.633054. [Online]. Available: https://doi.org/10.1145/964725.633054.

[46] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools", *IEEE Network*, vol. 17, no. 6, pp. 27–35, 2003. DOI: 10.1109/MNET.2003.1248658.

[47] J. Sommers, P. Barford, and W. Willinger, "A proposed framework for calibration of available bandwidth estimation tools", in *11th IEEE Symposium on Computers and Communications (ISCC'06)*, IEEE, Cagliari, Sardinia, Italy, 2006, pp. 709–718.

[48] E. Goldoni and M. Schivi, "End-to-end available bandwidth estimation tools, an experimental comparison", in *International Workshop on Traffic Monitoring and Analysis*, Springer, Zurich, Switzerland, Apr. 2010, pp. 171–182.

[49] A. K. Paul, A. Tachibana, and T. Hasegawa, "NEXT: New enhanced available bandwidth measurement technique, algorithm and evaluation", in *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, IEEE, Washington, DC, USA, 2014, pp. 443–447.

[50] A. K. Paul, A. Tachibana, and T. Hasegawa, "NEXT-FIT: Available Bandwidth Measurement over 4G/LTE Networks – A Curve-Fitting Approach", in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Crans-Montana, Switzerland, 2016, pp. 25–32. DOI: 10.1109/AINA.2016.24.

[51] D. Murray, T. Koziniec, S. Zander, M. Dixon, and P. Koutsakis, "An analysis of changing enterprise network traffic characteristics", in *2017 23rd Asia-Pacific Conference on Communications (APCC)*, IEEE, Piscataway, New Jersey, 2017, pp. 1–6. DOI: 10.23919/APCC.2017.8303960.

[52] C.-X. Chen and K. Nagaoka, "Analysis of the State of ECN on the Internet", *IEICE TRANSACTIONS on Information and Systems*, vol. 102, no. 5, pp. 910–919, 2019. DOI: 10.1587/transinf.2018NTP0006.

[53] H. Lim, S. Kim, J. Sippe, *et al.*, "A Fresh Look at ECN Traversal in the Wild", *arXiv preprint arXiv:2208.14523*, 2022.

[54] K. Edeline and B. Donnet, "Evaluating the Impact of Path Brokenness on TCP Options", in *Proceedings of the Applied Networking Research Workshop*, Virtual Event, Jul. 2020, pp. 38–44. DOI: 10.1145/3404868.3406662.

[55] CAIDA, *The CAIDA UCSD Anonymized Internet Traces - 2008-2016*, Website, Last accessed on 2024/02/26, 2019. [Online]. Available: http://www.caida.org/data/passive/passive_dataset.xml.

[56] K. Cho, K. Mitsuya, and A. Kato, "Traffic Data Repository at the WIDE Project", ser. USENIX 2000 FREENIX Track, San Diego, CA, USA: USENIX, 2000.

[57] F. Helfert, "Load Generation for L4 Traffic Analysis", *Master's Thesis in Informatics, Technische Universität München*, Aug. 2019.

[58] P. Barias, "Long Term Analysis of TCP Internet Traffic Characteristics", *Master's Thesis in Informatics, Technische Universität München*, Sep. 2020.

[59] S. Bauer, B. Jaeger, F. Helfert, P. Barias, and G. Carle, "On the Evolution of Internet Flow Characteristics", in *Proceedings of the Applied Networking Research Workshop*, Virtual Event, USA: ACM, Jul. 2021.

[60] S. Hülkenberg, "Large Scale TCP Root Cause Analysis in Go", *Master's Thesis in Informatics, Technische Universität München*, Aug. 2023.

[61] B. Veal, K. Li, and D. Lowenthal, "New methods for passive estimation of TCP round-trip times", in *International workshop on passive and active network measurement (PAM*

2005), Springer, Boston, MA, USA, Mar. 2005, pp. 121–134. DOI: 10.1007/978-3-540-31966-5_10.

[62] *Wireshark TCP Analysis*, Website, Last accessed on 2024/02/28. [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/ChAdvTCPAnalysis.html.

[63] D. A. Borman, R. T. Braden, and V. Jacobson, *TCP Extensions for High Performance*, RFC 1323, May 1992. DOI: 10.17487/RFC1323. [Online]. Available: https://www.rfc-editor.org/info/rfc1323.

[64] S. Floyd, J. Mahdavi, M. Mathis, and D. A. Romanow, *TCP Selective Acknowledgment Options*, RFC 2018, Oct. 1996. DOI: 10.17487/RFC2018. [Online]. Available: https://www.rfc-editor.org/info/rfc2018.

[65] S. Floyd, D. K. K. Ramakrishnan, and D. L. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*, RFC 3168, Sep. 2001. DOI: 10.17487/RFC3168. [Online]. Available: https://www.rfc-editor.org/info/rfc3168.

[66] *gopacket*, Website, Last accessed on 2024/02/28, 2020. [Online]. Available: https://github.com/google/gopacket.

[67] CAIDA. "Passive Monitor: equinix-chicago". Last accessed on 2024/02/26. (2019), [Online]. Available: http://www.caida.org/data/monitors/passive-equinix-chicago.xml.

[68] CAIDA. "Passive Monitor: equinix-nyc". Last accessed on 2024/02/26. (2019), [Online]. Available: http://www.caida.org/data/monitors/passive-equinix-nyc.xml.

[69] I. Tsareva, T. V. Doan, and V. Bajpai, "A Decade Long View of Internet Traffic Composition in Japan", in *2023 IFIP Networking Conference (IFIP Networking)*, 2023, pp. 1–9. DOI: 10.23919/IFIPNetworking57963.2023.10186393.

[70] *DPDK*, Website, Last accessed on 2024/02/28. [Online]. Available: https://www.dpdk.org.

[71] S. Bauer, K. Holzinger, B. Jaeger, P. Emmerich, and G. Carle, "Online Monitoring of TCP Throughput Limitations", in *2020 IEEE/IFIP Network Operations and Management Symposium (NOMS 2020)*, Budapest, Hungary (Virtual Event), 2020.

[72] S. Bauer, F. Wiedner, B. Jaeger, P. Emmerich, and G. Carle, "Scalable TCP Throughput Limitation Monitoring", in *2021 IEEE/IFIP Symposium on Integrated Network and Service Management (IM 2021)*, Bordeaux, France (Virtual Event), May 2021.

[73] P. Emmerich, M. Pudelko, S. Gallenmüller, and G. Carle, "FlowScope: Efficient packet capture and storage in 100 Gbit/s networks", in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, Rome, Italy, 2017, pp. 1–9. DOI: 10.23919/IFIPNetworking.2017.8264852.

[74] P. Emmerich, M. Pudelko, Q. Scheitle, and G. Carle, "Efficient Dynamic Flow Tracking for Packet Analyzers", in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, Tokyo, Japan, Oct. 2018. DOI: 10.1109/CloudNet.2018.8549214.

[75] K. Holzinger, "High Performance Online TCP Root Cause Analysis", *Master's Thesis in Informatics, Technische Universität München*, Aug. 2019.

[76] S. Woo and K. Park, "Scalable TCP session monitoring with symmetric receive-side scaling", KAIST, Daejeon, Korea, Tech. Rep., 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:14061661.

[77] F. Wiedner, "Large-scale TCP Throughput Limitation Analysis", *Master's Thesis in Informatics, Technische Universität München*, May 2020.

[78] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator", in *Proceedings of the 2015 Internet Measurement Conference*, ser. IMC '15, Tokyo, Japan: ACM, 2015. DOI: 10.1145/2815675.2815692. [Online]. Available: https://doi.org/10.1145/2815675.2815692.

[79] S. Bauer, B. Jaeger, M. Reimann, J. Fromm, and G. Carle, "Towards the Classification of TCP Throughput Changes", in *2022 IEEE/IFIP Network Operations and Management Symposium (NOMS 2022)*, Budapest, Hungary, Apr. 2022.

[80] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods", *Signal Processing*, vol. 167, 2020. DOI: 10.1016/j.sigpro.2019.107299.

[81] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blazek, and H. Kim, "A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods", *IEEE transactions on signal processing*, vol. 54, no. 9, pp. 3372–3382, 2006. DOI: 10.1109/TSP.2006.879308.

[82] A. G. Tartakovsky, A. S. Polunchenko, and G. Sokolov, "Efficient computer network anomaly detection by changepoint detection methods", *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 4–11, 2012.

[83] M. Alkasassbeh, "A novel hybrid method for network anomaly detection based on traffic prediction and change point detection", *arXiv preprint arXiv:1801.05309*, 2018.

[84] W. Li, S. Gao, X. Li, Y. Xu, and S. Lu, "TCP-NeuRoc: Neural Adaptive TCP Congestion Control with Online Changepoint Detection", *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2461–2475, 2021.

[85] C. Truong, L. Oudre, and N. Vayatis, "ruptures: change point detection in Python", *arXiv preprint arXiv:1801.00826*, 2018.

[86] J. Bai, "Least absolute deviation estimation of a shift", *Econometric Theory*, vol. 11, no. 3, pp. 403–436, 1995.

[87] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost", *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[88] P. Fryzlewicz, "Wild binary segmentation for multiple change-point detection", *The Annals of Statistics*, vol. 42, no. 6, pp. 2243–2281, 2014.

[89] P. Fryzlewicz, "Unbalanced Haar technique for nonparametric function estimation", *Journal of the American Statistical Association*, vol. 102, no. 480, pp. 1318–1327, 2007.

[90] B. Jaeger, D. Scholz, D. Raumer, F. Geyer, and G. Carle, "Reproducible Measurements of TCP BBR Congestion Control", *Computer Communications*, vol. 144, pp. 31–43, May 2019.

[91] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-defined Networks", in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Monterey, California: ACM, 2010. DOI: 10.1145/1868447.1868466. [Online]. Available: https://doi.org/10.1145/1868447.1868466.

[92] J. Fromm, "Classifying TCP Throughput Changepoints with TCP Transfer Periods", *Bachelor's Thesis in Informatics, Technische Universität München*, Jul. 2020.

[93] M. Reimann, "Validating CPD for TCP Transfer Period Classification ", *Bachelor's Thesis in Informatics, Technische Universität München*, Oct. 2021.

[94] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "emimic: Estimating HTTP-based video QoE metrics from encrypted network traffic", in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, IEEE, Vienna, Austria, 2018, pp. 1–8. DOI: 10.23919/TMA.2018.8506519.

[95] S. C. Madanapalli, H. H. Gharakhieli, and V. Sivaraman, "Inferring netflix user experience from broadband network measurement", in *2019 Network Traffic Measurement and Analysis Conference (TMA)*, IEEE, Paris, France, 2019, pp. 41–48. DOI: 10.23919/TMA.2019.8784609.

[96] C. López, D. Morato, E. Magana, and M. Izal, "Effective analysis of secure web response time", in *2019 Network Traffic Measurement and Analysis Conference (TMA)*, IEEE, Paris, France, 2019, pp. 145–152. DOI: 10.23919/TMA.2019.8784652.

[97] B. Jun, M. Varvello, Y. Zaki, and F. E. Bustamante, "WebTune: A Distributed Platform for Web Performance Measurements", in *2021 Network Traffic Measurement and Analysis Conference (TMA)*, Virtual Event: IEEE, 2021.

[98] N. Zilberman, A. W. Moore, B. Cooper, *et al.*, "NRG: A Network Perspective on Applications' Performance", in *5th Network Traffic Measurement and Analysis Conference, TMA 2021*, Virtual Event: IFIP, 2021. [Online]. Available: http://dl.ifip.org/db/conf/tma/tma2021/tma2021-paper13.pdf.

[99] S. Bauer, P. Sattler, J. Zirngibl, C. Schwarzenberg, and G. Carle, "Evaluating the Benefits: Quantifying the Effects of TCP Options, QUIC, and CDNs on Throughput", in *Proceedings of the Applied Networking Research Workshop*, San Francisco, CA, USA: ACM, Jul. 2023.

[100] *Alexa. Top 1M sites.* Last accessed on 15 Sep 2022. [Online]. Available: https://www.alexa.com/topsites.

[101] "About CrUX". Last accessed on 2024/02/28. (2024), [Online]. Available: https://developer.chrome.com/docs/crux/about/.

[102] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain, *TCP Fast Open*, RFC 7413, Dec. 2014. DOI: 10.17487/RFC7413. [Online]. Available: https://www.rfc-editor.org/info/rfc7413.

[103] D. Dittrich, E. Kenneally, *et al.*, "The Menlo Report: Ethical principles guiding information and communication technology research", *US Department of Homeland Security*, 2012.

[104] C. Partridge and M. Allman, "Addressing Ethical Considerations in Network Measurement Papers", *Communications of the ACM*, vol. 59, no. 10, Oct. 2016. DOI: 10.1145/2793013.2793014. [Online]. Available: https://doi.org/10.1145/2793013.2793014.

[105] M. Koster, G. Illyes, H. Zeller, and L. Sassman, "Robots Exclusion Protocol", Tech. Rep. 9309, Sep. 2022, 12 pp. DOI: 10.17487/RFC9309. [Online]. Available: https://www.rfc-editor.org/info/rfc9309.

[106] R. T. Fielding and J. Reschke, *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*, RFC 7230, Jun. 2014. DOI: 10.17487/RFC7230. [Online]. Available: https://www.rfc-editor.org/info/rfc7230.

[107] *Scrapy*, Website, Last accessed on 2024/02/22. [Online]. Available: https://scrapy.org.

[108] *Wget2*, Website, Last accessed on 2024/02/22. [Online]. Available: https://gitlab.com/gnuwget/wget2.

[109] D. Manrique, *ForcedIPHTTPSAdapter*, Website, Last accessed on 2022/06/20. [Online]. Available: https://github.com/Roadmaster/forcediphttpsadapter.

[110] *tcpdump and libpcap*, Website, Last accessed on 2024/02/22. [Online]. Available: https://www.tcpdump.org.

[111] *D.2. tshark: Terminal-based Wireshark*, Website, Last accessed on 2024/02/22. [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/AppToolstshark.html.

[112] Z. Durumeric and D. Adrian, *Cached Chrome Top Million Websites*, Website, Last accessed on 2024/02/28. [Online]. Available: https://github.com/zakird/crux-top-lists.

[113] *tcptraceroute(1) - Linux man page*, Website, Last accessed: 1 August 2024. [Online]. Available: https://linux.die.net/man/1/tcptraceroute.

[114] "University of Oregon Route Views Project". (), [Online]. Available: https://www.routeviews.org/routeviews/.

[115] A. Arturi, E. Carisimo, and F. E. Bustamante, "As2org+: Enriching AS-to-Organization Mappings With PeeringDB", 2023, 400–428, ISBN: 978-3-031-28485-4. DOI: 10.1007/978-3-031-28486-1_17.

[116] M. Prince, *Introducing: I'm under attack mode*, Website, Last accessed on 2024/02/28. [Online]. Available: https://blog.cloudflare.com/introducing-im-under-attack-mode.

[117] C. Schwarzenberg, "Measuring Performance and Path Characteristics of Webservers", *Master's Thesis in Informatics, Technische Universität München*, Jul. 2020.

[118] M. S. Shaukat, "Measuring the Impact of Transport Layer Protocols and Their Configuration on the Performance of Connections", *Master's Thesis in Informatics, Technische Universität München*, Apr. 2023.

[119] S. Bauer, J. Janelidze, B. Jaeger, P. Sattler, P. Brzoza, and G. Carle, "On the Accuracy of Active Capacity Estimation in the Internet", in *2023 IEEE/IFIP Network Operations and Management Symposium (NOMS 2023)*, Miami, USA, May 2023.

[120] F. Abut, "Through the Diversity of Bandwidth-Related Metrics, Estimation Techniques and Tools: An Overview", *International Journal of Computer Network and Information Security*, vol. 10, pp. 1–16, Aug. 2018. DOI: 10.5815/ijcnis.2018.08.01.

[121] A. B. Downey, "Using pathchar to estimate Internet link characteristics", *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 4, pp. 241–250, 1999.

[122] R. S. Prasad, C. Dovrolis, and B. A. Mah, "The effect of layer-2 store-and-forward devices on per-hop capacity estimation", in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, IEEE, vol. 3, 2003, pp. 2090–2100.

[123] R. L. S. De Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using Mininet for Emulation and Prototyping Software-Defined Networks", in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, IEEE, 2014, pp. 1–6.

[124] *iPerf - The ultimate speed test tool for TCP, UDP and SCTP*, Website, Last accessed: 27 February 2024. [Online]. Available: `https://iperf.fr`.

[125] *gRPC*, Website, Last accessed: 27 February 2024. [Online]. Available: `https://grpc.io`.

[126] A. Barbeiro, "Visualizing Multipath Networks with Dublin Traceroute", *Presentation at the MOCA Italian hacker camp*, 2016.

[127] *webfsd - A simple HTTP server for mostly static content written in C*, Website, Last accessed: 27 February 2024. [Online]. Available: `https://github.com/ourway/webfsd`.

[128] K. M. Salehin, V. Sahasrabudhe, and R. Rojas-Cessa, "Remote measurement of interrupt-coalescence latency of internet hosts", in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, IEEE, 2017, pp. 1–9.

[129] R. Ravaioli, G. Urvoy-Keller, and C. Barakat, "Characterizing ICMP rate limitation on routers", in *2015 IEEE International Conference on Communications (ICC)*, IEEE, 2015, pp. 6043–6049.

[130] H. Guo and J. Heidemann, "Detecting ICMP rate limiting in the Internet", in *International Conference on Passive and Active Network Measurement*, Springer, 2018, pp. 3–17.

[131] N. Beck, "Root Cause Analysis for Throughput Limitations of QUIC Connections", *Master's Thesis in Informatics, Technische Universität München*, Oct. 2023.

[132] P. Brzoza, "KPI Analysis of Webserver Traffic through Active Measurements", *Master's Thesis in Informatics, Technische Universität München*, Nov. 2021.

[133] J. Janelidze, "Analyzing Robustness and Intrusiveness of Capacity Estimation Tools in the Internet", *Bachelor's Thesis in Informatics, Technische Universität München*, Oct. 2022.

[134] P. Brzoza, "Implementation of a Network Bandwidth Estimation Tool", *Bachelor's Thesis in Informatics, Technische Universität München*, Mar. 2019.

[135] F. List, "Estimating the Narrow Link Position of Network Paths", *Bachelor's Thesis in Informatics, Technische Universität München*, Jun. 2020.

[136] E. Rosche, "Automated Testing of Capacity Estimation", *Bachelor's Thesis in Informatics, Technische Universität München*, Nov. 2021.