



# What's the Delay? Understanding Latency Across the Network

Simon Sundberg

Faculty of Health, Science and Technology

---

Computer Science

---

DOCTORAL THESIS | Karlstad University Studies | 2026:30

---

# What's the Delay? Understanding Latency Across the Network

Simon Sundberg

What's the Delay? Understanding Latency Across the Network

---

Simon Sundberg

---

DOCTORAL THESIS

---

Karlstad University Studies | 2026:30

---

urn:nbn:se:kau:diva-108434

---

ISSN 1403-8099

---

ISBN 978-91-7867-709-2 (print)

---

ISBN 978-91-7867-710-8 (pdf)

---

<https://doi.org/10.59217/qklv6836>

---

© The author

---

Distribution:

Karlstad University

Faculty of Health, Science and Technology

Department of Mathematics and Computer Science

SE-651 88 Karlstad, Sweden

+46 54 700 10 00

---

Print: Universitetstryckeriet, Karlstad 2026

---

**WWW.KAU.SE**

# What's the Delay? Understanding Latency Across the Network

SIMON SUNDBERG

*Department of Mathematics and Computer Science*

## Abstract

Network latency directly affects the performance of many applications that run over the Internet. While significant effort is spent on reducing network latency, the fundamental capability to observe latency remains limited in many network environments. Network operators who rely on active measurements do not get a complete view of the latency encountered by ordinary application traffic, while the coarse-grained measurements frequently used by network researchers fail to capture the rapid latency fluctuations in emerging wireless networks.

In this thesis, we explore the potential of technologies like eBPF and hardware timestamps to address the limited observability of latency in modern networks. To this end, we use eBPF to design two lightweight in-kernel passive monitoring solutions that network operators can use to monitor the end-to-end network latency and track the latency within the local end host network stack. Through deployments in an Internet service provider network and across servers in a global content distribution network, we demonstrate the feasibility of continuously monitoring the latency in production. Our analysis of the monitoring data highlights that last-mile access remains a significant source of latency in end-users' Internet connections, and shows how large latency spikes can occur already in the early parts of web servers' packet processing pipeline. Additionally, we develop methodologies that combine high-frequency active measurements with accurate hardware timestamps to gain a more detailed understanding of the latency characteristics in wireless networks. Using our high-fidelity measurements to study Starlink, we reveal how its link-layer scheduling and apparent use of front-drop queueing impact latency. By studying latency in various network environments, we thus advance our understanding of network latency, and by designing new tools to measure latency, we provide a foundation for future research on network latency.

**Keywords:** network latency, host latency, passive monitoring, active measurements, Linux networking, eBPF, hardware timestamps, Starlink



# Vad är det som dröjer? Att förstå latens i datornätverk

SIMON SUNDBERG

*Institutionen för matematik och datavetenskap*

## Sammanfattning

Nätverksfördröjningen påverkar prestandan för många av de applikationer som körs över internet. Trots att betydande insatser görs för att minska nätverkslatensen är möjligheten att observera latensen fortfarande begränsad i många nätverksmiljöer. Nätverksoperatörer som förlitar sig på aktiva nätverksmätningar får inte en fullständig bild av latensen för vanlig applikationstrafik, medan de lågfrekventa mätningar som ofta används av nätverksforskare inte lyckas fånga de snabba latensfluktuationerna i trådlösa nätverk.

I denna avhandling undersöker vi potentialen hos tekniker som eBPF och hårdvarutidsstämplar för att förbättra möjligheterna att observera latens i moderna nätverk. Vi använder eBPF för att designa två effektiva passiva övervakningslösningar som körs i operativsystemets kärna, vilka låter nätoperatörer övervaka latensen både över hela nätverket och inom den lokala nätverksstacken. Genom att installera dessa verktyg hos en internetleverantör och på servrar i ett globalt distributionsnätverk visar vi att det är möjligt att kontinuerligt övervaka latensen i produktionsmiljöer. Vår analys av mätdata visar att accessnätverk fortfarande orsakar en stor del av latensen i slutanvändarnas internetanslutningar, samt att stora fördröjningar kan uppstå redan tidigt i webservrars mjukvarustack för bearbetning av nätverkspaket. Vidare utvecklar vi metoder som kombinerar högfrekventa aktiva mätningar med exakta hårdvarutidsstämplar för att få en mer detaljerad förståelse av latensen i trådlösa nätverk. Genom att använda våra högupplösta mätningar för att studera Starlink avslöjar vi hur dess resursschemaläggning i länklagret och köhantering påverkar latensen. Genom att studera latensen i olika nätverksmiljöer fördjupar vi därmed vår förståelse av nätverkslatens, och genom att designa nya verktyg för att mäta latens lägger vi grunden för framtida forskning om nätverkslatens.

**Nyckelord:** nätverkslatens, ändsystemlatens, passiv övervakning, aktiva mätningar, Linuxs nätverksstack, eBPF, hårdvarutidsstämplar, Starlink



## Acknowledgements

In your hands, you hold the culmination of five years of my work. However, the work in this thesis is not mine alone. Numerous people have supported this work in various ways and allowed me to undergo this journey towards becoming a researcher. First, I would like to thank the Knowledge Foundation and Red Hat Research for funding my research. I also want to thank the AIDA (A Holistic AI-Driven Networking and Processing Framework for Industrial IoT) project partners Ericsson, Tietoevry, and, in particular, Red Hat, for helping distribute my research and providing valuable feedback. Additionally, I wish to thank the entire team at LibreQoS for adopting some of my work. Seeing how my research has played a small role in your effort to improve the Internet has been incredibly encouraging.

Furthermore, I wish to thank Toke Høiland-Jørgensen at Red Hat and Jesper Dangaard Brouer at Cloudflare for all the guidance and technical expertise they have provided. You have been my unofficial supervisors throughout my PhD studies. Thank you, Toke, for all the pedagogical feedback you have provided for my manuscripts and code. You make a great teacher. Thank you, Jesper, for all the ideas you have contributed, for our sessions where we have tracked down technical issues, and for your effort in deploying our work at Cloudflare. I hope I get to work with you again in the future. I also wish to thank Robert Chacón at LibreQoS and JackRabbit Wireless for enthusiastically deploying our experimental monitoring solution in his network and sharing the measurement data. I wish all Internet service providers were as passionate about their customers' experience as you.

Huge thanks to all my colleagues here at Computer Science at Karlstad University for your support. I hope that this is just the start of our journey together. I especially wish to thank my main supervisor, Anna Brunstrom, for her guidance, wisdom, and faith in me. Thank you, Anna, for always showing interest in my work, for assisting me with everything from course selection to handling issues with conference submissions, and for taking the time to help me even as the clock ticks past any scheduled meeting time. I also want to thank my co-supervisor, Simone Ferlin-Reiter, for all of her support. Thank you, Simone, for all the feedback and insights you have provided, for celebrating every success, and for your encouraging words when there were setbacks. Thank you, Johan Garcia, for setting me on this path to becoming a researcher in the first place. I have greatly enjoyed working together with you on all of our “not obviously impossible” projects. Thank you, Andreas Kassler and Tobias Pulls, for reviewing the introductory summary in this thesis (and my prior licentiate thesis). The parts of your detailed feedback I have been able to address certainly improved the quality of this work.

Additionally, I want to thank my friends and family. Without your support and encouragement, I would never have gotten to where I am today. Thanks, Mom and Dad, for raising me, loving me, and always having my back. I will always be your son, and I could not wish for better parents. Thank you, Frida, my beloved sister, for all the positive energy you bring. We are each other's

opposites in many ways, yet I could not be more proud of you. Never stop being you. To my old friend, Mattias Larsson, thank you for listening to my rants about academic publishing processes and other topics I am sure you find equally interesting. To all the members of my Christian cell group, thank you for all your prayers and companionship. You are my safe space in a chaotic world.

Last, but certainly not least, thank you, God, my Lord and Savior. You give me hope for the future and the strength to keep going in times of hardship. I am not your most loyal servant, but I pray that I shall never stray too far from your path. For thine is the kingdom, the power, and the glory, for ever.

Karlstad, April 21, 2026

Simon Sundberg

# Contents

List of Appended Papers	xv
<b>INTRODUCTORY SUMMARY</b>	<b>1</b>
1 Introduction	3
2 Background and Related Work	5
2.1 Network Latency . . . . .	5
2.1.1 One-Way Delays and Round Trip Times . . . . .	6
2.1.2 Sources of Latency . . . . .	6
2.2 Measuring Latency . . . . .	8
2.2.1 Active Latency Measurements . . . . .	9
2.2.2 Passive Latency Measurements . . . . .	10
2.2.3 Hybrid Latency Measurements . . . . .	11
2.3 eBPF . . . . .	12
2.4 Starlink . . . . .	14
3 Research Questions	15
4 Research Methods	17
4.1 The Empirical Research Method . . . . .	17
4.2 Types of Experimental Evaluation . . . . .	19
4.2.1 Test Environments . . . . .	20
4.2.2 Measurement Methodology . . . . .	21
4.3 Limitations . . . . .	21
5 Contributions	23
6 Summary of Appended Papers	32
7 Conclusion	34
<b>PAPER I:</b>	
<b>Efficient Continuous Latency Monitoring with eBPF</b>	<b>65</b>
1 Introduction	66
1.1 Ethical Considerations . . . . .	68
2 Design and Implementation	68
3 Results	70
3.1 RTT Accuracy . . . . .	70
3.2 Monitoring Overhead . . . . .	72

4 Conclusion	76
Appendices	80
A Effects of Using TCP Timestamps to Infer RTT	80

## **PAPER II: Evolved Passive Ping—Passively Monitor Network Latency from within the Kernel** 87

1 Introduction	88
2 Background	89
2.1 Passive Network Latency Monitoring . . . . .	89
2.1.1 Algorithm for Passive TCP RTT Measurement . . . . .	89
2.1.2 Avoiding Inflation from Transmission Gaps . . . . .	91
2.1.3 Handling Duplicate TSval . . . . .	91
2.1.4 Extending the Algorithm Beyond TCP Timestamps . . . . .	92
2.2 eBPF . . . . .	93
2.2.1 The eBPF Instruction Set . . . . .	93
2.2.2 The Verifier and JIT Compilation . . . . .	93
2.2.3 eBPF Attachment and Program Types . . . . .	94
2.2.4 eBPF Helpers and Maps . . . . .	94
3 Design of epping	95
3.1 User Space Program . . . . .	96
3.2 eBPF Program . . . . .	96
3.3 Supporting Both TC and XDP . . . . .	97
3.4 Supporting IPv4 and IPv6 . . . . .	98
3.5 Avoiding Local RTTs . . . . .	98
3.6 Sampling RTTs . . . . .	99
3.7 Aggregation . . . . .	100
4 Optimizations	101
4.1 Keeping Track of Last Seen TSval . . . . .	101
4.2 Merging Forward and Reverse Flow State . . . . .	102
4.3 Tracking Number of Outstanding Timestamps . . . . .	103
4.4 Minimizing Key Size for Aggregated Statistics . . . . .	103
4.5 Keeping Packet Timestamps as Part of the Flow State . . . . .	104
5 Challenges	104
5.1 Concurrency . . . . .	104
5.1.1 Per-Flow State . . . . .	105
5.1.2 Per-Subnet and Global State . . . . .	106
5.2 Clearing Out Stale Map Entries . . . . .	106
5.3 eBPF . . . . .	108

5.3.1	Rapid eBPF Evolution . . . . .	108
5.3.2	The Verifier . . . . .	109
5.3.3	Limitations Inherent to eBPF Programs . . . . .	112
<b>6</b>	<b>Conclusion</b>	<b>112</b>
<b>7</b>	<b>Acknowledgements</b>	<b>113</b>

**PAPER III:**  
**Measuring Network Latency from a Wireless ISP—Variations**  
**Within and Across Subnets** **121**

<b>1</b>	<b>Introduction</b>	<b>122</b>
<b>2</b>	<b>Passively Collecting RTTs</b>	<b>123</b>
2.1	An Overview of epping . . . . .	124
2.2	Capturing the RTT Distribution . . . . .	124
2.3	Runtime Overhead . . . . .	125
2.4	Concurrency . . . . .	125
2.5	Memory Footprint . . . . .	126
2.6	Collecting Additional Traffic Features . . . . .	126
<b>3</b>	<b>Measurement Setup</b>	<b>127</b>
3.1	Network Topology . . . . .	127
3.2	Measurement Configuration . . . . .	128
3.3	Measurement Overhead . . . . .	129
<b>4</b>	<b>Overall Traffic Distribution Characteristics</b>	<b>130</b>
4.1	Protocol Distribution . . . . .	130
4.2	ECN Occurrences . . . . .	131
4.3	Internal and External Latency Variation . . . . .	132
4.4	Latency Variation Over Time . . . . .	134
<b>5</b>	<b>Variations in the External Latency</b>	<b>138</b>
5.1	Variations Between /8 Address Blocks . . . . .	138
5.2	Variations Between Individual Subnets . . . . .	140
5.3	RTT Distribution for Common ASes . . . . .	142
<b>6</b>	<b>Related Work</b>	<b>145</b>
<b>7</b>	<b>Limitations</b>	<b>148</b>
<b>8</b>	<b>Conclusion</b>	<b>149</b>
	<b>Appendices</b>	<b>154</b>
<b>A</b>	<b>Ethics</b>	<b>154</b>

<b>PAPER IV:</b>	
<b>Waiting at the Front Door—Continuous Monitoring of Latency in the Host Network Stack</b>	<b>159</b>
<b>1 Introduction</b>	<b>160</b>
<b>2 Related Work</b>	<b>161</b>
<b>3 Background</b>	<b>162</b>
3.1 Linux Kernel Network Stack . . . . .	163
3.2 Linux Packet Timestamping . . . . .	164
<b>4 Netstacklat Design</b>	<b>164</b>
4.1 Core Idea . . . . .	165
4.2 The Netstacklat Probe Points . . . . .	165
4.3 Aggregation of Latency Values . . . . .	167
4.4 Handling TCP Head-of-Line Blocking . . . . .	168
<b>5 Evaluation in Testbed</b>	<b>169</b>
5.1 Experimental Setup . . . . .	170
5.2 Latency Distribution . . . . .	171
5.3 Correlation to End-to-End Metrics . . . . .	174
5.4 Overhead . . . . .	175
<b>6 Comparing Overhead of Monitoring Tools</b>	<b>176</b>
6.1 Tool Configuration . . . . .	177
6.2 CPU Overhead . . . . .	177
6.3 Impact on End-to-End Performance . . . . .	178
<b>7 Deployment at Cloudflare</b>	<b>180</b>
7.1 Latency Under Increasing Server Load . . . . .	180
7.2 Temporary Latency Anomaly . . . . .	183
<b>8 Limitations and Future Work</b>	<b>185</b>
<b>9 Conclusion</b>	<b>186</b>
<b>Appendices</b>	<b>192</b>
<b>A Ethics</b>	<b>192</b>
<b>B Additional Details from Testbed Experiments</b>	<b>192</b>
B.1 Latency Distributions . . . . .	192
B.2 End-to-End Performance . . . . .	195
B.3 System Load . . . . .	196
<b>C Tool Comparisons for Additional Workload Configurations</b>	<b>196</b>

<b>PAPER V:</b>	
<b>Characterizing Wireless Link Throughput with eBPF and Hardware Timestamps</b>	<b>203</b>
1 Introduction	204
2 IPD Tool Design	205
2.1 The Challenge with Link-Wide IPD . . . . .	206
2.2 Packet Timestamp Collection . . . . .	206
2.3 Sorting Packet Timestamps . . . . .	207
2.4 Efficient Encoding of IPD Information . . . . .	208
3 IPD Capture Evaluation	209
3.1 Experiment Setup . . . . .	209
3.2 IPD Quality . . . . .	210
3.2.1 Hardware Timestamps . . . . .	210
3.2.2 Software Timestamps . . . . .	210
3.3 Evaluation of Reordering . . . . .	213
3.3.1 Degree of Reordering . . . . .	213
3.3.2 Maximum Reordering . . . . .	214
4 Example of an IPD-Based Examination	215
5 Discussion	216
6 Conclusions	217
 <b>PAPER VI:</b>	
<b>A Detailed Characterization of Starlink One-way Delay</b>	<b>223</b>
1 Introduction	224
2 Background and Related Work	224
3 Measurement Collection	225
4 Day-Scale Variation	226
5 Second-Scale Variation	227
6 Millisecond-Scale Variation	228
7 Reconfiguration Delay Anomaly	230
8 Delay Variation Modelling	231
9 Conclusions	234

<b>PAPER VII:</b>	
<b>Characterizing the Configuration of Starlink Queuing</b>	<b>243</b>
1 Introduction	244
2 Background and Related Work	245
3 Measurement Setup	246
4 Per-Flow or Per-User Queues	247
5 Queuing Behavior Overview	248
6 Buffer Management	249
7 Queue Sizing	255
8 Load-Based Rate Adaptation	255
9 Conclusions	257
Appendices	264
A Ethics	264
B ECN Behavior	264
C Network Path	264
D Example Runs Randomly Selected Among Runs with Loss > 30%	267

## List of Appended Papers

1. **S. Sundberg**, A. Brunstrom, S. Ferlin-Reiter, T. Høiland-Jørgensen, and J. D. Brouer. Efficient Continuous Latency Monitoring with eBPF. *Passive and Active Measurement Conference*. Lecture Notes in Computer Science, vol. 13882. 2023.
2. **S. Sundberg**. Evolved Passive Ping—Passively Monitor Network Latency from within the Kernel. Technical Report, Karlstad University. 2024.
3. **S. Sundberg**, A. Brunstrom, S. Ferlin-Reiter, T. Høiland-Jørgensen, and R. Chacón. Measuring Network Latency from a Wireless ISP—Variations Within and Across Subnets. *Proceedings of the ACM Internet Measurement Conference*. 2024.
4. **S. Sundberg**, A. Brunstrom, S. Ferlin-Reiter, J. D. Brouer, and T. Høiland-Jørgensen. Waiting at the Front Door—Continuous Monitoring of Latency in the Host Network Stack. Accepted for presentation at the *ACM Internet Measurement Conference*. 2026.
5. **S. Sundberg**, J. Garcia, and A. Brunstrom. Characterizing Wireless Link Throughput with eBPF and Hardware Timestamps. *IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks*. 2023.
6. J. Garcia, **S. Sundberg**, and A. Brunstrom. A Detailed Characterization of Starlink One-way Delay. *Proceedings of the 3rd Workshop on LEO Networking and Communication*. 2025.
7. J. Garcia, **S. Sundberg**, and A. Brunstrom. Characterizing the Configuration of Starlink Queuing. Accepted for presentation at the *ACM Internet Measurement Conference*. 2026.

## Comments on my Participation

**Paper I** I am the first author and wrote most of the paper. The initial idea to implement passive ping in eBPF was proposed by co-authors Anna Brunstrom and Toke Høiland-Jørgensen. I am, however, responsible for most of the design and implementation of the tool presented in the paper, although all co-authors have contributed with feedback on various aspects of the tool, and Toke Høiland-Jørgensen and Jesper Dangaard Brouer provided technical expertise regarding implementation details. With support from my co-authors, I also planned, ran, and analyzed the results of all the experiments described in the paper.

**Paper II** I am the sole author of this technical report, and as such wrote the entire paper with some minor editing by my supervisor, Anna Brunstrom. The paper is based on my own experience from implementing the passive measurement tool. I primarily designed and implemented the tool myself. Likewise, the single experiment presented in the paper was done by me. However, as outlined in my participation for Papers I and III, the original idea for the tool is not mine, and I received support and feedback from Anna Brunstrom, Simone Ferlin-Reiter, Toke Høiland-Jørgensen, and Jesper Dangaard Brouer throughout the entire development process.

**Paper III** As the first author, I wrote the vast majority of this paper. I designed and implemented the aggregation functionality described in the paper with feedback from the co-authors. While I provided some assistance for setting up the measurements, Robert Chacón deployed the measurement tool in his ISP network and collected the data on which the study is based. I then processed, visualized, and analyzed the measurement data to derive the results presented in the paper.

**Paper IV** The original idea (as described in Section 4.1 of the paper) for the `netstacklat` tool we present in the paper, was provided by Jesper Dangaard Brouer. I am, however, the one who designed and implemented the tool and performed all the experiments for evaluating it in the controlled testbed setting. Jesper adapted the tool as needed to deploy it at Cloudflare and shared the observations from this deployment, presented in Section 7. I wrote the vast majority of the paper, although Simone Ferlin-Reiter and Høiland-Jørgensen wrote most of the background on the Linux network stack (Section 3.1), and all authors have edited various parts of the text.

**Paper V** As the first author, I am responsible for a large part of the underlying work in this paper. I and co-author Johan Garcia have contributed equally to the writing of the paper, where I primarily wrote Sections 2-3.2 and 5, while Johan wrote most of 3.3-4. Johan Garcia came up with the initial idea for the `IPD-tool` we present in the paper, while I implemented the tool and carried out all the measurements. The analysis in the paper is joint work, where I primarily focused on the evaluation of the IPD quality, while Johan carried out most of the analysis for packet reordering and for assessing the performance of the 5G link.

**Paper VI** For this paper, my main contribution is the development of the nanoprobe measurement tool and carrying out the measurement campaign, where the tool was used to collect all the data that the paper is based on. Johan Garcia, who is the first author, analyzed the measurement data and wrote the vast majority of the paper. I only wrote parts of the section on measurement collection and performed some editing throughout the paper.

**Paper VII** Similarly to Paper VI, first author Johan Garcia wrote the majority of the paper, analyzed the measurement data, and also implemented the queuing simulator used for the analysis. Meanwhile, I implemented the measurement tool and performed all the network measurements. Additionally, I wrote part of the section on the measurement setup, performed the per-flow queuing analysis (Figure 1 in the paper), and carried out some auxiliary experiments and analysis, like the ECN and path measurements described in the appendix.

## Other Publications

- S. Sundberg and A. Brunstrom. Poster: Using BPF to Measure Latency at High Link Speeds. *ACM Internet Measurement Conference*. 2021.
- S. Sundberg, A. Brunstrom, S. Ferlin-Reiter, T. Høiland-Jørgensen, and J. D. Brouer. Passive Monitoring of Network Latency at High Line Rates. *The 17th Swedish National Computer Networking Workshop*. 2022.
- J. Garcia, S. Sundberg, G. Caso, and A. Brunstrom. Multi-Timescale Evaluation of Starlink Throughput. *Proceedings of the 1st ACM Workshop on LEO Networking and Communication*. 2023.
- J. Garcia, S. Sundberg, and A. Brunstrom. Inferring Starlink Physical Layer Transmission Rates Through Receiver Packet Timestamps. *IEEE Wireless Communications and Networking Conference*. 2024.
- J. Garcia, S. Sundberg, and A. Brunstrom. Fine-Grained Starlink Throughput Variation Examined with State-Transition Modeling. *19th Wireless On-Demand Network Systems and Services Conference*. 2024.
- J. Garcia, M. Beckerle, S. Sundberg, and A. Brunstrom. Modeling and Predicting Starlink Throughput with Fine-Grained Burst Characterization. *Computer Communications*. Vol. 234. 2025.
- J. Garcia, S. Sundberg, and A. Brunstrom. TCP Congestion Control Performance over Starlink. *Proceedings of the 2025 Applied Networking Research Workshop*. 2025.



# Introductory Summary



“Every moment of our lives is precious, and every moment spent waiting on a computer, wasted.”

— *Dave Thät*  
*On reducing latencies below the perceptible (2013)*



# 1 Introduction

The Internet has become an integral part of modern society, today connecting over 6 billion users [1]. As the networks connecting us to the Internet have evolved, the focus has traditionally been on improving the rate at which they can transfer data. However, as data rates have steadily increased, with the average data rate for broadband connections now surpassing 100 Mbps in many countries [2–4], it has become increasingly clear that higher data rate alone will not be enough to satisfy the needs of all the applications we connect over the network. Instead, reducing the time it takes to deliver a single network packet across the network, the *network latency*, has become a key factor for improving network performance.

For one of the most common Internet applications, loading web pages, increasing the data rate beyond 20 Mbps has been shown to have little impact on the page load time, with network latency becoming the limiting factor instead [5–8]. Figure 1 shows how increasing data rate and reducing latency impact the time to load a webpage. As can be seen, increasing the data rate initially improves performance. However, once sufficient capacity is available, in this case past 3 Mbps, further increasing data rate has diminishing returns with only very minor reductions in page load time. Meanwhile, reducing latency consistently reduces load time. Network latency becomes even more important when we consider more interactive applications, with studies showing that network latency has a large impact on the end user's quality of experience (QoE) for audio and video calls [9–13], live video streaming [14–16], gaming [17–21], virtual reality (VR) [22, 23], and teleoperation of unmanned vehicles and robots [24–29].

Even though network latency has a large impact on many applications, network operators and service providers often lack the means to monitor the latency for their network traffic, leaving them blind to issues causing latency in their networks and unable to assess the quality of service (QoS) they are able to provide to their customers. Furthermore, continuous monitoring of network latency is needed to assess the impact that various solutions aimed at reducing latency, such as edge computing [30–33], changes to the end host network stack [34–37], active queue management (AQM) algorithms [38–40], and new radio transmission schemes [41, 42], have once deployed in production. Unfortunately, publicly available software solutions for monitoring the latency of application traffic have too high overhead to support continuous monitoring [43–46], are tailored for a distinct environment [47], or require specific hardware, like Tofino [48] switches, to deploy [49–51]. Therefore, network operators largely have to rely on active latency measurements [52–56], which may not accurately reflect the latency for real application traffic [57].

To address the limited visibility of network latency for application traffic, we in this thesis design novel solutions for passively monitoring network latency that efficiently run in-line with the Operating System (OS) network stack. We provide solutions both for monitoring the end-to-end latency across networks, as well as for tracking the latency caused by the end host itself. By leveraging

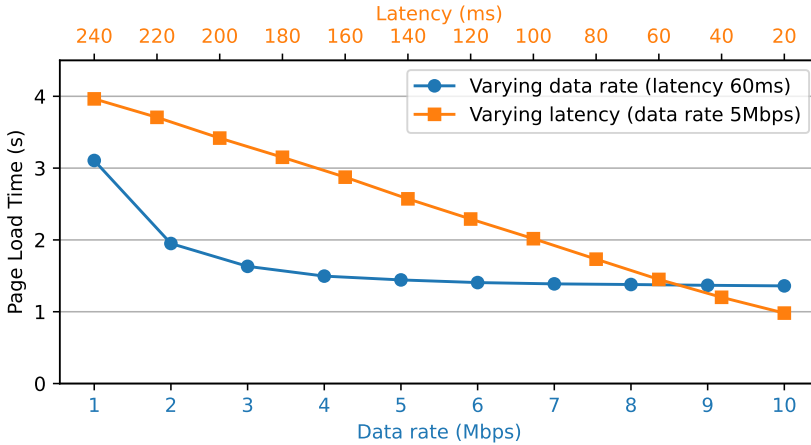


Figure 1: How the network data rate and latency impact the time to load a web page. Data from [5].

the recently emerging eBPF [58] technology in Linux, our implementations achieve low enough overhead to support continuous monitoring while also remaining compatible with the Linux network stack. Our work thereby enables general-purpose devices running Linux to efficiently monitor latency, supporting deployment across a wide range of network infrastructure, such as web servers [59–61], load balancers and proxies [62–66], network intrusion detection systems and firewalls [67–70], and routers [71–74]. To demonstrate the feasibility of continuously monitoring the latency of application traffic and the utility of the monitoring data, we deploy our solutions in real production networks. From these deployments, we, for instance, identify that the last-mile access remains a significant source of latency in an ISP network, and observe how an unbalanced packet processing workload can drastically increase the latency within web servers.

In recent years, we have also seen the deployment of new wireless network technologies, like the 5th generation (5G) cellular networks, WiFi 7, and the satellite network Starlink, offering users convenient high-capacity connections to the Internet. However, the performance for wireless networks can rapidly fluctuate due to changes in the physical radio environment [75] and the wireless resource allocation mechanisms [76]. How the network latency varies over short timescales in these wireless networks is not yet well understood, making it challenging to predict the network performance and accurately emulate the link behavior for latency-sensitive applications. This is further complicated by proprietary solutions where the internal details of the network are unknown, where, for instance, many aspects of how the Starlink network operates are not publicly known [77, 78]. To better understand the latency characteristics of these wireless connections, we design and implement new high-fidelity active measurement tools that can capture how the dynamic radio environment and

internal resource scheduling impact latency. Furthermore, we use these tools to perform extensive measurements over the Starlink network, revealing aspects of its internal operation and how its resource scheduling, reconfiguration intervals, and queuing behavior cause different levels of latency variation.

In summary, network latency on the modern Internet is still not well understood, as network operators often lack the means to monitor the latency of application traffic, and researchers have yet to fully grasp the latency characteristics of emerging wireless technologies. To improve the visibility of network latency, we design widely deployable passive monitoring solutions and high-fidelity active measurement tools. By utilizing these solutions to perform measurement campaigns in production networks and analyzing the results, we gain a better understanding of real-world network latency. The rest of this introductory summary is structured as follows: Section 2 provides relevant background on network latency, methods to measure latency, the eBPF technology that our work utilizes, and the Starlink network that we study. Section 3 lays forward the research questions this thesis addresses, while Section 4 explains the methods used to address them. The contributions of this thesis are outlined in Section 5, with Section 6 summarizing the appended papers. Finally, Section 7 concludes the introductory summary.

## 2 Background and Related Work

In this section, we provide relevant background for understanding the contents of this thesis and how it relates to other research. Section 2.1 further explains what network latency is and how it arises, while Section 2.2 covers different ways of measuring latency and how the work in this thesis addresses limitations with previous measurement solutions. In Section 2.3, we provide an overview of eBPF—a key technology that several of our measurement solutions rely on—focusing on its applications for networking. Finally, we briefly cover the Starlink network and the challenges with understanding its latency characteristics in Section 2.4.

### 2.1 Network Latency

There are several metrics that can be used to assess the performance of computer networks. The most common network metrics are throughput, latency, and packet loss. The throughput is the amount of data that is transferred over the network within a given time interval, typically measured in bits per second. In contrast, latency is the time it takes for a single network packet to traverse the network, typically measured in milliseconds. Packet loss is the fraction of network packets that are lost when traversing the network, and is usually expressed as a percentage. In this thesis, we focus on *network latency* due to its increasing importance for modern networked applications as outlined in the introduction. Additionally, we also consider the inter-packet delay (IPD), which is the time between two consecutive network packets being delivered by

the network, which in turn is a function of network throughput and latency variation.

### 2.1.1 One-Way Delays and Round Trip Times

Network latency can either be measured one-way or two-way. The one-way latency, also known as the one-way delay (OWD), is the time it takes a network packet to go from its source to its destination. The two-way latency, usually referred to as the round-trip time (RTT), is instead the time it takes a network packet to go from the source to the destination, and then for the destination to send a packet back to the source again. The RTT is hence the sum of the OWD from the source to destination, and the OWD from the destination to the source, together with any additional time it takes for the destination to send a reply upon receiving the original packet. While the OWD offers higher granularity, it is in practice often not feasible to measure, as it generally requires that the clocks at the source and destination are synchronized with each other. In contrast, the RTT can be measured from the source node alone and therefore avoids the need for synchronized clocks, making it a common alternative to OWD [79]. Furthermore, as the destination end host often responds to received packets, e.g., to acknowledge the reception of data or to respond to a request, the RTT also maps well to many networked applications and provides a lower bound for how long an application will have to wait for a response when communicating over the network. The OWD is sometimes estimated as half the RTT; however, this is often an inaccurate estimation, as the latency for the forward and reverse paths can differ [79–81].

In this thesis, we utilize the more granular OWD measurements when feasible, i.e., when we can ensure that the source and destination have synchronized clocks. This is the case when we measure the latency within the end host (Paper IV), where all points use the same system-wide clock, and for the latency measurements over Starlink (Paper VI and VII), where we use a looped network topology where the same host is used as both the source and destination. Meanwhile, when measuring the latency for general Internet traffic (Paper III), where we have no control over the end hosts, we rely on the RTT instead.

### 2.1.2 Sources of Latency

One important attribute of network latency is that, unlike throughput, it is additive [82]. The latency for each hop along the traversed network path adds up to form the total end-to-end latency across the whole network. Subsequently, the end-to-end latency can also be decomposed into smaller components, and achieving low end-to-end latency requires achieving low latency at each step of the way. Within each network hop, there are several potential sources that may contribute to the network latency, which are typically divided into the following four categories [83, 84]:

**Propagation delay** The propagation delay is the time it takes a network packet to travel (propagate) the distance across a network link. This delay is primarily affected by the geographical distance along the network path between the source and destination, and is a fundamental lower bound of the network latency posed by the physical speed of light. Note, however, that the network path is often significantly longer than the shortest geographical distance between the source and destination due to the placement of network infrastructure and non-optimal routing [33, 85, 86], which can cause a significant latency inflation, especially in regions with underdeveloped network infrastructure [81, 87] and for satellite networks [88].

The main approach to reduce propagation delay is to physically move the services closer to the end-user, which is achieved by, for example, leveraging Content Distribution Networks (CDNs) [88–91] and edge computing [30–33, 92]. However, several previous studies [90, 93–95], as well as our study in Paper III, indicate that large latencies often occur within the so-called last-mile networks, where propagation delay is not a major component. Moving content closer to end-users will therefore not be sufficient to achieve consistent low latency.

**Processing delay** Each element along the network path spends some time processing network packets as they arrive, adding processing delay. For switches and routers, this processing includes parsing the packet headers to determine where to forward the packet next, and the processing time is often assumed to be small and constant [96]. For end hosts, the processing time includes the Network Interface Card (NIC) transferring the packet to the CPU, the OS network stack allocating necessary packet buffers, performing protocol specific processing, and delivering the packet to the right application, and finally the application reading the packet payload. As the development of higher capacity networks has outpaced the increase in host resources (CPU, cache etc.), several studies have found that significant latency can accrue within the end host itself, both within the hardware interconnect between the NIC and CPU [97–99], as well as within the OS network stack [35–37, 100]. We further investigate the latency within the OS network stack in Paper IV, where we present a novel solution to monitor this processing delay in the end host.

**Queueing delay** Network devices use queues to manage traffic bursts were network packets under a short period arrive faster than they can be handled. This gives rise to queueing delay, where packets in the queue have to wait for all the preceding packets to be handled. The queueing delay will vary depending on the number of packets in the queue at any given moment. While small temporary queues are useful to absorb traffic bursts, a seminal article [101] from 2011 highlighted the presence of large and unmanaged queues that could add seconds of latency in Internet connections, a phenomenon termed bufferbloat. Since then, significant research and engineering efforts have gone into reducing queueing delays. For example, several new congestion control [102–105] and active queue management (AQM) [38–40, 106, 107] algorithms have been

developed. Notably, the Low Latency, Low Loss, and Scalable Throughput (L4S) [108] architecture, recently standardized by the Internet Engineering Task Force (IETF), combines both congestion control and AQM to reduce queueing delays.

As queueing delay only occurs when the network traffic load exceeds the capacity of some part of the network, the network latency when the network is under high traffic load, sometimes referred to as the *latency under load* or *working latency* [109], may be significantly higher compared to when the network is mostly idle. The traffic load is therefore important to consider when measuring network latency. Our study of latency for Internet traffic (Paper III) is based on real application traffic to capture the variations caused by the traffic under realistic networking scenarios. When studying the latency of the Starlink network, we deliberately generate different traffic loads to capture the latency variations inherent to Starlink both without queueing delays (Paper VI) and with queueing delays (Paper VII).

**Transmission delay** The time it takes to transmit all the bits of a network packet into the physical medium of the link is known as the transmission delay. In dedicated wired connections, this is often just the serialization delay given by  $\frac{\text{packet size}}{\text{link throughput}}$ , which can be lowered by increasing the link throughput [96]. However, for wireless links where the medium (electromagnetic spectrum) is shared, we also have to account for medium access control (MAC) and resource scheduling, which may for example delay the transmission to a scheduled time slot, thereby adding additional latency<sup>1</sup> [82]. For ultra-reliable low latency communication (URLLC) in 5th generation (5G) cellular networks, which aims to reliably achieve link latencies below one millisecond [110], new link-layer frames [41] and support for shorter time slots in resource scheduling [42, 111] have been introduced to reduce latency. As part of this thesis, we perform highly granular latency measurements over Starlink that reveal how its resource scheduling impacts the latency (Paper VI).

## 2.2 Measuring Latency

There are numerous ways to measure network latency. As already covered in Section 2.1.1, the latency can either be measured as the OWD or the RTT of the network. Furthermore, like all types of network measurements, latency measurements can be divided into *active* and *passive* methods, as specified by RFC 7799 [112]. Active methods generate their own packet streams, while passive methods only observe existing packet streams without interfering with them. Additionally, there are hybrid methods that combine aspects of active and passive methods.

---

<sup>1</sup>The additional delay to access the link is often separated out from the transmission time and referred to as medium access delay instead.

### 2.2.1 Active Latency Measurements

Active network latency measurement techniques generate network probe packets and measure the delay for the probes as they traverse the network. They thus measure the network latency by actively testing it. The most well-known method to actively measure network latency is likely ping [52]. To measure latency, ping sends an ICMP echo request packet to a specified host. As all hosts connected to the Internet are required to respond to ICMP echo requests [113], ping can in theory be used to measure the latency to any host on the Internet, although routers are known to filter [114] or rate-limit [115] ICMP messages, limiting the visibility in some networks. There are many additional tools that measure the latency by emitting probe packets in a similar manner, such as traceroute, IRTT [116], One-way Active Measurement Protocol (OWAMP) [117], Two-way Active Measurement Protocol (TWAMP) [118] and PerfTrace [119].

Most tools for actively measuring network latency rely on software timestamps, which are either generated by the kernel network stack [120] or by the measurement application itself. While software timestamps are accurate enough for measuring general Internet-scale latency, usually in the order of several milliseconds, they include additional processing latency from the end host that makes them less suitable for measuring small changes in latency at sub-millisecond scales. Previously, costly specialized measurement devices have been required to accurately measure network latencies at microsecond scales. However, nowadays many NICs can provide hardware timestamps for when packets arrive, enabling accurate latency measurements on commodity hardware [121]. Unfortunately, there are few publicly available tools that can use hardware timestamps for latency measurements. While the MoonGen [122] traffic generator supports hardware timestamps, it can only use hardware timestamps for a single outstanding packet at a time, limiting the frequency of hardware-assisted latency measurements to once per RTT. Meanwhile, HiPerConTracer [123] has no such frequency limit on its hardware-assisted latency measurements, but can only measure the RTT. In this thesis, we therefore develop the new active network latency measurement tool nanoprobe (based on nanoping [124]). Nanoprobe uses hardware timestamps on both the client and server side to provide accurate OWD measurements. We also use hardware timestamps to accurately measure the inter-packet delay with our IPD-tool, which can be combined with an external traffic generator to test how a network link impacts the packet arrival pattern.

While active latency measurements allow for highly accurate measurements under controlled conditions, they also have several drawbacks that make them less suitable for continuously monitoring the latency for regular application traffic in production networks. First off, the measurement probes add additional traffic to the network. Keeping updated latency measurements to all hosts in a large network without adding excessive network overhead requires sophisticated schemes [54, 125], where the latency is only measured between a subset of hosts and estimated for the rest. The additional network traffic also perturbs the network, creating an observer effect [126] where the latency for application

traffic may be impacted by the additional probe packets. Furthermore, active latency measurement tools that do not rely on ICMP messages typically require the installation of a server application capable of responding to the probe packets on the target host. This limits the usability of many active measurement tools for network operators that do not have control over the end hosts they deliver traffic to. Finally, the network latency measured for the probe packets may not accurately reflect the latency that ordinary application traffic encounters [57].

### 2.2.2 Passive Latency Measurements

Purely passive latency measurements neither generate new packets nor modify any existing packets to measure latency. Instead the latency is inferred by observing existing traffic on the network. Hence, in contrast to active measurements, passive measurements monitor the latency of ordinary application traffic rather than testing the latency with network probes.

One common way to passively measure RTTs is by measuring the time between matching message-reply packet pairs, where the later reply packet can be assumed to have been sent back as a response to the initial message packet. This is often done for TCP traffic as the TCP protocol sends back acknowledgements when receiving data packets, making it possible for any host along the path of the TCP traffic to monitor the latency. Current tools that can passively monitor TCP RTTs include `tcptrace` [127], `Wireshark` [43], `pping` [44], and `tstat` [128]. However, all these tools rely on traditional user space packet capturing that comes with significant overhead, making it challenging to monitor traffic in modern multi-gigabit networks [129, 130]. To achieve higher performance, some have turned to hardware acceleration, where a line of work [49–51] from Princeton University uses P4 [131] to implement passive TCP RTT monitoring on Intel Tofino [48] switches. The aforementioned `tstat` can also be configured to use Endace DAG [132] cards. While these hardware-accelerated solutions offer excellent performance, their deployment is limited to environments where such hardware is readily available. Furthermore, `tstat` has also been ported [133] to the Data Plane Development Kit (DPDK) [134], offering a high-performance software-based alternative, although it requires the host to be entirely dedicated to just monitoring the network. To enable passive latency monitoring in a wider set of environments, we use eBPF to implement TCP RTT monitoring in the Linux kernel, offering much lower overhead than the user space packet capturing solutions while still remaining fully compatible with the Linux network stack and any applications running on top of it.

Another approach to passively measure network latency is to observe the same packet from multiple points in the network, making it possible to calculate the OWD between the observation points. This approach may be used in a distributed context to measure OWD between different hosts [135, 136]. In this thesis, we instead use eBPF to apply a similar method within a single end host, monitoring the processing latency between different stages in the kernel network stack. While some prior tools [45, 46, 137, 138] are also able to monitor latency within the host in a similar manner, our solution

sacrifices some granularity to drastically reduce the overhead, making it feasible to continuously monitor the host latency in production.

As passive latency measurements are inferred directly from application traffic and avoid the drawbacks with activate latency monitoring covered in Section 2.2.1, passive approaches are generally well-suited for continuous monitoring of application perceived network latency. However, the passive nature also limits the measurements to what the existing network traffic covers. As passive monitoring gives up the control of the traffic the measurements are based on, it cannot uncover the latency to hosts to which no traffic is sent, nor explore how different traffic patterns other than the currently occurring ones will impact network latency.

### 2.2.3 Hybrid Latency Measurements

There exist some latency measurement techniques that fall between active and passive measurements, having some properties of both. The most prominent type of hybrid methods is the so-called in-band network telemetry, where switches or routers along the path provide telemetry information as packets traverse them. While there are different types of in-band network telemetry solutions, at a high level, they usually work as follows: First, a source node inserts a telemetry header into a packet. Intermediate nodes along the network path then fill in the telemetry header with the requested data as they forward the packet. This can include information like a path or port identifier, link utilization, delay, or timestamps. Finally, a sink node strips the telemetry header and collects the telemetry data, often forwarding it to a telemetry server [139].

While the concept of in-band network telemetry has existed for a long time [140], it started to grow popular with the advent of programmable data planes brought about by P4 programmable switches. The programmable data plane enabled rapid development and deployment of new in-band network telemetry solutions [141–143] in data center networks, further strengthened by the In-band Network Telemetry (INT) specification [144] for P4 programmable devices. In-band network telemetry can offer very accurate and fine-grained latency measurements, for example, hop-by-hop latency between switches. However, in-band network telemetry requires support at both source, intermediate, and sink nodes, and can therefore usually only be deployed within a network domain where the operator has full control over all involved devices, predominantly within data center networks. In-band network telemetry is therefore generally not applicable when studying the network latency across the wider Internet, and is not an approach we explore in this thesis. For a more extensive overview of in-band network telemetry and its use cases, see Tan et al. [139].

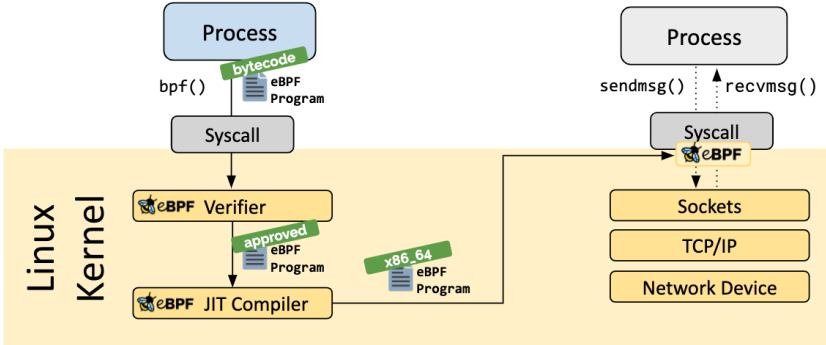


Figure 2: Overview of the process for loading an eBPF program. From [148] by eBPF.io authors, used under Creative Commons Attribution 4.0.

## 2.3 eBPF

The underlying technology we utilize in several of our solutions to efficiently monitor network packets inside the Linux kernel, is called eBPF<sup>2</sup>. The eBPF technology allows for safe and efficient execution of custom programs inside the Linux<sup>3</sup> kernel. This can be used to load programs that can observe or even modify kernel behavior at runtime, without needing to modify, recompile, and reboot the kernel itself, nor load any kernel modules.

The process for loading an eBPF program is illustrated in Figure 2. First, the eBPF program needs to be composed in an eBPF-specific instruction set [145]. This is typically accomplished by writing the program in C and then compiling it to a sequence of eBPF instructions called eBPF bytecode. The eBPF program can then be loaded into the kernel from user space by using the `bpf()` system call. When the kernel loads the eBPF program, a component called the verifier will statically analyze the program to ensure it is safe, for example, by verifying that it will never access any out-of-bounds memory or get stuck in an infinite loop. This verification process is one of the key advantages compared to kernel modules, as it ensures that an eBPF program can never crash or freeze the kernel [58]. If the eBPF program passes the verifier, a Just-In-Time (JIT) compiler will convert the eBPF instructions into native machine instructions to provide good runtime performance. The loaded eBPF program can then be attached to a hook inside the kernel, and will then run every time the hook is triggered. To keep state between invocations, as well as to communicate with other eBPF programs and with user space processes, eBPF programs can use special key-value stores provided by the kernel called eBPF maps. A more comprehensive overview of eBPF is provided in Section 2.2 of Paper II and by Gbadamosi et al. [58].

<sup>2</sup>Used to be an acronym for extended Berkeley Packet Filter, but as it is capable of far more than filtering packets, eBPF is nowadays considered a name and does not stand for anything.

<sup>3</sup>eBPF was originally developed for Linux, but is currently being standardized at the IETF [145] and adopted by other operating systems like Windows [146] and BSD [147].

eBPF can be used for many different purposes, depending on which type of hook the eBPF programs are attached to. The traffic control (tc-BPF) and eXpress Data Path (XDP) [149] hooks trigger every time a packet is received or transmitted, and allow eBPF programs to inspect the packet, modify packet content, and take actions on the packet, such as dropping it or redirecting it to a different interface. XDP and tc-BPF programs can thus be used to realize a programmable data plane in the Linux kernel [150]. This has been leveraged in the industry, for example, to implement the efficient Container Network Interface Cilium [151], used by Google and Amazon, the fast Katran [64] load balancer developed for Facebook, and the Unimog [65] load balancer used by Cloudflare. In academia, eBPF has been proposed for a wide range of network related tasks, including extending transport protocols [143, 152–154], supporting network function virtualization (NFV) [155–158], detecting malicious or anomalous traffic [69, 159–161], and implementing components of the 5G network [162–164].

In this thesis, we focus on using eBPF for monitoring network traffic rather than implementing new networking functionality. eBPF has several attributes making it an attractive choice for monitoring network packets. Compared to traditional user space packet capturing, like `libpcap` [165], eBPF does not need to clone packets and then process the cloned packets in user space. Instead, eBPF can directly inspect the original packets inside the kernel. Compared to packet processing techniques that bypass the kernel, like DPDK, eBPF still allows the packets to be processed by the Linux network stack instead of moving all networking functionality into user space. This makes it possible to transparently deploy eBPF applications on a wide range of devices already relying on the rich functionality offered by the Linux network stack without any change in tooling [166]. DPDK also dedicates a CPU core to busy polling, making eBPF a more suitable choice in environments where dedicating a full core for network monitoring is costly [167]. Furthermore, in addition to the aforementioned XDP and tc-BPF hooks, eBPF allows hooking into virtually any kernel function, a capability that enables us to observe network packets at multiple different points in the kernel network stack, enabling us to monitor latency within the end host.

There already exist several tools that utilize eBPF to monitor network latency in different ways. Several works [143, 155, 168–170] have implemented forms of in-band telemetry with eBPF, where switches or routers append timestamps or latency values to existing packets, and an eBPF program at the end host extracts the telemetry data to monitor the latency application traffic encounters along the path. However, as mentioned in Section 2.2.3, such solutions require a coordinated deployment across all hosts along the measurement path, typically limiting them to monitoring the latency within a network domain under the control of a single network operator. The `noobprobe` [171] tool implements active monitoring of RTTs by cloning a subset of existing network packets but limiting their time-to-live (TTL) to capture hop-by-hop latency in a similar manner as `traceroute`. As the RTT probes are based on cloned packets, they blend in with the normal application traffic, thereby

avoiding any differential treatment that active measurements are otherwise prone to. However, like other active measurements, the cloned packets still add additional network traffic, so the latency can only be monitored for a small subset of traffic to avoid excessive network overhead. For passive monitoring, the `tcprtt` tool from the BPF Compiler Collection (BCC) [172] and the `netobserv-ebpf-agent` [173, 174] both passively monitor TCP RTT by extracting the RTT estimates that the kernel TCP stack keeps for each connection. In this thesis, we also present a tool for passively monitoring TCP RTTs; however, our solution is based on inspecting the TCP packet headers and therefore works on any middlebox that TCP traffic passes through, not only the end hosts. Furthermore, there are several eBPF-based tools [45, 46, 137, 138] that can trace packets within the end host’s kernel network stack. However, these tools are primarily designed for temporary testing or debugging of the network stack, with their high overhead making them impractical for continuously monitoring the end host latency. In this thesis, we provide an alternative tool that also makes use of eBPF to monitor the latency of packets as they traverse the kernel network stack; however, we perform in-kernel aggregation of the measurements to drastically reduce the monitoring overhead.

## 2.4 Starlink

Starlink is a low Earth orbit (LEO) satellite network consisting of over 9 thousand satellites, providing Internet access to more than 9 million subscribers across 155 countries and markets [175]. Starlink subscribers access the network via their Starlink-supplied user terminal, which in turn connects to one of the satellites currently in range. The satellite then relays the user’s data to a ground station (possibly after multiple inter-satellite hops), which then connects to the terrestrial Internet through one of several globally distributed points of presence (PoPs) [78]. However, most details about how Starlink operates internally are not publicly known, forcing network researchers to treat Starlink like a black box, where the inner workings have to be deduced through external measurements [77, 78, 176]. Through various forms of measurements, researchers have, for instance, learned that the radio link between the user terminal and the satellite uses 1.33 ms long link-layer frames [77, 177], discovered the existence of a global controller reconfiguring the network every 15 seconds [78, 178, 179], and observed how Starlink usually routes all of a subscriber’s traffic through a single satellite hop to a per-user statically assigned ground station [176, 180, 181], although recent reports indicate that Starlink now supports inter-satellite routing with multiple satellite hops as well [78, 182].

From a latency perspective, the low altitude (~550 km) of the LEO satellites allows Starlink to achieve much lower latencies than prior geostationary orbit (GEO) satellite networks, avoiding the hundreds of milliseconds of propagation delay needed to reach a GEO satellite (35 786 km). There are, however, many aspects related to both the radio access technology and the satellite and ground station network that can influence Starlink’s latency. In addition

to studies on the general latency for Starlink in different regions [78, 183], researchers have, for instance, investigated how the routing between the (constantly moving) satellites and ground stations [180, 182], the location of terrestrial PoPs [88, 182, 184, 185], mobility [186–189], multi-user contention [190], and weather [191, 192] impact the latency. In this thesis, we further contribute to the understanding of Starlink's latency characteristics by performing high-fidelity measurements that reveal how Starlink's resource scheduling affects the latency (Paper VI), and also use latency as a means to infer the queuing configuration (Paper VII).

### 3 Research Questions

The primary objective of this thesis is to *improve the visibility and understanding of network latency in production networks*. To achieve this overarching goal, we set out to answer the following research questions (RQs):

RQ1 *How can we enable efficient continuous monitoring of the network latency encountered by application traffic?*

As covered in Section 2.2.1, active measurements are not well-suited for continuous network monitoring as they add additional network traffic and may not capture the latency for ordinary application traffic. Supporting continuous latency monitoring for application traffic, therefore calls for efficient passive measurement solutions that can run in a wide range of network environments, where prior passive solutions either have too high overhead or require specific hardware that often make them impractical to deploy.

Here, we identify eBPF as an enabling technology that allows observing network packets in-line with the kernel network stack, offering a low-overhead alternative to traditional packet capturing. Unlike kernel bypass technologies, eBPF-based monitoring remains fully compatible with the rich Linux networking ecosystem, making it feasible to deploy in many parts of the already existing network infrastructure. Furthermore, we find aggregating the measurements directly in the kernel to be a key approach for avoiding the overhead from pushing individual measurement values, as well as to keep the footprint of the monitoring data manageable. Based on these insights, we design two new passive monitoring solutions. Our first solution, `epping`, can passively monitor TCP RTTs from network middleboxes (Papers I, II, and III). This allows network operators that have no control over the end hosts, such as ISPs, to monitor the latency that their customers experience. Our second solution, `netstacklat`, instead monitors the latency within the end host itself, allowing service providers to observe how much latency is added by the kernel's traffic processing within their servers (Paper IV).

RQ2 *What insights about network latency for real application traffic can we obtain through continuous monitoring?*

For the continuous monitoring of application traffic to be worthwhile, we must be able to derive valuable insights from the monitoring data. We address this question by monitoring different network environments with our solutions from RQ1 and analyze the observed latency characteristics. Specifically, we analyze the network latency for all subscribers in a wireless ISP network, studying both the last-mile latency between the ISP and the subscribers, as well as the latency between the ISP and the external Internet endpoints that the subscribers connect to (Paper III). Furthermore, we analyze the latency within web servers, both in a testbed and for production servers at a global CDN provider, investigating how the traffic load impacts the server's latency (Paper IV).

RQ3 *How can we measure network latency with enough accuracy and granularity to gain a detailed understanding of the underlying network connection?*

Periodic active measurements, such as the commonly used ping probes at second-scale intervals, are useful for obtaining a high-level view of a network's latency. However, such infrequent measurements are not able to fully resolve the behavior of the underlying network. According to the Nyquist-Shannon theorem [193, 194], we need to sample a signal at twice the rate of its frequency to avoid aliasing. For wireless networks, where the radio conditions can change rapidly, and resource scheduling occurs at millisecond granularity, this implies that we need to measure the latency at sub-millisecond frequencies to fully capture the latency behavior. Furthermore, the latency measurements need to be highly accurate to be able to capture small variations in latency and provide useful results in networks where the overall latency is very low. For instance, 5G URLLC specifies a OWD of just 0.5 ms [110], necessitating sub-millisecond accuracy for meaningful measurements. To achieve such high measurement accuracy, we investigate the possibility of using hardware timestamps that modern NICs can provide. Due to a lack of publicly available tools that offer both high frequency-measurements and hardware-supplied timestamps, we design two new measurement tools that offer both sub-millisecond granularity and accuracy. Our `IPD-tool` captures the link-wide inter-packet delay on a per-packet basis, allowing for detailed analysis of the packet arrival patterns (Paper V). We also introduce `nanoprobe`, a ping-like traffic generator for measuring OWD, which can probe the network latency at very high frequencies and generate custom traffic bursts to test queueing behavior (Papers VI and VII).

RQ4 *What can detailed latency measurements reveal about the underlying network behavior of emerging wireless technologies?*

In the last few years, new wireless access technologies like WiFi 7, 5G, and Starlink have become widely available. Many factors may influence the latency in these networks, of which several may be hidden behind proprietary implementations. This can result in complex latency

variations that are not yet fully understood, making it challenging to simulate or predict the performance of these networks. For instance, a simulation-based study [195] suggests that the TCP congestion controls Cubic [196] and BBR (Bottleneck Bandwidth and Round-trip propagation time) [102] should offer similar throughput over Starlink. However, real-world measurements over Starlink instead show that Cubic obtains much lower throughput than BBR [197, 198].

To address this research question and gain a better understanding of the latency characteristics of these emerging wireless networks, we perform highly detailed measurements using the methodologies from RQ3. In this thesis, we focus on Starlink, which, as the first commercially available LEO network offers a completely new type of network. By probing Starlink's OWD at high frequencies, we investigate how the latency evolves at sub-millisecond scales and capture the impact of the radio link's resource scheduling (Paper VI). Furthermore, we test a large number of traffic burst patterns and analyze the resulting latency buildup to infer Starlink's queuing behavior (Paper VII).

## 4 Research Methods

Computer science is a very wide research field, which encompasses both more theoretical disciplines closely tied to mathematics, such as data structures and algorithms, and more applied disciplines tightly connected to engineering, such as software engineering and computer architecture [199]. This work takes place within the computer networking branch of computer science. We mainly use an applied approach similar to the natural sciences [200], where prior knowledge is used to form a hypothesis of how to solve a specific problem. The hypothesis is then tested by implementing a solution in the form of a software artifact, and experiments are then performed to test and evaluate the artifact.

### 4.1 The Empirical Research Method

At a high level, the work in this thesis can be mapped to the hypothetico-deductive empirical research method. Figure 3 illustrates the cycle of the empirical research method, and exemplifies how the work in Paper I can be mapped to it. In the initial *observation* phase, we identify a problem that is not sufficiently addressed by the current literature. In this case, we find that passive network latency monitoring solutions that depend on traditional packet capturing, such as `pping` [44], do not scale well to the high packet rates encountered in modern multi-gigabit networks. In the *hypothesis building* phase, we then formulate a hypothesis about how to solve the problem based on current knowledge of the topic. For Paper I, we hypothesize that by moving the passive monitoring logic into the kernel network stack with the help of eBPF, we can improve the performance by avoiding the overhead from packet cloning and frequent user-space and kernel space communication inherent to traditional packet capturing.

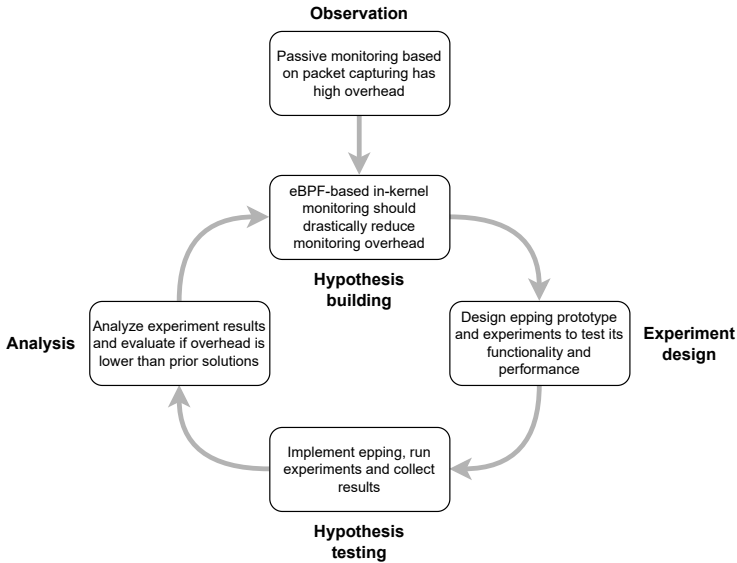


Figure 3: The work in Paper I mapped to the empirical iterative research method.

Next, we enter the *experiment design* phase, where we design experiments to assess if the hypothesized solution is able to solve the problem. This typically involves designing a proposed solution in the form of a software artifact based on a hypothesis, together with a set of experiments to evaluate the software artifact. If there is prior work addressing a similar problem the experiments are usually also designed to compare the proposed solution to prior state-of-the-art solutions. In the example with Paper I, we design our solution for eBPF-based in-kernel network monitoring and design experiments to both validate that our solution is able to accurately measure latency as well as to assess its runtime overhead and compare it to the prior pping tool. In the *hypothesis testing* phase, we implement the proposed software design or measurement methodology, and then run the experiments and collect the results. The results are then analyzed in the *analysis* phase to assess if they support the hypothesis or not. For Paper I the proposed solution is realized through the implementation of the epping tool. Our experimental results show that the eBPF based epping performs better than the prior packet capture based pping tool, which then supports our hypothesis and demonstrates that epping may be a viable solution to the problem.

As Figure 3 indicates, the empirical research process is an iterative process, where the cycle may have to be repeated several times before the work is ready for publication. If the analysis indicates that the hypothesis does not hold, the hypothesis may need to be modified and reevaluated, or the experiments or analysis may need to be refined to better test the hypothesis. While these

iterations are not clearly visible in the published papers, the presented work in this thesis has undergone several such iterations. For example, initial analysis showed that an early prototype of `epping` did not measure latency correctly, so the implementation had to be updated and reevaluated. Likewise, the aggregation functionality added to `epping` in Paper III required several iterations until a suitable solution was found. There is also a larger cycle, where the analysis leads to new observations, and thereby entirely new hypotheses to test. One such example is that we in Paper I found that pushing a large amount of individual measurement values caused significant overhead for `epping`. We therefore worked on aggregating the measurements in Paper III.

## 4.2 Types of Experimental Evaluation

There are several different approaches for testing a hypothesis in computer science, which can largely be broken down into: analytical modeling, which mathematically models the problem and solves it; simulation, which simulates a system using more complex models that are hard to analytically solve; and real-world experiments [199]. Additionally, there is also emulation, which can be seen as a hybrid between simulation and real-world experiments, where the experiment is conducted in a real system, but where parts of the environment are simulated.

As the work in this thesis largely focuses on evaluating the runtime performance of software tools and gaining insights about real-world network performance, all included papers mainly rely on real-world measurements. Accurately simulating software performance for the complex system architectures in use today is highly challenging, especially when there are interactions between multiple components (e.g. CPU, RAM, and NICs) and when applying relatively new technologies such as eBPF. As we evaluate different aspects of the performance for proposed software tools in Papers I, III, IV and V, we therefore directly measure the performance on real-world bare-metal servers, avoiding any confounding impacts from virtualization. Likewise, we find real-world experiments the most suitable choice for investigating under-explored aspects of real-world network performance. We therefore run experiments in production network environments when exploring the latency experienced by ordinary Internet traffic at a wireless ISP in Paper III, the latency within web servers at a CDN in Paper IV, and latency over the recently deployed Starlink network in Papers VI and VII. The real-world measurements are especially relevant for the latter, where the black-box nature of Starlink makes accurate emulation and simulation particularly challenging. We note, however, that our real-world measurements may help enable more accurate emulation or simulation of network latency in these network environments, making such approaches more feasible in future work.

In addition to real-world measurements, we also use emulation in Paper I to verify that our proposed `epping` tool is able to accurately report the latency under different network conditions. Here we use `netem` [201] to emulate different but known levels of latency and packet loss over a link, which is

impractical to achieve through physical network connections alone (e.g. to add a 10 ms one-way delay to a link would require a roughly 2000 km long fiber cable). Furthermore, we also use simulation in Paper VII, where real-world measurement results are compared to simulator output to find a theoretical model that matches the observed behavior.

#### 4.2.1 Test Environments

When running real-world experiments, the environment you run the experiment in has a large impact on the outcome and what conclusions you can draw from the results. For the work in this thesis, we have utilized both local testbeds and live deployments in production networks. In the testbeds, we have full control over both the hosts, the network connecting the hosts, and the traffic load running over the network. Meanwhile, in the production networks, we only have limited control over the hosts acting as our vantage points, and little to no control over the network and the traffic load.

We use local testbeds when assessing various aspects of the performance for our proposed tools `epping`, `netstacklat`, and `IDP-tool` in Papers I, IV, and V, respectively. The testbeds allow us to evaluate the tools under controlled conditions, where we can change a single variable at a time, and also allow us to collect any information we need from the involved hosts to get a detailed understanding of the performance. Furthermore, the controlled environment lets us repeat the tests multiple times, which also enables fair comparisons with alternative solutions. However, one limitation with the testbeds is that the environment may not reflect realistic scenarios in real-world deployments. For example, in Paper I we evaluate the run time performance of `epping` with elephant flows, which generate a high traffic load but do not accurately reflect the traffic mix you would generally expect to encounter on the Internet. While the testbed environments are hence well-suited for evaluating and comparing the performance of the tools, the latency measurements from the testbeds themselves do not provide us with very useful information about the latency on the wider Internet.

To gain useful insights about the latency behavior in real Internet traffic, we also deploy our `epping` tool in an ISP network to capture the network latency of customer traffic, and deploy `netstacklat` at a global CDN provider to monitor the host latency within web servers. In addition to allowing us to draw conclusions about the latency experienced by actual Internet traffic, these deployments also serve to demonstrate that the tools are feasible to deploy in real production environments. However, due to the lack of control over the network and traffic load in this type of environment, we cannot replicate these experiments. Additionally, this environment requires us to take additional care as the monitored network traffic may reveal sensitive information about the users generating the traffic or the operators processing it. For the ISP deployment, we aggregated the traffic for all subscribers so that it was not possible to attribute any data to a specific individual. For the deployment at the CDN provider, we never had access to the raw measurement data. Instead, we collaborated with one of their engineers to generate a handful of graphs

based on their monitoring dashboard.

The Starlink measurements in Paper VI and VII were performed in an environment that can be seen as a mix between a testbed and a deployment in a production environment. For these experiments, we have full control of the end hosts and the network traffic they generate, but we do not have any control over the Starlink network that we are studying. As the traffic runs over the real Starlink network, this setup still allows us to gain insights that are relevant for other Starlink users. Meanwhile, the full control over the end hosts allows us to generate the specific pattern of measurement traffic we need to gain a detailed understanding of how the Starlink network performs.

#### 4.2.2 Measurement Methodology

As covered in Section 2.2, network latency measurements can either be performed actively by sending test traffic over the network, or passively by observing existing network traffic. Throughout Papers I-IV we employ passive monitoring, as it allows us to get a better picture of the network latency that ordinary application traffic experiences on the Internet and avoids interfering with the monitored networks by adding additional traffic. However, for studying the Starlink network in Paper VI and VII we instead use active measurements. Here, active measurements are used because the focus is on studying the behavior of the underlying network rather than the application-perceived performance. We need control over the network traffic to ensure that the network packets are sent frequently enough to collect the fine-grained measurements needed to study the latency impact of Starlink’s resource scheduling, as well as to send traffic bursts of different sizes to infer the queueing behavior.

Finally, the measurements with the IPD-tool in Paper V can be considered active measurements. While the IPD-tool itself is passive, we use it in a setup where it monitors actively generated synthetic traffic from Ookla speedtests [202]. The passive nature of the IPD-tool hence only decouples it from the traffic generator used to perform active measurements. Once again, the reason for using active measurements is that we want to understand the performance of the underlying network link itself. For the measured IPD to reflect the performance of the network link, the link needs to be fully saturated with traffic, otherwise the IPD may reflect application behavior instead.

### 4.3 Limitations

Our choice of research methods also bring some inherent limitations. While the heavy emphasis on real-world measurements helps achieve accurate results for the tested scenarios, it may also limit the generalizability of the results. With emulation and simulation approaches, it is often possible to evaluate a wide range of scenarios. Meanwhile, the results we present in this thesis only tell us about the system and network performance in the environments the experiments were performed. The total runtime overhead we measure for different software tools is heavily dependent on the system they run on as well as the workload, although we expect the relative performance difference between

tools to at least partially generalize to other systems. Likewise, the absolute latency values we report are likely specific to the network environments we measure. However, several of the latency characteristics, such as long latency tails in both last-mile networks and end hosts, are likely to be found in other networks environments as well, which is also supported by similar findings in other studies.

While we cannot feasibly test every possible scenario, we test both different workloads and network environments to extend the generalizability. For instance, in testbed scenarios we vary the amount of concurrent flows for `epping`, try different server configurations when assessing `netstacklat`, and test different traffic burst patterns when investigating Starlink’s queuing configuration. In many cases, we also test in different network environments. For example, while we only perform controlled overhead tests for `epping` and `netstacklat` in our testbed environment, we also validate that they achieve low overhead in production environments under the mixed workload created by the real-world user traffic. For the Starlink measurements, the majority of the experiments are carried out from our vantage point here at Karlstad University. However, for the queuing evaluation we replicate the experiments from a secondary site at the University of Malaga, verifying that the results are not specific to the Starlink connection at Karlstad University.

Furthermore, some of our approaches and design decisions limit what our tools can measure and, consequently, what conclusions we can draw from the measurement results. For our passive monitoring solutions, we utilize aggregation to both reduce the runtime overhead and the volume of monitoring data. However, this aggregation also limits the resolution of the latency measurements. While the resolution is sufficient to capture the overall latency distribution and study the long latency tails, the lowered resolution prevents us from analyzing minor latency variations. The aggregation also inherently groups the data, preventing us from exactly pinpointing the origins of any individual latency sample. Future work could complement the aggregated data with a subsample of individual traces to offer more details while still maintaining a low footprint. Even very sparsely sampled latency traces (e.g. 1:128 000) can be highly useful to identify root cause issues [47].

Additionally, our passive monitoring tools also have limited coverage, only being able to monitor certain types of network traffic. Specifically, `epping` can only infer the RTT for ICMP echo and TCP traffic, while `netstacklat` only tracks the ingress latency for TCP and UDP packets. However, these coverage limitations are not inherent to the design of our tools, but rather come from the limited scope of their current implementations. The `epping` implementation could be extended to infer the latency for QUIC traffic through the spin bits [203] and certain application protocols running over UDP [204]. Likewise, `netstacklat` could be extended to monitor additional protocols as well as egress latency by identifying and hooking into additional functions in the kernel network stack. For active measurements, a notable limitation with the current implementation of `nanoprobe` is that it sends all probe packets over a single network flow. Being able to send traffic over multiple flows would be a

useful addition to test if and how networks employ per-flow queuing. When investigating Starlink's queuing, we instead complemented our nanoprobe measurements with less precise `iperf3` measurements to test for per-flow queuing. However, as we found that Starlink does not employ per-flow queuing, we had no immediate need for multi-flow support in nanoprobe.

Finally, the scope of our work also brings considerable limitations. In this thesis, we focus almost exclusively on network latency. However, from an application or end-user perspective there are additional factors that impact the total end-to-end delay. Network throughput, latency, and packet loss all interact and impact the total delay to transmit data over a network. Notably, packet loss has a large impact on the effective latency for reliable data transmissions, where lost packets need to be retransmitted. The interaction between different aspects of network performance are more holistically considered by recent Quality of Experience (QoE) [205] and Quality of Outcome (QoO) [206] metrics, where the focus is shifted from the raw performance of the network to how well the network is able to serve the applications running on top. Using the latency measurements from our work to calculate such QoE and QoO metrics can help make the results more insightful and actionable. However, for network researchers and engineers, we argue that it is still important to understand the fundamental capabilities of the network itself. For the end-user, the perceived latency is also impacted by application processing time and the latency for peripherals, such as touchscreens [207], which we do not consider in this work. For identifying the underlying cause for network latency, it is also useful to collect additional context along with the latency measurements. While we in several cases correlate our latency measurements to other collected metrics, we do not attempt to design any holistic monitoring framework that can automatically identify the root cause of the latency on its own.

## 5 Contributions

Figure 4 summarizes how the papers included in this thesis map to the research questions and contributions. Overall, we address the research questions from Section 3 by making the following contributions:

- C1 *We design an efficient in-kernel system for continuous passive monitoring of network latency.*

Network operators have limited means to monitor the latency for traffic passing through their network. Active measurements may not reflect the latency perceived by ordinary application traffic, while measurements from the applications or transport protocols themselves are only available on the end hosts. Passive latency monitoring could allow network middleboxes to observe the latency for any application traffic passing through them. However, currently available solutions for passively monitoring network latency either use traditional packet capturing and user space processing [43, 44, 127, 128], which do not scale well to multi-gigabit networks, or rely on hardware acceleration [43, 44, 127, 128], which

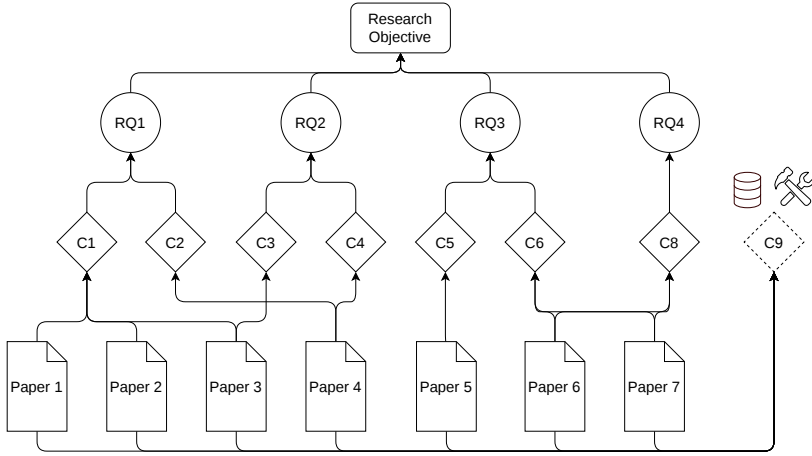


Figure 4: The mapping between the papers, contributions (C) and research questions (RQ) in this thesis

prevents them from being deployed in many network environments that lack the necessary hardware.

To address this gap and enable efficient latency monitoring that can run on general-purpose hardware, we design a passive network monitoring system that runs in-line with the kernel’s network stack. Our solution, Evolved Passive Ping (epping), is based on the prior pping [44]. However, we move the monitoring logic into the Linux kernel’s packet processing flow by utilizing eBPF, thereby avoiding the packet capturing overhead where packets are cloned and processed in user space. We modify the passive latency inference algorithm to enable it to run in the constrained eBPF environment, and make several optimizations, such as minimizing state lookup and supporting lock-less concurrent packet processing across multiple CPU cores, to allow it to scale to high packet rates. Furthermore, we find that with the monitoring moved into kernel space, pushing individual measurements to user space for reporting can incur a significant overhead. To further reduce the overhead, we therefore design a light-weight in-kernel aggregation mechanism. With this aggregation mechanism, the user space can fetch the entire latency distribution instead of processing each measurement value, reducing both the communication overhead and the amount of monitoring data. Through controlled testbed experiments, we demonstrate how our in-kernel epping design can successfully handle over 16 times higher packet rates than the packet-capture-based pping.

Our design of the epping tool, which helps address RQ1, is first introduced and evaluated in Paper I. Paper II provides a deeper look at the design challenges that we overcome, and Paper III further extends

epping by adding support for in-kernel aggregation of measurements.

C2 *We design an efficient solution for continuously monitoring the latency within the end host network stack.*

Network latency does not only occur in the network between the end hosts. Latency can also arise within the end host itself as the network packets are transferred from the NIC to the CPU, traverse the kernel network stack, and finally are read by the receiving application. Several works have studied various aspects of this host latency [35–37, 97–100, 208, 209]. However, despite this wealth of studies showing how host latency can cause significant delays, there are only a handful of tools that are able to passively monitor the host latency for ordinary application traffic. These prior solutions for monitoring host latency are either specialized for a very specific network environment and workload [47], or have too high overhead to support continuous monitoring in production [45, 46, 137, 138].

To address the limited visibility of host latency, we therefore design an efficient passive host latency monitoring solution. Like several prior solutions, we use eBPF to observe network packets as they reach several core functions in the Linux network stack, allowing us to follow packets as they traverse the network stack all the way until the receiving applications read the packet contents. However, prior eBPF solutions push individual timestamps as packets reach different points in the network stack to user space, and then calculate the host latency in user space by matching up all the timestamps for each packet. Building upon our insights from C1, we realize that pushing multiple messages per packet to user space may be costly, and that in-kernel aggregation presents an opportunity to drastically reduce the overhead. To make in-kernel aggregation possible, we use a novel design that allows us to efficiently calculate the host latency (and subsequently aggregate it) within the eBPF programs, which takes advantage of Linux's capability to timestamp packets early in the network stack. Furthermore, we design a filter to detect when applications may be head-of-line blocked, where the kernel TCP stack has to wait on out-of-order segments to arrive before it can deliver data to the application. We thereby avoid conflating head-of-line blocking with host latency.

We demonstrate that our design for `netstacklat` achieves a low monitoring overhead through several testbed experiments. Under a heavy workload scenario, where the prior solutions reduce end-to-end throughput by at least 17% and increase tail latency by over 100%, `netstacklat` only reduces throughput by 2% and does not inflate the tail latency by more than 6%. In a deployment on production servers at a global CDN provider, we find that `netstacklat` adds an overhead of just 0.04% CPU utilization during peak hours for the servers at the busiest location.

The `netstacklat` design is presented and evaluated in Paper IV, which addresses RQ1.

C3 *We measure and analyze the latency of ordinary Internet traffic in a wireless ISP network.*

While there are many studies [17, 32, 56, 86, 90, 183, 210–217] that actively measure network latency across different parts of the Internet, there are only a handful of studies [93, 95, 218, 219] that employ passive latency measurements from within the network to capture the latency for real application traffic. To get a better understanding of how latency for ordinary Internet traffic varies, we collaborate with a wireless ISP operator to deploy our passive monitoring solution from C1 inside their core network. Over the course of a one-month period, we monitor the network traffic from all of the ISP’s subscribers, collecting an extensive dataset containing over 9 billion latency measurements.

We perform an extensive analysis of the latency measurements, investigating how the latency distributions vary across different parts of the network. Similar to several prior studies [90, 93–95], we find a wide latency tail in the last-mile connection between the ISP and the end user devices, with 0.16% of the measurements exceeding one second. Like other studies [86, 95, 214], we also find a diurnal variation in the last-mile latency where the latency is correlated to the traffic load, increasing during the busier periods in the evening. However, given that this ISP already employs a state-of-the-art AQM solution, the high last-mile latency is unlikely to stem from bufferbloat in the ISP network, instead suggesting that the residential WiFi routers are the primary source of the large latency spikes. As a previous study [220] has demonstrated in a testbed, AQMs are insufficient for preventing high latencies in the presence of WiFi. Our measurements thereby indicate that residential WiFi remains a noticeable latency issue in the wild, preventing end users from obtaining consistent low latency even when their ISP takes measures to prevent bufferbloat.

In contrast to the last-mile segment, our analysis shows that the external leg between the ISP and the remote Internet endpoints that subscribers connect to, generally has a higher baseline latency, but a narrower latency tail. The external latency is generally stable over time. There is a weak decreasing trend during the busy afternoon periods, which is likely an effect of traffic shifting to a different set of services that to a greater extent is served from nearby data centers. There are large differences in the latency distributions depending on the destination of the traffic, with the overall external latency distribution having a multi-modal distribution focused around peaks that match the latency to nearby data centers. 42% of the latency samples fall within the initial peak, being less than 24 ms, having a similar magnitude as the typical last-mile latency. Meanwhile, for address blocks associated with mainly Asian and African regions, we not only see a much higher median latency, often exceeding 200 ms, but also much greater latency variations, with the standard deviation in several cases reaching 100 ms. We also examine the most popular autonomous systems (ASes). Here, we find that a majority of the traffic is served

from just a handful of ASes, several of which have a very consistent latency around 20 ms, although some ASes also show a multi-modal distribution where a fraction of the traffic is served from more distant data centers. Coinciding with a denial-of-service attack against some Amazon networks, we also detect a drastic increase in latency to those networks, exceeding 250 ms for over an hour.

This wireless ISP measurement study maps to RQ2, and is presented in further detail in Paper III.

C4 *We investigate the latency in the local kernel network stack and application layer for HTTP web servers.*

There are many studies [35, 36, 98, 100, 209, 221, 222] that examine various aspects of network latency within the end host in testbeds. However, these studies use methodologies that cannot easily be applied in production, such as modifying the kernel [221, 222] or relying on active end-to-end measurements [35, 36, 100, 209], making it challenging to explore how host latency impacts application traffic in production networks. While Fathom [47] has been used to break down the latency for several applications at Google, showing how the kernel network stack and application processing can make up the majority of the end-to-end latency in some cases, it is limited to remote procedure call (RPC) applications in a data center setting where both the client and server can be instrumented. To further explore the extent of host latency in production servers, we therefore utilize our passive host monitoring from C2 to investigate the latency in HTTP servers.

We first demonstrate our measurement methodology by analyzing the host latency for the commonly used nginx [60] and Apache [61] servers for various traffic loads in a testbed. In total, we examine 144 different combinations of server configurations, HTTP file sizes, and number of concurrent connections. Our experiments show that already in the very early parts of the kernel network stack, there can be large differences in how long it takes to process the packet. At all four layers in the kernel network stack that we study, we find that the maximum observed host latency to be at least three orders of magnitude greater than the minimum latency, indicating that very large inflations in host latency are possible. The relative discrepancy between the minimum and maximum latency and the probability of large latency inflations further increase at the higher layers of the network stack. We also note that latency at all four layers grows as the number of concurrent connections increases. Furthermore, we show that in our testbed setup, where the traffic generator is directly connected to the server, the host latency at the server is strongly correlated to the end-to-end request latency, indicating that host latency could be a useful load-balancing signal.

To analyze the host latency in a production network, we then deploy our host monitoring across all servers at Cloudflare, a global CDN provider. Here we find that the latency between the start of the network

stack until their nginx application reads the HTTP requests is typically between 64 to 256  $\mu$ s. In a semi-controlled experiment, we configure the load balancer to direct an increasing amount of traffic to a server until the traffic load exceeds the expected capacity of the server. While the growing traffic load does lead to a measurable increase in the host latency, especially for the tail latency, the increase is modest, with the 95th percentile remaining below 1 ms even when the traffic load exceeds the expected server capacity. The inflation in host latency due to the high traffic load is therefore unlikely to have a large impact on end-to-end latency over the Internet. However, during normal operation, we detect an anomaly with a much greater impact on the host latency, where the median latency on one server steadily grows from around 128 to 256  $\mu$ s to over 2 ms over the course of 3 hours, despite no considerable increase in traffic load. We identify the issue causing this latency surge as Linux’s Generic Receive Offload (GSO) mechanism failing to merge packets from a flow, causing a large increase in kernel-level packet processing on a single CPU core.

The measurement and analysis of web server latency is presented in Paper IV, and helps address RQ2.

C5 *We design a method to measure the link-wide inter-packet delay and assess the accuracy of software and hardware timestamps.*

While wireless links are a convenient method to connect a wide range of devices, the radio conditions for the wireless connection may change over time, causing the network performance of the link to vary [75]. Assessing the capacity of such links with coarse-grained throughput measurements, e.g., at 1-second intervals, risks hiding the dynamics of the network link behind averages. Highly granular measurements can instead use full packet traces to obtain per-packet data. However, such packet traces can have large storage footprints. Furthermore, packet traces may contain sensitive information, such as IP addresses, which are challenging to anonymize [223, 224]. It is therefore often impractical to collect and share full packet traces. To minimize the data collection while still retaining measurement granularity, we therefore propose collecting traces of only packet sizes and the IPD, as the ratio between them offers the highest possible granularity of throughput from a packet-level perspective. Such IPD-traces can be converted to throughput traces of arbitrary resolution, as often used for trace-driven emulation [225, 226], while also providing a means to assess the link’s burstiness and IPD-jitter. We provide a use-case example by collecting IPD-traces for a high-capacity 5G connection and identifying several multi-millisecond-long gaps with no data reception. As the methodology we present is network agnostic, we later use a similar IPD measurements setup to infer Starlink’s link-layer frame configuration [177, 227, 228]. The Starlink IPD measurements have also been used for throughput prediction [229].

To facilitate the collection of link-wide IPD-traces, we design the eBPF-

powered IPD-tool, which can capture the IPD and packet size using only 4 bytes per packet. In the process, we identify two major challenges for accurate link-wide IPD collection: timestamp accuracy and concurrent packet processing. Due to the short timescales of IPDs, they require highly accurate timestamps for when the packets arrive. To achieve the necessary accuracy, we rely on the hardware timestamps that modern NICs can provide. We conduct testbed experiments using both software and hardware timestamps to evaluate their suitability for IPD measurements. Our results show that with hardware timestamps, the IPDs closely match the theoretically expected values, even at sub-microsecond scales. In contrast, software timestamps frequently lead to impossibly low or much too high IPDs, only having sufficient accuracy to measure IPDs that exceed 100  $\mu$ s. Link-wide IPDs also require determining the order in which all packets arrive on the link. As modern systems use mechanisms like Receive Side Scaling (RSS) [230] to scale the packet processing across multiple CPU cores, packets may be processed in parallel or in a different order than they arrived in. The hardware timestamps from the NIC allow us to compare the order in which the packets are processed to the original arrival order. In our experiments, we observe that the processing order may diverge from the original packet order by up to 50 packets. This reordering forces us to buffer timestamps from multiple packets so that the correct arrival order can be established before computing the IPD. It is therefore not feasible to directly compute the link-wide IPD in the data plane, as has previously been done for per-flow IPDs [231, 232]. The design of the IPD-tool and the evaluation related aspects, such as timestamp accuracy and packet reordering, maps to RQ3. The work is presented in more detail in Paper V.

C6 *We design and implement a method to frequently and accurately measure one-way delays.*

To capture how events like resource scheduling, which often occur at millisecond timescales, affect the latency of wireless links, we need to measure latency at sub-millisecond intervals. Assessing how the latency evolves over such short time intervals in turn requires sub-millisecond measurement accuracy. While there are many ping-like tools [52, 54, 57, 116–118, 123, 233] for actively measuring network latency, most of them rely on software timestamps, which, as we show with C5, have poor accuracy at sub-millisecond timescales. Additionally, software timestamps inflate the latency measurements of the network link by including part of the time it takes for the end host to process the packets. The end host can add substantial delays, as we learned from C4. Publicly available latency measurement tools that support hardware timestamps either cannot support sufficiently high measurement frequencies [122, 124], or only capture the RTT instead of the more fine-grained OWD [123].

To support both the highly frequent and accurate OWD measurements needed for fully resolving the latency characteristics of a wireless link,

we develop a traffic generator with support for hardware timestamps called nanoprobe. While nanoprobe is based on nanoping [124], we have extensively modified it to support our needs and maximize the measurement accuracy. To achieve low and accurate sending intervals, we apply busy waiting between transmissions instead of relying on timers. Likewise, we busy-poll hardware timestamps to reduce the risk that they are overwritten before the kernel can notify the program. Furthermore, we pin threads to individual CPU cores to avoid interference from CPU migration, and delegate OWD and RTT calculations to post-processing to reduce delays from unnecessary online computation and communication. We also add new features, such as a duplex mode where OWD probes are concurrently sent in uplink and downlink directions independently of each other instead of in a synchronized request-response manner. Additionally, we make it possible to schedule the packet size and transmission interval on a per-packet basis. This enables the construction of any arbitrary traffic load, which is useful to test how the network responds to different traffic patterns. Finally, by combining nanoprobe with a looped network setup where the same NIC acts as both sender and receiver, we avoid clock synchronization and drift issues when measuring the OWD.

The design and implementation of nanoprobe addresses RQ3. We introduce nanoprobe in Paper VI, and extend it with per-packet scheduling in Paper VII.

*C7 We perform highly granular measurements and a detailed analysis of the latency in Starlink.*

As the first and largest Low Earth Orbit (LEO) satellite network, Starlink provides a new type of access network that already connects millions of devices to the Internet. However, with network traffic being routed through space using proprietary technologies, there are many hidden factors that can impact the latency. While several previous studies [78, 181–184, 234] have statistically quantified the latency over the Starlink network at a macroscopic level, the observed latency variations are not yet well understood. For instance, prior measurements [178, 235] at 10 ms or 20 millisecond intervals have shown latency bands where the latency varies between a few distinct values. While this behavior has been assumed to be an artifact of Starlink’s resource scheduling, no exact explanation has been provided. There is also a lack of studies investigating the queueing behavior in Starlink, despite queueing often being a considerable source of variable latency.

To better understand Starlink network latency, we use our highly accurate measurement tool from C6 to perform extensive OWD measurements over a Starlink connection. We use both highly frequent but non-saturating probing to measure the latency without congestion, as well as traffic bursts of various sizes and rates to measure the latency inflation as queues build up. Our analysis reveals a distinct reverse sawtooth

latency pattern as multiple packets across several milliseconds accumulate before all being delivered at once in a single link-layer frame, and demonstrate that the banding observed in prior studies is an aliasing effect from the low measurement frequency. We also observe drastic latency spikes upwards of 300 ms during Starlink's periodic reconfiguration events, diurnal latency variations, and that the uplink has larger latency variations than the downlink. Furthermore, we compare the latency and loss patterns to a queuing simulator, showing how Starlink appears to be using a front-drop queue with a queue size of around 1500 packets. Notably, this queuing configuration may explain the poor performance observed for the commonly employed Cubic congestion control.

This study of network latency over Starlink maps to RQ4. While our analysis focuses on Starlink, the general methodology with high-frequency accurate latency probing can be used to uncover the latency characteristics in any IP-network. The investigation of Starlink's OWD is presented in Paper VI while the queuing characteristics are covered in Paper VI.

C8 *We make our research artifacts publicly available.*

Making use of prior knowledge, i.e. “standing on the shoulder’s of giants” [236], is a core pillar enabling scientific progress. To enable others to replicate, learn from, and build upon our work, we make our tools and datasets publicly available in addition to describing the work in scientific publications. We have open sourced `epping`<sup>4</sup>, `netstacklat`<sup>5</sup>, the `IPD-tool`<sup>6</sup>, and `nanoprobe`<sup>7</sup>. Notably, the source code for `epping` has been used to integrate passive RTT monitoring in LibreQoS [237], a QoS platform used by over 1000 ISPs worldwide [238], and our work thereby contributes towards monitoring the latency for millions of ISP subscribers. Furthermore, in addition to datasets from our own testbeds, primarily useful for replication purposes, we also share novel datasets with latency measurements from real-world production networks. In particular, we share a dataset<sup>8</sup> with aggregated latency measurements derived from 356 billion packets of customer traffic at a wireless ISP, as well as a two datasets<sup>9</sup> with detailed per-packet OWD measurements from over 500 million probe packets sent over Starlink. These datasets enable other researchers to further study the latency in ISP and Starlink networks, or can be used as input for realistic trace-driven network emulation.

While sharing tools and datasets does not directly map to any of the research questions, it is an important aspect for all of them. We share relevant artifacts where possible for all included papers in this thesis.

<sup>4</sup><https://github.com/xdp-project/bpf-examples/tree/main/pping>

<sup>5</sup><https://github.com/xdp-project/bpf-examples/tree/main/netstacklat>

<sup>6</sup><https://git.cs.kau.se/simosund/ebpf-ipd-tool/>

<sup>7</sup><https://github.com/simosund/nanoprobe>

<sup>8</sup><https://doi.org/10.5281/zenodo.13388093>

<sup>9</sup><https://doi.org/10.5281/zenodo.16275284> and <https://doi.org/10.5281/zenodo.18868451>

In summary, our contributions include the design of several novel solutions to measure network latency, giving network operators and service providers efficient ways to continuously monitor their networks (C1 and C2), and enabling researchers to perform the highly granular and accurate measurements needed to capture the latency in new network technologies (C5 and C6). Leveraging these solutions, we measure and analyze the end-to-end latency in a Wireless ISP (C3), the host latency within web servers (C4), and the delays caused by resource scheduling and queueing in Starlink (C7), contributing to an increased understanding of latency across various parts of the modern Internet.

## 6 Summary of Appended Papers

### Paper I - Efficient Continuous Latency Monitoring with eBPF

In this paper, we propose using eBPF to implement passive network latency monitoring inside the kernel to avoid the overhead of traditional packet capturing, where packets are cloned and processed in user space, as used by previous passive monitoring solutions. We hypothesize this will allow passive monitoring to scale to higher packet rates while still being deployable on general-purpose hardware. To test our hypothesis, we implement an evolved passive ping, `epping`. We evaluate `epping` on a testbed, and show that the RTT measurements provided by the tool are consistent with those produced by previous solutions. Furthermore, we compare the runtime performance of our solution to the previous `pping` tool, which uses a similar algorithm but relies on packet capturing. Our results show that the original `pping` is only able to monitor a fraction of all packets at multi-gigabit rates, while our proposed `epping` processes every packet while maintaining a lower CPU overhead than `pping`, allowing it to effectively monitor packets at over an order of magnitude higher rates with the same CPU resources. Additionally, we discuss the implications of our solution relying on TCP timestamps and show scenarios that can lead to the RTT being slightly overestimated.

### Paper II - Evolved Passive Ping

This technical report complements Paper I and III by providing additional details on the design and implementation of `epping` and its various features. We cover many of the optimizations we have done to reduce the overhead of `epping`, but also outline some possible further improvements. Additionally, we go over many of the challenges encountered during the development of `epping` and how we solved them, focusing primarily on handling concurrency from processing multiple packets simultaneously and implementing all of the logic with the quickly evolving eBPF technology.

### **Paper III – Measuring Network Latency from a Wireless ISP**

In Paper I, we noted that our passive latency monitoring tool `epping` can produce a very large amount of measurements. To make it practical to continuously monitor network latency, we therefore extend `epping` in this paper with the capability to aggregate RTT measurements inside the kernel. To demonstrate the feasibility and utility of continuously monitoring network latency, we deploy our extended `epping` inside the network of a wireless ISP and passively monitor the network latency for all of its customer traffic over a one-month period, collecting over 9 billion RTT measurements. We perform an extensive analysis of the latency data, inspecting the latency distribution both within and across subnets and autonomous systems of various sizes, and show how the latency varies over time. Among others, we find that the internal network latency is lower but has a wider tail than the external latency, and that the internal latency increases during peak hours, whereas the external latency shows no such trend. Furthermore, we visualize network latency across the entire IPv4 address space, showing that address blocks managed by the African and Asian regional Internet registries have both high median latency and large latency variations, while the most popular autonomous systems have relatively low and narrow RTT distributions.

### **Paper IV – Waiting at the Front Door**

In this paper, we turn our focus towards the network latency that can accrue within the end host. We propose a new tool, `netstacklat`, that can continuously monitor the within-host latency as network packets traverse the kernel network stack until the payload is read by the receiving application. By making use of Linux's packet timestamping capabilities, eBPF, and in-kernel aggregation, `netstacklat` achieves a low overhead. We assess `netstacklat` with a wide variety of HTTP workloads on a testbed, demonstrating how its low overhead only inflates the tail latency by 6% in a heavily loaded scenario where prior solutions more than double the tail latency. We also examine the `netstacklat` latency measurements, finding that even in the early parts of the kernel network stack, the latency can be inflated by over three orders of magnitude, and that the measured host latency strongly correlates to the end-to-end application latency. Finally, we perform a fleet-wide deployment of `netstacklat` at a global CDN provider, among others, showing how `netstacklat` identified a temporary anomaly where the kernel network stack failed to merge packets from a flow, leading to a drastic increase in network stack latency.

### **Paper V – Characterizing Wireless Link Throughput with eBPF and Hardware Timestamps**

In this work, we leverage eBPF and NIC-supplied hardware timestamps to capture the inter-packet delay (IPD) for all packets traversing a link. This IPD data provides a highly granular view of throughput variations and IPD jitter, which is useful when assessing the performance of wireless links where the available capacity frequently changes. By utilizing a custom data format and only

collecting the IPD, our method only requires 4 bytes per packet measurement, at least a 4X reduction compared to traditional packet captures, while also avoiding sensitive information like IP addresses. Furthermore, we show that while hardware timestamps can produce accurate IPDs at a nanosecond scale, software timestamps are inadequate for capturing IPDs at sub-millisecond scales. Additionally, we utilize the IPD collection to analyze the performance of a speedtest running over a 5G link at nearly 1 Gbps, and observe frequent multi-millisecond gaps where no packets are received on the link. The measurement methodology presented in this paper also serves as the foundation for several studies [179, 227, 228] examining Starlink’s performance (not included in the thesis).

### **Paper VI – A Detailed Characterization of Starlink One-way Delay**

In this paper, we investigate the one-way delay (OWD) latency over a Starlink connection for both downlink and uplink traffic across multiple timescales. To achieve highly accurate measurements, we develop an isochronous packet generator, nanoprobe, that uses NIC-supplied hardware timestamps. Furthermore, by sending very frequent probe packets, we get highly granular data with multiple probes within each 1.33 ms Starlink link-layer frame period. We find that Starlink’s link layer scheduling results in a reverse sawtooth pattern as packets are delivered in batches, and show how a coarser sampling interval of 10 ms, used by some previous studies, causes an aliasing artifact. We also find that when Starlink reconfigures the network every 15 seconds, the latency may increase by over 100 ms, and that the latency variation is greater for uplink traffic than for downlink traffic.

### **Paper VII – Characterizing the Configuration of Starlink Queuing**

One of the major components of network latency is queuing, so to better understand the performance of the Starlink network, we investigate its queuing behavior in this paper. To do this, we extend the nanoprobe tool from Paper VI with the capability to send probe packets according to a configurable schedule and use it to send traffic bursts of varying rates and durations over Starlink. By comparing the resulting one-way delay and loss patterns to a queuing simulator, we find that Starlink employs a drop-front queue rather than a tail-drop queue. While this drop-front behavior results in lower queuing delays than traditional tail drops, it may also explain the poor performance for loss-based congestion controls, like Cubic, that have been observed for Starlink. Additionally, Starlink does not appear to use per-flow queuing, and the queue size seems to be limited in the number of packets rather than the number of bytes.

## **7 Conclusion**

In this thesis, we have explored using modern technologies like eBPF and hardware timestamps to improve the visibility of network latency. We have

shown how eBPF can be used to create efficient in-kernel passive monitoring solutions that network operators and service providers can deploy in existing general-purpose infrastructure to continuously monitor the latency in both their networks and end hosts. Additionally, we demonstrate how combining accurate hardware timestamps with high-frequency measurements allows network researchers to examine the latency characteristics of emerging network technologies in detail. We have leveraged our new measurement solutions to analyze the latency in various network environments, including both the last-mile access and external segments in a wireless ISP network, the local end host network stack in servers, and the recently deployed Starlink satellite network.

Across the different networks and parts of the networks we have examined, we consistently find long latency tails. While latency distributions are generally known to be long-tailed, it is interesting to note that we find this to not only be the case for the end-to-end latency as a whole, but to also hold true for each separate aspect of the networks we have analyzed, including the last-mile access, upstream connections to individual ASNs, each layer in the end host network stack, and the reconfiguration delays in Starlink. This highlights that achieving consistent low latency will require holistic changes across all parts of the network. When such network changes aimed at reducing network latency are deployed, the capability to observe latency will be crucial to assess if they have the desired impact.

Beyond simply observing, there is also an opportunity to leverage latency measurements themselves as a means to lower network latency. We often find that network latency is correlated with traffic load in various ways, and future work could use the live in-line latency measurements from our passive monitoring solutions in various network devices to reduce congestion induced latency. For example, traffic shapers in ISP networks could detect latency increases to individual subscribers and dynamically reduce the data rate to avoid congestion in the residential WiFi routers, similarly to how CAKE [39] and PURPLE [239] can shape traffic below the expected network capacity to avoid bufferbloat in external devices. Likewise, routers could use passive latency measurements to avoid congested or high-latency routes [240], and network management and orchestration systems can detect latency anomalies to locate, predict, and prevent network issues [241, 242]. Meanwhile, our monitoring of end host latency could be used as a signal for load-balancing server workloads [243]. A detailed understanding of the latency characteristics in specific network environments, such as the structural latency variations we have found for Starlink, can also be used to design specialized networking solutions for those environments. For example, such insight can be leveraged to develop new congestion control [197, 244, 245] and multipath-scheduler [246, 247] algorithms that are able to better adapt to Starlink's latency variations.

There still remains much work to fully understand the latency properties of all the heterogeneous networks that make up the modern Internet. In this work, we have provided a snapshot view of the latency in a few select network environments, but it is not clear how well our findings generalize to other networks and how it might change over time. In particular, it would be useful

with more extensive and long-term measurement studies, such as the one by Trevisan et al. [219], to further examine how the network latency evolves over time. The low-overhead passive monitoring solutions we present in this thesis make such studies more feasible, as they can run on existing network devices without requiring any dedicated monitoring infrastructure. For Starlink, our fine-grained analysis has revealed several details about how the internal mechanisms impact the latency. However, there are still aspects of Starlink that require further study, such as how its dynamic load-based resource allocation may impact the latency for traffic of different intensities. Furthermore, there are many upcoming wireless networks that also need to be studied, such as other non-terrestrial networks [248, 249], private network slices [250, 251], and beyond-5G cellular networks [252]. Here, the methodologies we have developed to analyze Starlink can be reapplied to learn the latency characteristics of other black-box networks where the internal workings are unknown. Our work thereby not only serves to improve the understanding of the latency in current networks, but also provides the means to understand the latency for the evolving networks of the future.

## References

- [1] We Are Social and Meltwater, “Digital 2026: Global overview report,” Tech. Rep., 2025. [Online]. Available: <https://datareportal.com/reports/digital-2026-global-overview-report>
- [2] Ookla LLC, “Speedtest global index – Internet speed around the world,” 2025, Accessed: 2026-01-19. [Online]. Available: <https://www.speedtest.net/global-index>
- [3] Cloudflare Inc., “Internet quality worldwide — Cloudflare radar,” Dec. 2025, Accessed: 2026-01-19. [Online]. Available: <https://radar.cloudflare.com/quality?dateRange=52w#connection-quality>
- [4] D. Howdle, “Worldwide broadband speed league 2024,” 2024, Accessed: 2026-01-19. [Online]. Available: <https://bestbroadbanddeals.co.uk/broadband/speed/worldwide-speed-league/>
- [5] M. Belshe, “More bandwidth doesn’t matter (much),” Apr. 2010. [Online]. Available: [https://www.chromium.org/spdy/More\\_Bandwidth\\_Doesn\\_t\\_Matter\\_2\\_\(2\).pdf](https://www.chromium.org/spdy/More_Bandwidth_Doesn_t_Matter_2_(2).pdf)
- [6] S. Sundaresan, N. Feamster, R. Teixeira, and N. Magharei, “Measuring and mitigating web performance bottlenecks in broadband access networks,” in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC ’13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 213–226. [Online]. Available: <https://doi.org/10.1145/2504730.2504741>

- [7] S. A. M. Mostafa, M. Wittie, and U. Goel, "Does more bandwidth really not matter (much)?" in *Proceedings of the 13th EAI International Conference on Mobile Multimedia Communications*. EAI, Nov. 2020.
- [8] D. D. Clark and S. Wedeman, "Understanding the metrics of Internet broadband access: How much is enough?" Rochester, NY, Aug. 2022. [Online]. Available: <https://papers.ssrn.com/abstract=4178804>
- [9] K. Schoenenberg, A. Raake, S. Egger, and R. Schatz, "On interaction behaviour in telephone conversations under transmission delay," *Speech Communication*, vol. 63–64, pp. 1–14, 2014. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167639314000302>
- [10] S. Garg, A. Srivastava, M. Glencross, and O. Sharma, "A study of the effects of network latency on visual task performance in video conferencing," in *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. Chi Ea '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: <https://doi.org/10.1145/3491101.3519678>
- [11] G. Bingol, L. Serreli, S. Porcu, A. Floris, and L. Atzori, "The impact of network impairments on the QoE of WebRTC applications: A subjective study," in *2022 14th International Conference on Quality of Multimedia Experience (QoMEX)*, Sep. 2022.
- [12] J. Chen and Z. Tao, "Quantitative analysis and mitigation of the impact of network latency on video conferencing communication efficiency," *Human Factors*, 2025. [Online]. Available: <https://doi.org/10.1177/00187208251398477>
- [13] D. W. Edwards, "Impacts of telecommunications latency on the timing of speaker transitions," *Speech Communication*, vol. 171, p. 103226, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S016763932500041X>
- [14] F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, p. 25, Dec. 2019. [Online]. Available: <https://doi.org/10.1145/3366704>
- [15] H. Wang, X. Zhang, H. Chen, Y. Xu, and Z. Ma, "Inferring end-to-end latency in live videos," *IEEE Transactions on Broadcasting*, vol. 68, no. 2, pp. 517–529, Jun. 2022.
- [16] M. Sun, H. Chen, and Z. Ma, "Quality-of-experience assessment for ultra-low latency live streaming videos," in *2023 IEEE 25th International Workshop on Multimedia Signal Processing (MMSP)*, 2023, pp. 1–6.

- [17] S. C. Madanapalli, H. H. Gharakheili, and V. Sivaraman, “Know thy lag: In-network game detection and latency measurement,” in *Passive and Active Measurement*, O. Hohlfeld, G. Moura, and C. Pelsser, Eds. Cham: Springer International Publishing, 2022, pp. 395–410.
- [18] D. Halbhuber, M. Seewald, F. Schiller, M. Götz, J. Fehle, and N. Henze, “Using artificial neural networks to compensate negative effects of latency in commercial real-time strategy games,” in *Proceedings of Mensch Und Computer 2022*, ser. MuC '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 182–191. [Online]. Available: <https://doi.org/10.1145/3543758.3543767>
- [19] D. Halbhuber, V. Schwind, and N. Henze, “Don’t break my flow: Effects of switching latency in shooting video games,” *Proc. ACM Hum.-Comput. Interact.*, vol. 6, Oct. 2022. [Online]. Available: <https://doi.org/10.1145/3549492>
- [20] S. Liu and M. Claypool, “The impact of latency on navigation in a first-person perspective game,” in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. Chi '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: <https://doi.org/10.1145/3491102.3517660>
- [21] X. Xu, S. Liu, and M. Claypool, “The effects of network latency on Counter-Strike: Global Offensive players,” in *2022 14th International Conference on Quality of Multimedia Experience (QoMEX)*, Sep. 2022.
- [22] S. Van Damme, J. Sameri, S. Schwarzmann, Q. Wei, R. Trivisonno, F. De Turck, and M. Torres Vega, “Impact of latency on QoE, performance, and collaboration in interactive multi-user virtual reality,” *Applied Sciences*, vol. 14, no. 2290, 2024. [Online]. Available: <https://www.mdpi.com/2076-3417/14/6/2290>
- [23] K. Z. Win, Y. Ishibashi, and K. H. Win, “QoE assessment of cooperative work in networked virtual reality environment with haptic sense: Influence of network latency,” in *Advances in Information Communication Technology and Computing*, V. Goar, M. Kuri, R. Kumar, and T. Senjyu, Eds. Singapore: Springer Nature Singapore, 2024, pp. 69–83.
- [24] M. J. Lum, J. Rosen, H. King, D. C. Friedman, T. S. Lendvay, A. S. Wright, M. N. Sinanan, and B. Hannaford, “Teleoperation in surgical robotics – network latency effects on surgical performance,” in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Sep. 2009, pp. 6860–6863.
- [25] B. Berberian, P. Le Blaye, C. Schulte, N. Kinani, and P. R. Sim, “Data transmission latency and sense of control,” in *Engineering Psychology and Cognitive Ergonomics. Understanding Human Cognition*, D. Harris, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 3–12.

- [26] A. Khasawneh, H. Rogers, J. Bertrand, K. C. Madathil, and A. Gramopadhye, "Human adaptation to latency in teleoperated multi-robot human-agent search and rescue teams," *Automation in Construction*, vol. 99, pp. 265–277, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0926580517310890>
- [27] A. Noguera Cundar, R. Fotouhi, Z. Ochitwa, and H. Obaid, "Quantifying the effects of network latency for a teleoperated robot," *Sensors*, vol. 23, no. 8438, 2023. [Online]. Available: <https://www.mdpi.com/1424-8220/23/20/8438>
- [28] F. Hernandez-Gobertti, R. Lozano, K. Kousias, Ö. Alay, C. Griwodz, and D. Gomez-Barquero, "Exploring performance and user experience in haptic teleoperation systems: A study on QoS/QoE dynamics on immersive communications," in *2025 IEEE 26th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, May 2025, pp. 188–194.
- [29] Z. Y. Motiwala, A. Desai, R. Bisht, S. Lathkar, S. Misra, and D. D. Carbin, "Telesurgery: Current status and strategies for latency reduction," *Journal of Robotic Surgery*, vol. 19, no. 1, p. 153, Apr. 2025. [Online]. Available: <https://doi.org/10.1007/s11701-025-02333-1>
- [30] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.
- [31] B. Charyyev, E. Arslan, and M. H. Gunes, "Latency comparison of cloud datacenters and edge servers," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Dec. 2020, pp. 1–6.
- [32] L. Corneo, N. Mohan, A. Zavodovski, W. Wong, C. Rohner, P. Gunningberg, and J. Kangasharju, "(How much) can edge computing change network latency?" in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–9.
- [33] M. Iorio, F. Risso, and C. Casetti, "When latency matters: Measurements and lessons learned," *SIGCOMM Comput. Commun. Rev.*, vol. 51, no. 4, pp. 2–13, Dec. 2021. [Online]. Available: <https://doi.org/10.1145/3503954.3503956>
- [34] S. Gallenmüller, F. Wiedner, J. Naab, and G. Carle, "Ducked tails: Trimming the tail latency of(f) packet processing systems," in *2021 17th International Conference on Network and Service Management (CNSM)*, Oct. 2021, pp. 537–543. [Online]. Available: <https://ieeexplore.ieee.org/document/9615532>
- [35] Q. Cai, M. Vuppapapati, J. Hwang, C. Kozyrakis, and R. Agarwal, "Towards  $\mu$ s tail latency and terabit Ethernet: Disaggregating the host network stack," in *Proceedings of the ACM SIGCOMM 2022 Conference*, ser. Sigcomm '22. New York, NY, USA: Association

- for Computing Machinery, 2022, pp. 767–779. [Online]. Available: <https://doi.org/10.1145/3544216.3544230>
- [36] P. Cai and M. Karsten, “Kernel vs. user-level networking: Don’t throw out the stack with the interrupts,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 7, no. 3, Dec. 2023. [Online]. Available: <https://doi.org/10.1145/3626780>
- [37] M. Jasny, M. El-Hindi, T. Ziegler, and C. Binnig, “A wake-up call for kernel-bypass on modern hardware,” in *Proceedings of the 21st International Workshop on Data Management on New Hardware*, ser. DaMoN ’25. New York, NY, USA: Association for Computing Machinery, Jul. 2025, pp. 1–5. [Online]. Available: <https://dl.acm.org/doi/10.1145/3736227.3736235>
- [38] T. Høiland-Jørgensen, P. McKenney, D. Thät, J. Gettys, and E. Dumazet, “The flow queue CoDel packet scheduler and active queue management algorithm,” Internet Engineering Task Force, Request for Comments RFC 8290, Jan. 2018. [Online]. Available: <https://datatracker.ietf.org/doc/rfc8290>
- [39] T. Høiland-Jørgensen, D. Täht, and J. Morton, “Piece of CAKE: A comprehensive queue management solution for home gateways,” in *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, Jun. 2018, pp. 37–42.
- [40] S. Jung, J. Kim, and J.-H. Kim, “Intelligent active queue management for stabilized QoS guarantees in 5G mobile networks,” *IEEE Systems Journal*, vol. 15, no. 3, pp. 4293–4302, Sep. 2021.
- [41] H. Ji, S. Park, J. Yeo, Y. Kim, J. Lee, and B. Shim, “Ultra-reliable and low-latency communications in 5G downlink: Physical layer aspects,” *IEEE Wireless Communications*, vol. 25, no. 3, pp. 124–130, Jun. 2018.
- [42] R. Ali, Y. B. Zikria, A. K. Bashir, S. Garg, and H. S. Kim, “URLLC for 5G and beyond: Requirements, enabling incumbent technologies and network intelligence,” *IEEE access : practical innovations, open solutions*, vol. 9, pp. 67 064–67 095, 2021.
- [43] Wireshark Foundation, “Wireshark: The world’s leading network protocol analyzer,” 2025, Accessed: 2026-01-07. [Online]. Available: <https://www.wireshark.org/>
- [44] K. Nichols, “pping: Passive ping network monitoring utility,” Oct. 2022, Accessed: 2024-01-15. [Online]. Available: <https://github.com/pollere/pping>
- [45] F. Maurer, “Investigating causes of jitter in container networking,” Master’s thesis, KTH Royal Institute of Technology, 2021. [Online]. Available: <https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-304351>

- [46] G. Liang and M. Pumptis, "Pwru - Linux kernel and BPF-based networking debugger," Sep. 2024. [Online]. Available: <https://lpc.events/event/18/contributions/1942/>
- [47] M. A. Qureshi, J. Yan, Y. Cheng, S. H. Yeganeh, Y. Seung, N. Cardwell, W. De Bruijn, V. Jacobson, J. Kaur, D. Wetherall, and A. Vahdat, "Fathom: Understanding datacenter application network performance," in *Proceedings of the ACM SIGCOMM 2023 Conference*, ser. *Acm Sigcomm '23*. New York, NY, USA: Association for Computing Machinery, 2023, pp. 394–405. [Online]. Available: <https://doi.org/10.1145/3603269.3604815>
- [48] Intel Corporation, "Explore the Power of Intel® Intelligent Fabric Processors," n.d., Accessed: 2026-04-21. [Online]. Available: <https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch.html>
- [49] X. Chen, H. Kim, J. M. Aman, W. Chang, M. Lee, and J. Rexford, "Measuring TCP round-trip time in the data plane," in *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, ser. *SPIN '20*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 35–41. [Online]. Available: <https://doi.org/10.1145/3405669.3405823>
- [50] S. Sengupta, H. Kim, and J. Rexford, "Fine-grained RTT monitoring inside the network," in *Measuring Network Quality for End-Users 2021*. IAB, Sep. 2021.
- [51] —, "Continuous in-network round-trip time monitoring," in *Proceedings of the ACM SIGCOMM 2022 Conference*, ser. *SIGCOMM '22*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 473–485. [Online]. Available: <https://doi.org/10.1145/3544216.3544222>
- [52] M. Muuss, "The story of the PING program," 2000, Accessed: 2026-01-06. [Online]. Available: <https://web.archive.org/web/20260101223546/https://ftp.arl.army.mil/~mike/ping.html>
- [53] K. Phemius and M. Bouet, "Monitoring latency with OpenFlow," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, Oct. 2013, pp. 122–125.
- [54] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. *SIGCOMM '15*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 139–152. [Online]. Available: <https://doi.org/10.1145/2785956.2787496>

- [55] K. Liu, Z. Jiang, J. Zhang, H. Wei, X. Zhong, L. Tan, T. Pan, and T. Huang, “Hostping: Diagnosing intra-host network bottlenecks in RDMA servers,” in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 15–29. [Online]. Available: <https://www.usenix.org/conference/nsdi23/presentation/liu-kefei>
- [56] D. E. Cameron, M. Odiathevar, A. C. Valera, and W. K. G. Seah, “Polus: Detecting and characterising latency under load in multi-bottleneck wireless internet service provider networks,” in *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, May 2024, pp. 1–7.
- [57] C. Pelsser, L. Cittadini, S. Vissicchio, and R. Bush, “From Paris to Tokyo: On the suitability of ping to measure latency,” in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 427–432. [Online]. Available: <https://doi.org/10.1145/2504730.2504765>
- [58] B. Gbadamosi, L. Leonardi, T. Pulls, T. Høiland-Jørgensen, S. Ferlin-Reiter, S. Sorce, and A. Brunström, “The eBPF runtime in the Linux kernel,” Oct. 2024. [Online]. Available: <http://arxiv.org/abs/2410.00026>
- [59] W3Techs, “Usage statistics and market share of Linux for websites,” 2026, Accessed: 2026-01-18. [Online]. Available: <https://w3techs.com/technologies/details/os-linux>
- [60] W. Reese, “Nginx: The high-performance web server and reverse proxy,” *Linux J.*, vol. 2008, no. 173, Sep. 2008.
- [61] R. Fielding and G. Kaiser, “The Apache HTTP server project,” *IEEE Internet Computing*, vol. 1, no. 4, pp. 88–90, Jul. 1997.
- [62] HAProxy Technologies, “HAProxy — The reliable, high performance TCP/HTTP load balancer,” 2025, Accessed: 2025-01-18. [Online]. Available: <https://www.haproxy.org/>
- [63] The Varnish Cache Contributors, “Introduction to Varnish — Varnish HTTP cache,” 2024, Accessed: 2026-01-18. [Online]. Available: <https://vinyl-cache.org/intro/index.html>
- [64] N. Shirokov and R. Dasineni, “Open-sourcing Katran, a scalable network load balancer,” May 2018, Accessed: 2024-01-18. [Online]. Available: <https://engineering.fb.com/2018/05/22/open-source/open-sourcing-katran-a-scalable-network-load-balancer/>
- [65] D. Wragg, “Unimog - Cloudflare’s edge load balancer,” Sep. 2020, Accessed: 2026-01-19. [Online]. Available: <https://blog.cloudflare.com/unimog-cloudflares-edge-load-balancer/>

- [66] J.-B. Lee, T.-H. Yoo, E.-H. Lee, B.-H. Hwang, S.-W. Ahn, and C.-H. Cho, "High-performance software load balancer for cloud-native architecture," *IEEE access : practical innovations, open solutions*, vol. 9, pp. 123 704–123 716, 2021.
- [67] M. Cheminod, I. Cibrario Bertolotti, L. Durante, L. Seno, and A. Valenzano, "Open-source firewalls for industrial applications: A laboratory study of Linux IPFire behavior," in *IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society*, 2022.
- [68] F. Palmese, A. E. C. Redondi, and M. Cesana, "Feature-sniffer: Enabling IoT forensics in OpenWrt based Wi-Fi access points," in *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*, Oct. 2022, pp. 1–6.
- [69] S. Miano, M. Bertrone, F. Risso, M. V. Bernal, Y. Lu, and J. Pi, "Securing Linux with a faster and scalable iptables," *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 3, pp. 2–17, Nov. 2019. [Online]. Available: <https://doi.org/10.1145/3371927.3371929>
- [70] Cisco, "Snort - Network intrusion detection & prevention system," 2026, Accessed: 2024-01-18. [Online]. Available: <https://www.snort.org/>
- [71] H. Redžović, A. Smiljanić, and B. Savić, "Performance evaluation of software routers with VPN features," in *2016 24th Telecommunications Forum (TELFOR)*, Nov. 2016, pp. 1–4.
- [72] A. Abdelsalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, and L. Veltri, "Implementation of virtual network function chaining through segment routing in a Linux-based NFV infrastructure," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, Jul. 2017.
- [73] D. Lebrun and O. Bonaventure, "Implementing IPv6 segment routing in the Linux kernel," in *Proceedings of the 2017 Applied Networking Research Workshop*, ser. Anrw '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 35–41. [Online]. Available: <https://doi.org/10.1145/3106328.3106329>
- [74] OpenWrt Community, "[OpenWrt Wiki] About the OpenWrt/LEDE project," 2025, Accessed: 2026-01-18. [Online]. Available: <https://openwrt.org/about>
- [75] G. Cerar, H. Yetgin, M. Mohorčič, and C. Fortuna, "Machine learning for wireless link quality estimation: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 696–728, 2021.
- [76] N. Zhang, P. He, Z. Wu, P. Chen, L. Wang, and Z. Ye, "Latency analysis and trial for 5G ultra reliable low latency communication," in *2023 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, 2023, pp. 1–6.

- [77] T. E. Humphreys, P. A. Iannucci, Z. M. Komodromos, and A. M. Graff, "Signal structure of the Starlink Ku-band downlink," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 5, pp. 6016–6030, Oct. 2023.
- [78] N. Mohan, A. E. Ferguson, H. Cech, R. Bose, P. R. Renatin, M. K. Marina, and J. Ott, "A multifaceted look at Starlink performance," in *Proceedings of the ACM Web Conference 2024*, ser. WWW '24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 2723–2734. [Online]. Available: <https://dl.acm.org/doi/10.1145/3589334.3645328>
- [79] Y. Liu, J. Yang, Z. Wang, L. Pan, L. He, J. Lin, G. Song, and C. Li, "What causes delay asymmetry: A large-scale one-way delay measurement and empirical study," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, Dec. 2022, pp. 6127–6132.
- [80] A. Pathak, H. Pucha, Y. Zhang, Y. C. Hu, and Z. M. Mao, "A measurement study of internet delay asymmetry," in *Passive and Active Network Measurement*, M. Claypool and S. Uhlig, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 182–191.
- [81] Z. S. Bischof, J. P. Rula, and F. E. Bustamante, "In and out of Cuba: Characterizing Cuba's connectivity," in *Proceedings of the 2015 Internet Measurement Conference*, ser. IMC '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 487–493. [Online]. Available: <https://doi.org/10.1145/2815675.2815718>
- [82] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I.-J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, "Reducing Internet latency: A survey of techniques and their merits," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2149–2196, 2014.
- [83] A. Steed and M. F. Oliveira, "Chapter 10 - Requirements," in *Networked Graphics*, A. Steed and M. F. Oliveira, Eds. Boston: Morgan Kaufmann, 2010, pp. 313–353. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123744234000100>
- [84] U. Bauknecht and T. Enderle, "An investigation on core network latency," in *2020 30th International Telecommunication Networks and Applications Conference (ITNAC)*, Nov. 2020, pp. 1–6.
- [85] R. Noordally, X. Nicolay, P. Anelli, R. Lorion, and P. U. Tournoux, "Analysis of Internet latency: The Reunion Island case," in *Proceedings of the 12th Asian Internet Engineering Conference*, ser. Aintec '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 49–56. [Online]. Available: <https://doi.org.bibproxy.kau.se/10.1145/3012695.3012702>

- [86] I. N. Bozkurt, A. Aguirre, B. Chandrasekaran, P. B. Godfrey, G. Laughlin, B. Maggs, and A. Singla, "Why is the Internet so slow?!" in *Passive and Active Measurement*, M. A. Kaafar, S. Uhlig, and J. Amann, Eds. Cham: Springer International Publishing, 2017, pp. 173–187.
- [87] O. Victor Babasanmi and J. Chavula, "Measuring cloud latency in Africa," in *2022 IEEE 11th International Conference on Cloud Networking (Cloud-Net)*, Nov. 2022, pp. 61–66.
- [88] R. Bose, S. Fadaei, N. Mohan, M. Kassem, N. Sastry, and J. Ott, "It's a bird? It's a plane? It's CDN!: Investigating content delivery networks in the LEO satellite networks era," in *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, ser. HotNets '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 1–9. [Online]. Available: <https://doi.org/10.1145/3696348.3696879>
- [89] R. Singh, A. Dunna, and P. Gill, "Characterizing the deployment and performance of multi-CDNs," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 168–174. [Online]. Available: <https://doi.org/10.1145/3278532.3278548>
- [90] T. K. Dang, N. Mohan, L. Corneo, A. Zavodovski, J. Ott, and J. Kangasharju, "Cloudy with a chance of short RTTs: Analyzing cloud connectivity in the Internet," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 62–79. [Online]. Available: <https://doi.org/10.1145/3487552.3487854>
- [91] J. Sinha, R. Nishad, Stuti, and M. Hashir, "Reducing latency using P2P CDN," in *2023 7th International Conference on Trends in Electronics and Informatics (ICOEI)*, Apr. 2023, pp. 518–524.
- [92] H. Zhang, S. Huang, M. Xu, D. Guo, X. Wang, V. C. Leung, and W. Wang, "How far have edge clouds gone? A spatial-temporal analysis of edge network latency in the wild," in *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*, Jun. 2023, pp. 1–10.
- [93] 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. New York, NY, USA: Association for Computing Machinery, 2009, pp. 90–102. [Online]. Available: <https://doi.org/10.1145/1644893.1644904>
- [94] C. Pei, Y. Zhao, G. Chen, R. Tang, Y. Meng, M. Ma, K. Ling, and D. Pei, "WiFi can be the weakest link of round trip network latency in the wild," in *IEEE INFOCOM 2016 - the 35th Annual IEEE International Conference on Computer Communications*, Apr. 2016, pp. 1–9.

- [95] T. Høiland-Jørgensen, B. Ahlgren, P. Hurtig, and A. Brunstrom, “Measuring latency variation in the Internet,” in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 473–480. [Online]. Available: <https://doi.org/10.1145/2999572.2999603>
- [96] J. Kurose and K. Ross, “Computer networks and the Internet,” in *Computer Networking: A Top-Down Approach*, 6th ed. Pearson Education, Inc., 2013.
- [97] S. Agarwal, R. Agarwal, B. Montazeri, M. Moshref, K. Elmeleegy, L. Rizzo, M. A. de Kruijf, G. Kumar, S. Ratnasamy, D. Culler, and A. Vahdat, “Understanding host interconnect congestion,” in *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, ser. HotNets '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 198–204. [Online]. Available: <https://doi.org/10.1145/3563766.3564110>
- [98] S. Agarwal, A. Krishnamurthy, and R. Agarwal, “Host congestion control,” in *Proceedings of the ACM SIGCOMM 2023 Conference*, ser. Acm Sigcomm '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 275–287. [Online]. Available: <https://doi.org/10.1145/3603269.3604878>
- [99] M. Vuppalapati, S. Agarwal, H. Schuh, B. Kasikci, A. Krishnamurthy, and R. Agarwal, “Understanding the host network,” in *Proceedings of the ACM SIGCOMM 2024 Conference*, ser. Acm Sigcomm '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 581–594. [Online]. Available: <https://doi.org/10.1145/3651890.3672271>
- [100] S. Awamoto and M. Honda, “Opening up kernel-bypass TCP stacks,” in *Proceedings of the 2025 USENIX Annual Technical Conference*, Boston, USA, Jul. 2025. [Online]. Available: <https://www.usenix.org/conference/atc25/presentation/awamoto>
- [101] J. Gettys and K. Nichols, “Bufferbloat: Dark buffers in the Internet: Networks without effective AQM may again be vulnerable to congestion collapse.” *Queue*, vol. 9, no. 11, pp. 40–54, Nov. 2011. [Online]. Available: <https://doi.org/10.1145/2063166.2071893>
- [102] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, “BBR: Congestion-based congestion control,” *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, Jan. 2017. [Online]. Available: <https://doi.org/10.1145/3009824>
- [103] V. Arun and H. Balakrishnan, “Copa: Practical delay-based congestion control for the Internet,” in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA:

- USENIX Association, Apr. 2018, pp. 329–342. [Online]. Available: <https://www.usenix.org/conference/nsdi18/presentation/arun>
- [104] B. Briscoe, K. De Schepper, O. Tilmans, M. Kühlewind, J. Misund, O. Albisser, and A. S. Ahmed, “Implementing the Prague requirements for low latency low loss scalable throughput (L4S),” in *Netdev 0x13*, 2019. [Online]. Available: <https://www.netdevconf.info/0x13/session.html?talk-tcp-prague-l4s>
- [105] G. Kumar, N. Dukkipati, K. Jang, H. M. G. Wassel, X. Wu, B. Montazeri, Y. Wang, K. Springborn, C. Alfeld, M. Ryan, D. Wetherall, and A. Vahdat, “Swift: Delay is simple and effective for congestion control in the datacenter,” in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 514–528. [Online]. Available: <https://doi.org/10.1145/3387514.3406591>
- [106] K. D. Schepper, B. Briscoe, and G. White, “Dual-queue coupled active queue management (AQM) for low latency, low loss, and scalable throughput (L4S),” Internet Engineering Task Force, Request for Comments RFC 9332, Jan. 2023. [Online]. Available: <https://datatracker.ietf.org/doc/rfc9332>
- [107] W. G. de Morais, C. E. M. Santos, and C. M. Pedroso, “Application of active queue management for real-time adaptive video streaming,” *Telecommunication Systems*, vol. 79, no. 2, pp. 261–270, Feb. 2022. [Online]. Available: <https://doi.org/10.1007/s11235-021-00848-0>
- [108] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, “Low latency, low loss, and scalable throughput (L4S) Internet service: Architecture,” Internet Engineering Task Force, Request for Comments RFC 9330, Jan. 2023. [Online]. Available: <https://datatracker.ietf.org/doc/rfc9330>
- [109] Broadband Internet Technical Advisory Group, “Latency explained,” Tech. Rep., 2022. [Online]. Available: [https://www.bitag.org/documents/BITAG\\_latency\\_explained.pdf](https://www.bitag.org/documents/BITAG_latency_explained.pdf)
- [110] ETSI and 3GPP, “5G: Study on scenarios and requirements for next generation access technologies,” Tech. Rep. TR 38.913, 2025. [Online]. Available: [https://www.etsi.org/deliver/etsi\\_tr/138900\\_138999/138913/19.00.00\\_60/tr\\_138913v190000p.pdf](https://www.etsi.org/deliver/etsi_tr/138900_138999/138913/19.00.00_60/tr_138913v190000p.pdf)
- [111] X. Li, W. Xie, and C. Hu, “Research on 5G URLLC standard and key technologies,” in *2022 3rd Information Communication Technologies Conference (ICTC)*, May 2022, pp. 243–249.
- [112] A. Morton, “Active and passive metrics and methods (with hybrid types in-between),” Internet Engineering Task Force, Request for

- Comments RFC 7799, May 2016. [Online]. Available: <https://datatracker.ietf.org/doc/rfc7799>
- [113] R. T. Braden, “Requirements for Internet hosts - Communication layers,” Internet Engineering Task Force, Request for Comments RFC 1122, Oct. 1989. [Online]. Available: <https://datatracker.ietf.org/doc/rfc1122>
- [114] S. Bano, P. Richter, M. Javed, S. Sundaresan, Z. Durumeric, S. J. Murdoch, R. Mortier, and V. Paxson, “Scanning the Internet for liveness,” *SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 2, pp. 2–9, May 2018. [Online]. Available: <https://doi.org/10.1145/3213232.3213234>
- [115] R. Ravaioli, G. Urvoy-Keller, and C. Barakat, “Characterizing ICMP rate limitation on routers,” in *2015 IEEE International Conference on Communications (ICC)*, Jun. 2015, pp. 6043–6049.
- [116] P. Heist, “IRT’T (Isochronous Round-Trip Tester),” 2023, Accessed: 2026-01-07. [Online]. Available: <https://github.com/heistp/irtt>
- [117] M. J. Zekauskas, A. Karp, S. Shalunov, J. W. Boote, and B. R. Teitelbaum, “A one-way active measurement protocol (OWAMP),” Internet Engineering Task Force, Request for Comments RFC 4656, Sep. 2006. [Online]. Available: <https://datatracker.ietf.org/doc/rfc4656>
- [118] J. Babiarz, R. M. Krzanowski, K. Hedayat, K. Yum, and A. Morton, “A two-way active measurement protocol (TWAMP),” Internet Engineering Task Force, Request for Comments RFC 5357, Oct. 2008. [Online]. Available: <https://datatracker.ietf.org/doc/rfc5357>
- [119] Y. Liu, L. Pan, C. Li, L. He, Y. Luo, G. Song, J. Yang, and Z. Wang, “PerfTrace: A new multi-metric network performance monitoring tool,” in *2022 18th International Conference on Network and Service Management (CNSM)*, Oct. 2022, pp. 240–246.
- [120] The kernel development community, “Timestamping — The Linux kernel documentation,” 2025, Accessed: 2025-09-25. [Online]. Available: <https://www.kernel.org/doc/html/latest/networking/timestamping.html>
- [121] M. Primorac, E. Bugnion, and K. Argyraki, “How to measure the killer microsecond,” *SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 5, pp. 61–66, Oct. 2017. [Online]. Available: <https://doi.org/10.1145/3155055.3155065>
- [122] 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. New York, NY, USA: Association for Computing Machinery, 2015, pp. 275–287. [Online]. Available: <https://doi.org/10.1145/2815675.2815692>

- [123] T. Dreiholz, "High-precision round-trip time measurements in the Internet with HiPerConTracer," in *2023 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Sep. 2023, pp. 1–7.
- [124] Y. UO and T. Asada, "Nanoping," 2018, Accessed: 2026-01-20. [Online]. Available: <https://github.com/ijj/nanoping>
- [125] S. A. Mohammed, S. Shirmohammadi, and S. Altamimi, "Artificial intelligence-based distributed network latency measurement," in *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2019, pp. 1–6.
- [126] K. Baclawski, "The observer effect," in *2018 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, Jun. 2018, pp. 83–89.
- [127] Shawn Ostermann, "Tcptrace," Jun. 2013, Accessed: 2022-05-03. [Online]. Available: <https://github.com/blitz/tcptrace>
- [128] A. Finamore, M. Mellia, M. Meo, M. M. Munafo, P. D. Torino, and D. Rossi, "Experiences of Internet traffic monitoring with tstat," *IEEE Network*, vol. 25, no. 3, pp. 8–14, May 2011.
- [129] V. Moreno, J. Ramos, J. L. Garcia-dorado, I. Gonzalez, F. J. Gomez-arribas, and J. Aracil, "Testing the capacity of off-the-shelf systems to store 10GbE traffic," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 118–125, Sep. 2015.
- [130] J. Li, C. Wu, J. Ye, J. Ding, Q. Fu, and J. Huang, "The comparison and verification of some efficient packet capture and processing technologies," in *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. Fukuoka, Japan: IEEE, 2019, pp. 967–973.
- [131] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014. [Online]. Available: <https://doi.org/10.1145/2656877.2656890>
- [132] Endace Technology Limited, "Endace DAG 9.2X2 full duplex 10Gb/s packet capture card announced," 2010, Accessed: 2026-04-21. [Online]. Available: <https://www.endace.com/dag-9.2x2-full-duplex-10gbs-capable-pcie-gen-2-packet-capture-card.html>
- [133] M. Trevisan, A. Finamore, M. Mellia, M. Munafo, and D. Rossi, "Traffic analysis with off-the-shelf hardware: Challenges and lessons learned," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 163–169, 2017.

- [134] DPK Project, “DPDK – The open source data plane development kit accelerating network performance,” 2026, Accessed: 2026-01-09. [Online]. Available: <https://www.dpdk.org/>
- [135] S. Niccolini, M. Molina, F. Raspall, and S. Tartarelli, “Design and implementation of a one way delay passive measurement system,” in *2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04ch37507)*, vol. 1, Apr. 2004, pp. 469–482 Vol.1.
- [136] H. ElBouanani, C. Barakat, W. Dabbous, and T. Turletti, “Passive delay measurement for fidelity monitoring of distributed network emulation,” *Computer Communications*, vol. 195, pp. 40–48, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0140366422002523>
- [137] Hengyoush, “Kyanos,” 2024, Accessed: 2026-01-20. [Online]. Available: <https://kyanos.pages.dev/>
- [138] A. Tenart, P. Valerio, and A. Moreno, “Challenges and limitations of debugging increasingly complex virtualized networks with Retis,” Zagreb, Croatia, Mar. 2025. [Online]. Available: <https://netdevconf.info/Ox19/sessions/tutorial/challenges-and-limitations-of-debugging-increasingly-complex-virtualized-networks-with-retis.html>
- [139] L. Tan, W. Su, W. Zhang, J. Lv, Z. Zhang, J. Miao, X. Liu, and N. Li, “In-band network telemetry: A survey,” *Computer Networks*, vol. 186, p. 107763, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128620313396>
- [140] A. Yaar, A. Perrig, and D. Song, “Pi: A path identification mechanism to defend against DDoS attacks,” in *2003 Symposium on Security and Privacy, 2003.*, May 2003, pp. 93–107.
- [141] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, “PINT: Probabilistic in-band network telemetry,” in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. Sigcomm ’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 662–680. [Online]. Available: <https://doi.org/10.1145/3387514.3405894>
- [142] Q. Huang, H. Sun, P. P. C. Lee, W. Bai, F. Zhu, and Y. Bao, “OmniMon: Re-architecting network telemetry with resource efficiency and full accuracy,” in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. Sigcomm ’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 404–421. [Online]. Available: <https://doi.org/10.1145/3387514.3405877>

- [143] J.-T. Hinz, V. Addanki, C. Györgyi, T. Jepsen, and S. Schmid, "TCP's third eye: Leveraging eBPF for telemetry-powered congestion control," in *Proceedings of the 1st Workshop on EBPF and Kernel Extensions*, ser. eBPF '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 1–7. [Online]. Available: <https://doi.org/10.1145/3609021.3609295>
- [144] The P4.org Applications Working Group, "In-band network telemetry (INT) dataplane specification," Nov. 2020. [Online]. Available: [https://p4.org/wp-content/uploads/sites/53/p4-spec/docs/INT\\_v2\\_1.pdf](https://p4.org/wp-content/uploads/sites/53/p4-spec/docs/INT_v2_1.pdf)
- [145] D. Thaler, "BPF instruction set architecture (ISA)," Internet Engineering Task Force, Request for Comments RFC 9669, Oct. 2024. [Online]. Available: <https://datatracker.ietf.org/doc/rfc9669>
- [146] P. Gaddehosur and D. Thaler, "Making eBPF work on Windows," May 2021, Accessed: 2024-03-08. [Online]. Available: <https://cloudblogs.microsoft.com/opensource/2021/05/10/making-ebpf-work-on-windows/>
- [147] Y. Hayakawa, "eBPF implementation for FreeBSD," in *The 15th Annual Technical BSD Conference (BSDCan 2018)*, Jul. 2018. [Online]. Available: <https://www.bsdcan.org/2018/schedule/events/963.en.html>
- [148] eBPF.io, "What is eBPF? An introduction and deep dive into the eBPF technology," 2024, Accessed: 2024-03-08. [Online]. Available: <https://ebpf.io/what-is-ebpf/>
- [149] T. Høiland-Jørgensen, J. D. Brouer, D. Borkmann, J. Fastabend, T. Herbert, D. Ahern, and D. Miller, "The eXpress data path: Fast programmable packet processing in the operating system kernel," in *Proceedings of the 14th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 54–66. [Online]. Available: <https://doi.org/10.1145/3281411.3281443>
- [150] M. A. M. Vieira, M. S. Castanho, R. D. G. Pacífico, E. R. S. Santos, E. P. M. C. Júnior, and L. F. M. Vieira, "Fast packet processing with eBPF and XDP: Concepts, code, challenges, and applications," *ACM Computing Surveys*, vol. 53, no. 1, Feb. 2020. [Online]. Available: <https://doi.org/10.1145/3371038>
- [151] Cilium, "Cilium - Cloud native, eBPF-based networking, observability, and security," 2024, Accessed: 2024-03-08. [Online]. Available: <https://cilium.io>
- [152] V.-H. Tran and O. Bonaventure, "Beyond socket options: Making the Linux TCP stack truly extensible," in *2019 IFIP Networking Conference (IFIP Networking)*, May 2019, pp. 1–9.

- [153] Q. De Coninck, F. Michel, M. Piraux, F. Rochet, T. Given-Wilson, A. Legay, O. Pereira, and O. Bonaventure, “Pluginizing QUIC,” in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. Sigcomm ’19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 59–74. [Online]. Available: <https://doi.org/10.1145/3341302.3342078>
- [154] M. Jadin, Q. De Coninck, L. Navarre, M. Schapira, and O. Bonaventure, “Leveraging eBPF to make TCP path-aware,” *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2827–2838, Sep. 2022.
- [155] M. Xhonneux, F. Duchene, and O. Bonaventure, “Leveraging eBPF for programmable network functions with IPv6 segment routing,” in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT ’18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 67–72. [Online]. Available: <https://doi.org/10.1145/3281411.3281426>
- [156] N. Van Tu, J.-H. Yoo, and J. Won-Ki Hong, “Accelerating virtual network functions with fast-slow path architecture using eXpress data path,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1474–1486, Sep. 2020.
- [157] S. Miano, F. Risso, M. V. Bernal, M. Bertrone, and Y. Lu, “A framework for eBPF-based network functions in an era of microservices,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 133–151, Mar. 2021.
- [158] S. Qi, Z. Zeng, L. Monis, and K. K. Ramakrishnan, “MiddleNet: A unified, high-performance NFV and middlebox framework with eBPF and DPDK,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 3950–3967, Dec. 2023.
- [159] P. Szynekiewicz, “Signature-based detection of botnet DDoS attacks,” in *Cybersecurity of Digital Service Chains: Challenges, Methodologies, and Tools*, J. Kołodziej, M. Repetto, and A. Duzha, Eds. Cham: Springer International Publishing, 2022, pp. 120–135. [Online]. Available: [https://doi.org/10.1007/978-3-031-04036-8\\_6](https://doi.org/10.1007/978-3-031-04036-8_6)
- [160] S.-Y. Wang and J.-C. Chang, “Design and implementation of an intrusion detection system by using Extended BPF in the Linux kernel,” *Journal of Network and Computer Applications*, vol. 198, p. 103283, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1084804521002769>
- [161] A. Osaki, M. Poisson, S. Makino, R. Shiiba, K. Fukuda, T. Okoshi, and J. Nakazawa, “Dynamic fixed-point values in eBPF: A case for fully in-kernel anomaly detection,” in *Proceedings of the Asian Internet*

- Engineering Conference 2024*, ser. AINTEC '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 46–54. [Online]. Available: <https://dl.acm.org/doi/10.1145/3674213.3674219>
- [162] F. Parola, F. Risso, and S. Miano, “Providing telco-oriented network services with eBPF: The case for a 5G mobile gateway,” in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, Jun. 2021, pp. 221–225.
- [163] T. A. N. do Amaral, R. V. Rosa, D. F. C. Moura, and C. E. Rothenberg, “An in-kernel solution based on XDP for 5G UPF: Design, prototype and performance evaluation,” in *2021 17th International Conference on Network and Service Management (CNSM)*, Oct. 2021, pp. 146–152.
- [164] X. Foukas, B. Radunovic, M. Balkwill, and Z. Lai, “Taking 5G RAN analytics and control to a new level,” in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, ser. ACM MobiCom '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: <https://doi.org/10.1145/3570361.3592493>
- [165] The Tcpdump Group, “Tcpdump & libpcap,” 2025, Accessed: 2026-01-19. [Online]. Available: <https://www.tcpdump.org/>
- [166] F. Parola, R. Procopio, R. Querio, and F. Risso, “Comparing user space and in-kernel packet processing for edge data centers,” *ACM SIGCOMM Computer Communication Review*, vol. 53, no. 1, pp. 14–29, Apr. 2023. [Online]. Available: <https://doi.org.bibproxy.kau.se/10.1145/3594255.3594257>
- [167] W. Tu, Y.-H. Wei, G. Antichi, and B. Pfaff, “Revisiting the open vSwitch dataplane ten years later,” in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 245–257. [Online]. Available: <https://doi.org/10.1145/3452296.3472914>
- [168] G. Jereczek, T. Jepsen, S. Wass, B. Pujari, J. Zhen, and J. Lee, “TCP-INT: Lightweight network telemetry with TCP transport,” in *Proceedings of the SIGCOMM '22 Poster and Demo Sessions*, ser. Sigcomm '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 58–60. [Online]. Available: <https://doi.org/10.1145/3546037.3546064>
- [169] T. Osiński and C. Cascone, “Achieving end-to-end network visibility with host-INT,” in *Proceedings of the Symposium on Architectures for Networking and Communications Systems*, ser. Ancs '21. New York, NY, USA: Association for Computing Machinery, 2022, pp. 140–143. [Online]. Available: <https://doi.org/10.1145/3493425.3502764>

- [170] Intel® Corporation, “Host-INT for packet-telemetry,” 2023, Accessed: 2026-01-19. [Online]. Available: <https://github.com/intel/host-int>
- [171] J. Sommers, N. Rudolph, and R. Durairajan, “Schooling NOOBs with eBPF,” in *Proceedings of the 1st Workshop on eBPF and Kernel Extensions*, ser. eBPF ’23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 21–27. [Online]. Available: <https://doi.org/10.1145/3609021.3609302>
- [172] IO Visor Project, “BCC - Tools for BPF-based Linux IO analysis, networking, monitoring, and more,” 2026, Accessed: 2026-01-19. [Online]. Available: <https://github.com/iovisor/bcc>
- [173] P. G. Kannan, S. M. Gupta, D. Behl, E. Raichstein, and J. Takvorian, “Designing a lightweight network observability agent for cloud applications,” in *Passive and Active Measurement*, P. Richter, V. Bajpai, and E. Carisimo, Eds. Cham: Springer Nature Switzerland, 2024, pp. 262–276.
- [174] D. Behl, J. Pinsonneau, and M. Mahmoud, “Network observability using TCP handshake round-trip time,” Feb. 2024, Accessed: 2026-01-19. [Online]. Available: <https://developers.redhat.com/articles/2024/02/27/network-observability-using-tcp-handshake-round-trip-time>
- [175] Starlink, “Starlink 2025 progress report: Expanding boundaries for humans on Earth,” Tech. Rep., 2026. [Online]. Available: [https://starlink.com/public-files/starlinkProgressReport\\_2025.pdf](https://starlink.com/public-files/starlinkProgressReport_2025.pdf)
- [176] J. Pan, J. Zhao, and L. Cai, “Measuring a low-earth-orbit satellite network,” in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2023, pp. 1–6.
- [177] J. Garcia, S. Sundberg, and A. Brunstrom, “Inferring Starlink physical layer transmission rates through receiver packet timestamps,” in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, 2024, pp. 1–6.
- [178] H. B. Tanveer, M. Puchol, R. Singh, A. Bianchi, and R. Nithyanand, “Making sense of constellations: Methodologies for understanding Starlink’s scheduling algorithms,” in *Companion of the 19th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT 2023. New York, NY, USA: Association for Computing Machinery, 2023, pp. 37–43. [Online]. Available: <https://doi.org/10.1145/3624354.3630586>
- [179] J. Garcia, S. Sundberg, G. Caso, and A. Brunstrom, “Multi-timescale evaluation of Starlink throughput,” in *Proceedings of the 1st ACM Workshop on LEO Networking and Communication*, ser. LEO-NET ’23. New York, NY, USA: Association for

- Computing Machinery, 2023, pp. 31–36. [Online]. Available: <https://doi.org/10.1145/3614204.3616108>
- [180] S. Ma, Y. C. Chou, H. Zhao, L. Chen, X. Ma, and J. Liu, “Network characteristics of LEO satellite constellations: A Starlink-based measurement from end users,” in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, May 2023, pp. 1–10.
- [181] J. Pan, J. Zhao, and L. Cai, “Measuring the satellite links of a LEO network,” in *ICC 2024 - IEEE International Conference on Communications*, Jun. 2024, pp. 4439–4444.
- [182] L. Izhikevich, M. Tran, K. Izhikevich, G. Akiwate, and Z. Durumeric, “Democratizing LEO satellite network measurement,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 8, no. 1, Feb. 2024. [Online]. Available: <https://doi.org/10.1145/3639039>
- [183] F. Michel, M. Trevisan, D. Giordano, and O. Bonaventure, “A first look at Starlink performance,” in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 130–136. [Online]. Available: <https://doi.org/10.1145/3517745.3561416>
- [184] J. Zhao and J. Pan, “LENS: A LEO satellite network measurement dataset,” in *Proceedings of the 15th ACM Multimedia Systems Conference*, ser. MMSys '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 278–284. [Online]. Available: <https://doi.org/10.1145/3625468.3652170>
- [185] Y. Wu, M. Cui, G. Gou, Y. Wei, G. Xiong, Z. Li, X. Ju, and W. Xia, “Beneath the heavens: A thorough measurement study of the Starlink terrestrial network,” in *2025 IEEE/ACM 33rd International Symposium on Quality of Service (IWQoS)*, 2025, pp. 1–10.
- [186] B. Hu, X. Zhang, Q. Zhang, N. Varyani, Z. M. Mao, F. Qian, and Z.-L. Zhang, “LEO satellite vs. cellular networks: Exploring the potential for synergistic integration,” in *Companion of the 19th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT 2023. New York, NY, USA: Association for Computing Machinery, 2023, pp. 45–51. [Online]. Available: <https://doi.org/10.1145/3624354.3630588>
- [187] C. Beckman, J. Garcia, H. Mikkelsen, and P. Persson, “Starlink and cellular connectivity under mobility: Drive testing across the arctic circle,” in *2024 Wireless Telecommunications Symposium (WTS)*, Apr. 2024, pp. 1–9.
- [188] A. Ghafoori, A. Famili, and A. Stavrou, “Stars and towers on the wheels: Global perspective on satellite networks vs. terrestrial 5G,” in

- GLOBECOM 2024 - 2024 IEEE Global Communications Conference*, Dec. 2024, pp. 301–306.
- [189] D. Laniewski, E. Lanfer, and N. Aschenbruck, “Measuring mobile Starlink performance: A comprehensive look,” *IEEE Open Journal of the Communications Society*, vol. 6, pp. 1266–1283, 2025.
- [190] T. B. Sørensen, P. Mogensen, and M. López, “Fixed broadband service via Starlink in under-served areas - the Danish case,” in *2024 IEEE 100th Vehicular Technology Conference (VTC2024-fall)*, Oct. 2024, pp. 1–6.
- [191] D. Laniewski, E. Lanfer, B. Meijerink, R. van Rijswijk-Deij, and N. Aschenbruck, “WetLinks: A large-scale longitudinal Starlink dataset with contiguous weather data,” in *2024 8th Network Traffic Measurement and Analysis Conference (TMA)*, May 2024, pp. 1–9.
- [192] E. Lanfer, D. Laniewski, D. Otten, and N. Aschenbruck, “Weather-based link prediction for LEO-satellite networks using the WetLinks dataset,” in *2024 IFIP Networking Conference (IFIP Networking)*, Jun. 2024, pp. 586–588.
- [193] H. Nyquist, “Certain topics in telegraph transmission theory,” *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, Apr. 1928.
- [194] C. Shannon, “Communication in the presence of noise,” *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, Jan. 1949.
- [195] G. Barbosa, S. Theeranantachai, B. Zhang, and L. Zhang, “A comparative evaluation of TCP congestion control schemes over low-Earth-orbit (LEO) satellite networks,” in *Proceedings of the 18th Asian Internet Engineering Conference*, ser. Aintec ’23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 105–112. [Online]. Available: <https://doi.org/10.1145/3630590.3630603>
- [196] S. Ha, I. Rhee, and L. Xu, “CUBIC: A new TCP-friendly high-speed TCP variant,” *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008. [Online]. Available: <https://doi.org/10.1145/1400097.1400105>
- [197] Z. Lai, Z. Li, Q. Wu, H. Li, W. Liu, Y. Liu, X. Xie, Y. Li, and J. Liu, “Mind the misleading effects of LEO mobility on end-to-end congestion control,” in *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, ser. HotNets ’24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 34–42. [Online]. Available: <https://doi.org/10.1145/3696348.3696867>
- [198] J. Garcia, S. Sundberg, and A. Brunstrom, “TCP congestion control performance over Starlink,” in *Proceedings of the 2025 Applied Networking Research Workshop*. New York, NY, USA: Association for Computing Machinery, 2025, pp. 70–77. [Online]. Available: <https://doi.org/10.1145/3744200.3744760>

- [199] G. Dodig-Crnkovic, "Scientific methods in computer science," in *Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, 2002. [Online]. Available: <http://www.es.mdu.se/publications/375->
- [200] A. H. Eden, "Three paradigms of computer science," *Minds and Machines*, vol. 17, no. 2, pp. 135–167, Jul. 2007. [Online]. Available: <https://doi.org/10.1007/s11023-007-9060-8>
- [201] S. Hemminger, "Network emulation with NetEm," in *Linux Conf Au*, vol. 5, 2005. [Online]. Available: <https://www.rationali.st/jittertrap-baby-steps-in-dsp/netem-shemminger.pdf>
- [202] Ookla, "Speedtest CLI: Internet speed test for the command line," 2023, Accessed: 2023-06-16. [Online]. Available: <https://www.speedtest.net/apps/cli>
- [203] I. Kunze, C. Sander, and K. Wehrle, "Does it spin? On the adoption and use of QUIC's spin bit," in *Proceedings of the 2023 ACM on Internet Measurement Conference*, ser. IMC '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 554–560. [Online]. Available: <https://doi.org/10.1145/3618257.3624844>
- [204] O. Michel, S. Sengupta, H. Kim, R. Netravali, and J. Rexford, "Enabling passive measurement of zoom performance in production networks," in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 244–260. [Online]. Available: <https://doi.org/10.1145/3517745.3561414>
- [205] ITU, "Latency measurement and interactivity scoring under real application data traffic patterns," Tech. Rep. Recommendation ITU-T G.1051 (03/2023), Mar. 2023. [Online]. Available: <https://www.itu.int/epublications/publication/itu-t-g-1051-2023-03-latency-measurement-and-interactivity-scoring-under-real-application-data-traffic-patterns>
- [206] B. I. Teigen, M. Olden, and I. Kunze, "Quality of outcome (QoO)," Internet Engineering Task Force, Internet Draft draft-ietf-ippm-qoo-07, Feb. 2026. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-ippm-qoo>
- [207] R. Jota, A. Ng, P. Dietz, and D. Wigdor, "How fast is fast enough? A study of the effects of latency in direct-touch pointing tasks," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 2291–2300. [Online]. Available: <https://doi.org/10.1145/2470654.2481317>

- [208] S. Qi, L. Monis, Z. Zeng, I.-c. Wang, and K. K. Ramakrishnan, “SPRIGHT: Extracting the server from serverless computing! High-performance eBPF-based event-driven, shared-memory processing,” in *Proceedings of the ACM SIGCOMM 2022 Conference*, ser. SIGCOMM ’22. New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 780–794. [Online]. Available: <https://dl.acm.org/doi/10.1145/3544216.3544259>
- [209] A. Daichendt, F. Wiedner, J. Andre, and G. Carle, “Applicability of hardware-supported containers in low-latency networking,” in *2024 20th International Conference on Network and Service Management (CNSM)*, Oct. 2024, pp. 1–7.
- [210] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, “Netalyzer: Illuminating the edge network,” in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC ’10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 246–259. [Online]. Available: <https://doi.org/10.1145/1879141.1879173>
- [211] V. Bajpai, S. J. Eravuchira, and J. Schönwälder, “Dissecting last-mile latency characteristics,” *SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 5, pp. 25–34, Oct. 2017. [Online]. Available: <https://doi.org/10.1145/3155055.3155059>
- [212] C. Lu, B. Liu, Z. Li, S. Hao, H. Duan, M. Zhang, C. Leng, Y. Liu, Z. Zhang, and J. Wu, “An end-to-end, large-scale measurement of DNS-over-Encryption: How far have we come?” in *Proceedings of the Internet Measurement Conference*, ser. IMC ’19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 22–35. [Online]. Available: <https://doi.org/10.1145/3355369.3355580>
- [213] R. Fontugne, A. Shah, and K. Cho, “Persistent last-mile congestion: Not so uncommon,” in *Proceedings of the ACM Internet Measurement Conference*, ser. Imc ’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 420–427. [Online]. Available: <https://doi.org/10.1145/3419394.3423648>
- [214] M. Candela, V. Luconi, and A. Vecchio, “Impact of the COVID-19 pandemic on the Internet latency: A large-scale study,” *Computer Networks*, vol. 182, p. 107495, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128620311622>
- [215] V. Luconi and A. Vecchio, “Impact of the first months of war on routing and latency in Ukraine,” *Computer Networks*, vol. 224, p. 109596, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128623000415>
- [216] G. Caso, M. Rajiullah, A. Brunstrom, L. De Nardis, Ö. Alay, and M. Neri, “A standard-compliant assessment of beyond-eMBB QoS/QoE

- in 5G networks,” in *2024 IEEE Conference on Standards for Communications and Networking (CSCN)*, Nov. 2024, pp. 230–236.
- [217] J. Schnitzer, D. Reed, and A. Gandhi, “Where has the time gone? Examining over a decade of broadband latency measurements,” in *2024 8th Network Traffic Measurement and Analysis Conference (TMA)*, May 2024, pp. 1–10.
- [218] T. Favale, F. Soro, M. Trevisan, I. Drago, and M. Mellia, “Campus traffic and e-Learning during COVID-19 pandemic,” *Computer Networks*, vol. 176, p. 107290, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128620306046>
- [219] M. Trevisan, D. Giordano, I. Drago, M. M. Munafò, and M. Mellia, “Five years at the edge: Watching Internet from the ISP network,” *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 561–574, Apr. 2020.
- [220] T. Høiland-Jørgensen, P. Hurtig, and A. Brunstrom, “The good, the bad and the WiFi: Modern AQMs in a residential setting,” *Computer Networks*, vol. 89, pp. 90–106, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128615002479>
- [221] J. Li, N. K. Sharma, D. R. K. Ports, and S. D. Gribble, “Tales of the tail: Hardware, OS, and application-level sources of tail latency,” in *Proceedings of the ACM Symposium on Cloud Computing*, ser. Socc '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 1–14. [Online]. Available: <https://doi.org/10.1145/2670979.2670988>
- [222] Q. Cai, S. Chaudhary, M. Vuppapapati, J. Hwang, and R. Agarwal, “Understanding host network stack overheads,” in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 65–77. [Online]. Available: <https://doi.org/10.1145/3452296.3472888>
- [223] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, and B. Plattner, “The role of network trace anonymization under attack,” *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 5–11, Jan. 2010. [Online]. Available: <https://doi.org/10.1145/1672308.1672310>
- [224] N. V. Dijkhuizen and J. V. D. Ham, “A survey of network traffic anonymisation techniques and implementations,” *ACM Computing Surveys*, vol. 51, no. 3, 2018. [Online]. Available: <https://doi.org/10.1145/3182660>
- [225] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, “Mahimahi: Accurate record-and-replay for HTTP,” in *Usenix Annual Technical Conference*, 2015, pp. 417–429.

- [226] J. Garcia and P. Hurtig, “KauNetEm: Deterministic network emulation in Linux,” in *The Technical Conference on Linux Networking (Netdev 1.1)*, 2016. [Online]. Available: <https://netdevconf.info//1.1/talk-deterministic-network-emulation-using-kaunetem-hurtig-johan-garcia.html>
- [227] J. Garcia, S. Sundberg, and A. Brunstrom, “Fine-grained Starlink throughput variation examined with state-transition modeling,” in *2024 19th Wireless On-Demand Network Systems and Services Conference (WONS)*, 2024, pp. 69–76.
- [228] J. Garcia, M. Beckerle, S. Sundberg, and A. Brunstrom, “Modeling and predicting Starlink throughput with fine-grained burst characterization,” *Computer Communications*, vol. 234, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0140366425000477>
- [229] D. Ukwen, J. Garcia, A. Brunstrom, and M. Rajiullah, “Examining the predictability of Starlink downlink throughput,” in *2024 IEEE 29th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Oct. 2024, pp. 1–6.
- [230] T. Herbert and W. Bruijn, “The Linux kernel documentation: Scaling in the Linux networking stack,” n.d., Accessed: 2023-08-24. [Online]. Available: <https://docs.kernel.org/networking/scaling.html>
- [231] S. K. Singh, C. E. Rothenberg, M. C. Luizelli, G. Antichi, P. H. Gomes, and G. Pongrácz, “HH-IPG: Leveraging inter-packet gap metrics in P4 hardware for heavy hitter detection,” *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.
- [232] F. G. Vogt, F. Rodriguez, C. Rothenberg, and G. Pongrácz, “Innovative network monitoring techniques through in-band inter packet gap telemetry (IPGNET),” in *Proceedings of the 5th International Workshop on P4 in Europe*, ser. EuroP4 ’22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 53–56. [Online]. Available: <https://doi.org/10.1145/3565475.3569077>
- [233] M. Muehleisen and M. Abdel Latif, “Precise one-way delay measurement with common hardware and software,” in *Mobilkommunikation; 28. ITG-fachtagung*, May 2024, pp. 101–105. [Online]. Available: <https://ieeexplore.ieee.org/document/10651565>
- [234] J. Chen, Y. Li, H. Chai, J. Wang, S. Wu, and J. Pan, “Minimum round-trip time prediction for low Earth orbit satellite networks,” *International Journal of Satellite Communications and Networking*, vol. 43, no. 4, pp. 309–317, 2025. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/sat.1557>

- [235] J. Zheng, T. H. Luan, G. Li, J. Zhao, Z. Yin, N. Cheng, and J. Pan, "Low Earth orbit satellite networks: Architecture, key technologies, measurement, and open issues," *IEEE Network*, 2025.
- [236] I. Newton, "Isaac Newton letter to Robert Hooke," 1675. [Online]. Available: <https://digitallibrary.hsp.org/Detail/objects/9792>
- [237] LibreQoS, LLC, "LibreQoS — Network quality of experience," 2025, Accessed: 2026-01-07. [Online]. Available: <https://libreqos.io/>
- [238] F. Borsik, "State of the bloat," 2025. [Online]. Available: <https://www.youtube.com/watch?v=bmhlcl01VM4&t=7455s>
- [239] D. E. Cameron, A. C. Valera, and W. K. Seah, "PURPLE CAKE: Dynamic control of the common applications kept enhanced scheduler," in *2025 IEEE Conference on Dependable, Autonomic and Secure Computing (DASC)*, Oct. 2025, pp. 23–30.
- [240] B. Renl, D. Guo, G. Tang, W. Wang, L. Luo, and X. Fu, "SRUF: Low-latency path routing with SRv6 underlay federation in wide area network," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2021, pp. 910–920.
- [241] J. Chen, M. Chen, X. Wei, and B. Chen, "Matrix differential decomposition-based anomaly detection and localization in NFV networks," *IEEE access : practical innovations, open solutions*, vol. 7, pp. 29 320–29 331, 2019.
- [242] J. Han, T. Liu, J. Ma, Y. Zhou, X. Zeng, and Y. Xu, "Anomaly detection and early warning model for latency in private 5G networks," *Applied Sciences*, vol. 12, no. 12472, 2022. [Online]. Available: <https://www.mdpi.com/2076-3417/12/23/12472>
- [243] B. Wydrowski, R. Kleinberg, S. M. Rumble, and A. Archer, "Load is not what you should balance: Introducing Prequal," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. Santa Clara, CA: USENIX Association, 2024, pp. 1285–1299. [Online]. Available: <https://www.usenix.org/conference/nsdi24/presentation/wydrowski>
- [244] L. Jiang, Y. Zhang, J. Yin, X. Zhang, and B. Liu, "LEOTP: An information-centric transport layer protocol for LEO satellite networks," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2023, pp. 579–590.
- [245] Z. Lai, Z. Li, Q. Wu, H. Li, J. Li, X. Xie, Y. Li, J. Liu, and J. Wu, "LeoCC: Making Internet congestion control robust to LEO satellite dynamics," in *Proceedings of the ACM SIGCOMM 2025 Conference*, ser. Sigcomm '25. New York, NY, USA: Association for Computing Machinery, 2025, pp. 129–146. [Online]. Available: <https://doi.org/10.1145/3718958.3750491>

- [246] X. Liu, H. Zhou, Z. Zhang, Q. Gao, and T. Ma, "Multipath cooperative routing in ultradense LEO satellite networks: A deep-reinforcement-learning-based approach," *IEEE Internet of Things Journal*, vol. 12, no. 2, pp. 1789–1804, Jan. 2025.
- [247] A. Yonchev, E.-R. Modroiu, M. Corici, T. Magedanz, and C. Cavadar, "Towards reliable and cost-effective backhauling for private 5G networks: A multipath terrestrial-satellite transport network leveraging multi-armed bandit algorithms," in *Advanced Information Networking and Applications*, L. Barolli, Ed. Cham: Springer Nature Switzerland, 2025, pp. 268–280.
- [248] J. Zhao, O. Perrin, A. Ahangarpour, and J. Pan, "Measuring the OneWeb satellite network," in *2025 9th Network Traffic Measurement and Analysis Conference (TMA)*, Jun. 2025, pp. 1–10.
- [249] O. Abbasi, A. Yadav, H. Yanikomeroglu, N.-D. Đào, G. Senarath, and P. Zhu, "HAPS for 6G networks: Potential use cases, open challenges, and possible solutions," *IEEE Wireless Communications*, vol. 31, no. 3, pp. 324–331, Jun. 2024.
- [250] J. García-Morales, M. C. Lucas-Estañ, and J. Gozalvez, "Latency-sensitive 5G RAN slicing for industry 4.0," *IEEE access : practical innovations, open solutions*, vol. 7, pp. 143 139–143 159, 2019.
- [251] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, H. D. Schotten, and X. Costa-Pérez, "LACO: A latency-driven network slicing orchestration in beyond-5G networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 667–682, Jan. 2021.
- [252] M. U. A. Siddiqui, H. Abumarshoud, L. Bariah, S. Muhaidat, M. A. Imran, and L. Mohjazi, "URLLC in beyond 5G and 6G networks: An interference management perspective," *IEEE Access*, vol. 11, pp. 54 639–54 663, 2023.



# What's the Delay? Understanding Latency Across the Network

Network latency impacts the user experience for many current Internet applications and remains a key challenge for enabling future network use cases. However, the observability of latency in many network environments is poor, leaving both network operators and researchers alike with a limited understanding of the delays in modern networks. To improve the visibility of network latency, this thesis explores how eBPF can be leveraged to passively and continuously monitor production traffic, and how hardware-supported high-frequency active measurements can be used to capture the rapid latency fluctuations in wireless networks. We utilize our solutions to study the latency inside a wireless Internet service provider network, the packet processing time within the web servers, and the delays across the Starlink network. Our measurements reveal long latency tails in the last-mile access, within the host network stack, and from the resource allocation mechanisms in the Starlink network. Our measurements thereby provide a better understanding of latency in current networks, while our novel measurement tools provide the means to study the latency in future networks.

ISBN 978-91-7867-709-2 (print)

---

ISBN 978-91-7867-710-8 (pdf)

---

ISSN 1403-8099

---

DOCTORAL THESIS | Karlstad University Studies | 2026:30

---