Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich

Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)

Winter Semester 2024/2025

August 5, 2024 – March 2, 2025

Munich, Germany

Editors

Georg Carle, Marcel Kempf, Daniel Petri, Stefan Genchev

Publisher

Chair of Network Architectures and Services

Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich

Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)

Winter Semester 2024/2025

Munich, August 5, 2024 – March 2, 2025

Editors: Georg Carle, Marcel Kempf, Daniel Petri, Stefan Genchev

Network Architectures
and Services
NET 2025-05-1

Proceedings of the Seminar
Innovative Internet Technologies and Mobile Communications (IITM)
Winter Semester 2024/2025

Editors:

Georg Carle
Chair of Network Architectures and Services (I8)
Technical University of Munich
Boltzmannstraße 3, 85748 Garching b. München, Germany
E-mail: carle@net.in.tum.de
Internet: https://net.in.tum.de/~carle/

Marcel Kempf
Chair of Network Architectures and Services (I8)
E-mail: kempfm@net.in.tum.de
Internet: https://net.in.tum.de/~kempfm/

Daniel Petri
Chair of Network Architectures and Services (I8)
E-mail: petriroc@net.in.tum.de
Internet: https://net.in.tum.de/~petri/

Stefan Genchev
Chair of Network Architectures and Services (I8)
E-mail: genchev@net.in.tum.de
Internet: https://net.in.tum.de/~genchev/

Cataloging-in-Publication Data

Seminar IITM WS 24/25
Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)
Munich, Germany, August 5, 2024 – March 2, 2025
ISBN: 978-3-937201-82-5

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2025-05-1
Innovative Internet Technologies and Mobile Communications (IITM) NET 2025-05-1
Series Editor: Georg Carle, Technical University of Munich, Germany

Preface

We are pleased to present to you the proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM) during the Winter Semester 2024/2025. Each semester, the seminar takes place in two different ways: once as a block seminar during the semester break and once in the course of the semester. Both seminars share the same contents and differ only in their duration.

In the context of the seminar, each student individually works on a relevant topic in the domain of computer networks, supervised by one or more advisors. Advisors are staff members working at the Chair of Network Architectures and Services at the Technical University of Munich. As part of the seminar, the students write a scientific paper about their topic and afterward present the results to the other course participants. To improve the quality of the papers, we conduct a peer review process in which each paper is reviewed by at least two other seminar participants and the advisors.

Among all participants of each seminar, we award one with the *Best Paper Award*. For this semester, the awards were given to Len Bioly with the paper *Assessing the Energy Consumption of Software* and Karoline Spohn with the paper *The State of DNS Delegation: Technical Challenges and Solution Approaches* .

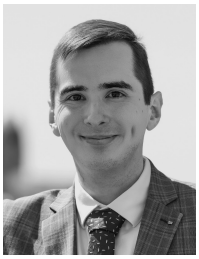
We hope that you appreciate the contributions of these seminars. If you are interested in further information about our work, please visit our homepage `https://net.in.tum.de`.

Munich, May 2025

Georg Carle

Marcel Kempf

Daniel Petri



Stefan Genchev

Seminar Organization

Chair Holder

Georg Carle, Technical University of Munich, Germany

Technical Program Committee

Marcel Kempf, Technical University of Munich, Germany
Daniel Petri, Technical University of Munich, Germany
Stefan Genchev, Technical University of Munich, Germany

Advisors

Jonas Andre (andre@net.in.tum.de)
Technical University of Munich

Tim Betzer (betzer@net.in.tum.de)
Technical University of Munich

Christian Dietze (diec@net.in.tum.de)
Technical University of Munich

Sebastian Gallenmüller (gallenmu@net.in.tum.de)
Technical University of Munich

Stefan Genchev (genchev@net.in.tum.de)
Technical University of Munich

Kilian Glas (glask@net.in.tum.de)
Technical University of Munich

Eric Hauser (hauser@net.in.tum.de)
Technical University of Munich

Max Helm (helm@net.in.tum.de)
Technical University of Munich

Kilian Holzinger (holzinger@net.in.tum.de)
Technical University of Munich

Benedikt Jaeger (jaeger@net.in.tum.de)
Technical University of Munich

Marcel Kempf (kempfm@net.in.tum.de)
Technical University of Munich

Holger Kinkelin (kinkelin@net.in.tum.de)
Technical University of Munich

Stefan Lachnit (lachnit@net.in.tum.de)
Technical University of Munich

Daniel Petri (petriroc@net.in.tum.de)
Technical University of Munich

Filip Rezabek (rezabek@net.in.tum.de)
Technical University of Munich

Patrick Sattler (sattler@net.in.tum.de)
Technical University of Munich

Leander Seidlitz (seidlitz@net.in.tum.de)
Technical University of Munich

Markus Sosnowski (sosnowski@net.in.tum.de)
Technical University of Munich

Johannes Späth (spaethj@net.in.tum.de)
Technical University of Munich

Lion Steger (stegerl@net.in.tum.de)
Technical University of Munich

Florian Wiedner (wiedner@net.in.tum.de)
Technical University of Munich

Seminar Homepage

`https://net.in.tum.de/teaching/ws2425/seminars/`

Contents

Blockseminar

Seminar

Assessing the Energy Consumption of Software

Len Bioly, Kilian Holzinger*, Johannes Späth*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: len.bioly@tum.de, holzingk@net.in.tum.de, spaethj@net.in.tum.de

***Abstract*—To lower costs and help the environment, green software engineering is becoming more and more critical to lower software's energy consumption. Therefore, this paper reviews methods to measure the energy consumption of software, including hardware-based methods like Intel RAPL, AMD APM, and an experimental approach named SEFlab. Furthermore, it covers software-based estimations created with eLens, GreenOracle, and Silicon Labs Energy Profiler. Hardware-based methods can achieve precise measurements at runtime but suffer from a long setup time. It is also difficult to detect which current comes from which software. The software-based methods are helpful in development because they are easy to set up and can visualize the energy consumption fine-grained in the code. The paper also includes information about CloudSIM and proprietary methods of market-leading cloud distributors like Amazon to reflect the current research results in energy-efficient cloud computing.**

***Index Terms*—energy consumption, software measurement, runtime monitoring, software-based estimation, distributed systems**

1. Introduction

The quantity of information and communication technology (ICT) is rising yearly; therefore, energy consumption and costs are breaking record after record. The SMARTer 2030 Report forecasts that, in 2030, ICT will be responsible for 2% of all carbon emissions [1]. Because carbon dioxide will be the primary contributor to global warming, it is essential to lower the energy consumption of software to combat the climate crisis [2]. To achieve this objective, the developers must take exact energy measurements of their software.

There can be many methods for analysing software energy consumption; therefore, this paper outlines some measurement methods. It contains black-box testing methods measuring energy consumption at runtime and white-box software-based estimation methods integrated into development environments. This review details Intel Running Average Power Limit (RAPL), AMD APM, and SEFlab as runtime measurements, besides GreenOracle, eLens, and Silicon Labs Energy Profiler as software-based estimation methods. Furthermore, it analyzes the accuracy and precision and lines out the difficulties and concerns of the mentioned methods.

Cloud Computing has risen exponentially over the last few years [3]. More complex measuring structures are needed to provide a platform for researchers and developers to test and optimize their systems. Therefore, this paper reflects CloudSIM 7G, a state-of-the-art, robust simulation toolkit for cloud computing. To introduce an example of monitoring methods at major cloud service providers, AWS CloudWatch is detailed.

2. Related Work

A paper by Felix Rieger and Christoph Bockrish [4] concludes a summary of different studies. They reviewed existing research on green software design and assessment methods, such as Silicon Labs and SEFlab, which are contained in this paper.

Andreas Schuler and Gabriele Kotsis [5] analyzed event-based, utilization-based, code-analysis, and measurement-based methods for mobile platforms like Android or iOS, thereby reviewing individual system parts' energy consumption. Within their study, they categorized existing methods and stated problems that must be solved. In total, they reviewed 134 studies between 2011 and 2021, most of them applying the Android platform.

3. Energy Measurement Methods

The following section introduces the different methods for analyzing software energy consumption. The passage will describe the mode of operation and the functional framework of these individual methods and also state some of their limitations.

3.1. Intel RAPL

Intel RAPL allows the user access to sensors inside the CPU, allowing the CPU's and DRAM's accumulated power consumption to be distinguished. Intel introduced the method with their Sandy Bridge lineup [6], [7]. The information is stored in Machine-Specific Registers (MSRs) [1], [7], [8]. Rather than capturing physical measurements, these registers store architectural events from the cores, processor graphics, and I/O, which are processed with energy weights to estimate the active power consumption of the package [9]. The collected consumptions are displayed in Joules and are updated on average every millisecond, therefore, the granularity is 15.3 µJ for SandyBridge [6]–[8] and 61 µJ for Haswell and Skylake architectures [7].

Table 1 shows the different relevant storing registers. With Intel RAPL, CPU package power, the total consumption of the processor cores and the consumption of the

 doi: 10.2313/NET-2025-05-1_01

TABLE 1: List of available RAPL sensors, Table 1 in [1]

RAPL_PKG	Whole CPU package
RAPL_PP0	Processor cores only
RAPL_PP1	*A specific device in the uncore*
RAPL_DRAM	Memory Controller

DRAM controller can be measured. A significant disadvantage of Intel RAPL is that there is no possibility of measuring the power consumption of individual cores [1].

Intel RAPL has severe security issues, as explored by Z. Zhang et al. [10]. Especially on Linux systems, unprivileged users can read the measurements offered by Intel RAPL through the "sysfs" interface. Furthermore, the same can be done on MacOS with specialized system calls.

They analyzed the memory power consumption using DRAM access procedures. With an AVX system call, they stored data in the DRAM and measured the energy consumption. They discovered that writing small segments consumes less power than writing larger segments. Then, they implemented a receiver and a sender, which can be placed, for example, one in the container and one in the management system. With some adjustments, they established a covert channel that can transmit 0 and 1 through those energy measurements while bypassing all the security implementations. On their testing systems, they achieved a bandwidth of 50 bps while maintaining an error rate below 2% on every system [10].

3.2. AMD Energy

In comparison, AMD Application Power Management (APM) can measure the energy consumed by each core and the total socket power. The socket power measurements differ from those of Intel's package power because it is not the sum of all cores but includes cache and other CPU internal parts. In contrast to Intel RAPL, it does not provide the consumed energy in Joules, but the average consumption over the last timeframe. On a system with AMD Opteron 6274, a timeframe was about 3.8 ms long and results in a granularity of 3.8 mW. Only the information of the last segment is stored in the registers. This approach is more accurate than Intel RAPL when measuring microscopic procedures because considering two timeframes instead of one does not have a tremendous impact. The disadvantage of AMD Energy is that no power measurement of the DRAM is possible [6].

3.3. SEFlab

Ferreira et al. [11] conducted further investigations to create the SEFlab, a hardware-based black-box measuring lab. It is especially suitable for processors produced before the introduction of Intel RAPL.

They tried to get exact measurements of both CPUs, memory, fans, mainboard, and HDD. With some additional testing, they could distinguish each wire to their consumer except the power for memory banks and fans because those are distributed directly on the motherboard. To address this problem, they measured the power consumption of the memory and fans and subtracted it from the measurements of the mainboard results. All these measurements were collected with a sampling frequency of 30kHz and then stored and processed in the data acquisition system (DAQ). The DAQ transmits the data to the measurement PC, where the data is visualized. Furthermore, they extracted a pulse via USB from the server and inserted it via a serial port. With that, they could get exact measurements when the software is executed on the server. Bram Visser did a validation analysis of SEFlab and discovered an error margin around 1% [11]. SEFlab is still an experimental approach that requires much work to adapt to other hardware.

3.4. eLens

A concept for estimating energy consumption is eLens. It combines *per-instruction energy modeling* and *program analysis* to trace executed paths. eLens bases the estimations on bytecode. The conversion process to bytecode is further introduced in Section 4 of this paper. eLens can be integrated into IDEs like Eclipse to estimate different application parts. Therefore, the developers do not need additional hardware if a SEEP described below is available. The algorithm can estimate the energy consumption per line of code, path, method, and application [12].

For eLens to work correctly, a software environment energy profile (SEEP) is needed. The SEEP contains the per-instruction energy costs of every hardware component in the target machine. The researchers of eLens hope that manufacturers will publish SEEPs of their products in their future. Since SEEPs are currently not available, the researchers generated their own SEEP with the LEAP setup [12].

The *Low Energy Aware Platform* (LEAP) is a testbed invented at the University of California, Los Angeles. It contains an Intel Atom CPU on a mini-ITX motherboard. With a Digital Acquisition Device (DAQ), they can sample at a rate of up to 10kHz. The system can be further adapted to measure all parts necessary for various application types. In cooperation with its nanosecond timestamp counter, it can report fine-grained energy consumption results while executing a task [13].

eLens contains three main parts, as shown in Figure 1. The Workload Generator converts the possible user interactions into path information, which can be processed further in the Analyzer and Source Code Annotator. The generator is needed, because it would be inaccurate if all paths were executed n times. The Analyzer then receives the paths from the workload generator and assigns each path's matching energy estimates extracted from the SEED. Afterwards, the Source Code Annotator visualizes the results from the Analyzer, which can then be integrated via the Eclipse plugin. For example, the single lines of code are highlighted in blue for lower consumption and then go up in steps to red representing enormous consumption [12].

3.5. GreenOracle

GreenOracle is an energy estimation software for Android applications. It uses machine learning based on a big data approach; the creation of the dataset is further

 doi: 10.2313/NET-2025-05-1_01

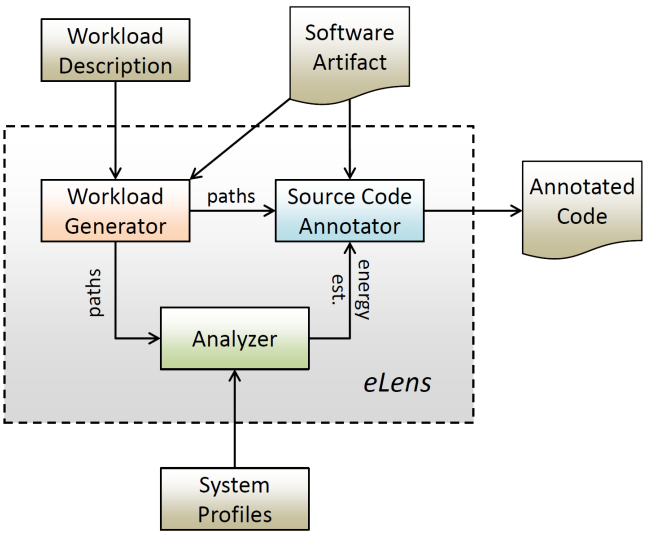


Figure 1: Structure of eLens, Fig. 1 in [12]

detailed in the accuracy and precision section of this paper. The software then calculates the energy consumption based on grouped system calls, processes with similar functions and energy consumption. With the fast execution and easy adaptation to new versions, app developers can assess energy-consuming parts of the code in progress. The results are promising, and it is universally applicable, while the achieved results are comparable to those of eLens. Nevertheless, GreenOracle is easy to use and does not require specialized hardware [14].

3.6. Silicon Labs Energy Profiler

In the world of "Internet of Things" (IoT), many household devices are connected to servers worldwide. Microcontrollers are needed to enable such functionality, which can manage and connect these devices while using only a little power. Silicon Labs provides a complete working environment with in-house software and hardware, which is available on the market. With their cooperating hardware the developer can assess the energy consumption of software since the 5th version of the Silicon Labs Studio. They manufacture 32-bit ARM Cortex cores, which can be programmed via the Silicon Labs SDK afterward [15].

The microcontroller EFM8 and EFM32 have a compatible debug interface which can be connected to a Silicon Labs starter kit containing a power supply. With the Advanced Energy Monitor (AEM) interface included in the SDK, the energy consumptions can be displayed within the IDE. The development environment has integrated the Silicon Labs Energy Profiler (SELP); therefore, multiple devices can be measured and compared simultaneously.

The visual user interface efficiently displays the live energy consumption with a waveform. The AEM interface provides a single-node and a multi-node view, where the user can see the animated live consumption. This presentation method allows the developer to efficiently assess all the needed information, make changes to the code, and measure again. Furthermore, the interactions between several devices can be measured and visualized. The system is modular overall and helps the developer getting complex measurements done in seconds, saving many work hours; e.g., the software allows the user to look into variable timeframes and modules, providing complete control and customization.

The IDE can manage the software autonomously, so the developer can start, pause, and end the measured software directly inside the development studio. Furthermore, recording the measurements is possible for later analysis, and the user can enable code correlation, whereby the software can assign the energy consumptions to each function because both code analysis and energy measurement run in parallel [16].

3.7. CloudSIM

Cloud SIM 7G is the 7th version of this open-source, java-based simulation tool. The system is not designed to assess the energy consumption of individual software but primarily to determine how to distribute the software most efficiently. The developers can simulate energy consumption of, for example, network components in geographically distributed systems while recreating the network traffic of their software. The developers of CloudSIM have integrated the CloudNetSIM++ software from OMNeT++ for this purpose. Using this, researchers can compare performances of, for example, star and mesh topology while connecting the data centers. It can manage TCP, UDP, and HTTP traffic. The system then outputs the calculated energy consumption per data center or hardware component type. With the extension ERouter, the consumption of switching and routing appliances can be assessed. It is an extensive system with an easy-to-use GUI (graphical user interface) that enables researchers and developers to simulate, optimize, and assess their software [17], [18].

3.8. AWS CloudWatch

AWS (Amazon Web Services), Google Cloud, and Microsoft Azure implement several proprietary measurement methods [19]–[21]. For example, AWS announced CloudWatch, allowing developers to see all system components' current hardware utilization. These approaches cannot measure absolute energy values but give the developers an indication of how energy-efficient those methods are compared to other software versions [19]. AWS CloudWatch works on all AWS EC2 systems, which all run Intel, AMD and proprietary AWS Graviton processors [22].

4. Accuracy and Precision

The following paragraphs will detail the experimental testing setups and measurement results, assessing the accuracy and precision of Intel RAPL, SEFlab, eLens, and GreenOracle.

The measurements of Intel RAPL improved from Sandy Bridge-EP to Haswell-EP [8]. The testing group of Khan et al. [7] did extensive research on the accuracy of Intel RAPL power measurements. They established a testing setup with an Intel Core i7-4770 @ 3.40Ghz, a Haswell workstation CPU, and an Intel i5-6500 @ 3.20Ghz, a Skylake Desktop CPU. Their analysis used Microbenchmarks, which address only a specific part of

the CPU, and application-level benchmarks like Stream or ParFullCMS. The benchmark Stream showed a strong correlation (coefficient of 0.99) between the RAPL package power, and the power drawn from the wall socket. It should be noted that this is only feasible at constant temperatures. Especially when testing the Haswell CPU, they observed a significant impact on longer benchmarks when the temperature is rising. They measured a correlation of 0.93 between package power and temperature reading. Therefore, if measuring with RAPL technology, the developers should remember that all the tests should have comparable core temperatures. Overall they estimated a mean error of 4% for Sandy Bridge and 1.7% for Haswell CPUs [7].

The accuracy of the SEFlab cannot be distinguished precisely because of the lack of testing different hardware. However, in their testing lab, they could measure very accurate results on runtime. In future work, it will also be possible to get precise predictions when enough data is collected [11].

The researchers compared the accuracy of eLens by comparing the measurements of the ground trough (GT) metered with LEAP and eLens. First, they downloaded unmodified Android applications from the Google Play Store and afterwards converted the Dalvik bytecode to Java bytecode using the *dex2jar tool*. Some applications cannot be transformed and were excluded from their validation process. Furthermore, the code has to run on the LEAP Platform and during the path measuring process, 0.01% of all paths threw an exception. All the remaining applications were given to eLens as input. They compared the estimations of eLens with the measured GT, but different problems occurred during this process. The GT counted waiting times on human input; the LEAP could not determine which energy consumption was just background noise, and the LEAP had only a sampling rate of 10 kHz. The rate is just enough to measure functions that run longer than 10 ms. This lack resulted in many functions where the GT cannot be distinguished. Therefore, they did not compare the measurements at the line of code granularity. The average error for the whole program level was 8,8% but was consistently below 10%. At the method level, the average was 7.1% and also below 10% in any case. For the hardware components, RAM and WiFi, they got a maximal error of 12% and one measurement with GPS, where an error of 8.1% occurred. These are excellent results in comparison to other software estimation methods. For example, the *average-bytecode* strategy at the whole program level had an average error of 133% and the *no-path-sensitivity* analysis 267%. Primarily because not only the CPU consumption is assessed, but also other hardware components can be included in the calculation [12].

To get high accuracy, the developers of GreenOracle collected 24 different Android applications with a total of 984 versions from *F-Droid* [23], *GitHub*, and partly from a direct website. Then they used the *Green Miner* [24], a hardware-driven energy profiler consisting of a Raspberry Pi, a Galaxy Nexus phone, and an Arduino Uno. The Raspberry Pi executes the tests on the phone and stores the measured data, while the Arduino Uno collects the energy consumption of the phone. The predefined tests consist of standard usage of the app and the user inputs are emulated with Unix shell. They executed all tests in airplane mode. After repeating each test 10 times, they collected all system calls with the *trace* command in several independent tests. Finally, they grouped similar system calls and created a table of only 13 different system calls. With advanced machine learning, they achieved an average error of 5.96% using super vector machine regression (SV) because other regression types generated worse results. However, they recommend ridge regression because the worst case is much better than SV regression. That concludes with a mean error of 6.17% and a worst-case error of 13%. These results are comparable to eLens stated above [14].

5. Error Analysis

During the testing of SEFlab, they encountered some contradictions with similar experiments, for example, an experimental analysis by H. Chen, S. Wang, and W. Shi. [11], [25]. They did not assess these issues within the paper. Furthermore, they did not address the impact of rising temperatures of the SOC when running more extended tests. They stated in the paper that CPU usage is highly correlated with its power draw. Figures 2 and 3 show that the utilization is initially at 100%, but the power draw is not at its peak. They explained this by *apparent contradictions* in their benchmarks and the *lack of scalability* reported in [11], [26]. For mobile usage, many processors are designed to be highly efficient at idle but are inefficient at peak performance. This structure is necessary because, on mobile devices, the idle time is much longer than the time when the peak performance is needed [26]. Another point that may impact the findings is that energy density on the internal head-spreader (IHS) rose over several chip generations. For example, a Xeon CPU from 2004 has a TDP of 103W on a heat spreader with a size of 42.5 mm x 42.5 mm (1806.25 mm^2) [27]. In contrast, a last-generation Intel Core like the 14900K has a TDP of 125W and a turbo of up to 253W on a 45 mm x 37.5 mm (1687.5 mm^2) IHS [28]. That is an increase of 29.9% per mm^2 of energy density. This increase can cause a higher temperature range of the SOC and, therefore, different scalabilities because the resistance of the SOC rises with rising temperatures and consumes more energy for the same performance [29].

6. Conclusion and Future Work

As shown in the review, there are many ways to assess software energy consumption. Nevertheless, to this date, no method has perfect accuracy and is easy to adapt to all scenarios. The hardware-based methods described in the first part are more precise than software-based approaches regarding runtime measurements. However, these hardware-based methods are time-consuming, and it is challenging to distinguish between total system power usage and power usage caused by the software. The software-based estimation methods are easy to adapt but have some uncertainties but still are suitable for more applications. eLens and GreenOracle are not yet available to standard developers and still need more straightforward integration with state-of-the-art software development sys-

TABLE 2: Overview of methods mentioned in the paper

Method	Platform Compatibility	Error	
INTEL RAPL	Only Intel CPUs since Sandy-Bridge + DRAM (since Sandy-Bridge Server)	4% (Sandy-Bridge) 1.7% (Haswell)	[6], [7]
AMD APM	Only AMD CPUs since 15h generation	No measured values	[6]
SEFlab	AMD & Intel CPUs as above + additional hardware components	~1%	[11]
eLens	Android based CPU + RAM, GPS, WiFi	< 10%	[12]
GreenOracle	Android based CPU	< 13%	[14]
SL E. Profiler	SL Microcontroller	No measured values	[16]
CloudSIM	distributed systems	No energy values	[17], [18]
AWS CloudWatch	AWS EC2 instances	No energy values	[19]

Note: In this table, SL stands for Silicon Laboratories.

tems. CloudSIM provides the most advanced, freely available technology in this paper, but the system has many possible future improvements; ideas could include running individual software in simulation, which would be analyzed in a manner comparable to eLens or GreenOracle. Big cloud distributors have advanced proprietary monitoring tools to shorten costs but they do not provide absolute energy values.

In conclusion, much work is needed in this segment to overcome the current boundaries, but more advanced technologies are being developed every year. Platform-independent and easy-to-use methods are not available to date. Only Intel RAPL, Silicon Labs Energy Profiler, CloudSIM, and AWS CloudWatch are commonly used in the industry. Especially for network applications, Intel RAPL is used because this method can deliver precise measurements, and most test setups in universities are Intel x86-based. However, more software-based methods with hardware integrations like eLens are also coming, making analyzing energy consumption much easier.

References

[1] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, "Measuring energy consumption for short code paths using rapl," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 3, p. 13–17, jan 2012.

[2] *Climate Change 2023: Synthesis Report (Full Volume) Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, 07.

[3] Statista Research Department, "Umsatz mit cloud computing** weltweit von 2010 bis 2023 und prognose bis 2025." [Online]. Available: https://de.statista.com/statistik/daten/studie/195760/umfrage/umsatz-mit-cloud-computing-weltweit/

[4] F. Rieger and C. Bockisch, "Survey of approaches for assessing software energy consumption," in *Proceedings of the 2nd ACM SIGPLAN International Workshop on Comprehension of Complex Systems*, ser. CoCoS 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 19–24. [Online]. Available: https://doi.org/10.1145/3141842.3141846

[5] A. Schuler and G. Kotsis, "A systematic review on techniques and approaches to estimate mobile software energy consumption," *Sustainable Computing: Informatics and Systems*, vol. 41, p. 100919, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210537923000744

[6] D. Hackenberg, T. Ilsche, R. Schöne, D. Molka, M. Schmidt, and W. E. Nagel, "Power measurement techniques on standard compute nodes: A quantitative comparison," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 194–204.

[7] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "Rapl in action: Experiences in using rapl for power measurements," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, no. 2, mar 2018.

[8] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, "An energy efficiency feature survey of the intel haswell processor," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, 2015, pp. 896–904.

[9] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, 2012.

[10] Z. Zhang, S. Liang, F. Yao, and X. Gao, "Red alert for power leakage: Exploiting intel rapl-induced side channels," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 162–175. [Online]. Available: https://doi.org/10.1145/3433210.3437517

[11] M. A. Ferreira, E. Hoekstra, B. Merkus, B. Visser, and J. Visser, "Seflab: A lab for measuring software energy footprints," in *2013 2nd International Workshop on Green and Sustainable Software (GREENS)*, 2013, pp. 30–37.

[12] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan, "Estimating mobile application energy consumption using program analysis," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 92–101.

[13] P. A. Peterson, D. Singh, W. J. Kaiser, and P. L. Reiher, "Investigating energy and security trade-offs in the classroom with the atom {LEAP} testbed," in *4th Workshop on Cyber Security Experimentation and Test (CSET 11)*, 2011.

[14] S. A. Chowdhury and A. Hindle, "Greenoracle: Estimating software energy consumption with energy measurement corpora," in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, 2016, pp. 49–60.

[15] Silicon Laboratories, "Silicon labs 32-bit microcontrollers." [Online]. Available: https://www.silabs.com/mcu/32-bit-microcontrollers

[16] ——, "Silicon labs energy profiler." [Online]. Available: https://docs.silabs.com/simplicity-studio-5-users-guide/1.0/using-the-tools/energy-profiler/

[17] R. Andreoli, J. Zhao, T. Cucinotta, and R. Buyya, "Cloudsim 7g: An integrated toolkit for modeling and simulation of future generation cloud computing environments," *arXiv preprint arXiv:2408.13386*, 2024.

[18] A. W. Malik, K. Bilal, K. Aziz, D. Kliazovich, N. Ghani, S. U. Khan, and R. Buyya, "Cloudnetsim++: A toolkit for data center simulations in omnet++," in *2014 11th Annual High Capacity Optical Networks and Emerging/Enabling Technologies (Photonics for Energy)*, 2014, pp. 104–108.

[19] I. Amazon Web Services, "Amazon cloudwatch." [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/finding_metrics_with_cloudwatch.html

[20] G. LLC, "Google cloud observability." [Online]. Available: https://cloud.google.com/monitoring/docs/monitoring-overview

[21] M. Corporation, "Microsoft azure monitor." [Online]. Available: https://learn.microsoft.com/de-de/azure/azure-monitor/best-practices-data-collection

[22] I. Amazon Web Services, "Amazon cloudwatch." [Online]. Available: https://aws.amazon.com/de/cloudwatch/

[23] F.-D. Contributors, "F-droid: Free and open source android app repository." [Online]. Available: https://f-droid.org/

[24] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, and S. Romansky, "Greenminer: a hardware based mining software repositories software energy consumption framework," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 12–21. [Online]. Available: https://doi.org/10.1145/2597073.2597097

[25] H. Chen, S. Wang, and W. Shi, "Where does the power go in a computer system: Experimental analysis and implications," in *2011 International Green Computing Conference and Workshops*, 2011, pp. 1–6.

[26] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[27] I. Corporation, "64-bit intel® xeon® processor 3.20 ghz, 1m cache, 800 mhz fsb." [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/28016/64-bit-intel-xeon-processor-3-20-ghz-1m-cache-800-mhz-fsb.html

[28] ——, "Intel® core™ i9 processor 14900k." [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/236773/intel-core-i9-processor-14900k-36m-cache-up-to-6-00-ghz.html

[29] R. Chu, R. Simons, M. Ellsworth, R. Schmidt, and V. Cozzolino, "Review of cooling technologies for computer products," *IEEE Transactions on Device and Materials Reliability*, vol. 4, no. 4, pp. 568–585, 2004.

The Potential of QUIC for Web Proxies Accelerating REST APIs

Kevin Fischer, Markus Sosnowski*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: kevin.fischer@tum.de, markus.sosnowski@tum.de

***Abstract*—The increasing demand for faster and more secure web services have driven significant advancements in networking protocols. This paper explores the potential of QUIC, a modern transport protocol, to accelerate web proxies and REST APIs, which are critical components in today's Internet architecture. We provide an overview of the REST (Representational State Transfer) paradigm and web proxies. Aswell, we explore basic and in-development features of QUIC and related work. The paper then analyses the problems, requirements and challenges of REST APIs, Web Proxies, and Content Delivery Networks and maps features of QUIC to them to show their potential for acceleration. The paper finds that QUIC can enhance the performance, security and reliability of REST APIs served through web proxies and CDNs. Still, there are connections between QUIC, REST APIs, and web proxies that need further research.**

***Index Terms*—quic, content delivery networks, rest, web proxies**

1. Introduction

In an era where web traffic is growing rapidly, the need for reliable, fast, and secure networking protocols has never been more critical. The most widely adopted solution for ensuring secure web communication is the Hypertext Transfer Protocol Secure (HTTPS), which integrates the Transmission Control Protocol (TCP) with Transport Layer Security (TLS). While HTTPS was and still is the standard for secure web traffic, its reliance on TCP has introduced several challenges and problems, particularly with the increasing and changing needs of modern web applications. To address these challenges, Google proposed QUIC in 2016 for standardization, among other things, with the goal of enhancing the performance and security of web communications beyond what TCP could traditionally offer [1].

When considering mechanisms for client-server communication, REST (Representational State Transfer) or RESTful APIs have become the predominant paradigm. REST outlines architectural requirements that APIs (Application Programming Interfaces) must follow [2]. In many implementations, requests to REST APIs are routed through web proxies. These web proxies can provide several advantages, such as improved security, reduced latency, and enhanced performance [3]. Given these benefits, the role of web proxies in optimizing the performance of REST APIs is of significant interest.

This paper explores the potential of QUIC to accelerate REST APIs, specifically in scenarios in which web proxies are involved. We aim to analyze how the features of QUIC can address the challenges and requirements of REST APIs when served by web proxies, thereby offering a more efficient and secure solution.

The remainder of this paper is structured as follows: Section 2 provides background information about QUIC, REST, Web Proxies, and CDNs (Content Delivery Networks), Section 3 presents related work and sets this paper apart from others. In Section 4, we analyze the problems, challenges, and requirements of REST APIs served by Web Proxies. In Section 5 we then map the features of QUIC to the problems that we found in Section 4. The paper concludes with a summary of our findings in Section 6.

2. Background

This section gives an overview of the relevant concepts for the following problem analysis and mapping of features to the problems.

2.1. QUIC Protocol

QUIC is a transport protocol, which was developed by Google and later standardized by the Internet Engineering Taskforce (IETF) [1]. Unlike traditional protocols like TCP in combination with TLS, QUIC is built on top of the User Datagram Protocol (UDP) and contains features that aim to reduce latency, enhance security and improve the overall efficiency of web communication.

2.1.1. Key Features of QUIC. QUIC introduces several features that differentiate it from the traditional TCP+TLS combination:

- **1-RTT and 0-RTT Handshakes:** While TCP requires at least one round-trip time (1-RTT) for the handshake and TLS needs 1-RTT as well, QUIC combines these steps [4]. Figure 1 shows how QUIC establishes a secure connection with a single round-trip. Moreover, QUIC supports 0-RTT handshakes for repeated connections, thereby data can be sent immediately [5].
- **Built-in Encryption:** QUIC mandates encryption by default and uses TLS 1.3 for secure communications [6]. From the start of the communication, the built-in encryption provides confidentiality, integrity, and authenticity. As a result, QUIC does

 doi: 10.2313/NET-2025-05-1_02

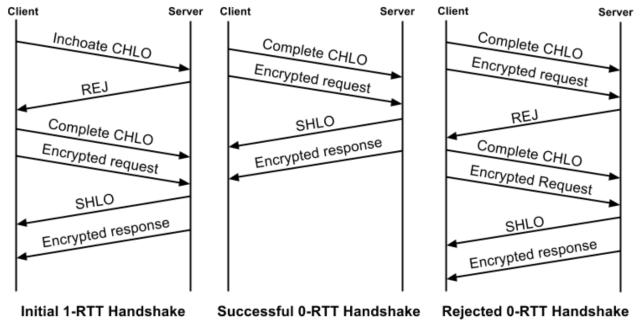


Figure 1: Timeline of QUIC's initial 1-RTT handshake, a subsequent successful 0-RTT handshake, and a rejected 0-RTT handshake [5].

not need an extra encryption layer like TLS on top of TCP.

- **Connection Migration:** QUIC allows to migrate a connection and continue it seamlessly when a client's IP address changes e.g. when switching from Wi-Fi to cellular data. While TCP uses a 4-tuple, QUIC utilizes a connection ID to identify a connection from a client to a server [7].
- **Stream Multiplexing:** QUIC can handle multiple streams of data within a single connection without the issue of head-of-line blocking, which occurs in TCP when packet loss in one stream can delay the delivery of data in other streams [8]. By using different `StreamIDs`, packet loss does not affect all streams, but only streams that are in the lost packet [9]. Figure 2 illustrates how QUIC behaves in comparison to TCP when encountering packet loss while having multiple streams.

The following two are extensions to QUIC that are particularly relevant for scenarios with a web proxy layer and CDNs. There is still ongoing work and development on these extensions:

- **MASQUE (Multiplexed Application Substrate over QUIC Encryption):** MASQUE is an extension for QUIC which allows to multiplex traffic of different applications with a single QUIC connection. It can create a secure tunnel to a proxy and because of that, it is particularly useful for applications like VPNs or proxy services with multiple streams, since the streams can be handled securely within one connection [10], [11]. An example scenario with a proxy server is demonstrated in Figure 3.
- **QUIC-Aware Proxying:** QUIC-Aware Proxying allows clients to tunnel QUIC connections and adds a new "forwarded" mode as stated in [12]. It allows to use special-purpose transforms rather than full re-encapsulation and re-encryption of QUIC packets when they are forwarded by a proxy.

2.2. REST APIs

REST is a paradigm that outlines principles for building scalable and stateless web services. It was first introduced by Roy Fielding in 2000 in [2], since then REST has become the dominant framework for designing APIs [14].

2.2.1. Key Principles of REST. As defined by Roy Fielding in [2], REST is based on the following principles:

- **Stateless:** Each request from a client to a server has to contain all necessary information, as the server does not memorize the client state between requests.
- **Client-Server Architecture:** REST separates concerns by defining distinct roles for clients (handling user interaction) and servers (managing data and logic).
- **Uniform Interface:** A consistent and standardized interface is used. Typically HTTP methods (GET, POST, PUT, DELETE) are used to interact with resources.
- **Code on Demand:** Web servers can send executable programs to clients. Oftentimes communication is needed in advance to assure that the client is able to process the offered ressource.
- **Layered System:** REST systems are layered, allowing intermediaries like proxies to manage requests without clients being aware.
- **Cache:** Responses can be marked as cacheable to enable reuse of responses for identical requests, thereby improving performance.

2.3. Web Proxies and HTTP Caching

Web proxies act as intermediaries between clients and servers. They forward requests from clients to servers and vice versa. While offering several benefits, such as enhanced security, traffic filtering and more, proxies can optimize network performance by caching frequently requested (static) content [15], [16]. Thereby proxies can reduce the load on the origin server and decrease response times.

The concept of HTTP caching enables web proxies to store copies of previously fetched content. When a client requests the same resource (e.g. a static web page), the proxy can serve it from the cache instead of sending another request to the origin server, which improves speed and efficiency. Caching mechanisms can be controlled by HTTP headers like `Cache-Control` and `Expires`, which can for example configure if you want your data to be cached or not [17]. Since caching eliminates the need for repetitive requests to the server, it significantly improves loading times and bandwidth usage [3].

2.4. Content Delivery Networks

A Content Delivery Network (CDN) is a network of distributed servers around the world that deliver web content to users. They deliver content based on their geographic location, the origin of the webpage and a content delivery server [18]. The main goal of a CDN is to serve content to clients fast and reliable. Oftentimes CDNs are used to deliver static content like images, stylesheets, and scripts, but also dynamic content like API responses. When requesting content from a website using a CDN, the request is redirected to the nearest CDN server (edge

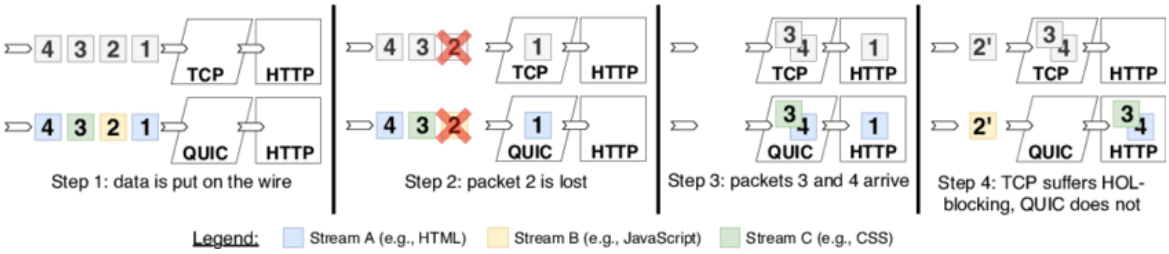


Figure 2: Head-Of-Line blocking in TCP vs QUIC [8].

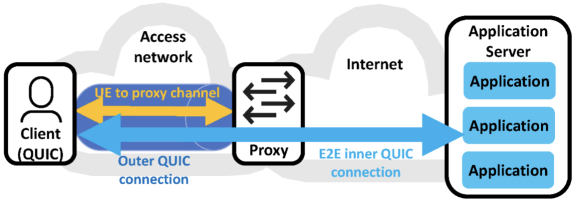


Figure 3: Proxy setup using MASQUE for QUIC-based tunneling [13], [10].

server) [19]. This redirection is typically achieved by performing AnyCast or a DNS resolution, where the domain name is mapped to the IP address that is closest to the edge server. This architecture that serves content from servers that are geographically closer to the client results in lower latency. By caching content on edge servers, CDNs can not only improve response times, but also reduce the load that the origin server has to handle [20].

3. Related Work

There are several papers that have explored the performance of QUIC's specific features. For instance, Kühlewind et al. [10] analyzed implications of MASQUE, while Cook et al. [21] examined how 0-RTT connection establishment of QUIC effects certain performance metrics in different conditions, like mobile networks. Other research provides general analyses of QUIC. Megyesi et al. [22] evaluated and compared QUIC's performance with other variants. This paper aims to provide a conceptual overview of how QUIC's features can address and improve the specific challenges of a web proxy to accelerate REST APIs.

4. Problem Analysis

This Section analyses the problems and challenges of REST APIs. These challenges come from limitations in the underlying transport protocols (most of the time TCP+TLS) or constrains of the REST paradigm.

4.1. Latency Due to Connection Establishment

REST APIs normally use HTTP over TCP, which results in a three-way handshake to establish a connection [23]. When adding TLS for encrypting data, additional two RTTs are needed for the handshake process before any data can be sent, although TLS 1.3 provides new features that allow a faster connection establishment [24]. This sequential handshake procedure results in latency, particularly bad for applications that require quick/real-time interactions. These circumstances imply an increased response time, where each connection results in a delay that affects user experience and general application performance. In scenarios with a web proxy layer, this additional layer could potentially increase latency and impact performance of the REST API.

4.2. Head-of-Line Blocking

HTTP/2 over TCP suffers from Head-of-Line Blocking due to in-order delivery of packets as illustrated in Figure 2 and explained in [21]. This phenomenon can cause delays in delivery of data, hence affecting the performance of a REST API that relies on punctual responses. In certain scenarios this phenomenon implies reduced throughput, longer wait times of API responses and especially has an impact on concurrent requests where multiple API requests are sent over a single TCP connection.

4.3. Connection Migration Challenges

REST APIs are frequently accessed by mobile clients that are likely to switch their network interfaces from time to time, such as from Wi-Fi to cellular data. Because of that the client's IP address can change during active sessions. Traditional TCP connections, identify a connection via the client's IP address and a port number, hence any change in the network interface leads to the termination of the TCP connection. If a client wants to request again from a REST API, it is necessary to establish a new connection. This process introduces additional latency due to the required TCP handshake and TLS negotiation. In scenarios with a web proxy layer or CDN is, this issue is further extended due to the additional layers of network traversal and new connections that need to be managed. Vargas et al. [25] highlights that a significant portion of API requests in CDNs consist of small, frequent JSON messages from mobile clients. The inability of TCP to handle connection migration efficiently leads to an increased latency and lower performance for these clients. This phenomenon implies reduced throughput and longer wait times for API responses.

4.4. Security Overhead

REST APIs typically use TLS to encrypt data, thereby ensuring confidentiality and integrity [23]. The overhead of encrypting and decrypting data consumes CPU resources of both clients and servers [4]. This can lead

to increased response times and reduced throughput in scenarios with many concurrent connections and high load. This overhead is particularly significant for REST APIs with frequent and small requests, where the relative cost of encryption / decryption is high compared to the actual payload size. Proxies that inspect or cache content often terminate TLS connections and act as intermediaries between client and server. This requires separate TLS handshakes for each leg, introducing additional latency and computational load.

4.5. Inefficient Multiplexing

REST APIs often utilize HTTP/2 to enhance performance through multiplexing, which allows multiple requests and responses to be sent concurrently over a single TCP connection, as described in [8]. But since HTTP/2 operates over TCP, packets must be delivered in order. This leads to head-of-line blocking, where the loss of a single packet delays all other packets until the lost packet is retransmitted. This limitation reduces the effectiveness of multiplexing and can cause delays in delivering API responses.

4.6. Caching

Cache validation mechanisms which use headers like `Cache-Control` and `ETag` often require additional RTTs for clients to confirm data freshness, increasing latency [26]. The statelessness of REST can lead to redundant data transmission (all meta informations need to be transfered) [27]. Encryption adds further complications, since the TLS connection between client and server is terminated at the proxy or CDN. This requires the proxy to decrypt and inspect the data, which can be computationally expensive and introduce additional latency.

5. Mapping QUIC Features to Problems

In this section, we map the features of QUIC introduced in Section 2 to the problems and challenges of Section 4. By mapping QUIC's features to the challenges of REST APIs served through web proxies and CDNs, we show how QUIC can enhance performance, security, and reliability.

5.1. Reducing Latency Due to Connection Establishment

QUIC combines the transport protocol and encryption layer. It reduces the number of round trips required to establish a secure connection. For new connections, QUIC can establish a secure connection in 1-RTT, and for repeat connections, it can achieve 0-RTT connection establishment, allowing data to be sent directly [1], [5]. By reducing the connection establishment latency, QUIC improves the responsiveness of REST APIs, especially in situations where short-lived connections to APIs via web proxies or CDNs are made.

5.2. Eliminating Head-of-Line Blocking

QUIC eliminates head-of-line blocking by implementing stream multiplexing at the application layer over UDP, which does not enforce in-order delivery. Streams of QUIC are independent, which means that the loss of packets in one stream does not affect packets in other streams [8]. As a result, REST APIs are able to handle multiple paralell requests more efficiently, with reduced latency and improved throughput.

5.3. Handling Connection Migration Challenges

QUIC connections are identified by a `ConnectionID` instead of a 4-tuple consisting of both IP addresses and ports [7]. This abstraction allows the connection to stay active even if the network address of the client changes. As a result, clients can seamlessly continue their sessions without the overhead of reconnecting and renegotiating cryptographic parameters. This feature is useful for improving the reliability and performance of REST APIs that are accessed by clients with mobile devices.

5.4. Addressing Security Overhead

By mandating encryption and including it into the transport layer, as described in [6], QUIC simplifies the security model. Additionally, QUIC's handshake process contributes to faster and more efficient secure connections. This improves security for REST APIs served via web proxies or CDNs. Apart from that, QUIC encrypts most of the header information, hence there are less possibilities for attacks, where attackers try to learn from the header information.

5.5. Multiplexing Efficiency

QUIC stream multiplexing allows REST APIs to handle multiple paralell requests with a single connection [9]. The independent streams prevent delays because of packet loss in one stream from affecting others, resulting in higher throughput and lower latency. This is beneficial for web proxies and CDNs that manage high volumes of simultaneous API requests.

5.6. Caching and QUIC

QUIC over HTTP/3 supports HTTP caching mechanisms. Eventhough QUIC encrypts most of the transport layer headers for security, HTTP/3 allows necessary header fields to remain accessible to intermediaries for caching [28]. Nevertheless, the encrypted nature of QUIC shows challenges for transparent caching proxies that rely on inspecting unencrypted headers.

5.7. CDN Integration with QUIC

CDNs can be improved with QUIC's features, such as connection migration. This persistence can reduce latency and improve the user experience. QUIC's handling of multiple streams and reduced connection establishment times allow CDNs to deliver content more effectively [20].

For instance, QUIC allows edge servers send content with lower latency and higher throughput, which is especially beneficial for REST APIs that rely on CDN infrastructure to reach a global audience. MASQUE and QUIC-Aware Proxying can also improve CDNs, since these functions are only partially available with the traditional TCP+TLS stack.

5.8. MASQUE and QUIC-Aware Proxying for Improved Proxy Performance

MASQUE allows clients to tunnel to proxies using QUIC and enable multiplexed traffic over a single QUIC connection [11]. This is useful for things like VPNs or proxy services that handle multiple streams within one connection. QUIC-Aware Proxying allows clients to tunnel QUIC connections or forward packets through an HTTP/3 proxy without terminating the connection at the proxy [12]. By using the extended CONNECT-UDP method over HTTP/3, the proxy can forward QUIC packets between the client and the target server transparently. The result is a reduced overhead associated with terminating and re-establishing connections at the proxy, but also the inability of CDNs to optimize things like caching.

6. Conclusion

This paper explored the conceptual potential of QUIC to accelerate REST APIs served through web proxies and CDNs. The paper identified key challenges / problems of REST APIs that mostly stem from traditional TCP-based protocols. Among these problems were latency from connection establishment, head-of-line blocking and connection migration issues. We mapped these problems to QUIC's features like including reduced handshake times, built-in encryption and connection migration.The paper demonstrated how QUIC can enhance performance, security and reliability. Adopting QUIC for REST APIs offers improvements, paving the way for faster and more secure web services that meet the evolving demands of today's internet infrastructure. Nevertheless there are still connections between QUIC, REST and web proxies that need further research, for example weither QUIC can be beneficial for certain properties of REST like uniform interfaces and statelessness.

References

[1] M. Bishop, "HTTP/3 and QUIC: Past, Present, and Future," https://www.akamai.com/blog/performance/http3-and-quic-past-present-and-future, 2021, [Online; accessed 29-August-2024].

[2] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[3] R. T. Fielding, M. Nottingham, and J. Reschke, "HTTP Caching," 2022, [Online; accessed 14-September-2024]. [Online]. Available: https://www.rfc-editor.org/info/rfc9111

[4] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The Cost of the "S" in HTTPS," 2014.

[5] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," 2017.

[6] M. Thomson and S. Turner, "Using to Secure QUIC," https://www.rfc-editor.org/info/rfc9001, 2021, [Online; accessed 29-August-2024].

[7] L. Tan, W. Su, Y. Liu, X. Gao, N. Li, and W. Zhang, "Proactive Connection Migration in QUIC," 2021.

[8] R. Marx, T. De Decker, P. Quax, and W. Lamotte, *Resource Multiplexing and Prioritization in HTTP/2 over TCP Versus HTTP/3 over QUIC*, 11 2020, pp. 96–126.

[9] T. Viernickel, A. Froemmgen, A. Rizk, B. Koldehofe, and R. Steinmetz, "Multipath QUIC: A Deployable Multipath Transport Protocol," in *2018 IEEE International Conference on Communications (ICC)*, 2018.

[10] M. Kühlewind, M. Carlander-Reuterfelt, M. Ihlar, and M. Westerlund, "Evaluation of QUIC-based MASQUE proxying," 2021.

[11] D. Schinazi, "The MASQUE Proxy," Tech. Rep., 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-schinazi-masque-proxy/04/

[12] T. Pauly, E. Rosenberg, and D. Schinazi, "QUIC-Aware Proxying Using HTTP," Tech. Rep., 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-masque-quic-proxy/03/

[13] Z. Krämer, M. Kühlewind, M. Ihlar, and A. Mihály, "Cooperative performance enhancement using QUIC tunneling in 5G cellular networks," 2021.

[14] RapidAPI, "2021 State of APIs Report," 2021, accessed: Month Day, Year. [Online]. Available: https://rapidapi.com/uploads/WP_2021_Developer_Survey_So_AP_Is_Report_f3832520be.pdf

[15] A. Luotonen and K. Altis, "World-Wide Web proxies," 1994.

[16] C. BasuMallick, "What Is a Proxy Server? Working, Types, Benefits, and Challenges," 2023, [Online; accessed 14-September-2024]. [Online]. Available: https://www.spiceworks.com/tech/data-center/articles/proxy-server/

[17] "HTTP caching," https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching, [Online; accessed 14-September-2024].

[18] R. Buyya, M. Pathan, and A. Vakali, *Content Delivery Networks*. Springer Berlin Heidelberg, 2008.

[19] E. Nygren, R. Sitaraman, and J. Sun, "The Akamai Network: A Platform for High-Performance Internet Applications," 2010.

[20] Cloudflare, "What is a CDN?" https://www.cloudflare.com/de-de/learning/cdn/what-is-a-cdn/, [Online; accessed 16-September-2024].

[21] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui, "QUIC: Better for what and for whom?" in *2017 IEEE International Conference on Communications (ICC)*, 2017.

[22] P. Megyesi, Z. Krämer, and S. Molnár, "How quick is QUIC?" in *2016 IEEE International Conference on Communications (ICC)*, 2016.

[23] E. Rescorla, "RFC2818: HTTP Over TLS," 2000.

[24] ——, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8446

[25] S. Vargas, U. Goel, M. Steiner, and A. Balasubramanian, "Characterizing JSON Traffic Patterns on a CDN," in *Proceedings of the Internet Measurement Conference*, 2019.

[26] R. T. Fielding, M. Nottingham, and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching," RFC 7234, 2014, [Online; accessed 29-August-2024].

[27] M. Massé, *REST API Design Rulebook*. O'Reilly Media, Inc., 2012.

[28] M. Bishop, "HTTP/3," RFC 9114, 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9114

Impact of Post Quantum Crypto on Networking

Ashkan Hassani, Holger Kinkelin*, Filip Rezabek*
* *Chair of Network Architectures and Services*
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: ashkan.hassani@tum.de, kinkelin@net.in.tum.de, rezabek@net.in.tum.de

***Abstract*—The Transport Layer Security (TLS) 1.3, IPsec, and DNSSec protocols, fundamental components of secure global communication, rely on public-key encryption, digital signatures, and fundamental exchange mechanisms. However, the advent of quantum computing poses a significant risk to these cryptographic systems, as quantum computers have the potential to break these schemes. The development and adoption of Post-Quantum Cryptography (PQC) are critical to address this emerging threat.**

This paper will discuss the transition to PQC, focusing on standardization efforts and the challenges related to networking protocols, public key infrastructure, and hardware limitations. Potential solutions, such as hybrid cryptographic systems, will also examined.

***Index Terms*—Post-Quantum Cryptography, Digital signatures, PKI, TLS, IPsec, DNSSec, certificates, Hybrid cryptography, X.509v3**

1. Introduction

Public key cryptography is the most essential part of keeping communications secure. The security of these encryption methods, like RSA, Diffie-Hellman, and elliptic curve cryptography (ECC), relies on solving complex mathematical problems that are difficult for regular computers to solve. These algorithms underpin protocols like TLS, IPsec, and DNSSec, safeguarding much of today's internet traffic [1].

However, the development of quantum computers threatens to break the mathematical foundations on which current cryptographic systems depend. Algorithms like RSA and DH would no longer be secure against quantum attacks2.

PQC, a newly designed cryptosystem, aims to stay secure even against powerful quantum computers, protecting global communication systems. The main challenge is transitioning to PQC while keeping security, efficiency, and compatibility with current infrastructure [2].

This paper examines why PQC is needed and the standardization work by **NIST**5. It also discusses how the transition to PQC affects networking protocols6, PKI systems8, and IoT devices9, focusing on performance, latency, and computational challenges. Additionally, it covers hybrid cryptographic systems7 for backward compatibility and multivariate-based cryptography for resource-limited environments.

2. The Power of Quantum Computers

Quantum information is based on qubits, analogous to bits in classical computers, but can exist in multiple states at a time, such as superposition. Qubits can become entangled, allowing quantum computers to perform many calculations in parallel. The efficiency of quantum algorithms depends on the availability and fidelity of qubits. In noisy environments, the fidelity of qubits is reduced, which can lead to incorrect calculations. Error correction is a technique to reduce such errors in quantum computations. With lower noise levels and better error-correction capabilities, quantum computers can execute quantum algorithms with the potential to break present cryptographic systems [3], [4].

We can't fully control or scale quantum computers, but we expect significant improvements soon. As these computers get more powerful and we get better at reducing errors and noise, they can run algorithms that could break today's cryptographic systems.

Two important quantum algorithms that demonstrate the power of quantum computers in this context are Shor's and Grover's algorithms. These algorithms can threaten today's cryptosystems and make them vulnerable.

2.1. Shor's Algorithm and Its Impact on Asymmetric Cryptography

In 1994, a scientist named Peter Shor introduced a quantum algorithm that allows quantum computers to solve mathematical problems such as prime factorization and discrete logarithms, which form the basis of widely used RSA, DH, and ECC.

RSA encryption and **digital signatures** are secure because factoring the product of two large prime numbers is infeasible for classical computers. Similarly, **DH key exchange** and **ECC** rely on the difficulty of solving discrete logarithms, either in a finite field or on elliptic curves. The best-known classical algorithms for both problems run in exponential time, ensuring these cryptosystems' security as long as key sizes are sufficiently large.

However, Shor's algorithm solves these problems in **polynomial time**, which means a powerful quantum computer could factor in large integers and break RSA encryption by deriving the private key from the public key. Similarly, quantum computers could break DH and ECC by solving the discrete logarithm problem in polynomial time, compromising key exchange and authentication in these systems [5].

 doi: 10.2313/NET-2025-05-1_03

2.2. Grover's Algorithm and Its Impact on Symmetric Cryptography

While Shor's algorithm severely threatens asymmetric cryptography, Grover's algorithm affects symmetric cryptography to a lesser extent.

Lov Grover introduced an algorithm that searches for an element in an unstructured database using the principle of quantum computers. This algorithm offers a quadratic speedup from $O(2^n)$ to $O(2^{(n/2)})$ over classical methods, which can also speed up brute force attacks by reducing the security of key to half its length, which speeds up the threat vulnerability of symmetric schemes such as DES and AES.

Doubling the key size can mitigate the impact of Grover's algorithm, as larger keys increase the time required for a successful attack. For example, AES-128 would offer only 64-bit security against a quantum attack, but AES-256 would still provide 128-bit security, making it an effective defense [6].

3. Impact of Quantum attacks on Networking

Key agreements and digital signatures are fundamental for secure communication and supporting protocols like TLS, SSH, IPsec, and digital certificates. Shor's algorithm threatens these public-key cryptosystems by solving the underlying mathematical problems they rely on. This would make it impossible to securely exchange symmetric keys using DH, while digital signatures generated by ECC could be reverse-engineered, exposing private keys from corresponding public keys.

In contrast, symmetric cryptography remains more resilient against quantum attacks. Grover's algorithm weakens symmetric encryption but does not fully compromise them. Recent research from the Center for Computational Quantum Physics in 2023 shows that implementing Grover's algorithm may not be as effective in practice as thought due to noise sensitivity and hardware limitation [7].

4. Urgency of Transition

Quantum computing is advancing rapidly, with record-breaking qubit counts and noise reduction and error correction improvements. While we can not predict when a cryptographically relevant quantum computer (CRQC) will be powerful enough to break current cryptographic algorithms, the uncertainty and rapid development emphasize transitioning to PQC [8]. If public key cryptography used in protocols, like TLS or IPsec, is broken by a quantum computer, it would also render all symmetric keys (used to encrypt data transmissions) vulnerable to decryption. This section investigates several attacks that become feasible with quantum computers.

One area that would be significantly affected is **DNSSec**. Since DNSSec is based on public key cryptography, all authoritative DNS name servers that want to sign their responses must first generate a public-private key pair before any response can be signed. It uses **chain of trust** to ensure that each zone's public key is validated by its parent zone. This hierarchical system of signing and verification up to the root DNS server creates a trust relationship that underpins the integrity of DNSSec.

Once quantum Computers can run cryptoanalytic attacks on public key algorithms, this entire chain of trust will be compromised, and DNSSec can no longer protect against **DNS Spoofing** attack.

The **Harvest now, decrypt later (HNDL)** attack is another particular concern. In this case, the encrypted data is collected today to decrypt in the future once quantum computers have advanced enough to break these cryptographic systems. Sensitive information, such as long-term corporate secrets, is at risk if the transition doesn't happen as soon as possible [9].

5. NIST Standardization

The NIST PQC standardization process for renewing cryptographic standards for key exchange and digital signatures, which started in December 2017, had three rounds of evaluation. Candidates were required to meet submission requirements and minimum acceptability criteria published by NIST [10]. Each round of the standardization process employed three primary evaluation criteria: security, Cost and performance, algorithm, and implementation characteristics.

For security evaluation, NIST analyzed the resistance of algorithms to side-channel attacks, perfect forward secrecy, and multi-key attacks. They also became more focused on real-world implementation and readiness for standardization.

For Cost and performance, the focus was on computational efficiency in key generation speed, memory requirements, and code size. Key considerations included side-channel resistance, constant-time performance, and memory optimization to ensure efficient real-world deployment.

They focused on efficiency and simplicity across multiple platforms for algorithm and implementation. NIST prioritized designs resistant to side-channel attacks and compatible with protocols like TLS and IPsec. The goal was to find algorithms that could be integrated with minimal performance lost [11]–[13].

Table 4. Standardized Algorithms

Old Name	New Name	Base
CRYSTALS–Kyber	**ML-KEM**	**lattice-based**
CRYSTALS–Dilithium	**ML-DSA**	**lattice-based**
FALCON	**FN-DSA**	**lattice-based**
SPHINCS+	**SLH-DSA**	**hash-based**

PQC algorithms are categorized in four categories based on the mathematical problems they use : 1) **Hash-based** 2) **Code-based** 3) **Multivariate-based** 4) **Lattice-based**

A detailed introduction about the pqc family and specifications for algorithms can be found in the corresponding FIPS standards: ML-KEM [14], ML-DSA [15], and SLH-DSA [16]. Each standard gives thorough guidelines on implementing them and what security settings to use.

 doi: 10.2313/NET-2025-05-1_03

5.1. Module Lattice-Based Key-Encapsulation Mechanism Standard

CRYSTALS–Kyber, standardized as **ML-KEM** in [14], is a PQ algorithm for key encapsulation mechanism (KEM). It enables the secure establishment of a shared secret key between two parties in a communication, which can be used for symmetric-key cryptography. This algorithm is a lattice-based cryptography, which means it's simple, efficient, and parallelizable. Its security is based on the hardness of the Module Learning with Errors (MLWE) problem, a generalization of Learning with Error (LWE). This emphasizes provable security based on the worst-case hardness of lattice problems [17], [18].

ML-KEM consists three algorithms:

1) **Key generation**: Produces public and private key
2) **Encapsulation**: Encrypts a shared secret key using the public key
3) **Decapsulation**: Decrypts a shared secret key using the private key

ML-KEM has three possible parameter sets, which make it flexible for different use cases, depending on the level of security required [14]:

1) **ML-KEM-512**
   - Encapsulation key size: **800 bytes**
   - Ciphertext size: **768 bytes**
   - Security strength: Equivalent to **AES-128**
2) **ML-KEM-768**
   - Encapsulation key size: **1,184 bytes**
   - Ciphertext size: **1,088 bytes**
   - Security strength: Equivalent to **AES-192**
3) **ML-KEM-1024**
   - Encapsulation key size: **1,568 bytes**
   - Ciphertext size: **1,472 bytes**
   - Security strength: Equivalent to **AES-256**

5.2. Module Lattice-Based Digital Signature Standard

ML-DSA is a PQ digital signature scheme based on CRYSTALS–Dilithium [15]. Digital signatures are most effective when they are bound to a specific identity. The signer must prove they own the matching private key to connect a public key to a verified identity. In PKI, this is achieved by issuing a certificate confirming the identity and the proof of private key possession. ML-DSA provides strong Security and is resistant to attacks that attempt to forge signatures by tricking the signer. This means even if an attacker tricks the signer into signing arbitrary messages; it still wouldn't be possible to create new valid signatures for other messages. This ensures that messages can not be faked or changed, which keeps the integrity and authenticity of digital signatures intact. The Security for lattice-based signatures relies on Module Learning with Errors (MLWE) and short integer solution (SIS) problems. SIS involves finding solutions to specific types of linear equations [15], [19].

ML-DSA consists of three main algorithms: 1) **key generation** 2) **Signing** 3) **Verifying**

ML-DSA comes in three different security levels, which ensure a balance between computational efficiency and cryptographic strength. This flexibility makes ML-DSA suitable for various uses, from securing digital communications to protecting long-term sensitive data [15].

1) **ML-DSA-44**
   - Public key size: **1,312 bytes**
   - Private key size: **2,560 bytes**
   - Signature size: **2,420 bytes**
2) **ML-KEM-65**
   - Public key size: **1,952 bytes**
   - Private key size: **4.032 bytes**
   - Signature size: **3,309 bytes**
3) **ML-KEM-87**
   - Public key size: **2,592 bytes**
   - Private key size: **4,896 bytes**
   - Signature size: **4,627 bytes**

5.3. Stateless Hash-Based Digital Signature Standard

SPHINCS+, standardized in [16], is a stateless hash-based digital signature scheme. It combines two key hash-based schemes:

1) **forest of random subsets (FORS)**: A few-time signature scheme [20]
2) **eXtended Merkle Signature Scheme (XMSS)**: A multi-time signature scheme [21]

An SLH-DSA is generated by first computing a randomized hash of the message. Part of this hash randomly chooses a FORS key, and the rest is signed with that key. The signature includes both the FORS signature and the data needed to verify the FORS public key. This verification data is created using the XMSS signature.

SLH-DSA also has three internal and external functions: 1) **Key generation** 2) **Signature generation** 3) **Signature verification**

The SLH-DSA key generation process requires three unique random values: **PK.seed**, **SK.seed** and **SK.prf**. The security parameter n maybe 16, 24, or 32, depending on the parameter set, which is:

1) $n = 16$
   - SLH-DSA-SHA2-128**s/f**
   - SLH-DSA-SHAKE-128**s/f**
2) $n = 24$
   - SLH-DSA-SHA2-192**s/f**
   - SLH-DSA-SHAKE-192**s/f**
3) $n = 32$
   - SLH-DSA-SHA2-256**s/f**
   - SLH-DSA-SHAKE-256**s/f**

6. Impact of transition to PQC on networking

The transition to PQC is critical for key establishment protocols like TLS, IPsec, and DNSSec. It presents several performance, efficiency, and integration challenges into existing systems.

 doi: 10.2313/NET-2025-05-1_03

6.1. Key establishment and TLS

PQC algorithms tend to have larger key sizes and signatures than classical cryptography systems, which impacts key establishment protocols.

Current systems like TLS and IPsec rely on RSA or ECDH for key exchanges, which use small keys, 32 bytes for X25519 in ECDH. However, PQC algorithms such as ML-KEM need larger key sizes, such as 1,184 bytes for client transmissions and 1,088 bytes for server transmissions. This increase can lead to performance bottlenecks, especially for network handshake latency, bandwidth requirements, and data transmission times. These performance impacts are noticeable in real-time implementation [22], [23].

6.2. DNSSec and PKI

WebPKI and DNSSec will also face difficulties. Current signatures and public keys are small enough to fit within the DNS packet's Maximum Transmission Unit (MTU). However, with PQC algorithms like ML-DSA, the size of signatures and public keys exceeds this limit, causing packet fragmentation, which increases latency and reduces network efficiency.

Recent experiments with new cryptography algorithms for DNSSec, like Falcon-512, SPHINCS+, and XMSS, have shown that while Falcon-512 performs better in terms of packet size and resolver compatibility, it still increases TCP traffic, which might strain DNS infrastructure. Larger signatures can also cause **SERVFAIL** responses in DNS because they exceed the allowed packet size limits.

For PKI, the larger public keys and signatures could slow down certificate validation and put more pressure on servers. This can cause necessary changes to the structure of certificates [24].

6.3. Impact on Hardware

In particular, the hardware that supports the current cryptosystem may not be constrained to handle the computational demands of PQ algorithms. Devices with older hardware, such as the Internet of Things (IoT), legacy systems, and embedded devices, have limited processing power, memory, and battery life, which may need to be improved for large key sizes and more complex calculations.

Also, older network infrastructures may need more power or memory to manage such large keys and complex computations, which could necessitate hardware replacements or firmware upgrades [25].

7. Hybrid cryptography

One potential solution is the use of hybrid cryptographic systems, which allow the simultaneous use of classical and post-quantum cryptography in protocols like TLS 1.3, IPsec, and DNSSec. This ensures that the resulting shared secret will be protected as long as one algorithm remains secure. It offers backward compatibility with existing systems and allows using post-quantum algorithms alongside traditional ones. Devices that support modern cryptography can default to post-quantum algorithms, while legacy devices can continue using classical algorithms until they become insecure. This flexibility allows for a gradual transition to post-quantum security, avoiding the need for an immediate system-wide switch. However, this cryptography also makes challenges about large key sizes and signatures, which can also lead to increased latency and higher bandwidth requirements [25], [26].

8. Transition Strategies for PQC in PKI

Transitioning to PQC in PKI is a complicated process, especially with X.509 certificates, since their role has been crucial in securing web communications using protocols like TLS. There are four main methods for integrating PQC into X.509v3 certificates [27]:

- **Quantum-Safe Certificates**: Replaces traditional public key with quantum-safe. This method is compatible with the current X.509v3 format, but it requires that systems trust quantum-safe algorithms. It also may not support backward compatibility with older systems.
- **Hybrid Certificates**: That combines traditional and post-quantum public keys and signatures in a single certificate, which ensures compatibility with old and new systems. However, they increase network traffic and complexity.
- **Composite Certificates**: It is similar to hybrid certificates but without using X.509. They combine multiple cryptographic algorithms into a single key or signature, requiring the attacker to break multiple algorithms simultaneously. It is suitable for high-security environments but demands more computational power.
- **Parallel Certificate Chains**: It issues two separate certificate chains, one for traditional and one for post-quantum cryptography, for the same entity. First, the traditional system is used. After switching to the new system, the post-quantum system takes over. This way, certificate sizes stay the same, but checking everything takes more time and computing power.

9. Multivariate-based Public Key Cryptography as a Solution for IoT

Multivariate-based Public Key Cryptography (MPKC) is a potential solution for IoT devices, which often operate with limited computational power and memory. The security of MPKC is based on solving multivariate systems of quadratic equations over a finite field. This problem is expected to remain secure against quantum attacks since no algorithm is currently known to solve it in polynomial time. It is fast and well-suited for devices that require quick computations. It also has a low computational overhead, and the short signature size makes it appropriate for limited bandwidth and storage applications [27].

10. Conclusion

Quantum computers are advancing quickly and could soon break the security systems we use today. Although

we do not know precisely when this will happen, we must switch to PQC. However, this change is complex and fast. It involves significant challenges in terms of performance, infrastructure, and integration.

Hybrid cryptography, which allows both classical and post-quantum algorithms, X.509v3 certificates with flexible methods for integration of PQC into existing PKI, and MPKC for IoT devices with limited resources are potential solutions. These solutions help keep security strong without needing to replace everything right away. While PQC is critical for future security, gradual implementation with compatible technologies such as blockchain or AI-driven security systems is essential.

References

[1] B. Westerbaan, "The state of the post-quantum Internet," *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 2024, https://blog.cloudflare.com/pq-2024/.

[2] S. J. Lily Chen, "Report on Post-Quantum Cryptography. NISTIR 8105," p. 15, 2016, https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf.

[3] A. Faz-Hernandez, "The Quantum Menace," 2019, https://blog.cloudflare.com/the-quantum-menace/.

[4] cloudflare, "What is quantum computing?" https://www.cloudflare.com/learning/ssl/quantum/what-is-quantum-computing/.

[5] P. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," 1999, https://fab.cba.mit.edu/classes/862.22/notes/computation/Shor-1999.pdf.

[6] e. s. mina zicu, "Threats to Modern Cryptography: Grover's Algorithm," 2020, https://www.preprints.org/manuscript/202009.0677/v1.

[7] "Grover's Algorithm Offers No Quantum Advantage," 2023, https://arxiv.org/pdf/2303.11317.

[8] "Quantum Threat Timeline Report," 2022, https://globalriskinstitute.org/publication/2022-quantum-threat-timeline-report/.

[9] "Harvest now, decrypt later," https://en.wikipedia.org/wiki/Harvest_now,_decrypt_later.

[10] "Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process," https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[11] "Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process. NISTIR 8240," 2019, https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf.

[12] "Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. NISTIR 8309," 2020, https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf.

[13] "Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. NISTIR 8413," 2022, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=934458.

[14] "Federal Information Processing Standards Publication Module-Lattice-Based Key-Encapsulation Mechanism Standard, FIPS 203," 2024, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf.

[15] "Federal Information Processing Standards Publication Module-Lattice-Based Digital Signature Standard, FIPS 204," 2024, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf.

[16] "Federal Information Processing Standards Publication Stateless Hash-Based Digital Signature Standard," 2024, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf.

[17] D. S. Adeline Langlois, "Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography," 2014, https://doi.org/10.1007/s10623-014-9938-4.

[18] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," 2005, https://doi.org/10.1145/1060590.1060603.

[19] "Short integer solution problem," https://en.wikipedia.org/wiki/Short_integer_solution_problem.

[20] VegeBun, "How do hash-based post-quantum digital signatures work?" 2022, https://research.dorahacks.io/2022/12/16/hash-based-post-quantum-signatures-2/.

[21] S. A. Huelsing, D.Butin, "XMSS: eXtended Merkle Signature Scheme," 2018, https://doi.org/10.17487/RFC8391.

[22] S. G. K. B. Muruganantham, P. Shamili, "Quantum cryptography for secured communication networks," 2020, https://core.ac.uk/download/pdf/329118533.pdf.

[23] "Post-quantum cryptography is too damn big," 2024, https://dadrian.io/blog/posts/pqc-signatures-2024/.

[24] P. van Dijk, "More PQC in PowerDNS: A DNSSEC Field Study," 2024, https://blog.powerdns.com/2024/07/15/more-pqc-in-powerdns-a-dnssec-field-study.

[25] "Design choices for post-quantum TLS," 2024, https://educatedguesswork.org/posts/pq-rollout/.

[26] B. Westerbaan, "NIST's pleasant post-quantum surprise," 2022, https://blog.cloudflare.com/nist-post-quantum-surprise/.

[27] j. W. Congli Wang, weijia Xue, "Integration of Quantum-Safe Algorithms into X.509v3 Certificates," 2009, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10176713.

Megaconstellations: Revolutionizing Internet Connectivity

Jonas Hohenstatter, Eric Hauser*, Leander Seidlitz*, Sebastian Gallenmüller*,
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: jonas.hohenstatter@tum.de, hauser@net.in.tum.de, seidlitz@net.in.tum.de, gallenmu@net.in.tum.de

***Abstract*—Recent ideas to launch thousands of interconnected satellites into orbit have initiated a space race of the 21st century. These large satellite systems aim to supply rural areas worldwide with fast, broadband Internet access. This paper is an introduction to this emerging topic, and it provides an overview of the history of artificial objects in space, current statistics of Starlink, and challenges the providers face. The second part features a literature research discussing the rather undisclosed technical details of Starlink and the hybrid routing mechanisms used in megaconstellations. This paper also compares the findings of two performance-based evaluations of Starlink and discusses the dimension of the performance gap between Megaconstellations and terrestrial networks. We also review the guides provided by SpaceX that explain the need for sustainable technology in the field of Megaconstellations to mitigate collision risks and challenges concerning increased space debris.**

***Index Terms*—megaconstellations, starlink, satellites**

1. Introduction

With the recent drop in space launch costs due to advanced manufacturing and reusability options, the potential for leveraging space technology to address modern-day challenges has significantly increased [1]. One of the most influential technologies in this field are satellites. Satellites are most commonly used for scientific earth observations, climate monitoring and communication.

The first satellites were mainly used to allow for transatlantic communication. These so-called geostationary satellites operate at an altitude of around 35 786 km (GEO) and are stationary relative to a fixed point on the Earth's surface. This ensures broad coverage by utilizing a small number of satellites [2].

On the other hand, a significant disadvantage is the high propagation delay that comes with the high altitude. Due to this trade-off, geostationary satellite connections cannot meet current standards of their terrestrial counterparts [3].

A solution to the high latency can be achieved by launching the satellites to a lower altitude. The closest possible orbit to the Earth lies within the low Earth orbit (LEO) region. This region ranges from 160 km to over 1600 km, with the lowest satellite currently orbiting at 167.4 km. LEO also inherits many other artificial objects like the International Space Station (ISS) at an altitude of 400 km [4].

The problem with this approach is the smaller coverage, which can be solved by increasing the number of satellites. This use of multiple satellites working together for a common purpose is known as a satellite constellation. In the context of the Internet, the ideal system leverages thousands of satellites. Such an enormous constellation is often described as a Megaconstellation [5].

2. History

This section focuses on the history of satellite-driven Internet. It discusses the history of the first geostationary provider up to modern Megaconstellations such as Starlink. It will also compare the most important providers.

2.1. Geostationary Internet

The first ideas for a communication satellite in GEO were published in 1945 by the fiction author Arthur C. Clarke. His article [6] presented the base for the early GEO satellites. The first reliable GEO satellites for communication were "Syncom 2" and "Syncom 3". Syncom 3's reliability was proven in 1964 when the Olympics in Japan were transmitted "live via satellite" to the US. Some of the first communication satellites were later updated to support Internet connections. A few ground stations connect these satellites with the terrestrial networks.

The first satellite with the sole purpose of providing broadband Internet (e-BIRD) was successfully launched in 2003 by Eutelsat. It still delivers Internet to Europe to date. Modern satellites utilize the high-frequency Ka-Band to maximize throughput. KA-SAT and Viasat-1 (2010) are the first to use this new technique [7], [8].

2.2. The Teledesic Network

This section corresponds closely to papers by Patterson [3] and Sturza [9] from the Teledesic Corporation.

The concept of launching Megaconstellations into LEO has been around since 1990. The "Teledesic Corporation" was the first company to practically attempt such a project. Teledesic planned to launch over 900 satellites to provide affordable broadband internet globally by 2002. The quality of service of this network was planned to be comparable to terrestrial networks with fiber-like delays and low bit error rates while providing 24-hour coverage for almost everyone on Earth. The capacity was planned to be equivalent to over 20 million users on traditional wired connections. Data rates on the Teledesic network

 doi: 10.2313/NET-2025-05-1_04

were meant to be adjustable with symmetrical and asymmetrical communication rates of up to 64 Mbit s^{-1}. Adjustable refers to them being scalable based on application demand.

The different satellites in the network can be seen as nodes communicating with eight neighboring nodes using a fast packet-switching technology. To ensure global coverage it was planned to split up the Earth into 20 000 supercells, with each being targeted by 64 transmit and receive beams from a satellite at all times. Data was planned to be distributed encrypted over the links in fixed-length packets of just 512 bit.

Due to financial difficulties, the project was suspended in 2002. Teledesic never launched any operational satellites [10].

2.3. Starlink

The current most influential Megaconstellation is operated as a subsidiary of the well-known aerospace company SpaceX under the name of "Starlink". Starlink was founded in 2014 by Elon Musk to supply rural areas with broadband internet and to provide global mobile broadband [11].

After the design phase, the first test launches were made in 2018 with two prototype satellites. The first 60 operational ones were launched in 2019, and Starlink went public in early 2021 [11], [12]. Because of the ability of SpaceX to launch the satellites using their own spacecraft and their advanced reusability options, they were able to launch a fleet of more than 6000 until August 2024. There are currently 4996 satellites in operation, and SpaceX plans to increase this number to around 40 000.

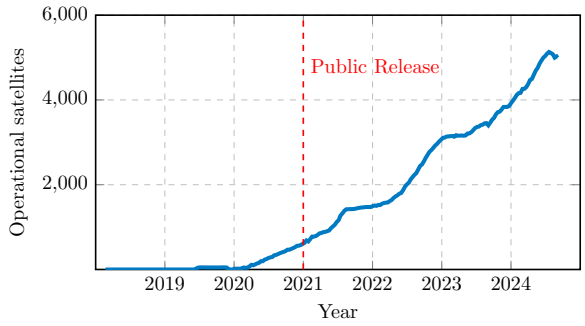


Figure 1: Starlink operational satellites over time [13]

These numbers were accumulated by Jonathan McDowell [13] with the operational satellites over time shown in Figure 1.

SpaceX uses their Falcon 9 rocket to deploy around 60 satellites per launch, with individual launches happening almost every week [11]. Starlink is currently in its second generation, with the second-generation satellites being four times more capable than those of the earlier generation [14]. In January 2024, SpaceX also launched new satellites with direct-to-cell capabilities. SpaceX plans to scale this new LTE Service network in the upcoming years with hundreds of new satellites [15]. Further explanations on the technical aspects of Starlink can be found in Section 3.

2.4. Provider Overview

Multiple other companies compete in this "space race" of the 21st century. Not only SpaceX, but also Amazon and Boeing have recently planned to launch their own constellations.

Companies like Iridium and Globalstar that have been around since the time of Teledesic, have also sent operational satellites into LEO.

Table 1 provides an overview of the most important providers. It shows information about the altitude of the satellites, planned and operational amount of satellites as well as the year when the constellation went into operation. It is closely based on data by Xingchi He [12].

TABLE 1: Important Megaconstellations [12]

Constellation	Planned	Operational	Since	Altitude (km)
Starlink	34224	4996	2020	≈ 500
OneWeb	648	616	2023	1200
Iridium-NEXT	66	80	2018	780
Globalstar-2	24	25	2013	1410
Kepler	360	16	2021	$550 - 650$
GW (China)	12992	0	-	$500 - 1200$
Amazon Kuiper	3236	0	-	$590 - 630$
Boeing	132	0	-	1056

The altitude plays a crucial role in this comparison, as it influences the amount of satellites needed. Starlink sticks out as the constellation with the highest number of planned and operational satellites.

Jonathan McDowell's statistics [13] state that there are currently 20 planned constellations with a total amount of planned satellites of 547 127. Only 7060 of them have already been launched.

3. Technical Details

This section will discuss the technical details as well as the performance of Megaconstellations. Starlink will serve as the primary example.

3.1. Starlink satellites

The newest Starlink satellites have a flat design and are relatively small with an overall mass of 250 kg. These second generation satellites have four times the capacity of the earflier generation. The generation 2 satellites are split up into two seperate versions. The "V2 mini" version that is not the full-size V2 satellite was designed to be compatible with the Falcon 9. Due to this design, up to 60 satellites can be distributed in one Falcon 9 launch. SpaceX is currently constructing the "Starship", a new rocket with a higher payload capacity to increase this number even more [14]. The Starship will be used to launch full-sized V2 satellites.

The new satellites utilize a Star Tracker to accurately calculate the position of each node. These new satellites also use inter-satellite links (ISL) to communicate with neighboring nodes. In the current generation, optical links in the form of three lasers are used. They can reach a transmission rate of up to 200 Gbit s^{-1}. There are eight antennas on each satellite that utilize the Ka-Band, Ku-Band and E-Band frequencies. A Starlink satellite can

 doi: 10.2313/NET-2025-05-1_04

adjust its path using an inbuilt ion engine based on argon gas. The engines can not be recharged in space yet and satellites without fuel will deorbit either naturally or controlled [16].

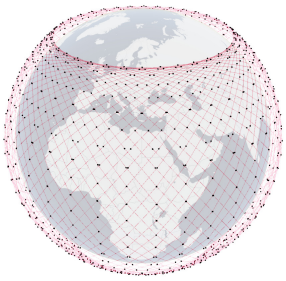


Figure 2: Starlink Shell one [17], [18]

Figure 3: Standard Dish with motors [23]

The satellites are launched into different groups, also known as shells. Each shell inherits different orbits in planes at different angles relative to the equator (inclination). The first Starlink shell has 72 orbits organized into orbital planes at a 53° incline [17]. Figure 2 shows the first Starlink shell with the majority of the current Starlink satellites. Other shells also serve regions near the poles with far fewer satellites [19].

A Starlink satellite can be operational for around five years before deorbiting naturally [20].

3.2. Starlink ground infrastructure

To function as intended, every constellation needs a connection to the Earth. These connections are usually established between a ground station (GS) and a satellite that is currently in view [21].

Most geostationary satellites only need a single GS, because of their fixed position relative to Earth. This is not the case for LEO satellites as most of them are only in view of a fixed point for around ten minutes [17]. LEO satellites do not have a fixed position to Earth to withstand gravitational pull. Various GSs have to be placed strategically around the globe to support all the satellites that orbit in and out of view and to operate the constellation effectively [21]. Starlink currently utilizes around 150 fixed GSs as gateways to the Internet, with most of them being located in the US and Europe [22].

Another big part of the ground infrastructure of Starlink are the private dishes that the subscribers use to connect to the network. These dishes communicate with currently visible satellites through a User Link (UL). The dish is wired to a router through Ethernet, which opens a local network for connections via WiFi or Ethernet [17].

For the visibility of satellites, the sky has to be mostly clear and the setup location of the dish has to be wide enough. The tilt of the dish also needs to be adjusted frequently with options for self-adjustment, as seen in Figure 3 [17]. There are currently six variants of the Starlink kit with options for an overall higher performance.

3.3. Routing and Traffic Management

Routing and Traffic Management are essential factors when evaluating networks. They are influencing factors on the efficiency and the overall connectivity of terrestrial and non-terrestrial networks.

Megaconstellations must have a sophisticated routing system in place to ensure accurate packet routing even with rapidly moving next hops (satellites). This is especially important in Megaconstellations where multiple hops take place using ISLs [24]. As shown in Section 2.2, Teledesic planned to incorporate a fast packet switching technology through ISLs.

Starlinks first generation approach was based on a bent-pipe strategy. Bent-pipe refers to a satellite only being used to relay a signal to a different point on Earth's surface. This was used, because generation 1 did not utilize ISLs yet. Indirect satellite communication was possible by using GSs as intermediate steps, but it was somewhat inconvenient. As shown by Ma et al. [17], a connection to Starlink generally only involved one hop due to this.

The signal from the dish was practically only redirected to the closest GS before switching to terrestrial services. This connection between the dish and GS can only be established successfully if their distance is at most 1000 km.

Sending a packet between two dishes in range of the same satellite was also tested in [17]. The experiment showed that the communication still had to be relayed through a GS instead of the ULs of the dishes communicating through the satellite.

As the generation 2 satellites are equipped with optical ISLs, it has become possible to route packets directly between satellites. This multi-hop approach can be beneficial in increasing the overall connectivity of the system. It increases the autonomy of the network as the routing is not dependent on a ground segment that may not have broadband connections in place. An advantage over ground infrastructure is also the vacuum as a propagation environment for wireless signals. The ultra-low latency of ISLs also enables real-time data sharing [24].

A handover to another satellite is needed if a node in a currently established link moves out of view, as visibility between nodes is crucial for ISLs to function [24].

The routing system for Megaconstellations can be set up in a hybrid manner. It consists of characteristics of static and dynamic routing.

Static routing might seem counterintuitive when looking at dynamically moving satellites, but the topology of

a Megaconstellation is predictable. The orbit period of the satellites can be cut into time slots that represent a stable snapshot of the network topology. Traditional routing algorithms like the Dijkstra algorithm can be utilized to find the optimal path in a given snapshot [24]. An algorithm that is optimized to constellations called "StepClimb" is described in [25].

Static routing cannot adapt to unpredictable real-time situations, such as defective satellites in the Megaconstellation. For these situations, a dynamic routing approach has to be used. The routing tables of the satellites have to be kept up-to-date. This can be achieved by periodically flooding the states of each node to all satellites, which can create an immense overhead. To solve this issue, satellites on a higher altitude acting as routing managers can be used [24]. This multi-layer approach with satellites in MEO (medium earth orbit) or GEO is yet to be used in Starlink.

Load balancing and congestion control are also crucial factors of networks and even more significant challenges for Megaconstellations. As the distribution and demand of the subscribers are uneven around the globe, there are imbalanced regional traffic loads. Minimum hop routing strategies and the resulting paths may be congested in high-traffic areas. The used routing algorithms have to account for this as well [24].

3.4. Performance

The Performance of Starlink has been evaluated in papers by Mohan et al. [19] and Ma et al. [17] This section sums up the most important findings and compares these papers.

SpaceX states the following: "Starlink users typically experience download speeds between 25 and 220 Mbit s^{-1}, with a majority of users experiencing speeds over 100 Mbit s^{-1}. Upload speeds are typically between 5 and 20 Mbit s^{-1}. Latency ranges between 25 and 60 ms on land, and 100+ ms in certain remote locations" [26]. Remote locations for Starlink are Oceans and Islands as well as the polar regions.

A measurement by Mohan et al. [19] using global online speed test data shows that the median latency of Starlink lies within 40 ms and 50 ms. In Well-Provisioned Regions like Seattle and the US, the latency is consistently well below 50 ms. This is likely the case because of the wide coverage of GSs in these regions. South American regions like Colombia and regions in Oceania show a low performance with latencies exceeding 100 ms. It was also shown that the latency of Starlink under load increased [19]. This performance analysis by Mohan et al. [19] reflects the promised performance closely.

The latency of Starlink has a very high variation in contrast to the stable latency in terrestrial networks. The reason for this is the continuous movement of the satellites and the handovers that are needed to keep connections active [17].

Sami et al. [17] features an experimental approach to measure the performance for connections from northern American terrain to a variety of AWS servers worldwide. The measurement has shown that the latencies using Starlink were higher, but the difference was mostly negligibly small. The throughput rates compared to terrestrial connections cannot be neglected. Starlinks upload rates only reach around 10% of the terrestrial connections.

Mohan et al. [19] have measured that Starlink achieves around 50-100 Mbit s^{-1} in download rates and 4-12 Mbit s^{-1} in upload rates. This corresponds to the data measured by Sami et al. [17]. These throughput rates - as for the latencies - lack stability.

There is currently a gap between the performance of Starlink and terrestrial connections, which occurs most prominently in well-provisioned areas. Starlink outperforms terrestrial connections in some underprovisioned areas like Columbia, where the local ISPs average 70 and 100 Mbit s^{-1} in latency, but Starlink averages 50 and 70 Mbit s^{-1}. In regions like the Philippines, Starlink performs worse than local ISPs, which is a result of the low distribution of ground stations. This same conclusion can be made from the above mentioned measurements in [17].

Users of Starlink experience higher overall latencies and lower throughput comparable to cellular connections. The current gap will most likely continue to shrink with the expanding infrastructure of Starlink in Space as well as on the ground [19].

4. Challenges

It is important to note that Megaconstellations do not only come with positive aspects, but also non-favorable impacts. SpaceX offers articles and guides that discuss solutions to these challenges.

There are currently 22 384 objects orbiting in LEO with a total mass of 6120.8 t. These objects stem from space missions since 1957, including mission payload and other debris. Some debris and particles also originate from so called fragmentation events due to collisions and explosions [27].

Since the age of Megaconstellations, the payload launch traffic into LEO has increased from under 500 to over 2500 launches. The expanding number of objects in space increases the risk of a collision in future space missions [27]. Starlink satellites have an inbuilt collision avoidance capability in place to mitigate this risk.

Objects within LEO reenter the atmosphere at any time and sometimes have unpredictable re-entry paths. SpaceX uses controlled and well-tracked deorbits of non-operational satellites to reenter them into the atmosphere [20].

Starlink satellites can be visible to the naked eye while being lifted into their operational orbit. They can still be visible to observatories on Earth when they are illuminated by the sun in their foreseen orbit, disrupting astronomical observations. Generation 2 of Starlink utilizes different materials to mitigate the brightness of the satellites [28].

5. Conclusion and future work

Megaconstellations is an emerging topic that has become more important in the last five years but still remains rather unknown. This paper provided an overall introduction to Megaconstellations and their importance.

SpaceX's project of "Connecting The Unconnected" aims to deliver Internet to areas around the globe with no connection to terrestrial services. Starlink can deliver broadband internet to the highest mountains and the

smallest isles in the ocean. Starlink can be utilized in emergencies and can act as a lifesaver.

Megaconstellations may not be the optimal way of connecting to the Internet when connecting from a well-connected area like Europe or the US, but they can revolutionize the connectivity in places like Colombia. With the performance gap getting smaller, Megaconstallations may be a viable option for well-provisioned areas in the future.

As the topic is relatively young, there is a great amount of potential for future work and optimizations. Future research could explore the simulation of constellations to find optimal routing solutions or technical improvements to mitigate collision risks. Exploring the environmental and social impact could also be an important topic.

References

[1] H. W. Jones, "The Recent Large Reduction In Space Launch Cost," International Conference On Environmental Systems, Inc. NASA Ames Research Center, July 2018.

[2] S. Liang and J. Wang, *Advanced Remote Sensing*, 2nd ed. Academic Press, 2019.

[3] D. P. Patterson, "Teledesic: A Global Broadband Network," in *Proceedings Of The IEEE*. IEEE, 1998, pp. 545–552, accessed from IEEE Xplore on August 27, 2024.

[4] K. Stewart, "Low Earth Orbit," *Encyclopedia Britannica*, 2024, accessed 1 September 2024. [Online]. Available: https://www.britannica.com/technology/low-Earth-orbit

[5] C. Pardini and L. Anselmo, "Effects Of The Deployment And Disposal Of Mega-Constellations On Human Spaceflight Operations In Low LEO," *Journal Of Space Safety Engineering*, vol. 9, no. 2, pp. 274–279, 2022.

[6] A. C. Clarke, "V2 For Ionosphere Research (Letter To The Editor)," *Wireless World*, vol. 51, no. 2, pp. 61–63, 1945.

[7] J. N. Pelton, "History Of Satellite Communications," in *Handbook Of Satellite Applications*, J. N. Pelton, S. Madry, and S. Camacho-Lara, Eds., 2017, pp. 31–71.

[8] Ground Control, "A Brief History Of Satellite Communications," https://www.groundcontrol.com/knowledge/guides/a-brief-history-of-satellite-communications/, accessed: 2024-09-01.

[9] M. A. Sturza, "The Teledesic Satellite System," in *Proceedings Of IEEE National Telesystems Conference - NTC '94*, 1994, pp. 123–126.

[10] The Seattle Times, "The Birth And Demise Of An Idea: Teledesic's 'Internet In The Sky'," *The Seattle Times*, accessed: 2024-09-01. [Online]. Available: https://archive.seattletimes.com/archive/?date=20021007&slug=teledesic070

[11] Mihir Tripathy, "How Is Starlink Changing Connectivity?" *Smithsonian Magazine*, 2022, accessed: September 5, 2024. [Online]. Available: https://www.smithsonianmag.com/science-nature/how-is-starlink-changing-connectivity-180980735/

[12] X. He, "Orbit Determination For Independent LEO Mega-Constellations," Ph.D. dissertation, Technische Universität München, 2024. [Online]. Available: https://mediatum.ub.tum.de/1721042

[13] J. McDowell, "Starlink Statistics," https://planet4589.org, 2024, accessed: 1 September 2024.

[14] SpaceX, "Second Generation Starlink Satellites," 2024. [Online]. Available: https://api.starlink.com/public-files/Gen2StarlinkSatellites.pdf

[15] SpaceX, "SpaceX Sends First Text Messages Via Its Newly Launched Direct To Cell Satellites," 2024. [Online]. Available: https://api.starlink.com/public-files/DIRECT_TO_CELL_FIRST_TEXT_UPDATE.pdf

[16] "Starlink," 2024, accessed: September 5, 2024. [Online]. Available: https://www.starlink.com

[17] 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*, 2023, pp. 1–10.

[18] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla, "Exploring The "Internet From Space" With Hypatia," in *ACM IMC*, 2020.

[19] 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*. New York, NY, USA: Association For Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3589334.3645328

[20] SpaceX, "Commitment To Space Sustainability," 2024. [Online]. Available: https://api.starlink.com/public-files/Commitment%20to%20Space%20Sustainability.pdf

[21] J. Kopacz, J. Roney, and R. Herschitz, "Optimized Ground Station Placement For A Mega Constellation Using A Genetic Algorithm," 2019, presented at the Pre-Conference Posters Session I, Utah State University, Logan, UT.

[22] Starlink Insider, "Starlink Gateway Locations: Everything You Need To Know," 2024, accessed: September 5, 2024. [Online]. Available: https://starlinkinsider.com/starlink-gateway-locations/

[23] SpaceX, "Starlink Router Image," 2024, accessed: September 5, 2024. [Online]. Available: https://www.starlink.com/de/specifications

[24] C. Wu, S. Han, Q. Chen, Y. Wang, W. Meng, and A. Benslimane, "Enhancing LEO Mega-Constellations With Inter-Satellite Links: Vision And Challenges," 06 2024.

[25] Q. Chen, L. Yang, Y. Zhao, Y. Wang, H. Zhou, and X. Chen, "Shortest path in leo satellite constellation networks: An explicit analytic approach," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 5, pp. 1175–1187, 2024.

[26] SpaceX, "Starlink Specifications," 2024, accessed: September 14, 2024. [Online]. Available: https://www.starlink.com/legal/documents/DOC-1400-28829-70

[27] European Space Agency, "Space Environment Statistics," 2024, accessed: September 16, 2024. [Online]. Available: https://sdup.esoc.esa.int/discosweb/statistics/

[28] SpaceX, "Brightness Mitigation Best Practices For Satellite Operators," 2024. [Online]. Available: https://api.starlink.com/public-files/BrightnessMitigationBestPracticesSatelliteOperators.pdf

Kata Container: Influence of Security onto Network Performance

George Jin, Florian Wiedner*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: george.jin@tum.de, wiedner@net.in.tum.de

***Abstract*—The introduction of technologies like 5G has increased the demand for high-performance, scalable networks. Network Function Virtualization (NFV) has emerged as a solution by virtualizing network functions on standard servers to meet these standards. However, it relies on robust security measures which simultaneously provide high network performance. This paper discusses whether Kata Containers integrated with Trusted Execution Environments (TEEs) is a viable solution for securing NFV environments. We dive into a comprehensive background of these technologies, analyze their security benefits and drawbacks, and their impact on network performance in NFV. We found that although Kata Containers and TEEs provide enhanced security through virtualization and hardware-based protection, they introduce performance overheads, especially regarding network latency and scalability. We conclude that further research is necessary to analyze the concrete security and performance implications of Kata Containers integrated with TEEs in NFV environments. This includes research in performance optimization techniques and strategies for determining the right balance between security and performance in NFV environments.**

***Index Terms*—Kata Containers, TEE, NFV**

1. Introduction

In recent years, the emergence of technologies like 5G and its use cases like IoT and edge computing have become increasingly relevant. With the progress they bring, they have introduced new requirements and challenges, particularly the need for highly scalable networks capable of delivering high network throughput and low latency while simultaneously dealing with a large number of devices in real-time [1], [2]. Consequently, NFV has become a key technology to meet these standards by virtualizing network functions on standard servers instead of relying on specialized hardware [3].

NFV, in turn, requires robust virtualization and containerization solutions to provide both high performance and high security [3]. Kata Containers has emerged as a promising approach to meet these requirements by combining the lightweight performance of traditional containers with the enhanced security and isolation characteristics of virtual machines (VMs) [4]. This makes them a potential solution for securing NFV environments. To further enhance the isolation of these containers, TEEs can be integrated to offer an additional layer of hardware-based security [5]. Together, they can provide an effective method for enhancing the security of virtualized network functions (VNFs), though their full impact requires further analysis.

However, the increased security and isolation offered by Kata Containers and TEEs come at the cost of performance overhead [6]. In environments where high network performance is critical, balancing the trade-off between enhanced security and optimal performance remains a major challenge. Therefore, understanding these challenges is crucial for evaluating the viability of Kata Containers and TEEs in NFV environments.

This paper provides an overview of the current research on Kata Containers and TEEs in NFV environments, investigating the security benefits and network performance drawbacks. The goal is to determine whether Kata Containers, with or without TEEs, present a viable solution for securing NFV environments while maintaining the necessary network performance requirements. To achieve this, Section 2 will provide a comprehensive background on Kata Containers, TEE, and NFV. In Section 3, we will explore the security benefits and drawbacks of Kata Containers and TEEs. This will be followed by Section 4, which discusses their network performance implications. And finally, we will compare the security implications with the performance drawbacks in Section 5.

2. Background

We will explore three key technologies important to our discussion: Kata Containers, TEEs, and NFV. Kata Containers aims to improve security by utilizing lightweight VMs. TEEs use a separated area on the CPU to provide hardware-level protection. NFV is a concept that moves network functions from specialized hardware to standard servers. A solid understanding of these technologies is essential for our evaluation.

2.1. Kata Containers

Traditional container runtimes like RunC achieve performance close to native levels [7]. However, by only relying on Linux namespaces and control groups (cgroups) while sharing the same host kernel, they offer little to no isolation, making them vulnerable to attacks. On the other hand, VMs offer strong isolation with their isolated kernel but introduce a significant performance overhead [8]. To bridge this gap, Kata Containers, an open-source container runtime introduced in 2017, aims to combine the high performance of lightweight containers with the security and isolation of VMs. This approach is the result of merging

 doi: 10.2313/NET-2025-05-1_05

Intel Clear Containers and Hyper.sh runV, technologies that run each container in its own optimized VM [4].

The main idea is to run each container within its own lightweight VM, highly optimized to minimize performance overhead and resource consumption using technologies like Guest Kernel Minimal and Guest Image [4] while providing kernel-level isolation. Additionally, it uses a specialized QEMU version named qemu-lite as the default hypervisor, which improves boot time and reduces memory footprint with features like Machine Accelerators, Kernel same-page merging, Hot Plug Devices, and Fast Template [4]. The architecture of Kata Containers consists of three main components: the Kata-runtime, Kata-agent, and Kata-shim. The Kata-runtime on the host creates the VM for running the container. The Kata-agent process, running in the guest kernel in the VM, sets up the environment and runs the container and processes within the container while receiving instructions from the host via gRPC [9]. The Kata-shim is a process on the host that is responsible for all container I/O streams [4]. An additional advantage of Kata Containers is their OCI-compliance, making them seamlessly integrate with containerization and container orchestration platforms like Docker, Kubernetes, and OpenStack by simply replacing runC with Kata-runtime, ensuring easy deployment for organizations and removing the need for major modifications of existing workflows [4].

2.2. Trusted Execution Environment

A TEE is a secure area within the processor that allows the safe execution of code and storage of data. It is segregated from the rest of the CPU, protecting its data from unauthorized access or tampering by code outside of that environment [10]. TEEs establish a chain of trust during the boot process, which guarantees the authenticity of the running software, the integrity of the runtime states, and the confidentiality of the code and data. TEEs also support remote attestation, allowing third parties to verify the integrity and trustworthiness of the TEE [11]. Prominent examples are Intel SGX and ARM TrustZone, each with its own approach to security. Intel SGX isolates data and code in areas called enclaves, while ARM TrustZone separates the whole processor into a secure and normal world [12]. In general, the core features of a TEE include strong hardware-based isolation, efficient scheduling and secure communication between secure and rich environment, and secure updates [5].

2.3. Network Function Virtualization

In traditional networks, network functions like firewalls, load balancers, and routers rely on specialized hardware. In times of fast-paced digital innovations and declining lifecycles of hardware, the frequent replacement and scaling of these systems pose a serious challenge for network service providers, particularly in fields like 5G [3]. NFV aims to solve this problem by virtualizing these network functions and running them on commercial off-the-shelf servers instead of specialized equipment, resulting in several advantages [3]. Firstly, it increases scalability and flexibility by enabling providers to scale the number of VNFs up or down depending on demand. Secondly, it offers better operating performance by dynamically allocating resources based on a given network load. Thirdly, it leads to shorter development cycles by replacing the necessity of installing new physical devices with the deployment of software updates. These benefits significantly reduce both capital expenditures and operational expenditures, making NFV a highly flexible and cost-effective solution for modern networks [3].

The NFV architecture consists of three components. A physical server provides computing and storage resources, a hypervisor that manages the virtual environment, and a virtualized environment for executing VNFs [3]. Even though NFV brings several advantages, the virtualization of network functions is expected to increase the potential for security attacks [3], requiring more sophisticated security measures. Firstly, it requires a protected hypervisor to prevent unauthorized access or data leakage [3]. Secondly, data communication and VM migration need a secure environment [13]. Thirdly, NFVs use application programming interfaces (APIs), which pose another security threat [14]. At the same time, the network performance must be comparable to traditional networks despite the additional virtualization layer. Therefore, a well-balanced trade-off between security and performance is necessary.

3. Kata Containers and Security

One of the main vulnerabilities of traditional containers is their dependence on the shared host kernel. If one container gets compromised, the attacker could potentially get access to the host and other containers. While the Linux kernel uses cgroups to isolate and limit the usage of physical resources for each container, it is shown that out-of-band workloads can break the cgroups' confinement [15], potentially making the whole system vulnerable to resource exhaustion attacks, such as Denial-of-Service. Next to the weak resource confinement, the shared kernel also poses the risk of privacy leakage through pseudo-filesystems, enabling attackers to gather sensitive information about the environment for further exploitation [16], [17]. In addition to the kernel layer, the container layer faces vulnerabilities like improper handling of symbolic links or insecure API handling, making container escapes possible [18]. In fact, 56.82% of vulnerability exploits could launch successfully from within a container [19], indicating the need for more advanced security solutions.

Kata Containers makes container escapes more difficult with its additional layer of security offered by lightweight VMs and kernel-level isolation, but research shows that escapes are still possible [18]. In particular, according to research, Kata Containers have three key vulnerabilities. First, Kata Containers did not properly enforce device cgroups, allowing attackers to access the /dev files on the VM inside the container, leading to CVE-2020-2023 [20]. The attacker could then overwrite the kata-agent, leading to container escapes and further compromises. The Kata Containers reuse the corrupted kata-agent, leading to CVE-2020-2025 [21]. Lastly, kata-runtime does not validate mount points in shared folders. That means it resolves any symbolic link and conducts the mount operation, leading to CVE-2020-2026 [22], which would allow the attacker to mount the root file

system to any part of the host system, thereby breaking the virtualized container despite the hardware virtualization in use [18]. To mitigate these vulnerabilities, Kata Containers can be integrated with TEEs. The isolation of the lightweight VM in Kata Containers, in combination with hardware-based protection by TEEs, can significantly increase the overall security of the system, offering protection not only from container escapes but also from malicious code from the host system itself. Additionally, remote attestation ensures that only verified code runs within the TEE, further reducing the attack surface [11]. However, the use of TEEs comes with new challenges as well. Even though TEEs provide secure hardware-based security, they are not immune to side-channel attacks, which make use of indirect system information like memory access, CPU load, or power consumption to infer information about other aspects of the system for further exploitation [11]. Nevertheless, while there is limited research in this area, combining Kata Containers and TEEs could be particularly useful for securing VNFs, making them less vulnerable compared to traditional network functions.

4. Network Performance Impact of Kata Containers

The shift from traditional network functions to an NFV environment introduces performance challenges, especially when implemented with Kata Containers. The critical performance metrics in NFV are network latency and network throughput. Latency refers to the time needed for data to travel from one point to another, while throughput refers to the amount of data able to be transmitted within a specified time window. Research has shown that the additional virtualization layer of Kata Containers notably reduces the network performance in certain cases compared to traditional containers like runC, introducing potential bottlenecks [6], [7], [9]. In particular, Kata Containers only shows slightly lower throughput compared to runC, less than 1% [6]. In more specific cases regarding TCP and HTTP throughput, it showed 1-18% lower throughput depending on the scenario [6], [7]. Similarly, Kata Containers only scored 20% of the throughput of runC in simple GET operations from in-memory data [8]. Similarly, latency is also affected, showing a performance loss of up to 35% compared to RunC [6], [9]. Despite these shortcomings, Kata Containers performs better than gVisor in both network throughput and network latency [6], [7]. These findings are further illustrated in Figure 1, adapted from [6], which compares the network performance of different container runtimes with an emphasis on latency. In particular, TCP_RR and UDP_RR are request-response tests, while TCP_CRR additionally includes the connection time and teardown time. runC and bare metal show the best performance, while gVisor shows the highest overhead, with Kata Containers performing in between.

Furthermore, in [7], the authors compared the scalability benchmarks of runC and gVisor across different container counts. The results of these scalability benchmarks are summarized in Figure 2, adapted from [7], which shows that while runC scales efficiently, gVisor suffers from significant network performance overhead with an increasing number of containers [7]. However, we observe that containerd/runsc does not show the same scaling issues as crio/runsc, which could be the result of specific optimizations in the containerd runtime. Nevertheless, gVisor relies on an additional isolation layer with its own user-space kernel. It can be expected that Kata Containers show similar behavior due to its additional VM layer. This means that as the number of containers grows, the network performance of Kata Containers could drop significantly. This scaling limitation could pose a serious challenge in NFV environments, where a large number of VNFs run simultaneously while requiring high network throughput and latency.

A way for addressing the performance drop of Kata Containers could be technologies like Single Root I/O Virtualization (SR-IOV), which allows direct access to network hardware, bypassing the virtualization layer [23]. Even though SR-IOV gives containers direct access to hardware, it does not directly violate the principles of NFV, as it allows the hardware to be shared between multiple VNFs [23]. Research shows that it can drastically reduce the performance overhead of the virtualization layer, almost achieving native network performance [23]. Consequently, SR-IOV could be a key solution to solve the network performance overhead and the scaling limitations of Kata Containers. Next to SR-IOV, there are software-based performance optimization techniques. One of them is the use of the kernel driver vhost-net, which offloads network packet handling from the VM to the host. This reduces context switches between guest and host, leading to higher throughput and lower latency while also allowing a higher number of VMs to run on the same host. This means more VNFs can run in parallel without performance degradation, improving scalability [24]. Another technique is a set of optimized libraries and drivers called Data Plane Development Kit (DPDK), which enables the network packets to bypass the host kernel entirely. By allowing distributed processing of network packets across multiple CPU cores, DPDK further increases network performance [25]. In terms of performance optimizations for TEEs, current improvements in ARM TrustZone include

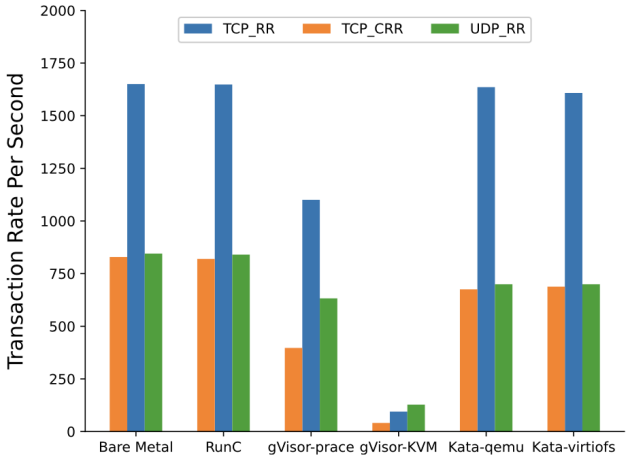
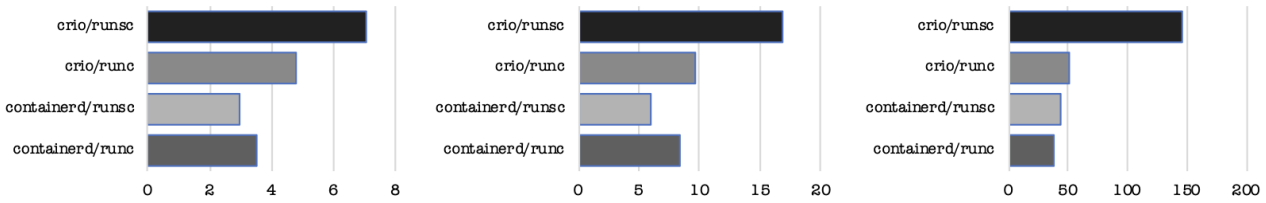


Figure 1: TCP_RR, TCP_CRR, UDP_RR transaction rates per second for different container runtimes. Adapted from [6].



(a) 5 Containers. (b) 10 Containers. (c) 50 Containers.

Figure 2: Benchmark comparing the network performance (in seconds) of RunC-based runtimes (crio/runc, containerd/runc) and gVisor-based runtimes (crio/runsc, containerd/runsc) across 5, 10, and 50 containers, based on 10 runs. Adapted from [7].

the simultaneous access to I/O devices for both secure and rich environment. This reduces context switching and increases the performance and scalability of I/O-heavy tasks like network packet processing in NFV [26]. However, Kata Containers could still be limited in their use in large-scale NFV environments because of their scalability limitations, and even optimization techniques like SR-IOV and vhost-net, which mitigate this limitation, could be insufficient to fully address this issue.

Additionally, when integrating Kata Containers with TEEs for additional security, there are also trade-offs in terms of network performance. One of the key issues is the frequent context switches between the TEE and the rich environment, especially in network-intensive tasks where large amounts of data are transported. These can lead to significant performance overheads and thus challenge the combined use of Kata Containers and TEEs in NFV environments. A solution is to minimize the performance overhead by limiting the TEE usage to critical parts of the network, thereby reducing the performance overhead while maintaining security [27]. Because of the additional overhead introduced by Kata Containers and TEEs, their usage in performance-critical applications in 5G networks could be limited.

5. Kata Containers and NFV

NFV environments allow the virtualization of network functions, leading to significant improvements in scalability, flexibility, and cost reductions in comparison to traditional networks. However, this shift comes at the cost of higher vulnerability to attacks. To mitigate these security risks, Kata Containers could offer a promising solution. In these environments, potentially multiple VNFs run on the same hardware, consequently requiring a strong level of isolation and security. While traditional containers like runC show high network performance, they are flawed regarding security and isolation, making them vulnerable to critical attacks like container escapes which could compromise the whole system. By using Kata Containers as an alternative, we could significantly reduce the risk of container escapes. This is the result of encapsulating each container in its own lightweight VM, offering kernel-level isolation. Running these Kata Containers in TEEs provides the benefit of an additional layer of hardware-based protection, even making attacks from a compromised host difficult. Nevertheless, TEEs are vulnerable to side-channel attacks, making use of secondary information for further compromise.

When it comes to network performance, VNFs require high throughput and low latency. However, securing NFV environments using Kata Containers comes at the cost of performance loss caused by the additional virtualization layer, making it a potential bottleneck for high-traffic scenarios. More specifically, while Kata Containers show comparable throughput to runC in the majority of cases, the network latency seems highly affected by the additional virtualization layer. Moreover, NFV requires easy scaling, but we have shown that Kata Containers are potentially limited in that regard, causing increasing performance overhead with a growing number of containers. This makes adopting Kata Containers difficult when handling high numbers of VNFs. Proposed solutions would be SR-IOV, vhost-net, or DPDK, which would enable Kata Containers to maintain high performance while retaining the security gains. However, with the use of SR-IOV or vhost-net, direct access to hardware could undermine the benefits of the additional virtualization layer and should only be considered when the need for performance outweighs the risk of reduced security and isolation. Additionally, Kata Containers could require more hardware resources than traditional containers due to the virtualization layer, increasing operational costs in terms of computing power and memory. Similarly, TEEs like Intel SGX and ARM TrustZone require specialized processors, potentially driving up capital costs. The decision whether to use Kata Containers and TEEs to secure NFV depends heavily on the specific use case. In areas where security is paramount, it could be a strong solution. On the contrary, in situations where performance is paramount, Kata Containers could pose a limitation with its performance overhead caused by the hardware virtualization and possible integration of TEEs, which further reduces the performance through the need for frequent context switches.

6. Conclusion and Future Work

This paper aims to provide a broad overview of the current research on Kata Containers, TEEs, and their security and performance implications on NFV environments. While Kata Containers make use of lightweight virtual machines to improve their isolation compared to traditional containers, they are still susceptible to vulnerabilities. These can be mitigated with the integration

of TEEs, which add an additional layer of hardware-based security, thereby creating a robust combination that can significantly reduce attacks in NFV environments. However, this enhanced security comes at the cost of overheads in network performance, particularly in network latency and scalability. Since VNFs rely on low latency, high throughput, and high scalability, these additional security layers could become a bottleneck in performance-critical scenarios. Therefore, further research is necessary to examine the exact security benefits and performance drawbacks in real-world deployments of Kata Containers integrated with TEEs in NFV environments. This includes further research in optimization techniques like SR-IOV for increased performance while maintaining security, as well as potential improvements, like the reduction of the frequency or cost of encryption cycles in Intel SGX [12]. Based on these results, future work should aim to determine the right balance of security and performance for secure NFV.

References

[1] N. Hassan, K.-L. A. Yau, and C. Wu, "Edge computing in 5g: A review," *IEEE Access*, vol. 7, pp. 127 276–127 289, 2019.

[2] I. Alawe, Y. Hadjadj-Aoul, A. Ksentini, P. Bertin, and D. Darche, "On the scalability of 5g core network: The amf case," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2018, pp. 1–6.

[3] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "Nfv: State of the art, challenges and implementation in next generation mobile networks (vepc)," *IEEE network*, vol. 28, no. 6, pp. 18–26, 2014.

[4] A. Randazzo and I. Tinnirello, "Kata containers: An emerging architecture for enabling mec services in fast and secure way," in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE, 2019, pp. 209–214.

[5] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/Ispa*, vol. 1. IEEE, 2015, pp. 57–64.

[6] X. Wang, J. Du, and H. Liu, "Performance and isolation analysis of runc, gvisor and kata containers runtimes," *Cluster Computing*, vol. 25, no. 2, pp. 1497–1513, 2022.

[7] L. Espe, A. Jindal, V. Podolskiy, and M. Gerndt, "Performance evaluation of container runtimes," in *Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER)*. IEEE, 2020, pp. 273–281.

[8] W. Viktorsson, C. Klein, and J. Tordsson, "Security-performance trade-offs of kubernetes container runtimes," in *2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2020, pp. 1–4.

[9] H. Z. Cochak, G. P. Koslovski, M. A. Pillon, and C. C. Miers, "runc and kata runtime using docker: A network perspective comparison," in *2021 IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE, 2021, pp. 1–6.

[10] Microsoft, "Trusted execution environment in azure," 2024, accessed: 2024-09-16. [Online]. Available: https://learn.microsoft.com/en-us/azure/confidential-computing/trusted-execution-environment

[11] S. Brenner, "Enhancing cloud security with trusted execution," Doctoral Dissertation, Technische Universität Carolo-Wilhelmina zu Braunschweig, Braunschweig, Germany, December 2020, disputation on 14.12.2020.

[12] N. Buchner, H. Kinkelin, and F. Rezabek, "Survey on trusted execution environments," 2022, accessed: 2024-10-14.

[13] V. Ashktorab, S. R. Taghizadeh *et al.*, "Security threats and countermeasures in cloud computing," *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, vol. 1, no. 2, pp. 234–245, 2012.

[14] C. S. Alliance, "The notorious nine cloud computing top threats in 2013," 2013, accessed: 2024-09-16. [Online]. Available: https://cloudsecurityalliance.org/artifacts/the-notorious-nine-cloud-computing-top-threats-in-2013

[15] X. Gao, Z. Gu, Z. Li, H. Jamjoom, and C. Wang, "Houdini's escape: Breaking the resource rein of linux control groups," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1073–1086.

[16] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *2010 proceedings IEEE infocom*. IEEE, 2010, pp. 1–9.

[17] H. Li, Y. Yang, Y. Dou, J.-M. J. Park, and K. Ren, "Pedss: Privacy enhanced and database-driven dynamic spectrum sharing," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1477–1485.

[18] Y. Yang, W. Shen, B. Ruan, W. Liu, and K. Ren, "Security challenges in the container cloud," in *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 2021, pp. 137–145.

[19] X. Lin, L. Lei, Y. Wang, J. Jing, K. Sun, and Q. Zhou, "A measurement study on linux container security: Attacks and countermeasures," in *Proceedings of the 34th annual computer security applications conference*, 2018, pp. 418–429.

[20] "CVE-2020-2023: Kata Containers device cgroup enforcement issue," https://nvd.nist.gov/vuln/detail/CVE-2020-2023, accessed: 2024-10-14.

[21] "CVE-2020-2025: Kata Containers reuse of corrupted kata-agent," https://nvd.nist.gov/vuln/detail/CVE-2020-2025, accessed: 2024-10-14.

[22] "CVE-2020-2026: Kata-runtime mount point validation failure," https://nvd.nist.gov/vuln/detail/CVE-2020-2026, accessed: 2024-10-14.

[23] Y. Dong, X. Yang, J. Li, G. Liao, K. Tian, and H. Guan, "High performance network virtualization with sr-iov," *Journal of Parallel and Distributed Computing*, vol. 72, no. 11, pp. 1471–1480, 2012.

[24] Red Hat, "Deep dive into virtio networking and vhost-net," https://www.redhat.com/en/blog/deep-dive-virtio-networking-and-vhost-net, 2019, accessed: 2024-10-14.

[25] Trenton Systems, "What is dpdk?" https://www.trentonsystems.com/en-us/resource-hub/blog/what-is-dpdk, 2023, accessed: 2024-10-14.

[26] N. Zhang, B. D. R. Hafshejani, D. Forte, and M. Tehranipoor, "Self-secured devices: Trustzone-based management of shared resources," p. 101934, 2021, accessed: 2024-10-14.

[27] F. Schwarz, "Trustedgateway: Tee-assisted routing and firewall enforcement using arm trustzone," in *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 56–71. [Online]. Available: https://doi.org/10.1145/3545948.3545961

Overview of Threshold PQC Schemes

Joon Kim, Filip Rezabek*, Dr. Holger Kinkelin†
Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
*Email: joon.kim@tum.de, *rezabek@net.in.tum.de, †kinkelin@net.in.tum.de*

***Abstract*—Threshold schemes distribute a signing key among multiple parties, requiring their collaboration to perform cryptographic tasks, thereby mitigating the risk of key compromise. As quantum computing advances, recent research has increasingly focused on combining these schemes with post-quantum secure cryptographic primitives such as lattices. This paper analyzes three significant advancements on post-quantum secure threshold schemes. First, Boneh et al.'s universal thresholdizer converts any cryptographic scheme into a threshold version, offering great flexibility, but it suffers from inefficiencies from homomorphically evaluating entire circuits. Second, Kamil et al. improve upon this by selectively applying homomorphic evaluation to an existing (n, n)-threshold scheme, extending it to a (t, N)-scheme. Lastly, Cozzo et al. thresholdize FALCON using multiparty computation techniques, but the mixture of linear and non-linear operations in FALCON results in relatively long signing times.**

***Index Terms*—Threshold Cryptography, Lattice-based**

1. Introduction

The rapid development of quantum computing poses a significant threat to the security of classical cryptographic systems [1] [2]. Traditional encryption schemes, such as RSA and ECC, rely on the computational hardness of problems, which, however, can be solved in polynomial time by quantum computers [3]. As a result, these cryptographic systems, widely deployed e.g. in securing the internet and financial transactions are no longer considered future-proof in the face of advancing quantum technology. This pressing concern has led to the emergence of post-quantum cryptography (PQC), which is based on mathematical problems, that are believed to be hard even for quantum computers [3].

In parallel with the need for PQC, there is a growing interest in threshold cryptography. This technique strengthens security in distributed systems by decentralizing control across multiple participants [4] [3]. This approach has a wide range of applications, including cloud computing and blockchains [5].

As organizations work to secure data against quantum and other emerging threats, the combination of PQC and threshold cryptography offers a compelling solution for achieving both quantum resilience and enhanced fault-tolerant security in critical systems. In light of this, this paper provides an overview of lattice-based cryptography and threshold cryptography, followed by a survey of recently proposed threshold PQC schemes. It focuses on the works of Boneh et al. [6], Kamil et al. [7], and Cozzo et al. [8].

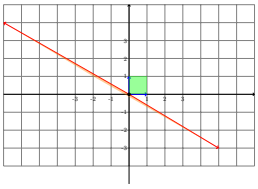


Figure 1: Two different basis vector pairs $\in \mathbb{R}^2$ build the fundamental domain $\mathbb{Z}^2$ [10].

2. Background

This section provides a brief overview of lattices and threshold cryptography.

2.1. Lattices

Lattices are as a key component of PQC due to their mathematical structure and the computational hardness of the problems they pose, which will be introduced in the following [9].

2.1.1. Lattice Fundamentals

Lattices are algebraic structures composed of points in n-dimensional space that are formed by the integer combinations of a set of basis vectors [9].

Definition 1. *A lattice $\mathcal{L}(B)$ is the span of its basis vectors $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ of $\mathbb{R}^n$ [9], so that*

$$\mathcal{L}(B) = \left\{ \sum a_i \mathbf{b}_i : a_i \in \mathbb{Z} \right\} \tag{1}$$

The cryptographic significance of lattices stems from the fact that a given lattice L can be represented by multiple bases [10]. While a "good" basis can simplify certain computational tasks, a "bad" basis can make these tasks exceedingly difficult [10]. For instance, in Figure 1, the red pair of long vectors and the blue pair of short vectors provide valid alternative bases for the same domain $\mathbb{Z}^2$. However, answering mathematical questions like "Is $(\mathbf{1}, \mathbf{0})^{\mathbf{T}} \in \mathcal{L}(B)$?" would be more challenging when using the long and nearly parallel vectors as B in Figure 1 (the "bad" basis). This is in contrast to the shorter, orthogonal vectors (the "good" basis). Beyond this, there exist other computational problems related to lattices that are considered hard and make lattices appealing for use in PQC, which will be presented in the following.

Definition 2. ***Shortest Vector Problem (SVP):*** *Given a lattice basis B and some norm $\|\cdot\|$, find a*

 doi: 10.2313/NET-2025-05-1_06

(nonzero) vector $\mathbf{v} \in \mathcal{L}(B)$ *such that* $\|\mathbf{v}\| = \lambda_{min}(\mathcal{L}(B))$*, where* λ_{min} *is the minimum distance in the lattice [9].*

Definition 3. ***Closest Vector Problem (CVP):*** *Given a lattice basis* B*, some norm* $\|\cdot\|$*, and an arbitrary vector* $q \in \mathbb{R}^n$*, find a lattice-point* $l \in L$ *such that* $\|l - q\|$ *is minimal [9].*

These problems can also serve as the foundation for other more practical, equation-based challenges. Two examples of such challenges follow.

2.1.2. Learning with Errors

The Learning With Errors (LWE) problem serves as the foundation for many PQC schemes [11]. It starts with a simple system of linear equations $A \cdot s = b$, where $A \in \mathbb{Z}^{n \times m}, s \in \mathbb{Z}^m, b \in \mathbb{Z}^n$. Solving this system can be done efficiently using standard techniques, such as Gaussian elimination. LWE complicates this by adding a small noise vector $e \in \mathbb{Z}^n$, leading to the system $A \cdot s = b + e$. The challenge now is to recover the secret vector s despite the added noise, which makes the problem computationally hard [11]. A formal definition of LWE follows.

Definition 4. ***Learning with Errors (LWE):*** *Let* $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ *be the ring of integers modulo* q*. Given are* A *and* b*, where:*

$$A \sim \mathbb{Z}_q^{n \times m} \quad \text{is a matrix sampled uniformly,}$$
$$b = A \cdot s + e \quad \text{a vector in } \mathbb{Z}_q^n \text{ with added noise } e$$

with $e \in \mathbb{Z}_q^n$ *as a small error vector. Recover the secret* $s \in \mathbb{Z}_q^m$ *[11].*

The hardness of LWE stems from its close connection to SVP and CVP [11]. For example, in an LWE instance of the form $A \cdot s = b + e$, where $A \in \mathbb{Z}_q^{n \times m}$, $s \in \mathbb{Z}_q^m$, $b \in \mathbb{Z}_q^n$, the matrix A can be viewed as a lattice basis, with each column representing a basis vector. As a lattice point is a linear combination of these basis vectors, $A \cdot s$ can be considered a lattice point, while b is a random vector in $\mathbb{Z}_q^n$. Given that the error vector e is small, solving an LWE instance to recover the secret vector s can almost always be done by solving CVP, searching for s with minimal $\|\mathbf{As} - \mathbf{b}\|$. A formal proof of LWE's hardness can be found in [11].

Moreover, the presented LWE-problem can be extended by introducing specific algebraic structures. For instance, the R-LWE uses a polynomial ring rather than a ring of integers as in standard LWE and uses polynomial multiplications, for which efficient algorithms that are similar to Fast-Fourier-Transformation can be used [11] [12]. A formal definition of R-LWE follows.

Definition 5. ***Ring Learning with Errors (R-LWE):*** *Fix a polynomial* $f(x)$*, consider the polynomial ring modulo* $f(x)$*, i.e.,* $\mathbb{Z}_q[x]/f(x)$*. Given are noisy samples* $(a_i(x), b_i(x))$*, where:*

$$a_i(x) \sim \mathbb{Z}_q[x]/f(x), \quad b_i(x) \leftarrow a_i(x) \cdot s(x) + e_i(x)$$

with $e_i(x) \in \mathbb{Z}_q[x]/f(x)$ *as a small error polynomial. Recover the secret* $s(x) \in \mathbb{Z}_q[x]/f(x)$ *[11].*

2.1.3. Inhomogeneous Short Integer Solution

Another practical hard problem is the Inhomogeneous Short Integer Solution Problem (ISIS) [13], which shares structural similarities with LWE but includes an additional constraint of a size bound. In fact, ISIS can also be reduced to CVP and SVP [13]. A formal definition of ISIS follows.

Definition 6. ***Inhomogeneous Short Integer Solution:*** *Given* $A \in \mathbb{Z}_q^{n \times m}$*,* $b \in \mathbb{Z}_q^n$*, and* $\beta \in \mathbb{R}$*, find* $s \in \mathbb{Z}_q^m$ *satisfying* $A \cdot s = y \mod q$ *with* $\|\mathbf{s}\|_2 \leq \beta$ *[13].*

2.2. Threshold Cryptography

In addition to lattices, threshold schemes can enhance security by distributing cryptographic operations or secrets across multiple participants [14]. The general concept of threshold cryptography will be introduced in the following.

2.2.1. Threshold Cryptography Fundamentals

In most large companies with hierarchical structures, significant decisions, such as signing major contracts, are typically made only after a majority of the board members reach an agreement. Threshold cryptography follows a similar principle of collaboration. In a (t, N)-threshold scheme, a secret key sk is split into N shares $(sk_1, ..., sk_N)$, with each share distributed to different participants. In cryptographic scenarios where the complete secret key sk is required, at least t participants must collaborate, combining their shares sk_i to reconstruct the key sk and complete their threshold task with it. This ensures that if an attacker compromises fewer than t participants or servers (e.g., $t-1$), they cannot reconstruct the full key sk and the system still remains secure [14] [15]. Main threshold tasks are introduced in the following.

Key generation (KGen): Two methods exist for KGen:

- *Generation by a trusted authority*: A trusted third party generates the public and private key pair (pk, sk). The secret key sk is then distributed among n participants using methods like (t, N) - Shamir Secret Sharing (SSS) [14]. This sharing method divides sk into N shares sk_i using Lagrange interpolation polynomials λ_i, ensuring that any $t-1$ shares reveal no information about the secret sk.
- *Distributed key generation (DKGen)*: Multiple participants jointly compute the public key pk and secret shares sk_i, which ensures that no single party has access to the complete secret key sk [16].

Threshold signatures: Any subset of t participants can collaborate to generate a signature, ensuring that the signature remains independent of the specific subset of t parties involved. Moreover, the signature size should be independent of t and N [15].

Threshold decryption: Any subset of t parties can decrypt a ciphertext ctx [15].

3. Analysis

This section presents and analyzes three influential contributions to lattice-based threshold PQC schemes.

3.1. Boneh et al. (2017)

The work by Boneh et al. [6] provides two primary contributions. First, they construct a threshold fully-homomorphic encryption (TFHE) scheme based on the LWE problem. Building on this framework, they con-

 doi: 10.2313/NET-2025-05-1_06

struct an "universal thresholdizer" [6], a tool capable of converting non-threshold cryptographic schemes into their threshold variant. These two key contributions are presented and analyzed in the following.

3.1.1. Threshold Fully Homomorphic Encryption

Boneh et al. construct a TFHE scheme, building on the existing FHE scheme developed by Gentry, Sahai, and Waters (GSW) [17], which is presented below.

Simplified GSW-FHE scheme [17]: Fix the message μ and the matrix G.

- **FHE.Setup** $\rightarrow (pk, sk)$: Sample a random matrix $\mathbf{A}$, a random vector $\mathbf{s}$, and a noise vector $\mathbf{e}$. Set $pk = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \end{pmatrix}$ and $\text{sk} = \begin{pmatrix} -\mathbf{s} & 1 \end{pmatrix}$.
- **FHE.Encrypt**$(pk, \mu) \rightarrow ctx$: Return ciphertext $ctx = \mathbf{A} \cdot R + \mu \cdot G$, where R is a random matrix with entries in $\{0, 1\}$.
- **FHE.Decrypt**$(pk, sk, ctx) \rightarrow \mu$: Compute the linear product $y = \langle sk, ctx^k \rangle$, where ctx^k is the kth column of the matrix ctx, and return 0 if y is small and 1 otherwise.

Using this scheme, one can encrypt the message with pk from FHE.Setup and evaluate an arbitrary cryptographic algorithm directly on the encrypted message, which explains its fully homomorphic capability. The decrypted result can be retrieved through FHE.Decrypt. The security of FHE.Encrypt relies on the hardness of the LWE problem, as a slightly modified LWE instance is constructed during FHE.Setup, with the introduction of a random matrix R during encryption. An example python implementation is available in [18].

The (t, N)-threshold variant of this scheme (TFHE) by Boneh et al. [6] consists of following steps: **TFHE.Setup**, **TFHE.Encrypt**, **TFHE.PartDec**, and **TFHE.FinDec**. In TFHE.Setup, instead of using a single secret key sk as in FHE.Setup, the secret is split into N secret shares sk_i, which are distributed by a trusted third party. For decryption, each party computes a partial decryption $p_i = \langle sk_i, ctx^k \rangle$ using their secret share sk_i during TFHE.PartDec. These partial decryptions p_i are then combined to reconstruct the full decryption p in TFHE.FinDec. For example, by employing (t, N)-SSS in TFHE.Setup, p can be successfully reconstructed in TFHE.FinDec using any t partial decryptions. However, directly combining the partial decryptions could potentially leak information about the secret shares due to the simple operations involved, such as the linear product. To address this, Boneh et al. introduce noise during TFHE.PartDec, modifying the decryption process to $p_i = \langle sk_i, ctx^k \rangle + noise$, which resembles the structure of a LWE instance $b = A \cdot s + e$.

3.1.2. Universal Thresholdizer

Using the constructed TFHE scheme and non-interactive zero-knowledge proofs (NIZK), Boneh et al. develop the universal thresholdizer [6], which is presentend in the following.

UT: Fix a circuit C of a cryptographic scheme and a subset U of t parties.

- **UT.Setup**$(\mu) \rightarrow (pp, sk_1, \ldots, sk_n)$: Generates the public key pk and secret shares sk_i from TFHE.Setup, computes the ciphertext ctx with TFHE.Encrypt(μ), and the commitment $\text{com}_i = \text{Com}(sk_i)$. The public parameters pp are defined as $(pk, ctx, \{\text{com}_i\}_{i \in [N]})$.
- **UT.Eval**$(pp, sk_i, C) \rightarrow (ctx', p_i, \pi_i)$: Evaluates the given circuit C on the ciphertext ctx, producing the evaluated ciphertext ctx' and computes the partial decryption $p_i = \text{TFHE.PartDec}(pk, ctx', sk_i)$. It then generates a NIZK proof π_i regarding the correctness of pp and sk_i.
- **UT.Verify**(pp, C, p_i, π_i): Checks the NIZK proof π_i.
- **UT.Combine**$(pp, \{p_j\}_{j \in U}) \rightarrow p$: Combines partial decryptions p_i and reconstruct p using TFHE.FinDec.

Using this scheme, any cryptographic protocol, such as a signature scheme, can be transformed into its threshold version. First, the cryptographic protocol is encoded into a circuit C composed of gates. After UT.Setup, each party generates a partial signature during UT.Eval by using the encoded circuit as input. The partial signatures are then verified and combined using UT.Verify and UT.Combine, respectively.

Although this approach of UT offers flexibilities with its capability to thresholdize non-threshold schemes, it also presents certain limitations. First, it relies on a trusted third party in UT.Setup, lacking DKGen, which is considered more secure. Second, executing an entire circuit in TFHE can be computationally expensive, particularly when the circuit involves heavy steps like challenges or rejection sampling. Moreover, the noise flooding in TFHE.PartDec leads to inefficient scaling complexities, with $\Omega(N \log N)$ for sizes of secret key shares and $\Omega(\lambda^3)$ for signatures, where λ indicates the security parameter [6]. These drawbacks are addressed in subsequent work.

3.2. Kamil et al. (2023)

Kamil et al. [7] address restrictions of UT [6] with the following solutions:

1) Instead of TFHE, they utilized a threshold linearly homomorphic encryption (THE) scheme, which is selectively applied only to the relevant steps of the protocol, rather than to the entire circuit. Additionally, the constructed THE scheme incorporates DKGen.
2) They combine the state-of-the-art (n, n) threshold pq-signature scheme by Damgård et al. [19] with the constructed THE and DKGen, extending it to a (t, N)-threshold scheme.

3.2.1. THE With DKGen

Kamil et al. construct a THE scheme as a threshold variant of the R-LWE-based HE scheme by Brakerski et al. [20] and extend this with DKGen based on (t, N)-SSS [14]. Morover, they take care to add noise when combining partial decryptions p_i, following the same rationale as Boneh et al. [6]. The simplified version of their built scheme is presented below.

 doi: 10.2313/NET-2025-05-1_06

- **THE.DKGen [7]:** Fix a ring element $a_E \in R_q$, a small prime number k, and a subset U of t parties.
  1) Every party P_i samples $s_i, e_i \in R_q$ and computes $b_i = a_E \cdot s_i + k \cdot e_i$ and its (t, N)-Shamir secret shares $s_{i,j}$ of s_i.
  2) P_i sends $(b_i, s_{i,j})$ to every other party P_j
  3) Every party P_i computes its public key $pk = (a_E, b_E = \sum b_j)$ and secret share $sk_i = \sum s_{j,i}$.
- **THE.Encrypt**$(pk, \mu) \rightarrow (ctx)$: Samples $r, e', e'' \in R_q$ and outputs $ctx = (u, v) = (a_E \cdot r + k \cdot e', b_E \cdot r + k \cdot e'' + \mu)$
- **THE.PartDec**$(ctx, sk_i) \rightarrow (p_i)$: Samples noise E_i and output partial decryption $p_i = \lambda_i \cdot sk_i \cdot u + k \cdot E_i$ with λ_i being the Lagrange multiplier for party P_i.
- **THE.FinDec**$(ctx, \{p_j\}_{j \in U}) \rightarrow (p)$: Outputs the complete decryption $p = (v - \sum_{j \in U} p_j) \bmod q \mod k$.

In essence, Kamil et al. extend the base HE scheme [20] to THE by including SSS and generating individual R-LWE instances, each with its error e_i. Accordingly, the security of the scheme relies on the R-LWE assumption from the base scheme [20]. C++ implementations of the base scheme and a similar version of THE are available in [21].

3.2.2. Combining THE with scheme by Damgård et al.

Based on the existing (n, n)-signature scheme by Damgård et al. [19], Kamil et al. utilize steps from the constructed THE to extend this to an arbitrary threshold scheme. The following presents the simplified protocols, with the modified steps highlighted in bold.

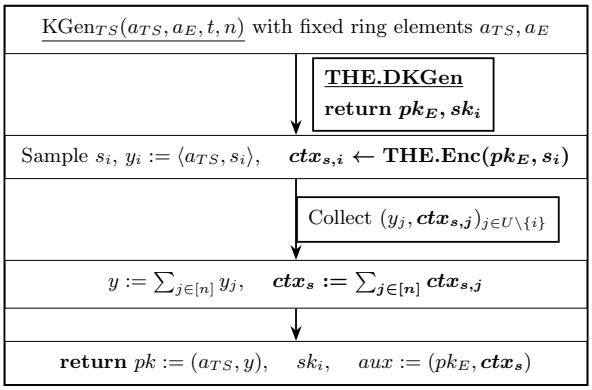


Figure 2: Passively secure (t, N) KGen protocol [22].

- **KGen:** The protocol first triggers THE.DKGen to generate the public key pk_E and the secret key share sk_i. Building on the existing protocol by Damgård et al. [19], it homomorphically encrypts the randomly generated s_i. The combined encrypted randomness, ctx_s, is then computed and returned as auxiliary information for subsequent operations.
- **Signing:** During signing, the protocol homomorphically encrypts the randomness r_i and computes the encrypted signature ctx_z by combining the encrypted random values $ctx_{r,j}$ from other parties, the auxiliary value ctx_s from KGen, and the challenge c. A subset of t parties then cooperatively decrypt ctx_z homomorphically, yielding the final signature z after combining the partial decryptions p_j.

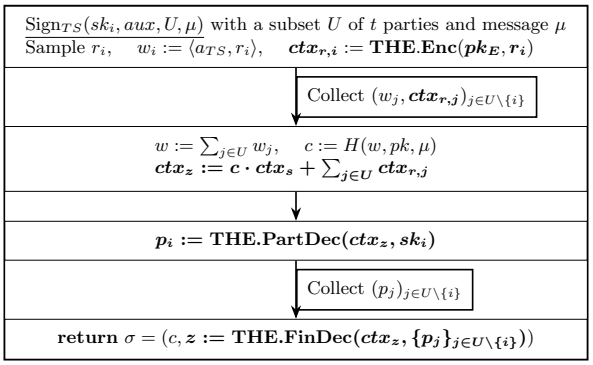


Figure 3: Passively secure (t, N) signing protocol [22].

In short, Kamil et al.'s modified protocol enhances the existing (n, n)-signature scheme by introducing randomness into KGen and generating the encrypted signature ctx_z using THE during signing, instead of the unencrypted signature z. This modification successfully extends the original protocol into a (t, N)-threshold variant. Furthermore, in contrast to UT, which requires a non-threshold scheme as input, this scheme is inherently a threshold scheme and does not depend on any external scheme. At the 128-bit security level with $t = 3$ and $N = 5$, this scheme produces signatures of size 46.6 kB of size 13.6 kB [7], offering a significant improvement in efficiency compared to other recent protocols such as Threshold raccoon [23].

3.3. Cozzo et al. (2019)

In contrast to the two presented works that utilize HE [6] [7], Cozzo et al. [8] discuss possibilities to thresholdize promising PQC schemes such as FALCON [24] based on secure multiparty computation (MPC) [25]. Non-threshold version of FALCON will be presented first.

Input : μ, $sk = B \in R_q^{2\times 2}$ with small $B_{i,j}$, $pk \in R_q^2$ with $B \cdot pk = \mathbf{0}$
Output: σ
1. $c \leftarrow H(r \parallel \mu)$ with random $r \in \{0,1\}^{320}$
2. Sample $t \in R_q^2$ such that $pk^\top \cdot t = c$.
3. $z \leftarrow \text{round}(t, B)$ such that $z \in \mathcal{L}(B)$
4. $s \leftarrow (t - z) \cdot B$
5. If s is not short, return to 3
6. Return $\sigma = (r, s)$

Figure 4: Simplified FALCON signing protocol [24]

FALCON follows the concept of hash-and-sign. It begins by computing a hash value c and search for t from an ISIS instance $pk^\top \cdot t = c$, which is possible by computing $t \leftarrow (c, 0) \cdot B^{-1}$ [24]. Afterwards, t is rounded to a close lattice-point $z \in \mathcal{L}(B)$ and the signature s is computed with the difference between t and z. Plus, a rejection sampling with the condition checking the shortness of s is included to enhance security of B. FALCON is fairly runtime efficient and compact, requiring only 0.3 miliseconds for signing and prodzcing signatures of 1.3 kB [24].

To thresholdize FALCON, Cozzo et al. [8] suggest utilizing MPC, enabling multiple parties to jointly perform computations using their individual inputs while ensuring the privacy of those inputs [25]. First, a linear secret sharing scheme (LSSS) can be used to distribute sk and pk, enabling linear operations to be performed

 doi: 10.2313/NET-2025-05-1_06

on the secret shares rather than directly on the secret key [26]. Furthermore, considering that FALCON involves both linear operations (e.g. step 2 in Figure 4) and non-linear ones (e.g. rejection sampling), LSSS-based MPC schemes are well suited for the linear operations, while garbled circuits (GC)-based MPC can handle the non-linear components [8].

This separation of needed MPC techniques requires costly conversions between LSSS and GC representations, leading to a major bottleneck and a longer signing time of 5.7 seconds [8]. Moreover, this threshold-FALCON possesses further limitations such as the absence of DKGen.

4. Conclusion

In conclusion, the presented three papers show recent advancements on threshold PQC schemes, presenting different ways to design threshold schemes. Boneh et al. [6] introduced the "universal thresholdizer," a tool capable of transforming any cryptographic scheme into its threshold variant through a black-box execution. Kamil et al. [7] identified remaining inefficiencies in the "universal thresholdizer" and proposed a new scheme using THE instead of TFHE. Cozzo et al. [8] utilize LSSS and MPC techniques, rather than HE, to thresholdize FALCON. Future work could focus on further optimizing the protocol proposed by Kamil et al. [7], e.g. by introducing compression techniques used in schemes like FALCON [24] or DILITHIUM [27] [28]. Moreover, exploring PQC schemes based on other mathematical primitives beyond lattices [29], such as hash functions or isogenies, presents an avenue for further research.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, p. 1484–1509, Oct. 1997. [Online]. Available: http://dx.doi.org/10.1137/S0097539795293172

[2] B. Brubaker, "Thirty years later, a speed boost for quantum factoring," 2023, quanta Magazine, October 17, 2023. [Online]. Available: https://www.quantamagazine.org/thirty-years-later-a-speed-boost-for-quantum-factoring-20231017/

[3] K. Sedghighadikolaei and A. A. Yavuz, "A comprehensive survey of threshold signatures: Nist standards, post-quantum cryptography, exotic techniques, and real-world applications," in *Proceedings of the arXiv Conference*, no. 2311.05514, 2024, pp. 1–2. [Online]. Available: https://arxiv.org/abs/2311.05514

[4] L. T. A. N. Brandão and R. Peralta, "Nist first call for multi-party threshold schemes (initial public draft)," National Institute of Standards and Technology (NIST), Tech. Rep. NIST IR 8214C IPD, 2023, january 2023. [Online]. Available: https://doi.org/10.6028/NIST.IR.8214C.ipd

[5] M. E. Manaa and Z. G. Hadi, "Scalable and robust cryptography approach using cloud computing," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 23, no. 7, pp. 1439–1445, 2020.

[6] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai, "Threshold cryptosystems from threshold fully homomorphic encryption," vol. 10, no. 1, 2024.

[7] K. D. Gur, J. Katz, and T. Silde, "Two-round threshold lattice-based signatures from threshold homomorphic encryption," in *Proceedings of the Cryptology ePrint Archive*, ser. Lecture Notes in Computer Science (LNCS), vol. 1318. Springer, 2023. [Online]. Available: https://eprint.iacr.org/2023/1318

[8] D. Cozzo and N. P. smart, "Sharing the LUOV: Threshold post-quantum signatures," Cryptology ePrint Archive, Paper 2019/1060, 2019. [Online]. Available: https://eprint.iacr.org/2019/1060

[9] J. H. Silverman, "An introduction to lattices, lattice reduction, and lattice-based cryptography," *IAS/Park City Mathematics Series*, pp. 1–5, 2023.

[10] T. Laarhoven, J. van de Pol, and B. de Weger, "Solving hard lattice problems and the security of lattice-based cryptosystems," in *Proceedings of the Cryptology ePrint Archive*, no. 533. International Association for Cryptologic Research (IACR), 2012, pp. 2–4. [Online]. Available: https://eprint.iacr.org/2012/533

[11] O. Regev, "The learning with errors problem," *Survey on Learning with Errors (LWE)*, pp. 1–23, 2005. [Online]. Available: https://cims.nyu.edu/~regev/papers/lwesurvey.pdf

[12] C. Peikert, O. Regev, and N. Stephens-Davidowitz, "Pseudorandomness of ring-LWE for any ring and modulus," in *Proceedings of the Cryptology ePrint Archive*, no. 258. International Association for Cryptologic Research (IACR), 2017. [Online]. Available: https://eprint.iacr.org/2017/258

[13] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," Cryptology ePrint Archive, Paper 2007/432, 2007. [Online]. Available: https://eprint.iacr.org/2007/432

[14] A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[15] Yvo Desmedt and Yair Frankel, "Threshold Cryptosystems," *Advances in Cryptology*, pp. 305–315, 1989, cRYPTO '89.

[16] C. Komlo, I. Goldberg, and D. Stebila, "A formal treatment of distributed key generation, and new constructions," in *Proceedings of the Cryptology ePrint Archive*, no. 292. International Association for Cryptologic Research (IACR), 2023. [Online]. Available: https://eprint.iacr.org/2023/292

[17] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Proceedings of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2013.

[18] K. Teranishi, "ECLib: An open-source homomorphic encryption library," 2024, accessed: 2024-09-22. [Online]. Available: https://github.com/KaoruTeranishi/EncryptedControl

[19] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi, "Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices," in *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I*, ser. Lecture Notes in Computer Science (LNCS), J. Garay, Ed., vol. 12710. Virtual Event, May 10–13: Springer, Heidelberg, 2021, pp. 99–130.

[20] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, S. Goldwasser, Ed. Cambridge, MA, USA, January 8–10: Association for Computing Machinery, 2012, pp. 309–325.

[21] A. A. Badawi, A. Alexandru, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, C. Pascoe, Y. Polyakov, I. Quah, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca, "OpenFHE: An open-source fully homomorphic encryption library," 2022, cryptology ePrint Archive, Paper 2022/915, Accessed: 2024-09-22. [Online]. Available: https://eprint.iacr.org/2022/915

[22] K. D. Gur, J. Katz, and T. Silde, "Two-round threshold lattice-based signatures from threshold homomorphic encryption," in *Proceedings of the Cryptology ePrint Archive*, ser. Lecture Notes in Computer Science (LNCS), vol. 1318. Springer, 2023, pp. 18–19. [Online]. Available: https://eprint.iacr.org/2023/1318

[23] R. del Pino, S. Katsumata, M. Maller, F. Mouhartem, T. Prest, and M.-J. Saarinen, "Threshold raccoon: Practical threshold signatures from standard lattice assumptions," Cryptology ePrint Archive, Paper 2024/184, 2024. [Online]. Available: https://eprint.iacr.org/2024/184

[24] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-fourier lattice-based compact signatures over ntru," Falcon Project, Tech. Rep., 2020, specification v1.2 — 01/10/2020. [Online]. Available: https://falcon-sign.info/

[25] Y. Lindell, “Secure multiparty computation (mpc),” *Unbound Tech and Bar-Ilan University*, 2019, accessed: 2024-10-18. [Online]. Available: https://eprint.iacr.org/2020/300.pdf

[26] R. Cramer, I. B. Damgård, N. Döttling, S. Fehr, and G. Spini, “Linear secret sharing schemes from error correcting codes and universal hash functions,” in *Proceedings of Eurocrypt 2019*. Springer, 2019. [Online]. Available: https://www.iacr.org/archive/eurocrypt2015/90560182/90560182.pdf

[27] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, “Crystals-dilithium: A lattice-based digital signature scheme,” *Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, Issue 1, pp. 238–268, 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/839

[28] C.-D. Team, “CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme (GitHub Repository),” 2017, accessed: 17 September 2024. [Online]. Available: https://github.com/pq-crystals/dilithium

[29] M. Buser, R. Dowsley, M. F. Esgin, C. Gritti, S. K. Kermanshahi, V. Kuchta, J. T. Legrow, J. K. Liu, R. C.-W. Phan, A. Sakzad, R. Steinfeld, and J. Yu, “A survey on exotic signatures for post-quantum blockchain: Challenges & research directions,” *Cryptology ePrint Archive*, vol. 1, no. 1, pp. 4–7, 2022. [Online]. Available: https://eprint.iacr.org/2022/1151.pdf

Current Limitations in Digital Map Modelling

Kilian Matheis, Johannes Späth*, Florian Wiedner*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: kilian.matheis@tum.de, spaethj@net.in.tum.de, wiedner@net.in.tum.de

***Abstract*—With the increased complexity of managing networks, the need for Digital Twins has been growing rapidly. These are based on digital maps of networks. The IETF has been working on standardising the modelling of networks and their topologies and proposed their base standard in RFC 8345. In this paper we therefore share our experience with modelling an Autonomous System based on RFC 8345. We summarise the current limitations and the possible solutions for them. The two main issues encountered were the lack of bidirectional links and the absence of links between networks. The proposed augmentations to RFC 8345 we and other researchers suggest based on shared experiences take away unnecessary complexity and offer backwards compatibility.**

***Index Terms*—RFC 8345, digital map modelling, digital twin, bidirectional links**

1. Introduction

The rapid growth of networks and rising customer demands require dynamic adaptation, therefore creating new challenges for operators. Managing these advanced networks and services is complex, and introducing innovations is risky and costly without reliable emulation platforms [1]. A rising technology to help manage these networks are Digital Twins. Zhou et al. [1] define a Network Digital Twin (NDT) as 'a digital representation that is used in the context of networking and whose physical counterpart is a data network' [1]. These NDTs can help in many ways. E.g., one of the main applications for an NDT is the testing of so called 'what-if' scenarios. Being able to test a network for possible points of failure or to test new functionalities without having to test in production is not only essential for a good consumer/user-experience but also for the maintaining institutions. Furthermore, there is the possibility to turn a network into an Autonomous Driving Network, which regulates the network autonomously using NDTs as possible test benches or simply as data input.

A Digital Map (DM) represents the topology information for a given network. It provides the core topological entities, their role in the network, core properties and relationships between networks and entities on a multi-level topology [2]. The building of digital maps is essential for digital twins [3].

We modelled a network based on the guidelines and standards provided by the IETF. Previous research has highlighted several limitations in the current version of RFC 8345, particularly regarding bidirectional links and cross-network connectivity [4] [5]. Various augmentations have been proposed to address these challenges, including improved guidelines for multi-layer network modeling [4]. Our work builds on these findings, offering a practical evaluation of the proposed solutions.

The rest of this paper is structured as follows: Chapter 2 presents necessary background information about the IETF standard RFC 8345 and Chapter 3.1 gives a short overview of our network we modelled. In Chapter 3 we cover the limitations we found and what proposed solutions and workaround we could find and also present our own opinion on how these limitations should be handled in the future and Chapter 4 summarises the limitations and the proposed solutions and gives an outlook into future work.

2. Understanding RFC 8345 – A YANG Data Model for Network Topologies

Because networks have grown in size and become more dynamic there was and still is a need for real-time topology data to support automation and programmability. To ensure interoperability and integration across multi-layer multi-vendor networks, the need for a standardised topology description has been growing. In 2018, to meet those demands, the IETF formalised their RFC 8345 [6], in which they proposed a standard describing network topologies in YANG, a data modelling language [7]. YANG organises data into a hierarchical tree structure using constructs such as containers, lists, leafs, and leaf-lists to represent complex configurations and state information. This modular approach allows for the creation of reusable and interoperable models that facilitate efficient network management and automation.

2.1. Structure of RFC 8345

The IETF introduces the data model in two divided parts. This design decision has been made to separate the integration of network topology and network inventory models. It allows the augmentation for inventory information without knowing anything about the underlaying topology of the network. According to the IETF the standard has purposefully been kept very abstract and generic so the model can represent any kind of network [6].

2.1.1. Base Network Model (ietf-network). The `ietf-network` module defines the base network data model. The model includes a container that holds a list

 doi: 10.2313/NET-2025-05-1_07

of networks. Each network has its own entry and is unique through its primary-key `network-id`. A network can pose multiple `network-types`. This container acts as a target for augmentation, therefore it is implemented as a container, rather than an empty leaf. This approach supports hierarchical representations of network subtypes.

To model network hierarchies, a network has a list of `supporting-networks` with references to underlay networks. For a Layer-3 network this could be a reference to a Layer-2 network.

To model parts of the network the RFC 8345 introduces so-called nodes. A node is intended as an abstract construct, meaning in one network it can model a physical device and in a different network it could be a processor or a router. Each node is uniquely identified by its node-id and strictly bound to its network. Each network has its own list of nodes. Just like networks can have supporting-networks, a node can have `supporting-nodes` in underlay networks [6].

2.1.2. Base Network Topology Model (ietf-network-topology). The `ietf-network-topology` module defines the base network topology data model. It augments `ietf-network` from above. To describe the topology of networks in an abstract way, it adds two crucial elements, links and termination points.

Nodes get extended by a list of termination-points (TPs). Just like nodes, a termination point is an abstract construct unique by its `tp-id` that represents one end of a link. This could be an interface. Termination points can have `supporting-termination-points` in underlaying nodes of underlaying networks.

Links are captured in a list in networks. Each link is uniquely identified by its key `link-id`. A link consists of a source and destination, both represented by the before mentioned termination-points. A link can also be supported by underlaying links in underlay networks. Links are point-to-point and only unidirectional [6].

2.2. Augmentations of RFC 8345

The YANG model, as mentioned above, has been kept very generic. This allows for YANG augmentations to the two YANG modules. The main focus in this paper is on augmentations on `ietf-network-topology`. Havel et al. [4] analysed a plethora of augmentations and came to the conclusion that work needs to be done on creating guidelines for augmentations as the current state is not consistent and therefore very hard to combine them into multi-layer networks based on one single standard.

3. Limitations of the Current RFC 8345 Version

To represent our test-network we wanted to describe our system using the proposed RFC 8345 standard. During this process, we experienced current limitations of the base model, which have also already been encountered by other researchers [5] [4]. This motivated us to go into further research and find existing workarounds or proposed solutions to RFC 8345 and evaluate them.

3.1. Our Autonomous System Network

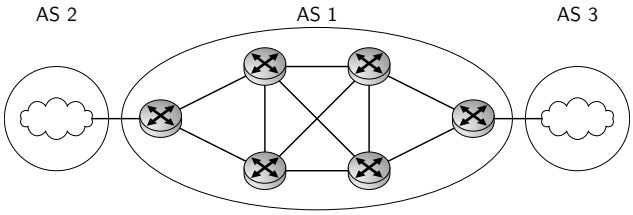


Figure 1: Network topology diagram

In our study of Autonomous Driving Networks, we modelled an Autonomous System (AS) comprising three interconnected systems. In our system we wanted to transfer files from AS2 to AS3 through AS1. Our AS1 consists of six routers. as1-host1 is connected to AS2 and as1-host6 is connected to AS3. The six routers are set up so that for data to flow from host-1 to host-6 it has to go through at least two more routers. All connections are wired, therefore bidirectional. Between the systems, the Border Gateway Protocol (BGP) is used, and our middle-system, AS1, uses the Intermediate System to Intermediate System Protocol (IS-IS).

3.2. Missing Bidirectional Links

The first limitation we experienced is that there is currently no way to model a link as bidirectional as "Links are point-to-point and unidirectional" [6]. This is an intentional design decision made by the IETF. The decision has been made to be in line with the philosophy to keep the standard on a generic level. The IETF further reasons that unidirectionality improves the applicability of graph algorithms. Although Havel et al. [5] regard this as inaccurate in practise, as in most cases further transformations have to be made before applying a graph algorithm of any kind.

3.2.1. Proposed Solutions. First it has to be said that the IETF, aware of the debate on this challenge, not only states their above mentioned reasons, but also offers their way of representing bidirectional-links. They want them to be modelled as two separate and independent uni-directional links [6]. This is a quite simple way to circumnavigate this challenge. From our perspective, this still seems rather unnecessary as it doubles the amount of links and therefore the networks complexity. This decision and the resulting independence of the two parts of a link that is supposed to be a one logical unit create the possibility of just one unidirectional link existing, therefore increasing the rate of errors.

This challenge can be solved simply through a small augmentation by adding a `direction-of-link` to the link container in `ietf-network-topology`. If one only wants to address this directionality challenge, this augmentation would be sufficient. This approach is fully backward compatible.

3.2.2. Evaluation. We would advise against resolving this limitation by oneself through augmentation and rather accept the additional complexity by modelling bidirectional

links with two unidirectional ones. As the solution to this limitation is fully backwards compatible and easy to implement, we recommend the IETF to implement it in an updated version of the RFC 8345. From our perspective, the benefits would outweigh the drawbacks and an augmentation regarding this would not go against the principle of generic design. Our view is in line with other papers [5] [4].

3.3. Missing Links Between Networks

The second limitation we encountered when we modelled our AS-System is the constriction on link paths. Links can only have nodes and termination points in the same network [6]. Therefore links between two (or more) separate networks are not allowed. This poses a problem for our intended description as our network to model has 3 different AS domains (AS1, AS2, AS3).

3.3.1. Proposed Solutions. If one wants to model links between two different networks using only RFC 8345, there is just one way to approach this problem. One needs to create a new, higher-level, network of domains [6]. In our case this would be a network with nodes AS1, AS2, AS3 and links connecting AS1 with AS2 and AS2 with AS3. Then describing those networks in detail in their own network. To model the links between them you would need to create duplicate nodes for the devices in the neighbouring networks and then create a link to this node from the corresponding note in the current network. This creates/emphasises another limitation that will be explained in chapter 3.5.

A more complex solution in the current state of RFC 8345 is to use the augmentation for Traffic Engineering (TE) [8]. TE-topologies are specialised augmentations used for traffic engineering, which often necessitate additional complexity due to their specific application in optimising traffic flow in complex networks. This adds the possibility to link nodes between two different networks by adding a network-ref to the source and destination. This change in reference makes it unfortunately impossible for programs that only understand RFC 8345 to retrieve the topology of the network [4]. This goes against the original intent of the standard. Additionally it adds possible unwanted complexity to the describing as the network has to be of type `te-topology`.

Modelling a IS-IS Topology makes this challenge even more apparent IS-IS areas and IS-IS Domains can currently not be modelled without using a workaround. De Dios et al. [9] use node-attributes to define the IS-IS area. This method makes it impossible for a RFC 8345 to understand the network and its topology without having prior IS-IS knowledge.

3.3.2. Evaluation. We recommend extending the RFC 8345 to allow links between nodes in separate networks by adding a network-ref in both source and destination of a given link. This solution would also be backward compatible [4]. This extension would minimise the complexity to model AS/IS-IS/OSPF networks, allow for simple RFC 8345 algorithms to understand those networks. We do not think that this addition goes against the spirit of having an abstract base model, because multiple connected networks are of high occurrence and not technology specific. This view is shared by Havel et al. [4]. Technology-specific augmentations, e.g. IS-IS and Open-Shortest-Path-First (OSPF), are also on hold regarding this challenge until it will be addressed in the base module.

3.4. Missing Option for Subnetworks/Partitioning

This limitation is closely related to the limitation on links between networks from Section 3.3. Currently, RFC 8345 does not provide a method for describing one network as part of another. This limitation is especially relevant in protocols such as OSPF and IS-IS, where networks are often divided into areas that need to be modeled as part of a larger structure. Havel et al. [4] propose to add a simple part-of relationship between networks to RFC 8345 . This proposal is aligned with our view, because it would greatly simplify the modelling of our own AS domain. It is also backward compatible.

3.5. Duplicate Nodes, TPs in Multiple Networks

In the current version of RFC 8345, nodes and TPs are only allowed to be in one network, meaning that if the same physical or logical node or TP exists in two networks, it must have different keys and therefore be two independent nodes from a modelling point-of-view. This would be the case for our AS-Domain as well, if we choose to use the workaround with two duplicate nodes in their respective network from Section 3.3. Havel et al. [4] state that this is the case with OSPF Networks and shows the possibility to have this challenge in IS-IS. This introduces the challenge of consistency when writing to a model, because one would have to assume that all duplicate nodes are in the same state. This is not guaranteed by the model as they are totally independent from the models point of view [4].

3.5.1. Proposed Solutions. In our case we would have to model the as1-host1, as2-host1 routers in the as1 AND the as2 network and the as2-host6 and as3-host1 routers in the as2 AND as3 network. The IETF is aware of the challenge and proposes the solution to have an extra network of physical nodes that represent the existing devices and are linked to the logical nodes from networks through the supporting-node logic [6]. In our case we would have to create a 5th physical-network with all our routers and then add those routers to their respective nodes in as1, as2 and as3.

Havel et al. [4] also propose to allow the definition of nodes outside the scope of networks. This solution would likely be backwards compatible but greatly alter the topology tree and therefore needs to be researched further.

3.5.2. Evaluation. In our opinion this limitation is, just like the limitation in Chapter 3.3, not technology specific and it would only be sensible to allow nodes to be defined outside of networks, to make them uniquely identifiable across multiple networks, as long as the solution is backwards compatible. Allowing nodes to exist independently of specific networks would streamline the

process of maintaining consistency across domains, as it would eliminate the need to synchronise duplicate nodes and their states/interfaces.

3.6. Missing Multi-Point Links

While modelling our Autonomous System, we did not initially encounter the absence of multipoint links as a limitation. However, during our research into the other shortcomings of RFC 8345, this challenge became apparent. Multipoint links, which enable the connection of multiple devices through a single logical link, are a critical feature in many modern network environments such as Layer 2 topologies. For instance, in Virtual-Local-Area-Network (VLAN) configurations, multipoint links allow multiple endpoints to communicate across a shared infrastructure. Unfortunately, RFC 8345 does not natively support this type of connection. Instead, it only allows for point-to-point links, which limits the ability to accurately model networks that require multipoint communication. This limitation originates from the core design of RFC 8345, where links are described as unidirectional and strictly point-to-point.

3.6.1. Proposed Solutions. The current workaround proposed by RFC 8345 is to model multipoint connections using pseudonodes [6]. Pseudonodes act as intermediary points to represent multipoint connectivity indirectly. By creating a new node for each multipoint connection, network designers can model the individual connections as point-to-point links between the devices and the pseudonode. However, this approach significantly increases network complexity, making the model more difficult to manage and prone to inconsistencies. The RFC acknowledges this limitation but offers no further improvements, stating that this method preserves the standard's generic, technology-agnostic design.

An alternative solution, discussed by Davis et al. [5], suggests augmenting RFC 8345 to allow links to directly have multiple termination points, enabling more effective support for multipoint connections. This approach would preserve backward compatibility, ensuring that existing models remain functional while simplifying the representation of networks using multipoint links. Like the bidirectional link augmentation mentioned earlier in Section 3.2, this enhancement would streamline the modeling process by reducing redundant link definitions, making network maps cleaner and more manageable. It introduces an additional, optional link-end definition, which fits within the current structure, ensuring that current point-to-point links continue to operate without modification. The simplicity of this solution would allow for broad adoption without disrupting existing network models.

A more advanced approach is also suggested to improve the integrity of the existing model [5]. This method involves extracting the current source and destination structures and then re-augmenting them back into the link, all within the same module. Since there is no namespace change, the augment remains within the module. While this solution is not fully backward compatible with YANG, it still produces the same instance structures (e.g., in JSON) and supports any existing augmentations for source and destination. One major advantage of this approach is that it allows for point inclusion to be controlled based on feature support, effectively separating the structures supporting existing capabilities from those designed to handle new functionalities [5].

3.6.2. Evaluation. From our perspective, we strongly advise the adoption of the simpler, backward-compatible solution into RFC 8345. This approach would resolve the limitation of multipoint links without increasing the complexity of the model unnecessarily, allowing for easier and more accurate network representation. Given the high occurrence of multipoint connections in modern networks, this change would make RFC 8345 far more applicable in real-world scenarios while preserving the standard's abstract and generic nature.

4. Conclusion

In summary, our evaluation of RFC 8345 has brought to light several limitations that complicate network topology modeling, particularly in Autonomous Systems. Key challenges include the lack of support for bidirectional links, restrictions on inter-network connectivity, and the absence of a mechanism for representing subnetworks or partitions. While manageable through workarounds, these issues unnecessarily increase complexity and risk of inconsistency.

To improve the applicability of RFC 8345, we believe the IETF should consider adjustments that remove these limitations while retaining backward compatibility. Specifically, enabling bidirectional links and allowing cross-network references would provide a more accurate and streamlined model for real-world networks. These enhancements would preserve the flexibility and abstraction of the standard, while making it more practical for advanced network architectures. Addressing these concerns would lead to more efficient implementations and broader adoption within the networking field.

By addressing these limitations, RFC 8345 would better support the development of Digital Twins and other advanced network architectures, ensuring its relevance in the evolving landscape of network management.

Further work has to be done on how augmentations for specific layers and protocols can or need to be changed when these improvements to RFC 8345 are implemented.

References

[1] C. Zhou, H. Yang, X. Duan, D. Lopez, A. Pastor, Q. Wu, M. Boucadair, and C. Jacquenet, "Network Digital Twin: Concepts and Reference Architecture," Internet Engineering Task Force, Internet-Draft draft-irtf-nmrg-network-digital-twin-arch-06, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/06/

[2] Benoît Claise, "Digital Map Modelling: IETF 117 Latest Developments," 2023, available online at https://www.claise.be/digital-map-modelling-ietf-117-latest-developments/; last accessed on 2024/09/18.

[3] O. Havel, B. Claise, O. G. de Dios, and T. Graf, "Digital Map: Concept, Requirements, and Use Cases," Internet Engineering Task Force, Internet-Draft draft-havel-nmop-digital-map-concept-00, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-havel-nmop-digital-map-concept/00/

[4] O. Havel, B. Claise, O. G. de Dios, A. Elhassany, and T. Graf, "Modeling the Digital Map based on RFC 8345: Sharing Experience and Perspectives," Internet Engineering Task Force, Internet-Draft draft-havel-nmop-digital-map-01, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-havel-nmop-digital-map/01/

[5] N. Davis, O. Havel, and B. Claise, "Some Refinements to Network Topologies (RFC8345)," Internet Engineering Task Force, Internet-Draft draft-davis-nmop-some-refinements-to-rfc8345-00, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-davis-nmop-some-refinements-to-rfc8345/00/

[6] A. Clemm, J. Medved, R. Varga, N. Bahadur, H. Ananthakrishnan, and X. Liu, "A YANG Data Model for Network Topologies," RFC 8345, Mar. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8345

[7] M. Björklund, "The YANG 1.1 Data Modeling Language," RFC 7950, Aug. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7950

[8] X. Liu, I. Bryskin, V. P. Beeram, T. Saad, H. C. Shah, and O. G. de Dios, "YANG Data Model for Traffic Engineering (TE) Topologies," RFC 8795, Aug. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8795

[9] O. G. de Dios, S. Barguil, V. Lopez, D. Ceccarelli, and B. Claise, "A YANG Data Model for Intermediate System to intermediate System (IS-IS) Topology," Internet Engineering Task Force, Internet-Draft draft-ogondio-nmop-isis-topology-00, Mar. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ogondio-nmop-isis-topology/00/

Usage of Path Property Emulation Tools

Julien Schiffer, Stefan Lachnit*, Sebastian Gallenmüller*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: schiffer.ju@tum.de, lachnit@net.in.tum.de, gallenmu@net.in.tum.de

***Abstract*—There are many different emulation tools, some with similar but different functions and goals. The most common ones are Mininet and NetEm. They are often used to test and validate the work of researchers. This includes new algorithms or programs. However, it is not always clear why a particular emulation tool was used. The following paper presents and categorizes the use of those tools. Meanwhile, it becomes clear that current emulation tools have technical limitations and are sometimes reaching their limits, which is why there will be even more powerful tools in the future.**

***Index Terms*—Network Emulation, Path property, Emulation tools, Mininet, Netem**

1. Introduction

As the Internet grows, the corresponding programs and requirements become more complex. It is not only important that applications work at all, but speed also plays a significant role. There are a lot of devices with different hardware on which the programs should still be usable quickly. Therefore, tests must be carried out to check how programs behave with other parameters, such as high latency and bandwidth. The aim is to recreate real network conditions. Entire virtual networks can also be created to carry out tests.

The use of such tools offers many advantages compared to testing on real networks. The most significant advantage is efficiency. Networks can be set up and used much faster. Moreover, the costs of real hardware are eliminated. Only with large and complex networks can the performance decrease, meaning the test results are not 100 % accurate. Nevertheless, the cost benefits outweigh the disadvantages in most cases, which is why emulation tools are used so frequently.

In the following paper, the emulation tools are presented and compared with each other, documenting their use over the last five years.

2. Emulation Tools

2.1. Function

Emulation tools make it possible to recreate real hardware or software environments to test their behavior in a controlled environment. Realistic networks or systems are simulated without the need for the underlying hardware. The tool must be as similar as possible to the original system. Network emulation tools, in particular, allow certain conditions to be emulated by changing individual parameters. This allows for better evaluation of experiments.

For example, it is possible to artificially increase the latency and see how the same application works on less powerful devices. Other limitations, such as low bandwidth or high packet loss, can also be emulated. It is also possible to emulate real networks, which function like real networks.

2.2. Most Common Emulation Tools

Mininet is by far the most frequently used tool in the papers. It can mimic a real network by creating virtual hosts, switches, controllers, and connections [1]. That is why it is primarily used when researchers want to focus on Software-defined Networks (SDN). It can also simulate targeted network failures. The simulations result in data sets that can be used to train models [2]. Because only one device is required, Mininet is very cost-efficient. However, Mininet-based networks cannot exceed the CPU or bandwidth available on a single server. This leads to bottlenecks when too many network components are emulated, resulting in increased latency, packet loss, or inaccurate bandwidths.

Another well-known tool is GNS3 [3]. Compared to Mininet, GNS3 is a more robust network emulator that is mainly used for simulating traditional networks. It can connect real network devices, such as Cisco routers and switches, to virtual machines (such as Linux servers). GNS3's emulation is more realistic because it uses official operating system images (such as Cisco IOS). However, these are often licensed, so it can be challenging to get hold of them. In addition, GNS3 is more resource-intensive due to emulating real devices, which is why it is slow on more extensive networks.

Emulab [4], meanwhile, connects physical and virtual networks. Real hardware (physical devices) and also virtual machines are used. Emulab enables the use of many physical machines in a distributed environment, which increases scalability.

Containernet is an extension of Mininet [5]. In addition to Mininet's functions, it enables the use of Docker containers as hosts, which allows the use of real services (e.g., microservices).

NetEm [6], on the other hand, focuses on changing specific parameters. It is possible to add latency, simulate packet loss and jitter, limit bandwidth, and much more. This makes the tool well-suited for performance testing and software optimization.

 doi: 10.2313/NET-2025-05-1_08

TABLE 1: Usage of Path Property Tools in Paper

	overall	Mininet	NetEm	GNS3	EmuLab	other
2020	10	5	1	0	3	1
2021	12	7	1	1	0	3
2022	6	5	1	0	0	0
2023	5	2	1	1	0	1
2024	6	2	2	0	0	2
sum	39	21	6	2	3	7

Dummynet [7] is similar to NetEm; it was developed only on FreeBSD, but nowadays, it is also usable on MacOS or Linux. NetEm, on the other hand, is based on Linux and is only installable on a Linux-based operating system. Dummynet focuses more on traffic shaping, which is why it is more flexible and efficient in those areas. However, that also needs a deeper configuration and more system resources. Both are applied on the network interface and act as an intermediate layer between devices and the network.

2.3. Installation

Setting up Mininet is easy. The first step is to install a Mininet VM image and then open it in a virtualization system(Quelle). This setup is quick because the Mininet VM has a pre-built environment that is ready to be used. NetEm, on the other hand, can only be installed on a Linux-based operating system such as Ubuntu because it is not a standalone program(Quelle). The corresponding websites provide 'codes' that make it possible to use those tools. For example, in NetEm, 10% packet loss is simulated with the code: `sudo tc qdisc change dev eth0 root netem loss 10%`.

Comparable programs can be installed on FreeBSD (Dummynet [7]).

3. Usage in papers

In the following, we reviewed the papers of ACM SIGCOMM and ACM CoNEXT and found all the papers in which path property emulation tools are used. Table 1 shows the use of the various tools from 2020 to 2024.

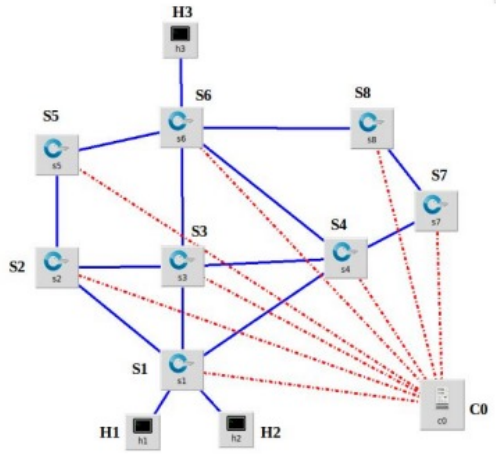


Figure 1: An Example Network Topology

3.1. Mininet

Mininet was used 21 times, representing about half of all papers using emulation tools. In references [8], the authors created an automatic Mininet topology generator. This creates "an evaluation testbed for any kind of measurements that require a ground-truth dataset" [8]. A ground truth dataset is a reference standard for evaluating models or algorithms. The accurate and verified dataset information must often be created manually, which the authors want to automate in this paper. The paper focuses primarily on Resource Public Key Infrastructure(RPKI) measurements, but they want to make the approach possible for all measurements requiring a ground truth dataset. The first step is to collect the required data from public sources to create a directed graph. Since Mininet is not powerful enough to simulate the entire topology, the framework must reduce the graph to approximately 4000 nodes while maintaining important properties such as the degree distribution of the nodes. The abstracted graph is then translated into a Mininet configuration file, which, in this case, generates a realistic RPKI deployment scenario. The developed program allows filtering on a self-selected set of nodes in the Mininet topology, which can create a ground truth dataset.

In Paper [9], an algorithm is created that finds the ideal path between two nodes in a network. It is not about the length of the path but about better traffic load distribution. Figure 1 shows a network created in Mininet, where different hosts are connected to a controller via many switches. Data is sent from the hosts with different bandwidths to check whether their algorithm will find the best path. There are three scenarios: once both hosts send with less than 50% of link bandwidth, once one host sends with more than 50%, and the last time both send the data with more than 50% bandwidth. In all three scenarios, the algorithm prevented overload, proving the quality of the algorithm.

It is also possible to train data sets with Mininet, done in [2], [10]. In the paper [2], datasets of network failures were simulated in Mininet to train decision-tree-based models. For this purpose, random traffic was emulated in selected topologies, and then failure was injected by manually setting the status of links and nodes to failure.

Sometimes, a network consisting of a client and a server, both dual-stacks, is emulated [11]. Then, the latency and bandwidth of the IPv4 and IPv6 paths can be adjusted.

It is also possible that the features of Mininet are not sufficient. For example, in Paper [12], IPMininet, an extension of Mininet that supports SRv6, is used. SRv6 is a modern and flexible routing technology that directly integrates the segment routing principle into the IPv6 protocol. This makes routing more efficient because packets carry explicit path information, and the behavior of networks is made programmable. IPMininet also allows the evaluation of network conditions, such as packet loss, by emulating network loss. This allows the authors to test their SRv6 plugin.

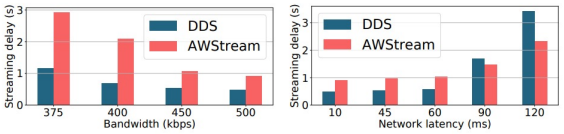
Primarily, Mininet is used to simulate networks with routers, switches, and hosts. Often, the bandwidth, latency, and other parameters are actively set. In Paper [13],

 doi: 10.2313/NET-2025-05-1_08

routers are used whose links have a bandwidth limit of 1000 Mbit s^{-1} and a constant delay of 2 ms.

The other papers also briefly address the use of Mininet to emulate topologies [14]–[25] but do not elaborate further.

In summary, Mininet can be used flexibly. Fundamentally, it is used to emulate networks on which tests are carried out. These include testing network performance or the functionality of one's algorithm. Sometimes, data sets are created from the information obtained, which are used to train models.



(a) Bandwidth vs. streaming delay **(b) Latency vs. streaming delay**

Figure 2: The response delay of AWStream and DDS with different network bandwidth and latency

3.2. NetEm

The use of NetEm in papers has been entirely one-sided. It is mainly described that parameters such as bandwidth and latency in networks have been varied [13], [26]. In paper [26], this was done to test the performance of DDS, a streaming technique. It is a model developed for video streams from cameras to be sent to servers, which will be processed using deep neural networks (DNNs). DNNs are an artificial neural network used to solve complex machine-learning tasks. In Figure 2, the streaming delay of DDS compared to an already established system (AWSStream), with different bandwidth and latency, is shown. The exact comparison is possible through the emulation in NetEm.

Other parameters, such as packet loss and long delay (due to a high round-trip time), can also be emulated [27], [28]. A 5G setup, a combination of increased jitter (fluctuations in the delay), lower bandwidth, and delays, and an LTE-M model were also simulated, which usually offers very low latency and high bandwidth.

In Paper [29], the bottleneck link's speed and the bottleneck buffer's size were configured to gain control over the network. This affects the network performance so that the authors can evaluate the response to network congestion and packet loss.

In summary, the users of this emulation tool are satisfied with the tool. There are limitations when the load is too high (for example, too much jitter), but this was not a problem in the papers mentioned.

3.3. Emulab

In Reference [30], extensive simulations are performed with Emulab. Different network topologies are emulated, with different conditions, to demonstrate the performance of "MPCC, a high-performance multipath congestion control architecture" [30]. It was explicitly declared that "[u]nless stated otherwise all link latencies, bandwidths, and buffer sizes are 30 ms, 100 Mbit s^{-1} and 375 kB (the Bandwidth-Delay Product), respectively." [30] All values are detailed and explained with potential limitations for other values. It is noted that the experiments in Emulab cannot be 100% transferred to real networks. Therefore, live experiments were also carried out in which files were downloaded from various locations in the cloud. This also implies that the tests in Emulab alone cannot provide complete information about the tool in practice.

The Paper [31] also emulates parameters such as latency, focusing on dynamic changes in network conditions.

It should be noted that papers that use Emulab conduct extensive and detailed experiments [17]. Emulab is designed for larger network emulations, increasing the complexity and time needed to understand the program.

3.4. GNS3

In Paper [32], [33], Gns3 was used similarly to Mininet to create virtual networks and test algorithms. Paper [32] is about Snowcap, which requires GNS3 to function. A detailed guide is provided explaining how to use Snowcap, one of the requirements being GNS3.

3.5. Other Tools

There is one paper that uses Dummynet [34]. Again, realistic network conditions were simulated, which were intended to represent mobile networks. These are known for fluctuating latencies, achieved in Dummynet by constantly adjusting the queue delays with target jitter values. The method caused latency fluctuations without letting packets arrive in the wrong order, which more realistically simulates mobile networks. Dummynet was used here because it is more focused on traffic shaping than NetEm, offering greater flexibility in emulation. However, the paper mentions that Dummynet is limited by a maximum queue size of 100, which does not allow for entirely flexible emulation.

Nanonet is an emulation tool conceptually based on Mininet and primarily aimed at segment routing experiments [35]. Segment routing is routing traffic in networks on predefined routers instead of making a new decision at each router. In Paper [35], they simulate network nodes connected to each other for a realistic simulation of network behavior. Nanonet then calculates the shortest path between the nodes. An even distribution of traffic on the paths was ensured to measure the maximum link utilization with a specific instance. This allowed testing and comparing different approaches for optimizing link weights.

Another extension of Mininet is G2-Mininet, which analyzes the Quantitative Theory of Bottleneck Structures (QTBS). QTBS is a mathematical theory developed to analyze and optimize communication networks. The paper simulated various network topologies (fat trees, folded clos, and Dragonfly) to test QTBS. More than 600 networks were simulated over more than 800 hours to confirm the correctness of the model.

In Paper [36], SimBricks, a Network System Evaluation with Modular Simulation, was introduced with ns-3 integrated. Ns-3 [37] is also an emulation tool that can process and synchronize packets using the Ethernet

network interface. SimBrick uses ns-3 to simulate network layers and connections between virtual and physical network topologies.

The paper [5] takes advantage of the fact that Containernet is based on containers that form a network of interconnected nodes. Such a network is created with each container running a KIRA routing server that provides IPv6 connectivity and some containers running additional 5G core network functions. This is part of the KIRA routing architecture intended to enable autonomous and fault-tolerant network control. Containernet is mainly used to emulate the network topology and test node failures.

A tool yet to be mentioned is BESS. It allows the user to control network properties such as latency at a more detailed level than other tools such as NetEm [38]. This enables fine-grained control of network traffic. In the paper [38], it is used "to control the access link speed, queue size, and add delay to ingress and egress packets" [38] of a switch. In addition, BESS can measure important data such as queue occupancy and packet loss, which enables more precise analysis.

The last paper introduces Klonet [39]. It is a new network emulation platform that was created for educational purposes. It is criticized that current emulation tools are inadequate for integrating network hardware due to insufficient scalability and other factors. The paper also creates a table with the tools currently in use and shows factors such as hardware support, container support, and VM support.

3.6. Summary

	Mininet	NetEm	GNS3	EmuLab
Function	Simulation of Software-Defined Networks (SDN)	Simulation of latency, packet loss, etc.	Simulation of physical and virtual devices	Testing and experimenting with real networks
Architecture	Virtual Switches, Hosts and Controllers	Works as part of Linux Traffic Control	Virtual machines, physical devices and switches	Hardware and software testbed
Scalability	Good for small to medium networks	Depending on the hardware	Very good for small to large networks	High scalability (both small and large networks)
Realism	High proximity to SDN-based networks	Not an exact replica, only simulates properties	Close to real networks	Very high realism through real hardware

Figure 3: Comparison of Path Property Emulation Tools

Many different emulation tools are used for different reasons. In Figure 3, the most commonly used tools are compared. For example, Mininet was often used in the papers, mainly to emulate small and medium-sized networks. NetEm was used to emulate network properties but was only mentioned briefly. Although GNS3 is more flexible in the size of the emulated networks, the tool was mostly only mentioned in passing. Papers that use Emulab have mainly carried out very extensive and detailed experiments that comprise several pages of the paper. The complexity of the tool can explain this. For example, while the functions of NetEm are limited to emulating parameters on one host, Emulab can emulate large networks with complex properties on several hosts, switches, and routers.

Nevertheless, the current tools are not perfect. Sometimes, the current applications do not meet the requirements. Very few tools can simulate specific hardware properties or modern technologies such as containers. Also, with large, realistic networks, many tools reach their limits. This has led to tools such as Klonet and Containernet, which are new emulation tools with more options that can be used in a broader variety of ways.

4. Conclusion

Path property emulation tools play a crucial role in network research and development. It is possible to precisely simulate network properties such as latency, bandwidth, loss, and jitter. Depending on the requirements, choosing the right emulation tool can be cost and time-efficient, mainly when a lot of data has to be processed.

However, since the established tools are imperfect, their results cannot always be 100% transferred to the real world. With the rapid development of the Internet, there will be even more powerful and diverse future tools capable of more. Technologies such as artificial intelligence also mean a lot is still possible in this area, making it possible that currently established tools will become outdated and no longer be used.

Nevertheless, the current tools make an essential contribution to the development of modern networks. Even if they are imperfect, most of the tools have been in use for several years and will continue to be used in various ways and on a wide scale.

References

[1] Mininet, https://mininet.org/overview/, 19.10.2024.

[2] X. Zuo, Q. Li, J. Xiao, D. Zhao, and J. Yong, “Drift-bottle: a lightweight and distributed approach to failure localization in general networks,” in *Conference: CoNEXT ’22: The 18th International Conference on emerging Networking EXperiments and Technologies*, 11 2022, pp. 337–348.

[3] GNS3, https://www.gns3.com/, 19.10.2024.

[4] Emulab, https://www.emulab.net/portal/frontpage.php, 19.10.2024.

[5] P. Seehofer, H. Mahrt, R. Bless, and M. Zitterbart, “Demo: Enabling autonomic network infrastructures with kira,” in *Conference: ACM SIGCOMM ’23: ACM SIGCOMM 2023 Conference*, 09 2023, pp. 1165–1167.

[6] NetEm, https://man7.org/linux/man-pages/man8/tc-netem.8.html#OPTIONS, 19.10.2024.

[7] DummyNet, https://cs.baylor.edu/~donahoo/tools/dummy/tutorial.htm, 19.10.2024.

[8] N. Rodday, R. Baaren, L. Hendriks, R. Rijswijk-Deij, A. Pras, and G. Dreo, “Evaluating rpki rov identification methodologies in automatically generated mininet topologies,” in *Conference: CoNEXT ’20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 530–531.

[9] K. Abiram and T. Kathiravelu, “Congestion avoidance in data communication networks using software defined networking,” in *Conference: CoNEXT ’21: The 17th International Conference on emerging Networking EXperiments and Technologies*, 12 2021, pp. 463–464.

[10] H. Mostafaei, S. Miri, and S. Schmid, “Poster: Reactnet: Self-adjusting architecture for networked systems,” in *Conference: 17th International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2021 Posters)*, 12 2021.

[11] F. Rochet, E. Assogba, M. Piraux, K. Edeline, B. Donnet, and O. Bonaventure, "Tcpls: modern transport services with tcp and tls," in *Conference: CoNEXT '21: The 17th International Conference on emerging Networking EXperiments and Technologies*, 12 2021, pp. 45–59.

[12] L. Navarre, F. Michel, and O. Bonaventure, "Srv6-fec: bringing forward erasure correction to ipv6 segment routing," in *Conference: SIGCOMM '21: ACM SIGCOMM 2021 Conference*, 08 2021, pp. 45–47.

[13] J. Zhang, C. Zeng, H. Zhang, S. Hu, and K. Chen, "Liteflow: towards high-performance adaptive neural networks for kernel datapath," in *Conference: SIGCOMM '22: ACM SIGCOMM 2022 Conference*, 08 2022, pp. 414–427.

[14] T. Jepsen, A. Fattaholmanan, M. Moshref, N. Foster, A. Carzaniga, and R. Soulé, "Forwarding and routing with packet subscriptions," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies)*, 11 2020, pp. 282–294.

[15] A. Sacco, F. Esposito, and G. Marchetto, "A distributed reinforcement learning approach for energy and congestion-aware edge networks," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 546–547.

[16] H. Nagda, R. Nagda, I. Pedisich, N. Sultana, and B. Loo, "Fdp: a teaching and demo platform for sdn," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 524–525.

[17] D. Senf, H. Shulman, and M. Waidner, "Performance penalties of resilient sdn infrastructures," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 528–529.

[18] P. Bol, R. Lunardi, B. B. França, and W. Cordeiro, "Modular switch deployment in programmable forwarding planes with switch (de)composer," in *Conference: SIGCOMM '21: ACM SIGCOMM 2021 Conference*, 08 2021, pp. 30–32.

[19] D. Guo, S. Chen, K. Gao, Q. Xiang, Y. Zhang, and Y. Yang, "Flash: fast, consistent data plane verification for large-scale network settings," in *Conference: SIGCOMM '22: ACM SIGCOMM 2022 Conference*, 08 2022, pp. 314–335.

[20] S. Sagkriotis and D. Pezaros, "Accelerating kubernetes with in-network caching," in *Conference: SIGCOMM '22: ACM SIGCOMM 2022 Conference*, 10 2022, pp. 40–42.

[21] S. Renganathan, B. Rubin, H. Kim, P. Ventre, C. Cascone, D. Moro, C. Chan, N. McKeown, and N. Foster, "Hydra: Effective runtime network verification," in *Conference: ACM SIGCOMM '23: ACM SIGCOMM 2023 Conference*, 09 2023, pp. 182–194.

[22] K. Namjoshi, S. Gheissi, and K. Sabnani, "Algorithms for in-place, consistent network update," in *Conference: ACM SIGCOMM '24: ACM SIGCOMM 2024 Conference*, 08 2024, pp. 244–257.

[23] C. Jiang, Z. Li, S. Rao, and M. Tawarmalani, "Flexile: meeting bandwidth objectives almost always," in *Conference: CoNEXT '22: The 18th International Conference on emerging Networking EXperiments and Technologies*, 11 2022, pp. 110–125.

[24] L. Brown, A. Gran Alcoz, F. Cangialosi, A. Narayan, M. Alizadeh, H. Balakrishnan, E. Friedman, E. Katz-Bassett, A. Krishnamurthy, M. Schapira, and S. Shenker, "Principles for internet congestion management," in *Conference: ACM SIGCOMM '24: ACM SIGCOMM 2024 Conference*, 08 2024, pp. 166–180.

[25] J. Yen, T. Lévai, Q. Ye, X. Ren, R. Govindan, and B. Raghavan, "Semi-automated protocol disambiguation and code generation," in *Conference: SIGCOMM '21: ACM SIGCOMM 2021 Conference*, 08 2021, pp. 272–286.

[26] T. John, P. Vaere, C. Schutijser, A. Perrig, and D. Hausheer, "Linc: low-cost inter-domain connectivity for industrial systems," in *Conference: SIGCOMM '21: ACM SIGCOMM 2021 Conference*, 08 2021, pp. 68–70.

[27] M. Sosnowski, F. Wiedner, E. Hauser, L. Steger, D. Schoinianakis, S. Gallenmüller, and G. Carle, "The performance of post-quantum tls 1.3," in *Conference: CoNEXT 2023: The 19th International Conference on emerging Networking EXperiments and Technologies*, 12 2023, pp. 19–27.

[28] A. Tahir, P. Goyal, I. Marinos, M. Evans, and R. Mittal, "Efficient policy-rich rate enforcement with phantom queues," in *Conference: ACM SIGCOMM '24: ACM SIGCOMM 2024 Conference*, 08 2024, pp. 1000–1013.

[29] M. Arghavani, H. Zhang, D. Eyers, and A. Arghavani, "Suss: Improving tcp performance by speeding up slow-start," in *Conference: ACM SIGCOMM '24: ACM SIGCOMM 2024 Conference*, 08 2024, pp. 151–165.

[30] T. Gilad, N. Rozen-Schiff, P. Godfrey, C. Raiciu, and M. Schapira, "Mpcc: online learning multipath transport," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 121–135.

[31] T. Meng, N. Rozen-Schiff, P. Godfrey, and M. Schapira, "Pcc proteus: Scavenger transport and beyond," in *Conference: SIGCOMM '20: Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 07 2020, pp. 615–631.

[32] T. Schneider, R. Birkner, and L. Vanbever, "Snowcap: synthesizing network-wide configuration updates," in *Conference: SIGCOMM '21: ACM SIGCOMM 2021 Conference*, 08 2021, pp. 33–49.

[33] M. Brown, A. Fogel, D. Halperin, V. Heorhiadi, R. Mahajan, and T. Millstein, "Lessons from the evolution of the batfish configuration analysis tool," in *Conference: ACM SIGCOMM '23: ACM SIGCOMM 2023 Conference*, 09 2023, pp. 122–135.

[34] N. Agarwal, M. Varvello, A. Aucinas, F. Bustamante, and R. Netravali, "Mind the delay: the adverse effects of delay-based tcp on http," in *Conference: CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies*, 11 2020, pp. 364–370.

[35] M. Parham, T. Fenz, N. Süss, K.-T. Foerster, and S. Schmid, "Traffic engineering with joint link weight and segment optimization," in *CoNEXT '21: Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 12 2021, pp. 313–327.

[36] H. Li, J. Li, and A. Kaufmann, "Simbricks: end-to-end network system evaluation with modular simulation," in *Conference: SIGCOMM '22: ACM SIGCOMM 2022 Conference*, 08 2022, pp. 380–396.

[37] ns 3, https://www.nsnam.org/, 19.10.2024.

[38] A. Philip, R. Athapathu, R. Ware, F. Mkocheko, A. Schlomer, M. Shou, Z. Meng, S. Seshan, and J. Sherry, "Prudentia: Findings of an internet fairness watchdog," in *Conference: ACM SIGCOMM '24: ACM SIGCOMM 2024 Conference*, 08 2024, pp. 506–520.

[39] J. Guo, D. Wu, C. Ma, Y. Hongfang, G. Sun, L. Luo, Y. Xu, and N. Zhang, "Poster: A hybrid virtual-real emulation platform for computer network education," in *Conference: ACM SIGCOMM Posters and Demos '24: ACM SIGCOMM 2024 Conference: Posters and Demos*, 08 2024, pp. 45–47.

Peer-to-Peer Network Attacks: Erebus and ConditionalExhaust

Hedi Baccouche, Kilian Glas*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: ge87qew@mytum.de, glask@net.in.tum.de

***Abstract*—The concept of Peer-to-Peer networks has become one of the most employed network architectures in the 21st century, particularly in the domain of cryptographic currencies. Blockchain networks operate with no central authority which engendered a security gap in the Peer-to-Peer layer. In this paper, we explore two attacks that target the *control plane* as well as the *data plane* of the P2P network. Specifically, we examine the Erebus attack on Bitcoin, which manipulates a Bitcoin node's outgoing connections through control plane vulnerabilities, and the ConditionalExhaust attack on Ethereum, which exploits data plane weaknesses to overwhelm network resources and impede the growth of the blockchain. This paper contributes to the understanding of the presented attacks and offers insights about some proposed countermeasures that leverage the resilience of these networks towards these attacks.**

***Index Terms*—Peer-to-Peer Networks, Erebus Attack, ConditionalExhaust Attack**

1. Introduction

Peer-to-Peer networks [1], also acronymized as P2P networks have become an alternative to the conventional architecture of networks, namely the Client-Server model. A P2P network is a decentralized network that enables the network nodes, also known as *Peers*, to communicate directly with each other without the oversight of a server authority. Each node in this network is considered simultaneously as a client and server. Multiple cryptocurrencies such as Bitcoin [2] and Ethereum [3] rely on P2P networks to communicate the state of the public ledger, a decentralized record of transactions known as blockchain [4], and validate transactions. Bitcoin and Ethereum opted for a decentralized architecture since the blockchain state is replicated and visible across peers, thus offering redundancy as well as transparency. Although this model offers valuable advantages, e.g. resilience to failure and high scalability in comparison to the Client-Server architecture, the lack of central authority has made it subject to multiple attacks. In fact, only relying on the P2P layer, an overlay network responsible for peer communication, can create security vulnerabilities, especially in the absence of strict node authentication. In this paper, we provide a detailed overview of two prominent attacks that are conducted on two different P2P networks, namely Bitcoin and Ethereum. We classified attacks into two categories: *control plane* and *data plane* attacks. Attacks that target the *control plane* aim to inhibit the node's discovery protocol by restricting a node's view of the whole network topology. *Data plane* attacks, on the other hand, focus on disrupting the actual data transmission or overloading the network with malicious traffic. The first attack, known as the Erebus attack [5] (Erebus means "darkness" in Greek), targets the *control plane* by inhibiting the node discovery of legitimate peers and enabling the attacker to manipulate all data exchanged between nodes. In this attack, the adversary is usually a Tier-1 AS [6, Chapter 2] that aims to redirect all victim nodes' connections to a preselected set of Bitcoin nodes in order to include his AS in the victim-to-node path. The second attack, known as the ConditionalExhaust attack [7, Section 4], can be classified as a *data plane* attack since the attacker overloads the Ethereum network with violating transactions that exhaust the victim's resources before it is rejected.
The remainder of this paper is structured as follows: Section 2 introduces the classes used to categorize the attacks. Section 3 includes both attacks in separate subsections, where each subsection is further divided into other subsubsections designated for the attack's phases as well as the proposed countermeasures to alleviate the attack's impact.

2. Categorization of Attacks

In this section we categorize the attacks based on the layer of the P2P network that will be mainly targeted by the attacker. Accordingly, we can differentiate two general planes of attacks namely *data plane* attacks and *control plane* attacks [8].

2.1. Control Plane Attacks

The *control plane* is responsible for the management of the network traffic through using routing protocols such as BGP [9] and OSPF. In P2P networks, the *control plane* is crucial for defining how peers discover each other through the use of protocols such as Kademlia DHT [10]. It ensures that the data follows the optimal path while being transmitted to the destination. Attacking the *control plane*, e.g. Eclipse attack [11], can make the P2P network vulnerable to misrouting and lead to the partitioning of peers [8], meaning that the node becomes isolated and has a restricted view of the entire network.

2.2. Data Plane Attacks

The *data plane*, also known as the forwarding plane, refers to the data transmission mechanism between peers

 doi: 10.2313/NET-2025-05-1_09

in P2P networks. It is responsible for forwarding data packets based on the routing decisions made by the *control plane* protocols since it operates at a lower abstraction level. Disrupting the *data plane*, e.g. ConditionalExhaust attack [7], means overloading the network, leading to degraded performance or even the failure of the whole network [8].

3. Attacks on Peer-to-Peer Networks

This section introduces two attacks targeting the Bitcoin and Ethereum P2P networks. In each subsection, we categorize an attack and depict its phases in detail. Finally, we enumerate some proposed countermeasures and evaluate their effectiveness to mitigate the corresponding attack.

3.1. Erebus Attack

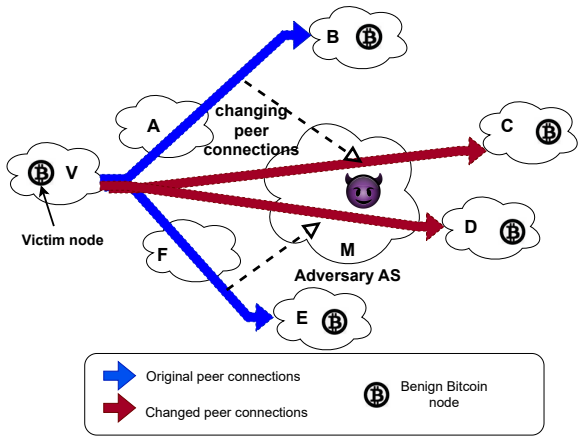


Figure 1: Illustration of Peer Connection Manipulation in Erebus Attack [12, Introduction]

Muoi et al. [12], carried out the Erebus attack by implementing a Python script. This attack exploits a critical vulnerability in P2P networks, which is the failure to verify the legitimacy of peers properly. This attack can be categorized as a *control-plane* attack. It aims to redirect all the outgoing connections of a Bitcoin node through an adversary AS (the autonomous system of the attacker) , therefore isolating the victim which becomes unable to form legitimate connections with honest nodes.
In the Erebus attack, we can distinguish two main phases that describe a successful attack conduction namely the reconnaissance phase and the attack execution phase.

3.1.1. Reconnaissance Phase. In this phase, the attacker seeks to gather information about the network topology. They aim to identify which Autonomous Systems (ASes) force the victim's connection to pass through the adversarial AS. Therefore, the adversary AS can be considered a man-in-the-middle network, which manipulates the data traffic of the victim node. These ASes are characterized as *shadow ASes* [12, Section IV, Subsection A]. In order to infer the default AS paths of the AS hosting the targeted node, the attacker can model the entire topology of the Internet and simulate BGP advertisement messages. This allows the attacker to observe how the targeted AS sets its default routes to other ASes. After identifying the *shadow ASes*, the attacker collects all the IP addresses associated with these ASes and stores them in a database. These IP addresses are referred to as *shadow IPs*. The attacker should also ensure that the path from the victim node to the *shadow IPs* traverses the adversarial AS. This can be achieved by the attacker through establishing a TCP connection with the victim node using the *shadow IPs* and waiting to receive a synchronization message from the victim during the TCP-handshake [13]. Receiving a packet with the SYN flag set from the victim is valid proof that the path from the victim to the shadow IPs traverses the adversary's network.

3.1.2. Attack Execution Phase. In the second phase, the attacker manipulates the victim node's outgoing connections, ensuring they are directed to the harvested *shadow IPs*. As the *shadow IPs* are controlled by the the attacker, they impersonate them and establishes a Bitcoin-handshake [14] with the victim node. The attacker then floods the victim with a large number of *shadow IPs* through *addr* messages [15] that serve for other peers advertisement. The victim node subsequently stores the received IPs in its *new table*[1] for future connections. The attacker intends at first to fill most of the *new table* slots with his harvested *shadow IPs* therefore increasing the chance that the victim node initiates an outgoing connection to one of them. The victim node inserts the *shadow IPs* in the right bucket of *new table* by hashing the IP prefix group *IPshad_pre* of the received IP address with the IP prefix *peer_pre* of the node that provided the IP address using a private Key SK and SHA-256 as the hash function H :

$$h_1 = H(\text{SK}, \textit{IPshad\_pre}, \textit{peer\_pre})$$

$$h_2 = H(\text{SK}, \textit{peer\_pre}, h_1 \mod 64)$$

$$\textit{new\_bucket} = h_2 \mod 1024$$

The node computes the slot number by hashing the bucket's number *new_bucket* with the IP address IP, to be added, and then applying the modulo operation (each bucket contains 64 slots):

$$\textit{slot\_number} = H(\text{SK}, \mathcal{N}, \textit{new\_bucket}, \textit{IP}) \mod 64$$

If two IPs collide in the same slot the node checks the timestamp as well as the number of the old connection attempts related to the already saved one. If the existing IP seems to be old (more than 30 days) [12, Section V, Subsection B] and not reliable, i.e. failed to establish some previous connections, it will be overridden by the new one. The victim node maintains another table, known as the *tried table*[2] which also represents a target for the attacker. Since the victim node randomly selects an IP from one of the two tables to initiate an outgoing connection, the attacker aims to occupy as many slots as possible in both tables. As an option, the

1. A table that stores IP addresses learned from other nodes for future connections. It contains 1024 buckets and each bucket contains 64 slots.
2. A table that contains moved IPs from the *new table* to which a successful connection has been made

 doi: 10.2313/NET-2025-05-1_09

attacker runs a script that triggers a repetitive reboot of the victim node when the likelihood of establishing an outgoing connection to one of the *shadow IPs* becomes significantly high (e.g. 30% or 50%) [12, Section V, Subsection C]. After each reboot, the node needs to reestablish all its outgoing connections, which provides the attacker an opportunity to overwhelm it again with new connection requests on behalf of the *shadow IPs*. This strategy can be adopted by the attacker to occupy all the node's outgoing connections and reduce the attack's duration as well. Since the attacker cannot access the two tables of the victim node at the same time due to some deployed measurements against eclipse attack, they can use the *trickledown* attack strategy [12, Section V, Subsection C] to indirectly flood the *tried table* with *shadow IPs*. In this newly adopted strategy, the attacker floods the *new table* and then waits for the *shadow IPs* to be inserted automatically in the *tried table*. The migration of IP addresses to the *tried table* can happen in two cases: (1) when the Bitcoin node establishes a successful outgoing connection to an IP address from the *new table*; (2) The Bitcoin node probes within predefined time intervals (2 minutes) the reachability of a random IP address from the *new table* through an ephemeral connection known as *feeler-connection* and add it in the *tried table* if it accepts the connection [12, Section V, Subsection C].

3.1.3. Countermeasures. In [12] Muoi Tran et al. discuss two effective countermeasures that may mitigate the Erebus attack. In this section, we also provide a graph that the authors presented to illustrate the effectiveness of both countermeasures.

a. Reduction of Table Size. One of the suggested countermeasures is to lower the size of both *new* and *tried* tables since it can significantly reduce the probability that all outgoing connections of the victim node are made to the *shadow IPs*. Usually, the adversary AS is a Tier-1 AS, which means that it has coverage over multiple continents in the world and disposes of multiple shadow ASes. This enables the attacker to harvest a large set of shadow IPs to execute the attack. Reducing both table sizes means that the node has fewer available outgoing connections, which reduces the likelihood that a shadow IP is chosen for establishing a connection.

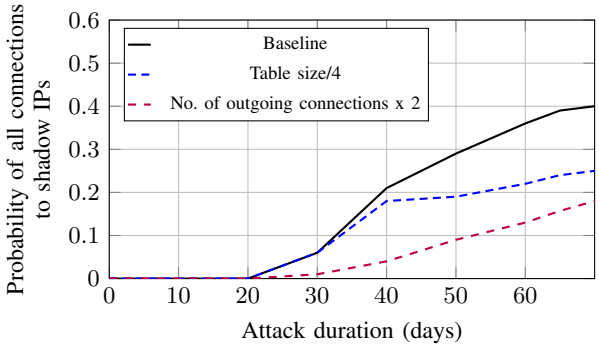


Figure 2: Probability of compromising all outgoing connections [12, Section VII, Subsection C]

Figure 2 illustrates well the impact of dividing the size of both tables by four on the probability of compromising all outgoing connections.

b. Higher Number of Outgoing Connections. Another effective countermeasure that was discussed was increasing the number of outgoing connections. A Bitcoin node has normally 8 outgoing connections. Increasing the number of these connections may cost the attacker a longer time to achieve his goal of compromising all the outgoing connections. According to figure 2 we can lower the probability of the nodes initiating all 8 outgoing connections to shadow IP from 40% to 18% during the same attack period of 60 days. However, this measure may be inconvenient because it can lead to the congestion of the Bitcoin P2P network.

3.2. ConditionalExhaust Attack in Ethereum

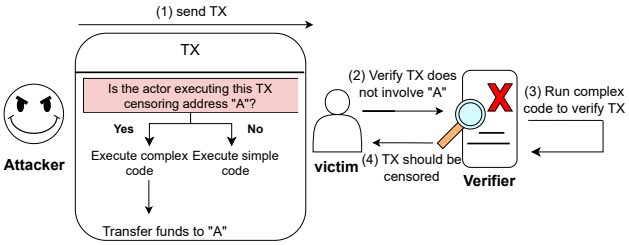


Figure 3: Illustration of a ConditionalExhaust Attack [7, Section 4]

The ConditionalExhaust attack is a Resource Exhaustion Attack, acronymized as REA, that targets multiple actors of the Ethereum network, forcing them to consume high computational resources in vain. This attack is considered a data plane attack since the attacker overloads the network with heavy computational transactions, which are then processed by different actors of the Ethereum P2P network without updating the actual state of the blockchain. Therefore, the network becomes congested, and actors spend additional overhead to process legitimate transactions. Before diving into the attack details, we will define some important concepts to gain a clearer understanding of the attack.
In the Ethereum blockchain, there are four key actors involved in transaction processing and validation, namely the *searchers*, *builders*, *relays*, and *validators*, also known as *proposers*. *Searchers* play a crucial role in identifying profitable transactions, i.e., transactions with high fees, from the pending transaction pool known as *mempool*. The corresponding profit is called Maximal Extractable Value (MEV) [16], which is extracted by the *validators* through choosing and controlling the order of transactions within a block. These selected transactions are then assembled in *bundles* and sent to the *builders*, who are responsible for creating "MEV maximal blocks" by choosing which transactions to include based on the MEV. *Relays* are intermediate actors between builders and validators. They receive "MEV maximal blocks" from *builders* and forward only the most profitable and valid blocks, i.e., blocks with valid structure and signature, to the *proposers*. *Validators* select at first the most rewarding blocks from different *Relays*. Additionally, they confirm the validity of the transactions by checking whether

they have a valid nonce [3, Section IV] (transaction's unique identifier) and signature. *Validators* also inspect the compliance of the transactions to their censorship policy by checking some specific transaction fields, e.g., checking whether the transaction's recipient address contains a sanctioned address. Finally, *proposers* should execute the transactions to check further if they interact with some censored entities.

The measurement unit in Ethereum is called gas. A user's transaction can cost different amounts of gas (measured in units) depending on its complexity. A transaction fee is calculated by multiplying the number of gas units used by the transaction with the gas price [17, Introduction] per unit, which is a variant price advertised by the sender of the transaction to specify the fee they are willing to pay for each unit of gas consumed. The first validator to include a transaction in a block is then awarded its related fee [7, Section 3].

Aviv et al. presented in [7] two attack variants, but we will only deal with one variant where the attacker is a non-*proposer*. This attack targets mainly two types of actors, which are the "sanction-compliant" *builders* and *validators* [7, Section 4]. These two actors adhere to rules that sanction some entities or addresses and forbid interaction with them. In the attack, we assume that α is the set of *validators* that censor a certain address β. The proposed attack's variant is also called *coinbase* [7] since the attacker already knows the addresses of the censoring validator and the sanctioned addresses by these actors. For a broader impact, the attacker can use different addresses censored by different validators to target a larger number of these actors.
The ConditionalExhaust attack consists of two phases namely the Deployment phase and the Execution phase.

3.2.1. Deployment Phase. In this phase, the attacker creates a smart contract, a program written in a high-level language such as Solidity [17, Subsection 1.1.1]. The attacker's smart contract includes a single function with two distinct control flows, each designed to target a specific validator type. If the targeted actor is a censoring validator, i.e., belongs to the set α, the function executes at first a code that demands high computational resources and then interacts with one of the entities sanctioned by the actor's censorship policy. However, if the actor is a non-censoring validator, i.e., not part of the set α, a simple code that does not incur high transaction fees will be executed. This can be achieved by the attacker by implementing a mapping (address -> boolean) in the smart contract, which maps each validator's address with a boolean value that indicates if they are a censoring validator or not [7, Subsection 4.1]. The executed control flow is then chosen based on this boolean value, i.e. if the boolean value is true the complex code will be executed otherwise the simple code is chosen.

3.2.2. Execution Phase. In this phase, the attacker creates many transactions that invoke the function of the deployed smart contract. The *gossip protocol* [18] of the Ethereum Layer is employed for sending the attacker's transactions. If the current actor belongs to α, they will start by checking the transaction recipient address to ensure that it does not correspond to any sanctioned entity. Furthermore, the transaction's compliance with the actor's rules cannot be verified without executing it. Therefore, the victim will waste excessive computational resources during the execution and then invalidate the transaction since it interacts with a forbidden address which is β. While executing the transaction, the validator incurs a large amount of gas units (proportional to the complexity of the transaction) and will not be awarded since the transaction cannot be included in the current block. As a notice, the attacker can have the possibility to resend the transaction if the targeted actor discards the transaction from the *mempool*, which defines the pool of the transactions that must be validated before including them in a block. However, if the targeted actor does not discard invalid transactions from their *mempool*, the attacker should increase the nonce of each sent transaction by 1, since every transaction should have a unique nonce (i.e. nonce of the last sender's validated transaction +1) [7, Section 3]. Otherwise, transactions with duplicate nonce will be discarded by the validator.
In case the receiving actor is a non-censoring validator, the simple code of the other control flow will be executed incurring insignificant fees.

3.2.3. Countermeasures. Yaish et al. proposed two countermeasures in [7, Section 8] that may mitigate the ConditionalExhaust attack and evaluated their effectiveness. They also pointed to the limitations of these measures.

a. Strict Access Lists. The authors suggested employing strict *access lists*, in which all addresses that are involved in a transaction are enumerated. This way both *builders* and *proposers* can verify a transaction's compliance before executing it by checking the related *access list*. Any interaction with an address that is not mentioned in the *access list*, obliges the sender to pay the incurred fees during the execution and leads to the transaction's rejection. Therefore *builders* and *proposers* can avoid resource waste by first checking the *access list*. Moreover, they can now benefit from the fees incurred during the execution of a violating transaction even though it will rejected. However, establishing such compliant and accurate lists is complex since the transaction should consider all possible states to specify these addresses [19]. Aside from avoiding resource waste and additional block construction overhead, this measure may lead to a higher rejection rate of legitimate transactions due to inaccurate access lists.

b. Random Transaction Selection. An additional proposed countermeasure is to impose a random selection of transactions regardless of their fees. Based on the normal *greedy* selection algorithm, *builders* prioritize the attacker's transactions since they lure them with high transaction fees. However, with this introduced measure the attacker should generate more transactions to conduct the attack successfully. Even though this strategy can lower the chance of a ConditionalExhaust attack, it contradicts the searcher's functionality as explained above.

4. Conclusion and Future Work

In this paper, we introduced two different attacks targeting two of the most prominent P2P networks, namely Bitcoin and Ethereum. We first categorized these attacks based on which plane they jeopardize (*control plane* or *data plane*). We began by introducing the Erebus Attack, its phases, and the threats a malicious Autonomous System (AS) can engender. We also discussed the countermeasures suggested by the authors and evaluated their effectiveness in lowering the success rate of the attack. In Addition, we presented the ConditionalExhaust attack, which cripples the blockchain update process by flooding the Ethereum network with transactions that break the censorship policy. We then outlined two countermeasures proposed by the authors, highlighting their effectiveness and limitations. In conclusion, the decentralized network architecture eliminated the need for a central authority, however, it introduced new vulnerabilities that must be addressed. Due to the lack of centralized control, P2P networks are more susceptible to attacks. To strengthen the resilience of these networks more efforts should be dedicated to establishing strict peer authentication and reconsidering existing P2P protocols. In the context of blockchain, researchers should focus on enhancing node partition detection protocols as well as finding alternative transaction verification methods.

References

[1] F. Musiani, "Network Architecture as Internet Governance," *Internet Policy Review*, vol. 2, 2013.

[2] K. John, M. O'Hara, and F. Saleh, "Bitcoin and Beyond," *Annual Review of Financial Economics*, vol. 14, pp. 95–115, 2022.

[3] D. Vujičić, D. Jagodić, and S. Ranđić, "Blockchain Technology, Bitcoin, and Ethereum: A Brief Overview." IEEE, March 2018, pp. 4,5.

[4] M. D. Pierro, "What Is the Blockchain?" *Computing in Science & Engineering*, vol. 19, no. 5, pp. 92–95, 2017.

[5] Muoi Tran and Inho Choi and Gi Jun Moon and Anh V. Vu and Min Suk Kang, "A Stealthier Partitioning Attack against Bitcoin Peer-to-Peer Network," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, p. 901.

[6] P. Kriegel, "Determining the Location of Autonomous System Relations Among Top Tier Internet Service Providers in the United States," Ph.D. dissertation, University of Oregon, 2016, Bachelor of Science Thesis, Department of Computer and Information Science and the Robert D. Clark Honors College.

[7] A. Yaish, L. Zhou, A. Gervais, K. Qin, and A. Zohar, "Speculative Denial-of-Service Attacks in Ethereum," 2024.

[8] Baptiste Prêtre, "Attacks on Peer-to-Peer Networks," 2005.

[9] M. Caesar and J. Rexford, "BGP Routing Policies in ISP Networks," *IEEE Network*, 2005.

[10] B. Spori, "Implementation of the Kademlia Distributed Hash Table," Master's Thesis, ETH Zurich, 2006, Advisor: Marcel Baur, Professor: Prof. Dr. Bernhard Plattner, SA-2006-19.

[11] Y. Yang and L. Yang, "A Survey of Peer-to-Peer Attacks and Counter Attacks," 2012.

[12] Muoi Tran and Inho Choi and Gi Jun Moon and Anh V. Vu and Min Suk Kang, "A Stealthier Partitioning Attack against Bitcoin Peer-to-Peer Network," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 897–898.

[13] F.-H. Hsu, Y.-L. Hwang, C.-Y. Tsai, W.-T. Cai, C.-H. Lee, and K. Chang, "TRAP: A Three-Way Handshake Server for TCP Connection Establishment," *Applied Sciences*, 2016, Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan.

[14] J. Tapsell, R. N. Akram, and K. Markantonakis, "An Evaluation of the Security of the Bitcoin Peer-to-Peer Network," in *2018 IEEE Congress on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics*. Egham, United Kingdom: IEEE, 2018, ISG-SCC, Royal Holloway, University of London, Egham, United Kingdom.

[15] Ethan Heilman and Alison Kendler and Aviv Zohar and Sharon Goldberg, "Eclipse Attacks on Bitcoin's Peer-to-Peer Network," in *Proceedings of the 24th USENIX Security Symposium*. Washington, D.C.: USENIX Association, 2015. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-heilman.pdf

[16] V. Gramlich, D. Jelito, and J. Sedlmeir, "Maximal Extractable Value: Current Understanding, Categorization, and Open Research Questions," *Electronic Markets*, vol. 34, no. 49, Oct 2024.

[17] C. Signer, "Gas Cost Analysis for Ethereum Smart Contracts," Master Thesis, ETH Zurich, November 2018.

[18] S. Naderi Mighan, J. Misić, and V. B. Mišić, "On Block Delivery Time in Ethereum Network." IEEE, 2022.

[19] L. Heimbach, Q. Kniep, Y. Vonlanthen, R. Wattenhofer, and P. Züst, "Dissecting the EIP-2930 Optional Access Lists," *arXiv preprint arXiv:2312.06574*, 2023.

AI and Machine Learning for Wi-Fi Optimization

Yousssef Baccouche, Jonas Andre*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: ge87qet@mytum.de, andre@net.in.tum.de

***Abstract*—Wireless Local Area Networks (WLANs), standardized in IEEE 802.11 and known as Wi-Fi are fundamental in daily internet communication. However, collision and interference may hinder Wi-Fi performance. This paper delves into models applied in layer 1 and 2 to mitigate performance issues. It presents 3 Machine Learning (ML) and Artificial intelligence (AI) based approaches. We exhibit Chen et al.'s [1] model called *Aifi* to remove interference as well as Ali et al.'s [2] approach to reduce collision. Then we proceed with Coronado et al.'s [3] mechanism that utilizes ML to determine the frames' length for each station.**

***Index Terms*—Wi-Fi, WLAN, IEEE 802.11, machine learning, deep learning, artificial intelligence, physical layer, data link layer.**

1. Introduction

The paper illustrates the deployability of AI and ML in Wi-Fi optimization, especially interference mitigation, collision avoidance, and frame aggregation. It demonstrates how each approach mitigates and enhances Wi-Fi performance. Wireless interference significantly impedes Wi-Fi performance, and mitigating its effects remains a critical challenge. Most available solutions to this issue rely on *CSMA/CA* [4]–[6], which lacks efficiency in removing interference. To diminish the issue, Chen et al. in [1] create an interference cancellation technique called *Aifi* that leverages the power of artificial intelligence to estimate and remove the interference. The models estimate the interference based only on the available physical information. *Aifi* shows promising results in reducing load latency. Beyond interference mitigation, channel access represents the most extensively addressed topic regarding the influence of ML on Wi-Fi performance. Ali et al. [2] propose a machine learning-based approach to dynamically choose the optimal contention window value to avoid collisions, while simultaneously accounting for the dynamic Wi-Fi environment. This model [7, Section A.1] shows steady throughput in dense networks. Frame aggregation in IEEE 802.11 is another area of improvement in layer 2. The two approaches provided by the 802.11 standard fail to accommodate the Wi-Fi dynamics. Thus, Coronado et al. [3] use supervised learning techniques to provide the optimal frame size for each station. The authors' model has significantly lowered the retransmission rate. The paper further analyzes in Section 2 Chen et al.'s [1] approach to remove interference. Section 3.1 studies in depth Ali et al.'s [2] model and its outcome whereas Section 3.2 decomposes the strategy adopted by Coronado et al. [3] and elucidates how ML is deployed in Wi-Fi. Then the paper recapitulates and outlines the main challenges that endanger AI and ML deployability in Section 4.

2. AI and ML in Optimizing the Physical Layer

Artificial intelligence and machine learning play a crucial role in mitigating the trade-offs and the issues that we face in the physical layer. In this section, we address interference and specify how ML and AI mitigated it based on the physical layer information.

2.1. Interference and its Mitigation

The approach to mitigate interference is based on *CSMA/CA* [4]–[6] . At first, the sender senses the carrier to determine if another device is sending at that moment. This process is called the *carrier sense (CS)*. If the channel is busy, *collision avoidance (CA)* is triggered, resulting in a delay as the transmission is postponed. However, *CSMA/CA* is limited as it can not fully eliminate the interference. Chen et al. in [1] come up with the idea to analyze the information available in the physical layer instead of probing in the air.

2.2. Background and Motivation

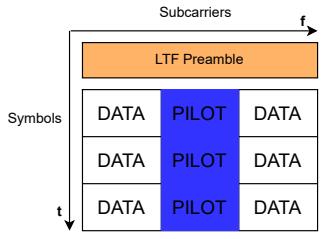


Figure 1: Physical Information in a Wi-Fi Frame [1]

The Figure 1 illustrates how the physical information is encapsulated in a Wi-Fi frame. As shown above, on the x-axis the frequency is divided into subcarriers. There is always an alternation between a block of data that carries the user's data and a pilot block used to observe the difference between the received signal and an ideal one. *Channel State Information (CSI)* [8] and *Pilot Information (PI)* [8] are extracted from the pilot subcarrier. A *Long Training Field (LTF)* preamble is spanned over all the subcarriers to synchronize them. The *LTF* preamble provides detailed specifications about the channel, mainly

 doi: 10.2313/NET-2025-05-1_10

used for channel estimation. Symbols stored in the subcarriers and the *LTF* preamble are illustrated by the y-axis. *CSI* gives the overall channel response and provides information on how the channel affected the signal. *PI* is a subset that specifically focuses on phase shifts and is used for phase synchronization. Since the pilot subcarrier and the *LTF* preamble are based on a *Binary Phase Shift Keying (BPSK)* [1, Section 2.1], the difference between two consecutive *PI* samples or *CSI* samples is always a *fixed Phase* [1]. *BPSK* is a modulation technique based on two phases that are seperated by 180° and can also be termed *2-PSK*. However, interference may alter the phase variation in *PI* and the frequency domain, represented by *CSI*, from linear to nonlinear. This nonlinearity is explained by the channel estimation [1] as follows:

$$H_I = \frac{Y_I}{X} = \frac{HX + I}{X} = H + \frac{I}{X} \tag{1}$$

In (1) H_I illustrates the channel estimation with the presence of the interference I, whereas H refers to the interference-free channel estimation $H = Y/X$ with Y reflecting the non-interfered signal. Y_I refers to the received *LTF* signal in the presence of interference and X is the predefined *LTF* signal. The fraction I/X characterizes the effect of interference and the origin behind the phase variation's nonlinearity.

2.3. System Overview

Chen et al. in [1] contribute a mechanism called *Aifi* that starts by extracting the interference features from the *PI* and *CSI* information to estimate interference on different data subcarriers with the help of a regression neural network. Regression neural network is a machine learning technique. Depending on the training data, the regression neural network predicts the outcome, i.e., the interference estimation. In this case, the training data is a set of interfered Wi-Fi signals [1, Section 3] that are subject to non-identical interference patterns and channel conditions to guarantee the generality and adaptability of the Neural Networks (NNs). These predictions are then improved via a refinement neural network to acquire more precise interference estimations. *Aifi* conducts an interference removal process on the data subcarriers and corrects the data encoding errors across the subcarriers.

2.3.1. Interference Estimation. *Aifi* extracts the interference features [1, Section 3.1] as follows:

$$I = (H_I - H)X \tag{2}$$

When the transmitting signal X is known, I can be uniquely identified by the channel estimation [9]–[11] without interference H and with interference H_I. For this matter, *Aifi* uses a *Convolutional Neural Network (CNN)*. The idea behind *CNN* is to use the provided training data to figure out what the filters also called kernels should be. In addition, it helps with pattern recognition which in this case refers to the interference patterns. The *CNN* starts by extracting features from the interfered and non-interfered signals and then proceeds with the subtraction as shown in (2). This demonstrates that *Aifi* does not require prior knowledge about patterns of the interfering signal. The training data that serves as an input to our NNs, *Aifi* starts by collecting non-interfered *PI* and *CSI* information from Wi-Fi frames with a minimum *Signal Interference and Noise Ratio (SINR)* of 23 dB. The higher the *SINR*, the better the quality of the signal, and thus the lower the relative impact of interference and noise on the signal. The estimations collected from the interfered and non-interfered signals used in training are stored as pairs. That way *Aifi* can remove the channel estimation triggered by irrelevant factors, e. g. device mobility. If the NNs are well trained to extract the features, *Aifi* can be efficient even though the training does not englobe all the possible interfering signals. After collecting the interference features from *PI*, *Aifi* proceeds with an RNN also known as *Deconvolutional Neural Network (DeNN)*. *DeNN* performs reverse operations of *CNN* and captures the non-linearity of the data subcarriers by focusing on their *generic features* [1, Section 3.1].

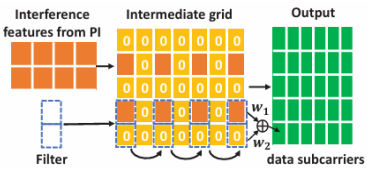


Figure 2: Regression NN [1, Section 3.1]

$$Output = w_1 \cdot x_1 + w_2 \cdot x_2 \tag{3}$$

The interference features extracted from the *PI* are expended into an intermediate grid with 0 padding, as depicted in Figure 2, to match the size of the data subcarriers. A one-Dimensional (1D) filter is applied to the intermediate grid. In a 1D filter, only the rows or the columns contribute to the output. In Figure 2, the filter is applied to the columns. The specificity of the 1D filter is its one-dimensional array kernel. The kernel length is set to 2 to process every pair of adjacent elements. In Figure 2, 1D filter has two trainable weights w_1 and w_2. Those weights are initially initialized randomly and then updated during the training process. The 1D filter slides across the intermediate grid operating as shown in (3). In this operation, x_1 and x_2 illustrate the adjacent elements over which the filter is applied. x_1 is weighted by w_1 likewise to x_2, i.e., weighted by w_2 and their weighted sum provides the output. The size of the 1D filter is chosen in a way to leverage the *continuity* [1, Section 3.1] between every two consecutive subcarriers.

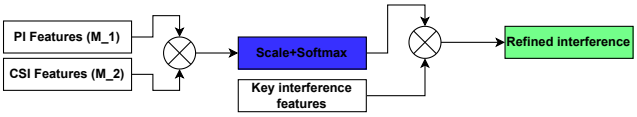


Figure 3: Feature Refinement NN [1, Section 4]

$$W = \text{softmax}\left(\frac{M_1 \cdot M_2}{\sqrt{\text{Scale}}}\right) \tag{4}$$

However, the estimation based only on the interference features captured from the *PI* lacks accuracy. To solve this issue Chen et al. [1] use the interference features from *CSI* to enhance the estimations' accuracy. This is done

 doi: 10.2313/NET-2025-05-1_10

via *stacked refinement NNs* to ensure the highest learning power level. The main purpose of the *refinement NN* is to learn the weight matrix (W) as depicted in (4). (W) reflects the correlation [1, Section 4] between *PI* and the *CSI* interference features. As shown in Figure 3 the *PI* features (M_1) and the *CSI* features (M_2) serve as input to the softmax function. The $\sqrt{\text{Scale}}$ in (4) is a constant that prevents the correlation from growing in magnitude, leading to a smaller gradient. The weight matrix is then applied to the key interference features extracted from either the *PI* or the *CSI*. As a result, a refined interference is obtained.

2.3.2. Interference Removal.

$$X = (X_I - I) \cdot W \tag{5}$$

Aifi uses a fully connected NN to imitate the equalization process in the commodity Wi-Fi that utilizes *Zero-Forcing (ZF)* [9], [12]. *ZF* tackles the signal's interference by applying the inverse of channel estimation to the signal [1, Section 3.2]. The fully connected NN ensures flexibility by computing multiple iterations and learning the optimal weights W. In addition, it emulates the equalization in the feature space offering efficient fine-tuning abilities to remove the interference. As shown above in (5), the (X_I) refers to the received signal that contains the interference, while (I) is the estimated interference from Section 2.3.1. By multiplying the subtraction with the learnable weights, the interference-free signal X is obtained.

2.3.3. Data Payload Correction. *Aifi* starts with residual interference detection and handling. Indeed, when the interference is too strong, the interference-removing process may fail to obliterate it. This is due to the *limited resolution* [1, Section 3.3] of the signal in *PI* and *CSI*. To mitigate this issue, *Aifi* incorporates a supplementary neural network to correct the decoding errors repercussed by the interference. *Aifi* then proceeds to emulate the Wi-Fi encoder. In this stage, *Aifi* utilizes a *Long-Short-Term Memory (LSTM)* network. *LSTM* is a type of a reccurent network used to mitigate the issue of long-term dependency in sequential data. In the recurrent NN the output of the previous input is fed into the node as part of the input. *LSTM* enhances this structure by adding an internal state mechanism. The state comprises three gates: the forget gate, the input gate, and the output gate. Each gate can be assigned a number between 0 and 1. *LSTM* learns the dependency between consecutive symbols and in case of an error, the *LSTM* network attempts to correct it by rebuilding the correct data payload. *Aifi* takes the output of the *LSTM* network and transforms it into the final data payload with the help of a demodulation neural network. This type of neural network tends to mimic the decoding behavior in the commodity Wi-Fi.

2.4. Real-Word Experimentation

The relevance of *Aifi* in boosting the network's performance in real-world applications, especially the loading of web pages is evaluated by Chen et al. [1]. The experiment is performed on a couple of well-known web pages e. g. Google.com, Delta.com, Twitter.com, Twitch.com [1, Section 8.2] and a predefined Wi-Fi transmission of 24Mbps. Before we delve into analyzing the outcome of the experience, we need to explain the *Frame Reception Rate (FRR)*. *FRR* is defined as:

$$\text{FRR} = \frac{\sum \text{successfully received frames}}{\sum \text{transmitted frames}} \tag{6}$$

As shown in (6), low *FRR* reflects a high impact of interference. Interference engenders high packet loss. High packet loss repercusses elevated latency. As depicted in Figure 4, in the case of low *FRR* ($FRR \leq 30\%$) we can see that *Aifi* prevails the most. When the interference is too strong ($FRR = 0\%$), we can remark that Twitter had the highest load latency (200 seconds). The loading functionality of Twitter was nearly unresponsive. However, with *Aifi* deployed, the load latency plummeted (80 seconds). The loading functionality of Twitch is now usable. All the web pages (Google.com, Delta.com, Twitter.com) experienced this huge load latency reduction with *Aifi* deployed, when the interference is prevailing. When the *FRR* is medium (between 40% and 70%), the load latency is reduced from around 160 seconds to around 50 seconds on average for all the web pages. And even in the case of high *FRR* ($FRR > 70\%$) *Aifi* ensures that the load latency is less than 3 seconds. It offers a good web browsing experience. To sum up, the model can transform the web pages' loading outstandingly.

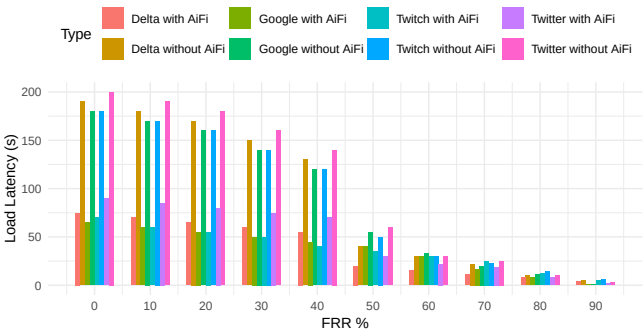


Figure 4: Latency of webpages loading [1, Section 8.2]

3. AI and ML in Optimizing the Data Link Layer

The main strength of AI and ML relies upon their power to gain knowledge rapidly, generalize it, and learn from previous experiences. In this section, we delve into the panoply of approaches that use AI and ML to optimize the data link layer.

3.1. Channel Access

Most applied approaches are tightly linked to the *Distribute Coordination Function (DCF)*. *DCF* is a *CSMA/CA*-based channel access mechanism [13]. Devices randomly select a backoff counter from the *Contention Window (CW)* range. This backoff counter represents the waiting time of this device before accessing the channel to avoid collision. The *CW* range is the following $\{0, 1, 2, \ldots, \min\{2^{c+k-1} - 1, 255\}\}$. (c) is a constant related to the physical configuration and (k) illustrates the number of transmission attempts starting from $k = 1$. One

 doi: 10.2313/NET-2025-05-1_10

of the commonly known trade-offs is the choice of the *CW* values. Indeed, choosing small *CW* values provide quicker transmissions but increase the collision risk and thus reduce input, whereas large *CW* values minimize the probability of collision but increase the idle time of the channel leading to a reduction in the throughput [7, Section A]. The following sections provide the major findings on how to mitigate this issue with the help of ML.

3.1.1. Collision Reduction. In high-density 802.11ax WLANs, Ali et al. [2] use a combination of *Reinforcement Learning (RL)* [14] and *Intelligent Q-learning-based Resource Allocation (IQRA)*. The purpose of *RL* is to optimize a policy to yield maximum reports. The report in our case refers to the increase of throughput and the reduction of collision probability. An *RL* consists of an agent, a set of states (S), and Actions ($\mathcal{A}$). When the agent acts $a \in \mathcal{A}$, the agent transitions from one state to another. As opposed to *DCF* which resets the *CW* value every time the channel is idle, the *CW* value is now calculated by analyzing the channel collision probabilities that are based on the *Channel Observation-based Scaled Backoff (COSB)* [15] mechanism. Before scaling the *CW*, *COSB* measures how often the channel is busy. It then predicts the likelihood of collisions according to failed transmissions or retransmission attempts while accounting simultaneously for the number of active devices. That way the *IQRA* mechanism can manage between *exploring new actions* [7, Section A.1] like new *CW* adjustment and choosing the *optimal actions* [7, Section A.1] by increasing or reducing the *CW* (according to COSB). *Optimal actions* reflect the safest strategy of leveraging the already available information to be efficient in the current environment. The *IQRA* mechanism always accounts for the unstability of the Wi-Fi environment.

3.1.2. Real-Word Experimentation. Results from the ns-3 network simulator [7, Section A.1] for dense networks (with 50 stations) as shown in Figure 5 illustrate the positive effect of *IQRA* on the network throughput compared to the standard 802.11 protocol. *IQRA* provides a nearly constant network throughput during the whole Simulation time (from 0s to 60s) around $38\,\mathrm{MB\,s^{-1}}$ contrary to the standard protocol that shows relatively low throughput. In the standard protocol, the network throughput decreases from $38\,\mathrm{MB\,s^{-1}}$ to around $27\,\mathrm{MB\,s^{-1}}$.

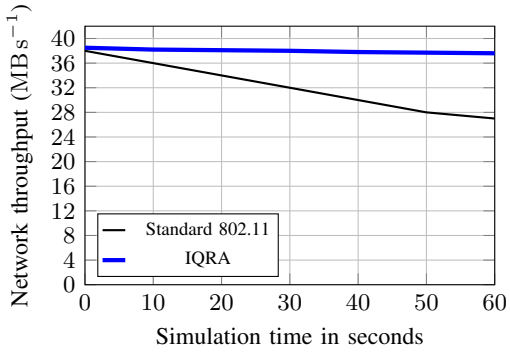


Figure 5: Network Throughput Comparison [7, Section A.1]

3.2. Frame Aggregation

Frame aggregation consists of combining smaller frames into a bigger one to improve efficiency in communication. Larger frames can reduce the overhead, however, they are more prone to transmission errors. One error in one of the subframes can cause the retransmission of the large frame. To tackle this trade-off, the 802.11 standard introduced two basic approaches. The first method is called *Aggregated Mac Service Data Unit (A-MSDU)* [16]. *A-MSDU* is more efficient as it has only one *Frame Check Sequence (FCS)* but is less robust. If one subframe is corrupted the entire frame is discarded. The second method is the *Aggregated Mac Protocol Data Unit (A-MPDU)* [16]. *A-MPDU* offers more robustness because each subframe has its own *FCS* but it introduces more overhead. The optimal approach is to apply those methods simultaneously to harmonize between robustness and efficiency. The 802.11 standard does not account for the *CSI*'s dynamic nature. Thus, to optimally choose the frame size, we leverage the dynamic features of ML. To optimally select the frame size under dynamic conditions, Coronado et al. [3] implement a *low computational complexity* [3, Section 1] technique based on a *Random Forest Regressor (RFR)* for the Modulation and Coding Scheme (MCS) settings and the frame aggregation. The MCS is a metric that reflects several parameters between the Access Point (AP) and the station including data rate, channel width, etc. The model is deployed on the management plane and fed periodically with the control plane's knowledge, e. g. channel conditions and user state. This model is considered to have *low computational complexity* because the type of ML model deployed requires minimal resources, reduced training time, and provides fast outcome predictions. A *Random Forest (RF)* is a ML model that uses an ensemble of decision trees to make its prediction. Each random forest is characterized by three parameters: the size of the nodes, the number of trees, and the number of features.

3.2.1. Model Description. The model utilizes the features like the station details as input [3, Section 4.3.5]. It assigns an *RFR* for each MCS and limits the depth of the tree to 3. By this limitation, the model avoids overfitting and reduces the complexity of the tree. The model then undergoes a tenfold cross-validation. Indeed, the data is split into 10 parts. The model is trained on 9 folds and is evaluated on the fold left. The model is deployed on the management plane. The management plane is responsible for controlling the network. An appropriate frame length is constantly provided for a specific MCS and in the next iteration, the model reconfigures itself by using the feedback from the real obtained goodput. In due course, the model can correct the next prediction with a prediction error factor [3, section 4.3.5].

3.2.2. Real-Word Experimentation. The setup consists of an AP, a Software-Defined Networking (SDN), and the ML model deployed on the management plane. A SDN ensures the configuration and monitoring of an efficient and dynamic network environment. The experiment [3, Section 5.1] consists of moving the stations within a $30\,\mathrm{m}$ radius of the AP while varying simultaneously the mobility and the MCS index (MCS-0, MCS-2, MCS-7). Each MCS index illustrates a unique combination of modulation

type, coding rate, and several other parameters. Performance is evaluated on the goodput metric [3, Section 5.2]. The simulation is repeated 10 times with 14 seconds runs. The model illustrated its ability to dynamically improve goodput by about 18.36% compared to static techniques like max aggregation and no aggregation. The model tends to outperform all the other approaches, especially when the bitrate is high ($25\,\mathrm{MB\,s^{-1}}$) and the MCS is equal to 7. The approach has significantly lowered the retransmission rate.

4. Conclusion and Challenges

The impact of ML and AI is undeniable in improving Wi-Fi performance. This paper provides a comprehensive overview of 3 recent ML-based approaches deployed in the first two layers. It starts with the approach of Chen et al. [1] called *Aifi*. This approach shows its efficiency in mitigating interference and reducing 80% bit errors. *Aifi* improves the *FRR* by a factor of 18. However, the applicability to various wireless technologies may pose both an impediment and a challenge for *Aifi*. Indeed, this approach applies to only OFDM-based systems [1, section 9]. Some wireless systems have different physical structures and may provide their physical information differently. Then the paper proceeds by analyzing Ali et al.'s [2] approach. This model ensures steady network throughput by reducing the chances of collision based on a combination of *RL* [14] and *IQRA*. For the frame aggregation, we scrutinize Coronado et al.'s [3] approach. This model provides the optimal frame length for each station and ensures a low retransmission rate. However, ML and AI may raise privacy and security concerns. The pace of AI and ML deployability outran the pace of their regulations. The success of AI and ML depends on future standardizations to avoid data transfer that involves sensitive information used for the training of ML models.

References

[1] K. H. Ruirong Chen and W. Gao, "AiFi: AI-Enabled WiFi Interference Cancellation with Commodity PHY-Layer Information," *ACM Conference on Embedded Networked Sensor Systems (SenSys '22)*, pp. 134–148, 2022.

[2] R. Ali, N. Shahin, Y. B. Zikria, B.-S. Kim, and S. W. Kim, "Deep reinforcement learning paradigm for performance optimization of channel observation—based mac protocols in dense wlans," *IEEE Access*, vol. 7, pp. 3500–3511, 2019.

[3] E. Coronado, A. Thomas, and R. Riggio, "Adaptive ML-based frame length optimisation in enterprise SD-WLANs," *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 850–881, Oct 2020.

[4] G. Bianchi, L. Fratta, and M. Oliveri, "Performance evaluation and enhancement of the csma/ca mac protocol for 802.11 wireless lans," in *Proceedings of the 7th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 1996, pp. 392–396.

[5] E. Felemban and E. Ekici, "Single hop ieee 802.11 dcf analysis revisited: Accurate modeling of channel access delay and throughput for saturated and unsaturated traffic cases," *IEEE Transactions on Wireless Communications*, vol. 10, no. 10, pp. 3256–3266, 2011.

[6] X. Wang and K. Kar, "Throughput modelling and fairness issues in csma/ca based ad-hoc networks," in *Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1. IEEE, 2005, pp. 23–34.

[7] S. Szott, K. Kosek-Szott, P. Gawlowicz, J. Torres Gómez, B. Bellalta, A. Zubow, and F. Dressler, "Wi-fi meets ml: A survey on improving ieee 802.11 performance with machine learning," p. 51, 09 2022.

[8] *IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std. 802.11-2012, March 2012.

[9] A. Klein, G. K. Kaleh, and P. W. Baier, "Zero forcing and minimum mean-square error equalization for multiuser detection in code-division multiple-access channels," *IEEE Transactions on Vehicular Technology*, vol. 45, no. 2, pp. 276–287, 1996.

[10] Y. Jiang, M. K. Varanasi, and J. Li, "Performance analysis of zf and mmse equalizers for mimo systems: An in-depth study of the high snr regime," *IEEE Transactions on Information Theory*, vol. 57, no. 4, pp. 2008–2026, 2011.

[11] P. Schniter, "Low-complexity equalization of ofdm in doubly selective channels," *IEEE Transactions on Signal Processing*, vol. 52, no. 4, pp. 1002–1011, 2004.

[12] Y. Ding, T. N. Davidson, Z.-Q. Luo, and K. M. Wong, "Minimum BER block precoders for zero-forcing equalization," *IEEE Transactions on Signal Processing*, vol. 51, no. 9, pp. 2410–2423, 2003.

[13] N. Shenoy, J. Hamilton, A. Kwasinski, and K. Xiong, "An improved ieee 802.11 csma/ca medium access mechanism through the introduction of random short delays," in *2015 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2015, pp. 331–338.

[14] C. K. Lee and S. H. Rhee, "Collision avoidance in IEEE 802.11 DCF using a reinforcement learning method," in *Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC)*, Oct. 2020, pp. 898–901.

[15] R. Ali, N. Shahin, Y.-T. Kim, B.-S. Kim, and S. W. Kim, "Channel observation-based scaled backoff mechanism for high efficiency WLANs," *Electronics Letters*, vol. 54, pp. 663–665, May 2018.

[16] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A tutorial on ieee 802.11ax high efficiency wlans," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 197–216, first quarter 2019.

Machine Learning Techniques for Self-Managing Networks

Bechir Boujelbene, Johannes Späth*, Max Helm*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: bechir.boujelbene@tum.de, spaethj@net.in.tum.de,helm@net.in.tum.de

***Abstract*—As modern network systems become increasingly complex and dynamic, traditional approaches to Root Cause Analysis (RCA) encounter inherent limitations when confronted with the needs of real-time analysis, scalability and the processing of vast volumes of generated input data. RCA refers to the process of identifying the root causes of observed failures within a network system. In this regard, Machine Learning (ML) approaches have emerged as a compelling alternative, capitalizing on their ability to process large-scale data and uncover complex patterns and distributions. This paper presents popular ML techniques and discusses their applicability to derive models for RCA in network fault management, highlighting their strengths, scalability and limitations.**

***Index Terms*—network failure diagnosis, root cause analysis models, telemetric data, machine learning**

1. Introduction

With the growing complexity and expansion of modern network topologies and the rising amount of generated traffic and connected devices [1], efficient network management has become more needed than ever. Network management encompasses all applied processes and tools designed to ensure the reliability, efficient performance and security of the network infrastructure [2]. In particular, root cause analysis (RCA) is considered to be a central aspect of network management. RCA models are designed to backtrack and identify the set of potential root causes that are responsible for a network failure [3].

Traditionally, constructing such models relies on the domain expertise of human operators, with the eventual aim of deriving a knowledge base of rules to diagnose network failures [4]. These rules depend on telemetric data measurements collected from various network devices and monitoring tools. The relevant input data for RCA models may include interface statistics or system logs from routers, or performance metrics such as CPU and memory usage from servers and endpoints [5].

However, as modern network traffic becomes more complex, vast amounts of data are being generated, which makes it impossible for humans to entirely process it. This results in traditional models failing to exploit all of the collected data and ignoring certain features that could potentially decrease the diagnostic output accuracy. Moreover, the process of constructing these rules is time consuming, unsuitable for environments where real-time reaction is essential, and not scalable to larger networks [4].

In this context, and in light of the recent success of machine learning (ML) applications in many areas of technology and science [6], ML techniques have emerged as a promising alternative to traditional RCA models. ML models feed off large volumes of input data [7], which is increasingly available on modern networks. In this paper, we present and discuss popular ML approaches that can be applied for deriving RCA models in networks based on input telemetric data.

The remainder of this paper is structured as follows: Section 2 presents key concepts of RCA in the context of network management. Section 3 highlights briefly previous related works. Section 4 introduces various suitable approaches for deriving RCA models and discusses their strengths and limitations.

2. RCA in Network Management

This section presents the fundamental concepts and terminology related to the field of network management and RCA, which are essential to understand the subsequent discussions of the approaches used to derive RCA models.

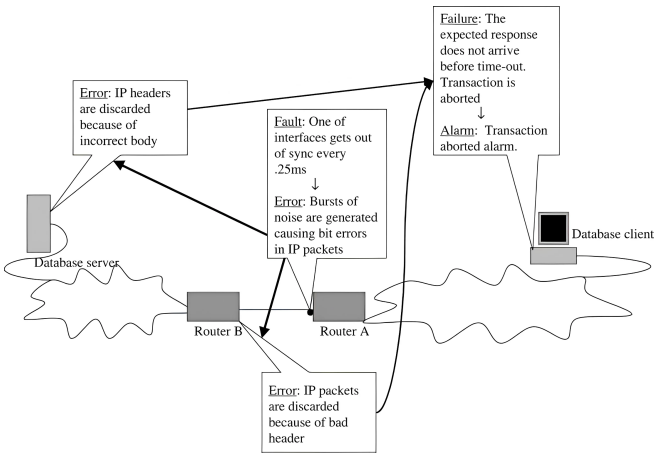


Figure 1: Illustration of different terminologies in RCA process (Figure from [8])

2.1. Terminology

We introduce the following terminologies, based on the previous works [9] [4]:
Network Error: is defined as the discrepancy between a condition of the network system and its theoretically

 doi: 10.2313/NET-2025-05-1_11

correct condition and is caused by one or many faults.
Network Faults: are also called root causes. These are network errors that can cause other errors but are not themselves caused by other errors. In other words, a network fault is the root cause of some error.
Network Failure: is an error that is observable from outside the system through external indicators called symptoms such as alarms raised upon anomaly detection. Upon detecting the failure, the telemetric data and statistics generated by network devices are then collected and used as input for the RCA model for diagnosis and producing the most likely root causes as outputs.
Root cause analysis: RCA is the process of determining the set of faults or root causes that generated originally the network failure observed by the given set of symptoms and associated with the generated telemetric data. Figure 1 illustrates these concepts using a network scenario where a failure occurs when a client attempts to access a database server. The failure is detected through a raised alarm, triggered by the absence of a response from the server. The fault originates from a hardware issue in the interface of a router along the path between the client and the server, causing bit errors in packets sent by the client, which are subsequently discarded by the server. This scenario also demonstrates how faults can originate at locations far from where their failure manifestations are observed.

2.2. RCA Workflow

The complete RCA process, starting from the model construction to the inference of probable root causes for a network failure, adheres to the workflow depicted in Figure 2. The first step would be to collect labeled telemetric data, which are historically observed data instances in network devices upon detecting symptoms of a network failure. Each instance is annotated with the corresponding fault or set of faults as labels. Combined with domain and system knowledge, an appropriate RCA model is constructed. These additional knowledge sources, however, are not always necessarily used, particularly when large datasets are available and the RCA models rely entirely on ML approaches with complex architectures. These models act from the outside as black boxes, extracting patterns from large datasets without requiring domain or system knowledge. Once training is complete and the RCA model is constructed, it can be used for inference. Telemetric data generated in response to new network failures is fed into the model to produce as an output the expected root causes. Furthermore, if a change in the network system occurs upon for example removing or adding new devices, the RCA model is updated.

3. Related Work

The preceding survey [3] highlighted the existing RCA models in various IT systems disciplines and not specifically in the context of network management. The survey emphasized the generation and inference algorithms of the models, with particular attention paid to performance aspects. In addition, previous papers [10] [8] discussed the challenges of fault localization in complex modern network systems and presented an overview of recent techniques and models as proposed solutions.

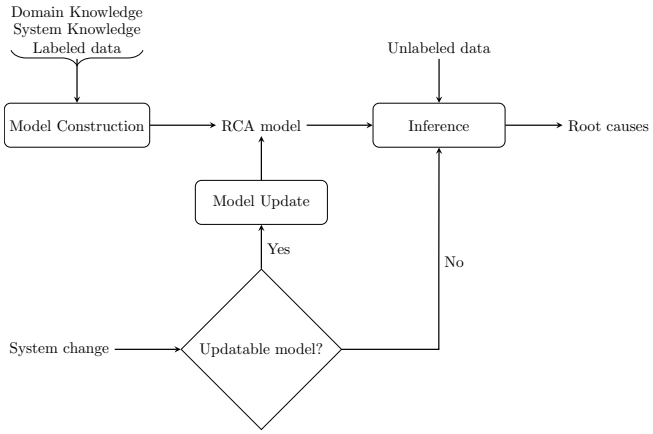


Figure 2: RCA workflow (Diagram was adapted from [3])

4. Approaches for RCA Models Derivation

This section provides an overview of various suitable approaches and techniques for deriving RCA models. In particular, we start by briefly introducing non-ML models and then proceed to delve in depth into ML models, highlighting their advantages, disadvantages, and scalability as summarized in Table 1.

4.1. Non ML Models

Non-ML models for RCA are based on deterministic approaches that do not involve any training algorithms or optimization techniques on the input telemetric data given the corresponding labels. While they are often easier to interpret and understand, their effectiveness is limited in modern, dynamic networks where it becomes quite challenging for such models to describe the complex distribution of the correlation between symptom data and faults. A review of the literature reveals numerous examples of such models that were used as the primary solution for RCA and localizing the root causes of network failures. In the following we list two widely applied approaches.

Rule-based Models: These models rely on predefined logical rules that are derived from human expertise in domain and system knowledge [11]. The rules are often expressed as if-then statements and the models rely on forward-chaining inference to produce potential faults as an output by executing the rules that were triggered, i.e., those whose conditions matched the input data [8]. One common method for representing the rules is through the use of codebooks, which map each network fault to a set of symptom data that should be observed in the faulty component itself and any affected components resulting from the original fault. The underlying root causes are then diagnosed by identifying the closest match to the observed input data. Reali et al. [12] employed this technique within a real Next Generation Network (NGN) that deployed wireline Voice over Internet Protocol (VoIP).

Pattern Mining-based Models: Pattern mining is a central task in the subfield of data mining that aims to analyze data in order to extract recurring patterns and strong

 doi: 10.2313/NET-2025-05-1_11

TABLE 1: Summary of Advantages and Disadvantages of ML Approaches for RCA

ML Approach	Advantages	Disadvantages
Decision Trees	Human-interpretable models; logical rules are easy to extract and align with domain knowledge; efficient inference execution.	Susceptible to overfitting and noisy network data; require ensemble methods for better generalization; limited in scalability.
Artificial Neural Networks	Capable of handling large-scale network data and capturing complex distributions; perform well in presence of noisy telemetric data.	Acts as a black box; lacks interpretability; training process can be expensive especially for complex architectures.
Support Vector Machines	Effective for high-dimensional telemetric data in network's RCA; can deal with non linear complex distributions.	Training is computationally expensive for large scale datasets; can underperform when trained to predict a large number of root causes.
Clustering	Useful when no historical labeled telemetric data is available.	Requires pre-selection of the number of clusters; unable to identify specific root causes but can only group based on similar observations.
Bayesian Networks	Human-interpretable models; readable dependencies; can combine domain knowledge with data-driven inference.	Model construction is expensive for large-scale networks; requires expert knowledge to define graph structure.

correlations [13]. This helps to facilitate decision-making for many tasks such as classification and prediction. In the context of network management, pattern mining techniques have been applied to analyze telemetric network data, enabling the discovery of meaningful patterns that assist in fault localization and RCA. For instance, Lozonavu et al. [14] applied sequential pattern mining [15] to discover correlations between network alarm instances. This approach constructs a directed weighted graph, where the nodes and directed edges represent the relations between different alarms and the associated weights illustrate the strength or also called the confidence of these relations. By deploying these dependencies, the model starts its inference with a network entity that reported an alarm. It then determines which other alarms are correlated, enabling the system to pinpoint the faulty network elements more precisely.

4.2. ML Models

ML is a scientific discipline concerned with the design of models capable to learn patterns and distributions of input data. The latter is typically partitioned into three complementary subsets: training, validation and testing sets [16]. The training set is used to train the model by optimizing its parameters to minimize a loss function between true outputs and predicted outputs by the model. The validation set fine-tunes hyperparameters and prevents overfitting, while the testing set assesses accuracy and generalization on unseen data [16]. This subsection reviews popular ML techniques that can be applied for RCA in network systems and summarizes the advantages, disadvantages and scalability of these approaches, as illustrated in Table 1.

Decision Trees: Decision Trees have been widely applied to fault localization and RCA in network systems. In particular, it is a supervised learning technique that requires labeled telemetric data annotated with the corresponding network faults [17]. Each distinct network fault or set of faults is represented by a class and the collected metrics such as interface statistics, bandwidth and memory usage are referred to as attributes.

A decision tree is a tree-like model that classifies data instances into classes represented by leaf nodes based on their attribute values. Internal nodes represent a test of an attribute and each outgoing branch from an internal node represents a possible range of values of this attribute. Learning a decision tree involves deciding which attribute and its associated test should be selected at each internal node to optimally split the data into branches. In general, optimal splits are picked by maximizing the "Gain" in information. The gain can be computed using various criteria, such as Entropy or Gini Index but the choice itself should not affect the ultimate model performance [18]. The inference algorithm for new instances is then applied by traversing the learned tree from the root node and following the branches based on the attribute values until a leaf node is reached which acts as the predicted root cause(s) of the model.

Chen et al. [17] deployed a decision tree based model combined with post processing performed on paths of the tree in order to identify causes of failures in large internet systems. In the field of RCA in network management, decision trees can be preferred as they have the advantage of yielding human interpretable results, which makes it easier to follow and understand the decisions made along the output path [10]. Furthermore, the model exhibits efficient runtime performance for the inference algorithm, rendering it well-suited for systems where time sensitivity and real-time analysis are crucial factors [3]. However, the scalability of these techniques can be limited to using only specific attributes of the input data, and the accuracy of the model can be severely degraded in the presence of noisy input data, a problem that is exacerbated in large-scale networks [17].

In fact, to address the last limitation and to improve generalization on unseen data, Random Forest (RF) techniques can be employed. RF is an ensemble learning method and its core concept lies in constructing multiple decision trees during the training process, rather than deriving only one. The inference outputs of the trees are then combined typically through majority voting, to make more robust and accurate predictions. For instance, Sauvanaud et al. [19] implemented an RF algorithm to localize root causes in Virtual Network Functions (VNFs).

Artificial Neural Networks (ANNs): ANNs are computational models inspired by the structure and learning mechanisms of biological neural networks (NNs) in the human brain. An ANN consists of multiple layers of interconnected neurons. Each neuron receives multiple inputs

 doi: 10.2313/NET-2025-05-1_11

from the previous layer, processes them, and generates a single output that is fed to each neuron in the next layer. This processing involves a weighted sum of the inputs, an addition of a bias, and the application of an activation function. The weights and biases represent the learned parameters of the model [20].

In the context of RCA in networks, the neurons of the input layer represent the telemetric data during network failures and the output neurons correspond to the root causes. The root cause associated with the output neuron exhibiting the highest value represents the predicted fault of the model. Wietgrefe et al. [21] developed a system called Cascade Correlation Alarm Correlator (CCAC) based on an ANN to predict the root causes of alarms in cellular phone networks. Each input neuron represents an alarm type and takes a binary value (active or inactive), while each output layer neuron corresponds to a failure's cause. The findings of [21] indicate that CCAC yields high prediction accuracy even in the presence of noise in the training data such as missing or irrelevant alarms. Furthermore, in a comparative study conducted by Wietgrefe et al. [22], it was demonstrated that CCAC is more accurate at predicting alarm causes than traditional approaches such as rule-based reasoning models. In general, ANN approaches have the ability to process the large-scale telemetric data produced by modern networks and can produce accurate predictions even with the presence of noisy data.

The above advantages may justify the fact that ANNs have been the subject of extensive research and are widely employed in numerous domains and fields [20]. However, it seems that ANNs have not achieved the same dominance in the area of RCA in network systems, unlike other disciplines. The primary reason behind this is that these models, especially those possessing complex architectures like deep ANNs, act from outside like black boxes and return predicted root cause(s) as an output, but it is almost impossible for human operators to backtrack and provide a logical explanation for it [3]. Moreover, such approaches use exclusively the labeled input dataset to construct the RCA models and are difficult to combine with available domain knowledge to derive meaningful and interpretable rules [3].

Support Vectoring Machines: Support Vector Machine (SVM) is another popular ML technique that can be applied to RCA and fault management in networks. SVM is essentially a linear supervised classifier that is based on the margin maximization principle [23]. To deal with non linear problems, which is the case in network's RCA, the input data can be preprocessed and mapped to higher dimensions using kernel methods. This process is called non-linear SVM [4].

Based on the training labeled telemetric data, an SVM is learned to find optimal separating hyper planes with each plane representing a network failure root cause(s). In the literature, we can find the application of SVM methods to a variety of network management tasks. For instance, the study conducted by Zidi et al. [24] applied an SVM-based model to detect failures in Wireless Sensor Networks (WSNs). WSNs consist of autonomous devices collaborating together through a wireless channel. The training dataset included both normal data measurements as well as measurements associated with different types of faults. Once the SVM model is trained, the inference algorithm predicts if new observations belong to a normal or a faulty case. The experimental results conducted by Zidi et al. [24] show that their SVM method achieves high accuracy rates.

In general, SVMs have the advantage of performing well in scenarios with high-dimensional input data [3], making them suitable for analyzing complex telemetric datasets in network management. However, SVMs seem to underperform when trained to predict a relatively large number of output classes, a common scenario when representing the diverse root causes of network failures [25]. As a result, their applicability can be limited to specific network failure scenarios.

Clustering: Clustering methods are unsupervised learning techniques that group data instances into clusters based on similar features or patterns, without the need for labeled data [9]. This is useful in network's RCA when the training telemetric data is not accompanied by the corresponding root causes. Such scenarios may arise due to a lack of historical labeled data or the presence of excessively noisy data, making supervised learning impractical.

Sozuer et al. [26] applied clustering techniques to identify correlated alarms belonging to the same cluster during network failure. This helps pinpoint the faulty network elements more precisely and localize the root causes. However, clustering models require predefining the number of clusters, and without expertise knowledge of the underlying network system, an incorrect choice can result in meaninglessly grouping telemetric measurements that do not originate from the same root cause(s) [9].

Bayesian Networks: A Bayesian Network (BN) is a probabilistic graphical model that represents variables and their conditional dependencies through a directed acyclic graph (DAG) [25]. The conditional probabilities are learned using techniques like Maximum Likelihood Estimation (MLE) or Bayesian Estimation [3]. In the context of RCA in networks, Bayesian Networks can model the causal relationships between the telemetric data, symptoms, and faults that act as variables of the model. For example, if an interface fault in a router causes increased latency and packet drops, a BN can capture these dependencies and help infer the root cause when these symptoms are observed.

Ruiz et al. [27] developed a BN model to identify the root causes of network failures at the optical layer. Khanafer et al. [28] proposed a failure diagnosis model using BN approach for Universal Mobile Telecommunications System (UMTS) networks. The dependencies between the variables in BN models are intuitively easy for human operators to understand. Moreover, the explicit representation of causes and effects enhances readability, making it easier to derive meaningful rules and integrate them with domain and system knowledge. However, constructing BNs can be computationally expensive, particularly for large-scale networks, and requires significant expertise to define the graph structure and which variables are included. Additionally, their scalability may be limited when dealing with high-dimensional datasets [25].

5. Conclusion

In this paper, we examined various approaches that can be applied to derive RCA models in network systems. Initially, we showed that traditional non-ML models, while easier to read and interpret, face major limitations when employed in dynamic and large-scale networks. We then proceeded to explore in depth ML models. Our review demonstrated that these techniques are able to handle large amounts of input telemetric data and identify the root causes of network failures more accurately and adaptively. Nevertheless, challenges such as explainability, computational cost, and exclusive reliance on input data still remain. A possible future work could aim to address these limitations by exploring hybrid models that combine the strengths of ML techniques with domain knowledge from traditional approaches, leading potentially to more robust RCA solutions.

References

[1] A. M. Odlyzko, "Internet traffic growth: Sources and implications," in *Optical transmission systems and equipment for WDM networking II*, vol. 5247. SPIE, 2003, pp. 1–15.

[2] L. Tawalbeh, *Network Management*, 04 2020, pp. 99–115.

[3] M. Solé, V. Muntés-Mulero, A. I. Rana, and G. Estrada, "Survey on models and techniques for root-cause analysis," *arXiv preprint arXiv:1701.08546*, 2017.

[4] M. Nouioua, P. Fournier-Viger, G. He, F. Nouioua, and Z. Min, "A survey of machine learning for network fault management," *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics: Theories and Applications*, pp. 1–27, 2021.

[5] T. Wang and G. Qi, "A comprehensive survey on root cause analysis in (micro) services: Methodologies, challenges, and trends," *arXiv preprint arXiv:2408.00803*, 2024.

[6] R. Pugliese, S. Regondi, and R. Marini, "Machine learning-based approach: global trends, research directions, and regulatory standpoints," *Data Science and Management*, vol. 4, pp. 19–29, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666764921000485

[7] "Training Data Quality: Why It Matters in Machine Learning — v7labs.com," https://www.v7labs.com/blog/quality-training-data-for-machine-learning-guide, [Accessed 03-12-2024].

[8] M. łgorzata Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, no. 2, pp. 165–194, 2004, topics in System Administration. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642304000772

[9] S. P. Kavulya, K. Joshi, F. D. Giandomenico, and P. Narasimhan, "Failure diagnosis of complex systems," *Resilience assessment and evaluation of computing systems*, pp. 239–261, 2012.

[10] A. Dusia and A. S. Sethi, "Recent advances in fault localization in computer networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 3030–3051, 2016.

[11] T. Marques, "A symptom-driven expert system for isolating and correcting network faults," *IEEE Communications Magazine*, vol. 26, no. 3, pp. 6–13, 1988.

[12] G. Reali and L. Monacelli, "Definition and performance evaluation of a fault localization technique for an ngn ims network," *IEEE Transactions on Network and Service Management*, vol. 6, no. 2, pp. 122–136, 2009.

[13] P. Fournier-Viger, W. Gan, Y. Wu, M. Nouioua, W. Song, T. Truong, and H. Duong, "Pattern mining: Current challenges and opportunities," in *International Conference on Database Systems for Advanced Applications*. Springer, 2022, pp. 34–49.

[14] M. Lozonavu, M. Vlachou-Konchylaki, and V. Huang, "Relation discovery of mobile network alarms with sequential pattern mining," in *2017 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2017, pp. 363–367.

[15] J. Pei, "Mining sequential patterns efficiently by prefix-projected pattern growth," in *Proc. of 17th International Conference on Data Engineering (ICDE 2001)*, 2001, pp. 215–224.

[16] O. Hazzan and K. Mike, *Core Concepts of Machine Learning*. Cham: Springer International Publishing, 2023, pp. 209–224.

[17] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *International Conference on Autonomic Computing, 2004. Proceedings.*, 2004, pp. 36–43.

[18] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[19] C. Sauvanaud, K. Lazri, M. Kaâniche, and K. Kanoun, "Anomaly detection and root cause localization in virtual network functions," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, 2016, pp. 196–206.

[20] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research," *Journal of Pharmaceutical and Biomedical Analysis*, vol. 22, no. 5, pp. 717–727, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0731708599002721

[21] H. Wietgrefe, K.-D. Tuchs, K. Jobmann, G. Carls, P. Fröhlich, W. Nejdl, and S. Steinfeld, "Using neural networks for alarm correlation in cellular phone networks," in *International Workshop on Applications of Neural Networks to Telecommunications (IWANNT)*. Citeseer Stockholm, Sweeden, 1997, pp. 248–255.

[22] H. Wietgrefe, "Investigation and practical assessment of alarm correlation methods for the use in gsm access networks," in *NOMS 2002. IEEE/IFIP Network Operations and Management Symposium.'Management Solutions for the New Communications World'(Cat. No. 02CH37327)*. IEEE, 2002, pp. 391–403.

[23] M. M. Adankon and M. Cheriet, "Support vector machine," *Encyclopedia of biometrics*, pp. 1303–1308, 2009.

[24] S. Zidi, T. Moulahi, and B. Alaya, "Fault detection in wireless sensor networks through svm classifier," *IEEE Sensors Journal*, vol. 18, no. 1, pp. 340–347, 2018.

[25] N. G. Lo, J.-M. Flaus, and O. Adrot, "Review of machine learning approaches in fault diagnosis applied to iot systems," in *2019 International Conference on Control, Automation and Diagnosis (ICCAD)*, 2019, pp. 1–6.

[26] S. Sozuer, C. Etemoglu, and E. Zeydan, "A new approach for clustering alarm sequences in mobile operators," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 1055–1060.

[27] M. Ruiz, F. Fresi, A. P. Vela, G. Meloni, N. Sambo, F. Cugini, L. Poti, L. Velasco, and P. Castoldi, "Service-triggered failure identification/localization through monitoring of multiple parameters," in *ECOC 2016; 42nd European Conference on Optical Communication*. VDE, 2016, pp. 1–3.

[28] R. M. Khanafer, B. Solana, J. Triola, R. Barco, L. Moltsen, Z. Altman, and P. Lazaro, "Automated diagnosis for umts networks using bayesian network approach," *IEEE Transactions on vehicular technology*, vol. 57, no. 4, pp. 2451–2461, 2008.

Experiment-Based Reverse Engineering of Signing Protocols for Smart Cards

Dennis Evtushenko, Stefan Genchev*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: ge95ved@mytum.de, genchev@net.in.tum.de

***Abstract*—Smart cards dominate the market in many different aspects, such as digital payment with a credit card or for any application for security, authentication or identification. In order for them to be used so widely, manufacturers offer interfaces to communicate with their respective cards even though they might implement completely different protocols for a variety of applications. These protocols can be public or proprietary. The main concern is a manufacturer decommissioning their product as part of the software development lifecycle, which might lead to electronic waste. This paper looks into designing a general setup and procedure to intercept and interpret the communication of any smart card implementing a PKCS#11 compatible library of a manufacturer. We introduce multiple tools for intercepting the traffic of these cards and present experiment-based strategies for reverse engineering their implementation details. Our paper could serve as a basis for drastically increasing the lifespan of smart cards. The presented strategies could then be used by the open source community to keep the cards updated, while also contributing to reducing the amount of unnecessary electronic waste.**

***Index Terms*—smart card, integrated circuit card, pkcs#11**

1. Introduction

Smart cards are widely used. From being used for banking with a credit card all the way to healthcare, smart cards lay the foundation of our digital society. The ISO/IEC 7816 [1], [2] standard defines characteristics, like the commands, to avoid the issue of having incompatible Integrated Circuit Cards (ICC) across different countries and to enable interoperability in interindustry. The implementation of the protocol used by the card is not necessarily open source and can be proprietary. Therefore, the manufacturers offer an interface to be able to communicate with the card.

We follow the assumption of manufacturers complying with the product development lifecycle, where decommissioning is part of the cycle [3]. The fundamental issue of proprietary cards is the users being fully reliant on the manufacturer for updates. Once the manufacturer decommissions their product, it can lead to cards having no support for newer operating system versions, rendering them unusable without the required updates, and therefore turning them into electronic waste.

This paper looks into an experiment-based method for reverse engineering these naturally discontinued and outdated proprietary cards to make them open source, granted that the manufacturer does not already release the protocol of their card to the open source community voluntarily. The idea behind that is being able to extend the lifetime of these ICCs by having the possibility of updating their middleware.

The remainder of this paper has the following structure: First, in Section 2 we provide background information by outlining the necessary key concepts. In Section 3, we proceed with evaluating and explaining the different challenges and choices made for our approach. Section 4 presents the setup for reverse engineering and goes into detail for a variety of experiments to extract information. Lastly, we compare our paper to a related paper by A. Nicolić et al. [4] in Section 5 and then briefly summarize the most important findings in Section 6.

2. Background

In this section, we give an overview of the key concepts needed to understand the basics of the communication of smart cards.

2.1. PKCS#11

The Public-Key Cryptography Standard #11 defines a platform-independent Application Programming Interface (API), which is called "Cryptoki". The API specifies a set of functions to perform cryptographic operations. It is object-based and offers all the functionality to create, use, modify and delete cryptographic objects such as RSA key pairs, certificates and domain parameters for DSA or Diffie-Hellman. Cryptoki is essential for manufacturers because they can provide the API as a dynamically linked library for the C programming language to users, e.g., in the form of a `.dll` file. This way, the manufacturer can abstract the implementation details while still allowing to bridge the communication between the ICC and the client with the provided library. By replacing the `.dll` file, the manufacturer can update the implementation of their ICC while leaving the ICC's functionality unchanged.

Alongside cryptographic objects, the standard provides multiple relevant definitions for this paper. Tokens are defined as devices with the ability of executing cryptographic functions as well as being able to store these cryptographic objects. A Slot is a reader that can hold a Token. For example, a smart card is considered a Token, while a smart card reader is viewed as a Slot. These two can have a connection with each other, which is defined as a Session [5].

 doi: 10.2313/NET-2025-05-1_12

2.2. APDU

An Application Protocol Data Unit (APDU) is a data packet used for the communication between ICCs and a card reader. An APDU is a byte array containing information defined in the ISO/IEC 7816-4 standard [1].

TABLE 1: Command APDU

Field	Description	Length in bytes
CLA	Class of command	1
INS	Instruction code	1
P1-P2	Parameters	2
L_c	Length of Command data	0, 1 or 3
Command data	Command data	Variable length
L_e	Max. length of Response data	0, 1, 2 or 3

TABLE 2: Response APDU

Field	Description	Length in bytes
Response data	Response data	Variable, at most L_e
SW1-SW2	Status bytes	2

There are command and response APDUs. Their format is shown in Tables 1 and 2, respectively. The first is responsible for sending the necessary data of the operation, which entails the instruction, parameters and more as described in Table 1. Some operations do not require any parameters or data. The latter returns the response to the host machine. This response contains two status-bytes SW1-SW2 indicating whether the command has been successful or not. Depending on the operation, this response can contain optional data. For reference, Figure 2 in Section 4.2.1 shows an example for command and response APDUs. Each command is strictly defined with its own corresponding sections for data in the byte array. For example, NIST SP 800-73 Pt.2-5 PIV [6] has its own set of public commands, defining the byte values for the APDUs for each command, which still follow the ISO/IEC 7816 standard. On the other hand, there can be manufacturers keeping these specifications private [1], [7].

3. Analysis

The problem statement is to reverse engineer as many details of the signing protocol of ICCs as possible. For example, this includes all of the supported mechanisms and parameters for each ICC as well as the corresponding mapping to the byte values in the command and response APDUs. The difficulty of reverse engineering lies in finding a starting point and commonalities within the protocols. Manufacturer-independent interfaces such as PKCS#11 and Minidriver [8] can solve these problems. They define the functionality as well as mechanisms used for cryptographic functions, as described in Section 2.1. Additionally, they offer a selection of variable parameters that can be modified and used to extract information out of the protocols, which is further explained in Section 3.3. The structure provided by the manufacturer independent interfaces is what makes the problem statement approachable. This lays the foundation for the different experiment-based approaches for reverse engineering the structure of commands presented in this paper.

3.1. PKCS#11 and Smart Card Minidriver

Smart card minidrivers [8] are interfaces, similar to PKCS#11, that can be written by smart card manufacturers. They are exclusive to Microsoft, which is a disadvantage in comparison to PKCS#11 because not all manufacturers or providers support it. Additionally, there is no option to debug the communication on Linux. In comparison to PKCS#11, minidriver offers a less clear command mapping. In conclusion, the minidriver is less flexible and more difficult to reverse engineer. For these reasons, we exclusively cover ICCs with an existing implementation of PKCS#11 in the context of this paper.

3.2. Virtual Card Reader and Logging

The process of reverse engineering in the context of this paper involves intercepting the data sent between the card reader and the ICC. We identified two viable possibilities to intercept and view this data, which are the virtual card reader and logging.

A virtual card reader implements a driver interface of a card reader with no underlying hardware. This software can only relay the data to a card reader, resulting in the possibility of intercepting the sent data. More specifically, this can then be used to view and interpret the data, which consists of command and response APDUs. An example for an implementation of a virtual card reader is *vpcd* [9]. This approach would be compatible with the existing PKCS#11-compatible software, therefore not requiring a test driver.

Logging, on the other hand, is intercepting the data between the card and the card reader using software such as *pcscd* on Linux or *APDUPlay* on Windows. Therefore, both Windows and Linux are viable options for this approach.

Both presented options for intercepting are suitable, however, in the context of this paper, we decided to use logging. On Linux, logging APDUs can be done with *pcscd*, which is a Personal Computer/Smart Card Daemon, using the `--apdu` and `--debug` options [10]. An alternative to pcscd is *APDUPlay* for Windows [11].

Since the PKCS#11 API is provided in C, the code for the experiments is also written in C. A different option is to use a wrapper such as *TUGraz IAIK* [12] in order to be able to use the manufacturers library in a different programming language such as Java or Python. However, using a wrapper is only preference and not necessary since the functionality stays the same.

3.3. Approach to Interpret Commands

In this section, we give an overview of selected PKCS#11 functionality, which serves as a basis on how to:

- Find functions that are able to be reverse engineered.

 doi: 10.2313/NET-2025-05-1_12

- Find modifiable parameters within these functions.
- Choose fitting and relevant values for these parameters in order to extract information.
- Interpret the data of these commands.
- Design and conduct further experiments for targeted information disclosure.

Firstly, we need to choose a command or a series of commands to send to a card. The APDUs have different fields where we can input different parameters.

Since the ICC follows the protocol of a vendor specific library implementing the PKCS#11 standard, there is a given list of all possible parameters for all functions defined in the standard. In order to reverse engineer the implementation of the ICC, all possible combinations of the functions and parameters have to be tried. The goal is to find out all of the supported functionalities as well as the mappings between them and their respective bytes.

`C_GenerateKeyPair` generates new private and public key objects. It is possible to choose the mechanism, such as RSA, and create a template for the respective keys, where the attributes can be further specified. When considering RSA for example, the private or public exponent as well as the key length (modulus bits) can be set to their desired values. Additionally, the attributes can specify what each key will be used for: The private key can be set to be used for signing, while the public key can be set to be used for verifying. This is intended for preventing improper uses. Furthermore, the amount of attributes for each template is a parameter of the function, which can be changed for different key pairs.

Similarly to `C_GenerateKeyPair`, `C_GenerateKey` generates a secret key for, e.g., AES or a set of domain parameters for, e.g., Diffie-Hellman. Analogously to the generation of key pairs, the mechanism, the attributes and the amount of the attributes can be specified. Some examples for attributes in this context are specifying the uses for keys to encryption or decryption as well as the key length and key type.

There are cryptographic functions defined by the PKCS#11 standard, such as `C_Encrypt`, `C_Decrypt`, `C_Digest`, `C_Sign` and `C_Verify`. Despite having different functionalities, these functions share many similarities by being split into multiple function calls and because of their parameters. Each function can be suffixed with `Init`, to initialize the respective operation, such as `C_EncryptInit`. This step is required and speficies the mechanism, such as `RSA` or `Elliptic Curve Cryptography (ECC)` for the `C_Sign` operation as well as the respective key for the selected operation. Afterwards, the operation itself is called where the data and the location of the output with their respective lengths can be defined [5].

PKCS#11 also defines multiple hashing and signing mechanisms, where the padding can be changed, such as `CKM_SHA256_RSA_PKCS`, performing SHA-256 hashing and RSA signing with PKCS#1 v1.5 padding or `CKM_SHA256_RSA_PKCS_PSS` with PSS padding [13].

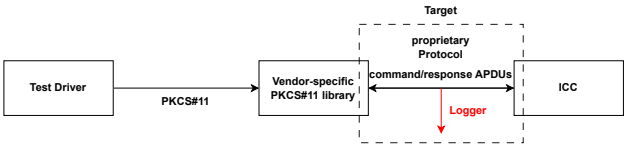
All of these various parameters for each command can then be used to interpret the bytes of the command APDUs as well as the response APDUs. Since we know exactly which parameters we put into the function we can specifically look for the changing bytes in the log. Considering there are many different options, the approach yielding the most consistent results is to only change one parameter value at a time. Upon only changing a single parameter value, most of the bytes within the APDU stay the same, while specific values will differ with every change of the respective parameter, narrowing down the location of the bytes of the input parameter over multiple function calls. This limitation allows for a more precise interpretation of these values and mapping them to their respective algorithm in both the command and response APDU.

4. Design

In this section, we present a specific soft- and hardware setup to be able to conduct experiments. These experiments are designed to extract information out of the proprietary implementation details of a card with a corresponding vendor library implementing the PKCS#11 standard.

4.1. Software and Hardware Setup

Figure 1: Experimental Setup



The experiments are conducted as shown in Figure 1. Firstly, a card with an instance of the PKCS#11 standard, such as PIV [6] or ISO/IEC 7816-15 [14] is needed. Additionally, a card reader is required to be able to communicate with the card. This reader is connected to a machine, where the raw communication between the card and reader is intercepted and logged. An application interacting with the PKCS#11 library on the computer is the client, which is acting as a test driver in this case. This test driver is responsible for the control and management of the experiment. The test driver is a program calling a series of selected commands for the current experiment in order to communicate with the card. These commands should be appropriately selected to disclose the details of the routines of the card as well as the concrete values of the APDU fields, as discussed in Section 3.3.

4.2. Experimental Approach

The following experiments are designed by the authors for possible approaches for the problem statement. All experiments are conducted using the soft- and hardware setup described in Section 4.1. Each experiment should be designed in a way such that a single operation with all of the associated functionality can be disclosed. For example, this would include not only encryption but also the initialization of the operation as well as the creation of the respective key with specified attributes. This potentially leads to a lot of data in the form of multiple APDUs, that has to be analyzed. Therefore, it can be effective to keep the same environment, such as keys or data for

multiple iterations as well as to minimize the changes of parameters. In the following, we present step by step approaches to extract information of the implementation of any card using the PKCS#11 protocol.

4.2.1. Password Encoding. The ISO/IEC 7816-15 [14] standard defines attributes for passwords, which are also used in ICCs following protocols of vendor specific libraries implementing the PKCS#11 standard. It defines five different possible encodings for passwords: `binary coded decimal`, `half-nibble binary coded decimal`, `ascii-numeric`, `utf8` and `iso9564-1`. Since passwords can have variable lengths, padding can be necessary. PKCS#11 allows password values to contain any valid UTF-8 character, which can be restricted to a subset depending on the Token. In this context, we assume that passwords can only hold numerical values with six characters.

The `VERIFY` command, which can be used to translate the `C_Login` command, is defined in ISO 7816-4 [1]. It is more like a recommendation for the structure and values of the APDUs. Therefore, manufacturers can make their implementations completely different. This fact makes reverse engineering more difficult but not impossible.

There are two relevant PKCS#11 functions for this experiment: `C_OpenSession` and `C_Login`. The first one is responsible for creating a Session between the Token and the Slot. The latter is for sending the password to log the user into the Token [5].

The login allows for a limited amount of attemps to enter the correct password before locking the user out of the Token. Once locked out, the supervisor can unlock the card again. Since we are working on a proprietary card, we do not require access to the supervisor actions. Therefore, we need to send different passwords while also avoiding locking the card in order to find the encoding. For this experiment, we choose an arbitrary password such as `123456` and call `C_Login`.

Figure 2: Possible APDUs for C_Login

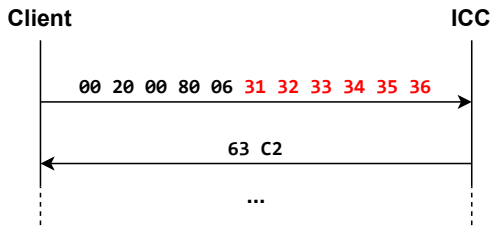


Figure 2 shows possible command and response APDUs for the password `123456`. In this case it is trivial to find the password within the bytes, which is the last 6 bytes of the command APDU marked in red. The password in this example is encoded as `ascii-numeric`. If it is not clear where the password is located on the first attempt, we can modify a single digit, e.g., `123455` and call `C_Login` once more. The resulting bytes can now be compared again. For more data, we create the initial situation, meaning the correct password has to be entered, to reset the password retry counter and `C_Logout` has to be called, to log the user out of the Token. Afterwards, the process can be repeated again while changing a single digit of the password. With enough collected data, the sent password's byte values can now be mapped to the password encoding used by this card.

4.2.2. C_Sign. The goal for this experiment is to iterate through all combinations of mechanisms that are compatible with a fixed key pair. In order to compute signatures, we have to generate a key pair and provide data that should be signed. For this experiment, the data can be fixed in the beginning, since there are no further implementation details that can be disclosed with more data sets. The next step is to choose a key type, such as RSA or ECC, for which we can generate a private key and public key pair using `C_GenerateKeyPair`. However, before calling the function, we define the template for the key pair. When considering RSA for example, this can include setting attributes such as the private and public exponent to valid arbitrary values, e.g., 3 as well as the key length to, e.g., 2048 or 4096. With the specified parameters, we can now create a Session using `C_OpenSession`, followed by logging into the Token with `C_Login`, if required, and call the `C_GenerateKeyPair` function.

The relevant parameters for the signature function are the mechanism, the signature key and the data. The generated key pair can now be used for multiple signatures. For each generated key pair from the previous step, we iterate through all possible mechanisms such as `CKM_SHA256_RSA_PKCS_PSS`, `CKM_SHA384_RSA_PKCS` and `CKM_ECDSA`, where the scheme, digest and padding are dynamic. Upon specifying all of the needed parameters, the signature can be done using `C_SignInit` and `C_Sign` and we can proceed with creating a new key pair and repeating the steps. For the purpose of reverse engineering the signature operation itself, it is sufficient to generate a fixed key pair for each iteration through the mechanisms. For more details about the structure and values of the attributes for keys, there should be a new key pair generated for every single possibility of valid options provided by the PKCS#11 standard.

Since cards implementing a PKCS#11 compatible library do not have to support all mechanisms defined by that standard, the return value can be checked to see if the selected mechanism is supported by the card [5], [13]. Granted it is supported, we gain some information about the implementation with this approach. We can inspect the APDUs to find out the mapping for each mechanism in the command and response APDUs. This process can be continued, by trying all possibilities for mechanisms and attributes during the key creation. However, the changes each iteration should be kept to a minimum to maximize the amount of information that can be confidently mapped.

5. Related Work

The research presented in the paper of A. Nikolić et al. [4], describes tools and strategies for reverse engineering smart card middleware of proprietary manufacturers. In contrast to this paper, their work focusses on the middleware to gain more information about the Windows smart card architecture as opposed to our research, where we present tools and techniques for analyzing smart cards

implementing PKCS#11 specifically. Nikolić et al. show multiple different approaches such as performing static analysis through disassembly, dynamic analysis and the analysis of the communication traffic, consisting of command and response APDUs, as described in this paper. It should be noted, however, that their research is limited to smart card middleware following the Windows Smart Card Minidriver Specification. Their paper successfully applies their described methods to the Serbian Electronic ID Card, where a platform-independent library has been developed to allow for use of this card on other operating systems.

6. Conclusion

This paper addresses the problem of manufacturers decommissioning their products without publishing their implementation for possible updates. In our research, we discuss multiple tools to intercept the flow of communication between smart cards and card readers. Additionally, we highlight the challenges of reverse engineering smart cards and their solutions through manufacturer independent libraries such as PKCS#11. Most importantly, our research introduces multiple step by step strategies to extract information about the implementation details of smart cards. This paper can serve as a basis for reducing the amount of electronic waste by enabling the possiblity of updates for the middleware.

Future research may include the reverse engineering of smart cards following different standards, using similar concepts as described in this paper. Furthermore, there is the possiblity to automate parts of the experiments by creating a program that can automatically generate all possible parameter combinations for the operations.

References

[1] ISO/IEC 7816-4:2020(E), "Identification cards — integrated circuit cards — part 4: Organization, security and commands for interchange," International Organization for Standardization, Geneva, CH, Standard, May 2020.

[2] ISO/IEC 7816-8:2021(E), "Identification cards — integrated circuit cards — part 8: Commands and mechanisms for security operations," International Organization for Standardization, Geneva, CH, Standard, August 2021.

[3] SAND2015-9022, "Model of the product development lifecycle," Sandia National Laboratories, Albuquerque, New Mexico, Report, September 2015.

[4] A. Nikolić, G. Sladić, and B. Milosavljević, "Reverse engineering smart card middleware," Novi Sad, Serbia, 2013.

[5] PKCS11-base-v2.40, "Pkcs #11 cryptographic token interface base specification version 2.40," Edited by S. Gleeson and C. Zimmerman, OASIS Standard, April 2015, http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html.

[6] H. Ferraiolo, K. Mehta, S. Francomacaro, R. Chandramouli, and S. Gupta, "Interfaces for personal identity verification: Part 2 – piv card application card command interface," National Institute of Standards and Technology, Gaithersburg, MD, NIST Special Publication (SP) NIST SP 800-73pt2-5, 2024, https://doi.org/10.6028/NIST.SP.800-73pt2-5.

[7] Yubico, "Apdus," https://docs.yubico.com/yesdk/users-manual/yubikey-reference/apdu.html, n.y., [Online; accessed 21-December-2024].

[8] "Smart card minidriver specification, v7.07," https://learn.microsoft.com/en-us/previous-versions/windows/hardware/design/dn631754(v=vs.85), Microsoft Corporation, 2017, [Online; accessed 16-December-2024].

[9] F. Morgner, "Virtual smart card," https://frankmorgner.github.io/vsmartcard/virtualsmartcard/README.html, 2024, [Online; accessed 17-December-2024].

[10] D. Corcoran and L. Rousseau, "pcscd - pc/sc smart card daemon," https://linux.die.net/man/8/pcscd, n.y., [Online; accessed 12-December-2024].

[11] P. Svenda, "Pc/sc apdu inspection and manipulation tool (apduplay)," https://github.com/crocs-muni/APDUPlay, 2019, [Online; accessed 12-December-2024].

[12] "Iaik pkcs#11 wrapper 1.6.9 api," https://javadoc.sic.tech/pkcs11_wrapper/current/index.html, IAIK at Graz University of Technology, 2023, [Online; accessed 21-December-2024].

[13] PKCS11-curr-v2.40, "Pkcs #11 cryptographic token interface current mechanisms specification version 2.40," Edited by S. Gleeson and C. Zimmerman, OASIS Committee Specification 01, September 2014, https://docs.oasis-open.org/pkcs11/pkcs11-curr/v2.40/cs01/pkcs11-curr-v2.40-cs01.html. Latest version: http://docs.oasis-open.org/pkcs11/pkcs11-curr/v2.40/pkcs11-curr-v2.40.html.

[14] ISO/IEC 7816-15:2016(E), "Identification cards — integrated circuit cards — part 15: Cryptographic information application," International Organization for Standardization, Geneva, CH, Standard, May 2016.

Optimization of QUIC Congestion Control with ECN

Rico Finkbeiner, Marcel Kempf*, Benedikt Jaeger*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: rico.finkbeiner@tum.de, kempfm@net.in.tum.de, jaeger@net.in.tum.de

***Abstract*—QUIC plays an important role in today's Internet by providing several benefits over TCP. In this paper, we explain how ECN can be used to optimize QUIC's congestion control. ECN tries to avoid retransmissions by explicitly notifying the sender about congestion in the network instead of silently dropping packets. When using QUIC, this can be done by including ECN counts in the ACK frames to mirror information about incipient congestions back to the sender, who can then reduce its sending rate. We also take a look at the support of ECN with QUIC in the Internet based on related work. ECN with QUIC is currently barely used, mainly because of missing support in common QUIC implementations and failures in the ECN validation stage.**

***Index Terms*—QUIC, ECN, congestion control**

1. Introduction

QUIC, a protocol introduced by Google [1], provides many benefits over the Transmission Control Protocol (TCP). By providing a zero round-trip time (0-RTT) handshake and avoiding head-of-line blocking delays, QUIC significantly reduces latency [1], [2]. Among other reasons, this has led to a wide adoption of QUIC in the Internet. Today, 8.4% of all websites use QUIC [3].

QUIC can be used with different congestion control algorithms [1], [2]. Traditional, loss-based congestion control algorithms like CUBIC [4] treat lost packets as a sign of congestion and consequently reduce the sending rate while retransmitting lost packets. These retransmissions increase the latency and reduce the available bandwidth.

By using Explicit Congestion Notification (ECN), the sender can be notified of congestion before packets have to be dropped. Routers can set a codepoint in the IP header to signalize incipient congestion. When a packet with this codepoint set arrives at the receiver, the receiver has to mirror this information back to the sender. The sender then reduces its sending rate. To achieve this, support of higher layer protocols, such as TCP or QUIC, is necessary. [5]

In the following, we show how ECN can be used with QUIC to optimize QUIC's congestion control. Section 2 provides background information about QUIC and how ECN can be used with TCP. Section 4 focuses on how ECN works with QUIC. In Section 5, we take a look at the support of ECN with QUIC in the Internet. Next, Section 6 discusses an idea of how congestion control with ECN could be further improved. Section 7 concludes this paper by providing a short summary.

2. Background

To understand how ECN with QUIC works, we first take a look at the QUIC protocol itself and how ECN can be used with TCP.

2.1. QUIC

Langley et al. [1] demonstrated their experiences at Google with QUIC in 2017. According to them, using the transport layer protocol TCP comes with various drawbacks. First, using Transport Layer Security (TLS) on top of TCP increases the delay by requiring both a TCP and TLS handshake. Second, multiplexing TCP streams can lead to head-of-line blocking delays.

Making changes to TCP to cope with these challenges is difficult [1]. Since TCP is implemented in the kernel, deploying changes takes time. Creating a completely new transportation protocol on layer 4 is challenging due to middleboxes like Network Address Translations (NATs) or Firewalls, as they would explicitly need to be adapted to support the new protocol. QUIC circumvents this by building on top of the User Datagram Protocol (UDP).

QUIC solves the previously mentioned problems of TCP [1]. To establish a new, secure connection, QUIC only needs a one round-trip time (1-RTT) handshake by combining the TLS and transport layer handshake. Under certain conditions, subsequent connections to the same server can be established using a 0-RTT handshake.

A single QUIC packet can consist of multiple frames, each containing data of a specific stream. Losing a UDP datagram only affects the streams contained in that datagram. Therefore, head-of-line blocking can be avoided. [1]

QUIC does not require a specific congestion control mechanism, making it possible to use different algorithms [1].

2.2. ECN

When using loss-based congestion control, nodes drop packets in case of a full buffer to signalize congestion [5, Section 1], [6]. The sender is able to detect congestion due to duplicate acknowledgments or timeouts and consequently reduces its congestion window [6].

Loss-based congestion control algorithms tend to keep buffers full, leading to a queuing delay [5], [7]. Both full buffers and retransmissions increase the latency. Active Queue Management (AQM) algorithms [8] like Random Early Detection (RED) [9] are trying to avoid the build-up of large queues at routers. While this reduces the queuing

 doi: 10.2313/NET-2025-05-1_13

delay, ECN mitigates the issue of retransmissions due to dropped packets [5].

2.2.1. ECN on the IP layer. ECN makes it possible to inform the sender about congestion without dropping packets. This can be done by setting the Congestion Experienced (CE) codepoint in the IP header [5].
RFC 3168 [5] refers to a packet with the CE codepoint set as a "CE packet". In the following, we use the same definition.

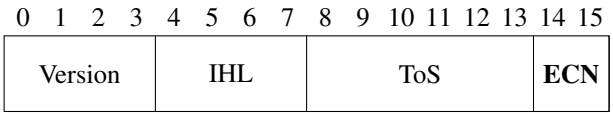


Figure 1: First 2 Byte of the IPv4 header. Based on RFC 791 [10, Figure 4] and RFC 3168 [5, Figure 2]

RFC 3168 defines four ECN codepoints by using bits 6 and 7 of the type of service (ToS) and traffic class field of the IPv4 and IPv6 header, respectively [5, Section 5]. Figure 1 shows the first two bytes of the IPv4 header, including the ECN field. The codepoint "00" is set if ECN is not being used. "11" is the CE codepoint. Either ECT(0) ("01") or ECT(1) ("10") are set by the transport protocol endpoints if they support ECN. Originally, both codepoints were handled equally by routers and served as a one-bit nonce. [5]

More recently, however, ECT(1) serves as a Low Latency Low Loss and Scalable Throughput (L4S) identifier [11]. Routers supporting L4S set the CE codepoint earlier, enabling faster reactions by the endpoints.

2.2.2. Compatibility of ECN. Since the adoption of ECN by routers and hosts happens gradually and is therefore not supported or used by every router and host, it is important that ECN works alongside existing congestion control algorithms. Thus, and to ensure fairness, hosts have to react to an ECN in the exact same way as to a dropped packet. In addition, routers are only allowed to set the CE codepoint if the packet would have been dropped to signal congestion when not using ECN. Routers should deal with a CE packet just as with any other packet. If a queue is entirely full, routers still have to drop incoming packets, even when ECN is used. [5]

2.2.3. ECN with TCP. If an endpoint receives a CE packet, the endpoint has to mirror this information back to the sender so the sending rate can be reduced. To achieve this, ECN requires support from the Transport Layer. [5]

First, when setting up a connection, both endpoints have to be able to signal their capability and willingness to use ECN (Section 2.2.4). Then, after agreeing to use ECN, both endpoints have to react to CE packets by reducing their sending rate (Section 2.2.5). [5]

TCP supports ECN by introducing two flags. Bits 8 and 9 of the reserved field of the TCP header are used for a congestion window reduced (CWR) flag and ECN-Echo (ECE) flag, respectively. Figure 2 shows the CWR and ECE flags. [5, Section 6]

2.2.4. Negotiating the use of ECN. If a host is willing and capable of using ECN, it has to send a packet with

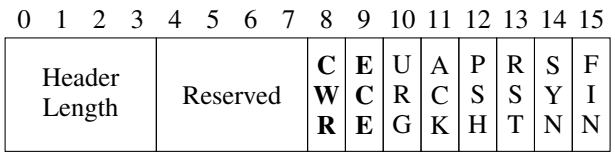


Figure 2: TCP header supporting ECN.
Based on RFC 3168 [5, Figure 4]

the ECE, CWR, and the SYN flag set in the TCP header. The receiver of this packet can then signal their support of ECN by setting the ACK, SYN, and ECE flags but not the CWR flag in the response. This response is then acknowledged by a packet with the ACK flag set. After completion, both endpoints have to react appropriately to CE packets and to TCP segments with the ECE flag set. However, they do not have to set the ECN-capable Transport (ECT) codepoint in the IP header themself. [5]

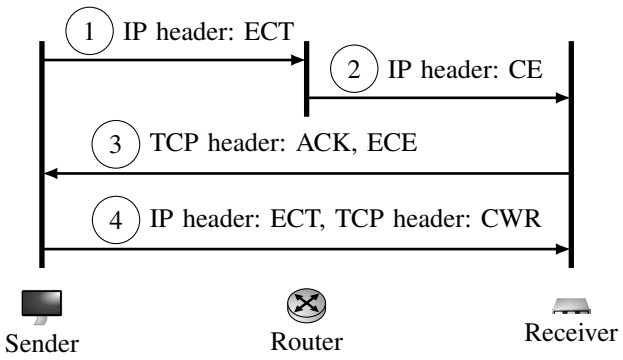


Figure 3: ECN with TCP

2.2.5. Using ECN. Figure 3 illustrates the use of ECN with TCP. When sending a packet, the sender sets the ECT(0) or ECT(1) codepoint in the IP header (1). Then, a router on the network path experiences congestion. Since the ECT codepoint in the IP header of our packet is set, the router can mark the packet with the CE codepoint (2). When receiving a CE packet, the receiver sets ECE in the TCP header (3) to mirror the information back to the sender. Upon receiving a TCP segment with the ECE flag set, the sender reduces its congestion window and informs the receiver about that by setting the CWR flag (4). As soon as the receiver processes this flag, it stops setting ECE in the TCP header. [5]

3. Related work

Most of the related, previous work has focused on ECN with TCP. However, more recent work has also studied improvements and the support of using ECN with QUIC.

ECN with TCP. Floyd [12] has conducted simulations to point out several advantages of using ECN with TCP. For instance, in one of their LAN simulation scenarios, they were using ten telnet connections, a 0.1 msec TCP clock, packet-based RED gateways (with and without ECN), and a 64 kB maximum TCP window. In this experiment, they were able to decrease the average delay from about 20ms to nearly zero by using ECN.

 doi: 10.2313/NET-2025-05-1_13

Different simulation scenarios show similar results: by using ECN, fewer packets were dropped, which decreased delays. They also pointed out potential disadvantages, mainly focusing on misbehaving endpoints and losing acknowledgment (ACK) packets.

A variety of other studies have focused on the support of ECN on the Internet. Lim et al. [13] have shown in 2022 that over 85% of Alexa Top 100K web servers support ECN with TCP. Bauer et al. [14] have observed a similar number while conducting Internet measurements regarding the effect of TCP options. This is a massive increase from what previous work has shown earlier. For instance, Bauer et al. [15] conducted experiments using the Alexa Top 1M list in 2011. Only about 17% of web servers supported ECN at that time.

ECN with QUIC. While ECN with TCP is widely deployed in today's Internet, Sander et al. [16] were able to show that the opposite holds for QUIC. By conducting extensive research and experiments on the support of ECN with QUIC in the Internet, they were able to demonstrate that by the time of their studies, ECN could be used with less than 2% of QUIC hosts. We take a closer look at their results in Section 5. Uchida et al. [17] proposed a method to leverage ECN with QUIC to improve the fairness of two competing hosts using QUIC with CUBIC and BBR [7], respectively. By adapting how CUBIC reacts to ECN codepoints and by adapting BBR's phase transitions, they were able to improve fairness in their experiments. For instance, when using a bottleneck link buffer of 1 Mbit, they improved Jain's fairness index value from less than 0.7 to nearly 1.

In this paper, we mainly focus on how ECN with QUIC works (Section 4) and what current impediments toward a wide deployment of ECN with QUIC are (Section 5).

4. ECN with QUIC

This section is based on RFC 9000 [2], which contains specifications on how ECN can be used with QUIC. Similar to using ECN with TCP, the sender decreases its sending rate when receiving a CE packet. In contrast to using ECN with TCP, ECN with QUIC can also be used in only one direction. In addition, the receiver informs the sender not only about CE packets but also about the ECT(0) and ECT(1) codepoints it receives. In order to use ECN with QUIC, a sender first has to confirm that both the receiver and intermediate nodes support ECN.

4.1. Mirroring ECN Counts

To be able to use ECN with QUIC, the receiver needs to be able to access bits 6 and 7 of the ToS and traffic class field of the IPv4 and IPv6 header, respectively. The receiver then maintains counts for the ECT(0), ECT(1), and CE codepoints it has observed. These counts can be mirrored back to the sender using specific fields in the ACK frames.

Similar to TCP, ACK frames in QUIC are used to confirm successfully transmitted packets [2, Section 19.3]. To support ECN, QUIC introduced the ACK frame type 0x03, as shown in Figure 4. ACK frames of this type additionally include the ECN counts of the receiver for the packet number space it acknowledges. By using this ACK frame type, the receiver is able to inform the sender about the total number of ECT(0), ECT(1), and CE codepoints it received.

```
ACK Frame {
    Type = 0x03,
    Largest Acknowledged,
    ACK Delay,
    ACK Range Count,
    First ACK Range,
    ACK Range ...,
    ECN Counts {
        ECT0 Count,
        ECT1 Count,
        ECN-CE Count,
    }
}
```

Figure 4: ACK frame format of type 0x03 in QUIC. Based on RFC 9000 [2, Figure 25 and Figure 27]

4.2. Example: ECN with QUIC

Figure 5 illustrates how QUIC with ECN works. After establishing a connection and agreeing on the use of ECN, the sender sets the ECT(0)/ ECT(1) codepoint in the IP header (1) to signal the use of ECN to routers. In (2), the router experiences a congestion. Instead of dropping the packet, it sets the CE codepoint in the IP header (3). The receiver maintains the ECN counts n, m, and k for the number of ECT(0), ECT(1), and CE codepoints received (4). The receiver includes these counts in the ACK frame (5, 6) when acknowledging the received packet. Due to the increase in the CE count, the sender then reduces its sending rate.

4.3. ECN validation

To determine whether the network path supports ECN, the sender sets ECT(0) (or ECT(1)) in the first few packets. If none of these packets are acknowledged, the sender assumes the packets have been dropped and that the path does not support ECN. The sender then disables ECN.

If the sender receives an ACK frame containing the ECN counts, he has to validate them [2, Section 13.4]. It is important that all ACK frames being used for ECN validation increase the largest acknowledged packet number.

There are several scenarios that lead to a failed ECN validation. In the following, we define *A_ECT_0* as the ECT(0) count in the current ACK frame. *S_ECT_0* stands for the total number of ECT(0) codepoints set by the sender. Similarly, Δ *A_ECT_0* defines the difference between the ECT(0) count in the current ACK frame and the ECT(0) count in the previous ACK frame. Δ *S_ECT_0* represents the number of packets that are newly acknowledged and were sent with the ECT(0) codepoint set. In the same way, *A_ECT_1*, Δ *A_ECT_1*, Δ *A_CE* and *S_ECT_1* are defined.

As specified in [2, Section 13.4], the following conditions are verified:

 doi: 10.2313/NET-2025-05-1_13

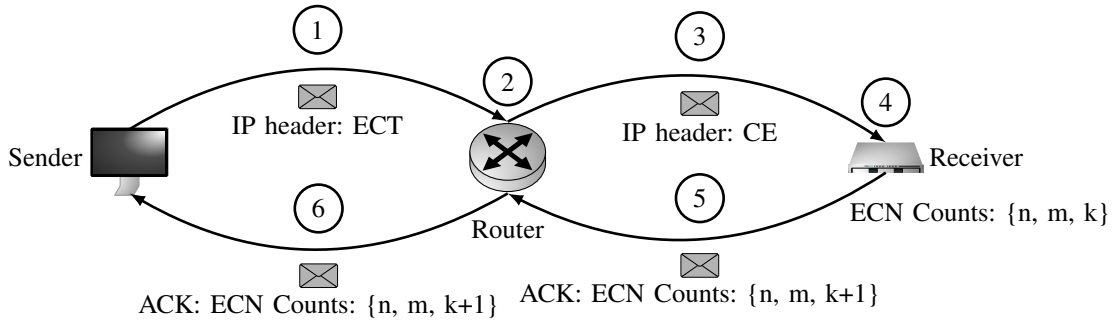


Figure 5: ECN with QUIC

1) The ACK frame contains ECN counts
2) ΔA_ECT_0 + ΔA_ECT_CE $\geq$ ΔS_ECT_0
3) ΔA_ECT_1 + ΔA_ECT_CE $\geq$ ΔS_ECT_1
4) A_ECT_0 $\leq$ S_ECT_0
5) A_ECT_1 $\leq$ S_ECT_1

If any of these conditions do not hold, ECN has to be deactivated. 1 validates that the receiver mirrors the ECN codepoints and that the ECN field of the IP header does not get cleared from a node on the network path. 2 and 3 are used to detect the remarking of ECN codepoints. For instance, a node on the network path could change a CE codepoint to ECT(0), although the packet was initially sent with an ECT(1) codepoint. Since ACK frames can get lost, it is possible that e.g. ΔA_ECT_0 >= ΔS_ECT_0 holds, which explains the inequality used in 2 and 3. However, the total number of A_ECT_0 / A_ECT_1 can never exceed S_ECT_0 / S_ECT_1. Thus, 4 and 5 are also used to detect the remarking of ECN codepoints.

5. Support of ECN with QUIC

One of the biggest challenges of using ECN over QUIC is the lack of support. Sander et al. [16] have been able to show that by the time of their study in 2023, QUIC with ECN could only be used with less than 2% of the hosts they investigated. In order to use ECN with QUIC, the codepoints have to be mirrored (see Section 4.1), and the validation of the ECN codepoints has to succeed (see Section 4.3). By conducting several experiments, they were able to pinpoint the causes of the low support of ECN with QUIC. This section discusses the main findings of Sander et al. [16].

5.1. Missing ECN counts

In case it is possible to access the ECN codepoints of the IP header, the QUIC standard [2, Section 13.4] defines the mirroring of ECN codepoints as a MUST. However, a previous statement in the standard makes QUIC support seem to be optional, causing ambiguities [18]. Actually, only 20% of QUIC hosts that were tested by Sander et al. include the ECN counts in the ACK frames. The interop runner [19] shows that only 6 out of 17 tested QUIC implementations support ECN.

5.2. Undercounting of ECN codepoints

Mirroring the ECN codepoints is not sufficient for using ECN with QUIC. Instead, the sender also validates the reported ECN counts (see Section 4.3). Sander et al. [16] were able to show that over 90% of the domains that support ECN mirroring fail this validation stage. Over one-half of them acknowledged fewer ECN codepoints than they had been sent with. They were able to show that this is mainly because of issues in the QUIC implementation used by the receiver and not due to nodes in the network.

5.3. Remarking of ECN codepoints

About one-third of the domains investigated report ECT(0) codepoints as ECT(1) codepoints. This is, however, not caused by the used QUIC stack but mainly by the network operator Arelion (ASN 1299). When repeating the measurements from various geographically distributed origins, the overall observed pattern stays the same. In fact, globally, only about 0.3% of the tested domains meet all the requirements during the validation phase. Even if the validation is successful, it does not mean that the endpoint or routers on the network path actually use ECN. [16]

6. Possible Improvement

The current QUIC standard [2] specifies the use of ACK frames with the type 0x03 when using ECN. As explained in Section 4.1, these ACK frames contain the total count of ECN codepoints the receiver has observed. This information, for instance, is used by the Prague Congestion Control Algorithm [20]. However, it still lacks the information on which packet was marked with which codepoint. This fine-grained information could be valuable to react even more efficiently and effectively to CE packets, according to Seemann et al. [21]

Therefore, they proposed a new QUIC ACK frame type, which includes the ECN codepoint that was set in the packets of each ACK range. If necessary, ranges have to be split into multiple ranges such that all packets within a range share the same ECN codepoint. [21]

It remains to be seen if this idea will be included in future versions of QUIC and if congestion algorithms can actually profit from this fine-grained information.

7. Conclusion

In this paper, we have explained how ECN works and how it can be used with QUIC. ECN is used to notify a

sender about congestion in the network. To achieve this, a router can mark a packet with the congestion experienced codepoint. QUIC then mirrors the codepoint back to the sender using ACK frames, and the sender can reduce its sending rate.

We have also discussed that the biggest impediment of using ECN with QUIC is the missing support in common QUIC implementations and misbehaving network operators.

While ECN can optimize QUIC's congestion control by trying to avoid retransmissions, it is currently barely used. Having more QUIC implementations supporting ECN would be essential for a wide adoption and usage of ECN with QUIC. In order to demonstrate the impact of using ECN with QUIC on throughput, latency, and packet drops quantitatively, performance measurements and comparisons could be conducted as part of future work. Depending on the results, this could speed up the support and use of ECN with QUIC.

References

[1] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 183–196. [Online]. Available: https://doi.org/10.1145/3098822.3098842

[2] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[3] W3Techs, "Usage statistics of QUIC for websites," https://w3techs.com/technologies/details/ce-quic, [Online; accessed 29-November-2024].

[4] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks," RFC 8312, Feb. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8312

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

[6] E. Blanton, D. V. Paxson, and M. Allman, "TCP Congestion Control," RFC 5681, Sep. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5681

[7] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, p. 20–53, Oct. 2016. [Online]. Available: https://doi.org/10.1145/3012426.3022184

[8] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," RFC 7567, Jul. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7567

[9] L. Zhang, D. C. Partridge, S. Shenker, J. T. Wroclawski, D. K. K. Ramakrishnan, L. Peterson, D. D. D. Clark, G. Minshall, J. Crowcroft, R. T. Braden, D. S. E. Deering, S. Floyd, D. B. S. Davie, V. Jacobson, and D. D. Estrin, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, Apr. 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2309

[10] J. Postel, "Internet Protocol," RFC 791, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc791

[11] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture," RFC 9330, Jan. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9330

[12] S. Floyd, "TCP and explicit congestion notification," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, p. 8–23, Oct. 1994. [Online]. Available: https://doi.org/10.1145/205511.205512

[13] H. Lim, S. Kim, J. Sippe, J. Kim, G. White, C.-H. Lee, E. Wustrow, K. Lee, D. Grunwald, and S. Ha, "A Fresh Look at ECN Traversal in the Wild," *arXiv preprint arXiv:2208.14523*, 2022.

[14] S. Bauer, P. Sattler, J. Zirngibl, C. Schwarzenberg, and G. Carle, "Evaluating the Benefits: Quantifying the Effects of TCP Options, QUIC, and CDNs on Throughput," in *Proceedings of the 2023 Applied Networking Research Workshop*, ser. ANRW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 27–33. [Online]. Available: https://doi.org/10.1145/3606464.3606474

[15] S. Bauer, R. Beverly, and A. Berger, "Measuring the state of ECN readiness in servers, clients, and routers," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011, pp. 171–180.

[16] C. Sander, I. Kunze, L. Blöcher, M. Kosek, and K. Wehrle, "ECN with QUIC: Challenges in the Wild," in *Proceedings of the 2023 ACM on Internet Measurement Conference*, 2023, pp. 540–553.

[17] N. Uchida, D. Nobayashi, D. Cavendish, and T. Ikenaga, "Poster: Fairness improvement method using ECNs with different congestion control algorithms within QUIC," in *2023 IEEE 31st International Conference on Network Protocols (ICNP)*. IEEE, 2023, pp. 1–2.

[18] "Technical Errata Reported RFC9000 ECN," https://mailarchive.ietf.org/arch/msg/quic/lsz4X-cZql71Ba56uQhNQz4NzGc/, 2023, [Online; accessed 28-November-2024].

[19] M. Seemann, "QUIC Interop runner," https://interop.seemann.io/, [Online; accessed 01-March-2025].

[20] K. D. Schepper, O. Tilmans, B. Briscoe, and V. Goel, "Prague Congestion Control," Internet Engineering Task Force, Internet-Draft draft-briscoe-iccrg-prague-congestion-control-04, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-briscoe-iccrg-prague-congestion-control/04/

[21] M. Seemann and V. Goel, "QUIC Accurate ECN Acknowledgements," Internet Engineering Task Force, Internet-Draft draft-seemann-quic-accurate-ack-ecn-01, May 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-seemann-quic-accurate-ack-ecn/01/

Analysis of Domain Name Sales and Auctions

Anton Ge, Christian Dietze*, Patrick Sattler*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: anton.ge@tum.de, diec@net.in.tum.de, sattler@net.in.tum.de

***Abstract*—The second-hand market for domain names is a special market as domains are intangible and unique goods with a wide price range. Most research papers focus on the appraisal of domain names, however, how and when domains are sold or auctioned has not been analyzed before to our knowledge. This paper presents a more holistic examination of the domain name market by analyzing auction and sales data from multiple marketplaces. We provide insights into properties and trends of the domain market, as well as characteristics of the domains that change hands.**

***Index Terms*—domain name, domain registrar, second-hand domain market, domain auction**

1. Introduction

Domain marketplaces allow the trading of domains that have expired or which current owners do not wish to use. On these marketplaces, prices of domains can vary greatly. To give a recent example, in September 2024, the domain `rocket.com` was traded for 14 million USD [1] while the majority of domains sell for far less. As such, domains present a special type of intangible good: According to Wu et al. [2], traditional accounting methods cannot be used to calculate the marketvalue of a domain due to its non-financial properties. In addition, each marketplace can employ their own policies on selling and buying, and offer a variety of other services. For sellers, it is crucial to know whether investing in domains is lucrative and which marketplace is most suited. For buyers on the other hand, knowing what to expect from the second-hand market can help decide if they should pursue their desired domain or register an alternative domain.
This paper analyzes five second-hand marketplaces for domain names. Using data collected from auctions and sales, we investigate how and when domains change ownership by analyzing bidding behaviors of potential buyers, characteristics of sold domains, and properties of the domain marketplaces. The rest of the paper is structured as follows: Chapter 2 presents background information on domain registration and a general overview of the domain providers. Chapter 3 summarizes related work. Subsequently, we present the data collection and processing, and display the results of the analysis in Chapter 4.

2. Background

In this chapter, we give a brief overview of domain names and their life cycle in order to understand the domain name market. Afterward, the domain providers and their most important policies are introduced.

2.1. Domain Names and their Life Cycle

Domain names are organized in a hierachical structure that is read from right to left: A domain starts at the DNS root (.), followed by the top-level domain (TLD) and second-level domain (SLD) separated by a period [3], [4]. Every combination of TLD and SLD is allowed exactly once in a domain name registry, and the length of each segment is limited to 63 characters [4].
Domain names are sold through registrars in cooperation with registry operators, organizations that manage individual TLDs and register domain names in a DNS database [3]. The registration duration usually lies between one and ten years at a time. To retain possession of a domain after that period, a renewal fee needs to be paid [5]. If a domain owner decides not to renew their registration, an expiration process will be triggered which can vary depending on the TLD. Website visitors might see a parked page showcasing information on where the domain can be acquired and advertisements to generate revenue. For generic TLDs, the expiration process goes roughly as follows: Up to 45 days after expiring, the domain owner can pay the standard renewal fee to keep the domain. If that does not happen, the domain enters the redemption grace period which lasts 30 days, and a renewal will incur an additional fee [5]. If the owner lets the deadline pass, the status of the domain will change to `pending delete`. Several days later, the domain will be deleted from the registry's database and becomes available for re-registration [5]. Some registries of country-code TLDs will immediately start the redemption grace period upon expiry [6].

2.2. Provider Overview

The auction and sales data was collected from the providers Sedo, Sav, Namecheap, GoDaddy and SnapNames. Below are their most important characteristics.

Market Listing Types. All providers offer auction and 'Buy now' types of sales on their platforms. Sedo, GoDaddy and SnapNames additionally have a 'Make offer' type that allows for negotiations between sellers and buyers.

Auctions by Private Sellers. Selling domains through auctions as an individual is only possible on Sedo and

 doi: 10.2313/NET-2025-05-1_14

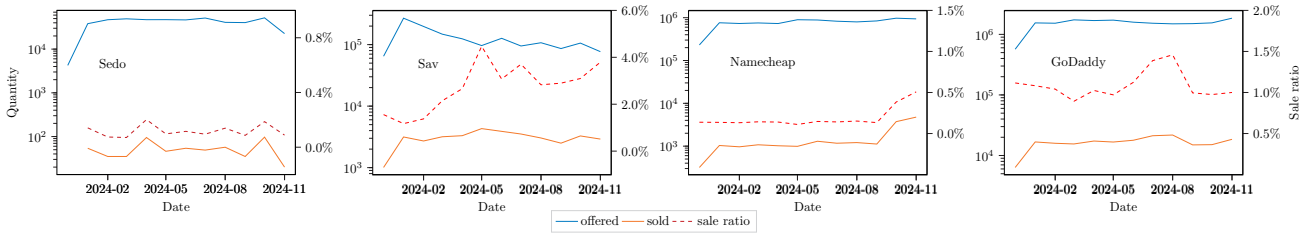


Figure 1: Domains offered and sold per month in marketplaces (on a logarithmic scale) and the sale ratio in percent.

Sav. On Namecheap, GoDaddy and SnapNames, private sellers are limited to sale-type listings.

Fees. Sav charges the lowest fee at 4% of the final sales price while GoDaddy has the highest rate at up to 25% and a 15 USD minimum for each sale. SnapNames and Namecheap have a fee of 20% and 10%, respectively. On Sedo, it costs 59 EUR to put a domain up for auction. Alternatively, it can start as a 'Make offer' that converts into an auction listing free of charge after the first offer was received. In both cases the fee is 15% of the final price.

Cross-listing. Sedo explicitly forbids cross-listing domains for sale in other marketplaces. Namecheap, GoDaddy and SnapNames cross-list auctions and sales from their partner platforms to promote a successful and faster sale.

Auction Format. All five providers use a soft-close auction format whereby an auction is extended by five to ten minutes everytime a bid is submitted five to ten minutes before it ends.

3. Related Work

There are several publications on domain appraisal. In one of the earlier works on this matter, Wu et al. [2] realized that, while domain names should be treated as intangible assets likewise to intellectual property, other methods of appraisal are needed. They identified multiple factors affecting the value of a domain, among the more significant ones, the domain structure, i.e., letters and numbers in a domain, its length and the TLD. However, their model includes some subjective criteria, such as impression and creativity.
Subsequent works by Dieterle and Bergmann [7], Bikadi et al. [8] Moro-Visconti [9] and Tang et al. [10] use very similar domain structure criteria. In addition, the number and types of keywords, and their respective search result volumes are contributors to a domain's value [8], [10]. Other metrics include revenue from advertisements [9] and various page ranks [8]–[10], e.g. Alexa Rank and Google Page Rank.
Thies [11] provides an overview of the domain market which describes the price index of domains in general and a trend in domain registrations from 2006 to 2013. According to Thies, domain registrations and the price index show an overall upward trend that rises and falls synchronously with the development of the IT economy [11]. Furthermore, their study provides a relative ranking of top-level domain prices which found the `com` TLD to be the most valuable one when SLDs are the same [11].

4. Evaluation

Sav, Namecheap, GoDaddy and SnapNames offer snapshots of their respective marketplace in the form of `CSV` and `JSON` files. Sedo is more protective of their data and restricts access to the files from the 'expired domains' category. Data collection on SnapNames has begun late in 2024 and is, thus, limited to the months of October and November. Several marketplaces include data for domains about to expire or domains about to go on auction. For the analysis, we consider domains that were in an ongoing auction or sale when the snapshot was taken.
The files were downloaded once per day because of GoDaddy's update rate. As a result, certain inaccuracies arise from the granularity of our data: A file includes auctions and sales that have not ended at the time the file was created. If bids were submitted between file creation and the end of an auction, the final bid count and auction price would not be captured by the market snapshot the next day. For this reason, the final bid count and auction price are referred to as 'minimal final bid count' and 'minimal final price'. Similarly, one can only assume that an auction without bids will lead to a 'likely unsold domain'.
The analysis starts at a macro level at which the overall numbers and characteristics of marketplaces are examined. Afterwards, we zoom in and take a closer look at the details of auctions and the properties of sold domains.

4.1. Providers

The data was collected from auctions and sales of 32,422,508 unique domain names across five providers from the end of December 2023 until November 2024. To put that number into perspective: according to Verisign's quarterly domain report, 362 million domains are registered across all TLDs [12]. Figure 1 shows the trends of domains offered and sold on each platform and the monthly sale ratios. Every auction with a final bid count of at least one counts as sold as bids are binding. Among the five providers, GoDaddy appears to be the biggest marketplace with an average of 1.6 million domains offered per month. It is followed by Namecheap and Sav with around 830K and 130K per month, respectively. Sedo has a monthly average of 44K in expired domains. SnapNames has been excluded from Figure 1 as there is only data from two months. In the time period considered, Sedo, Namecheap and GoDaddy appear to have equilibriums between the number of offers and sales: The monthly sale ratios vary by less than 0.5% for these three. Sav, on the other hand, looks more volatile as the ratio changes by up to 1.8% between months. Table 1 displays the average sale ratios.

 doi: 10.2313/NET-2025-05-1_14

TABLE 1: Average monthly sale ratios in marketplaces.

Sedo	Namecheap	Sav	GoDaddy
0.1%	0.2%	2.7%	1.1%

Cross-listing had a small effect on the final numbers as 26425 cross-listed domains were found in the dataset. Most cross-listings occurred between GoDaddy and Sav and GoDaddy and Namecheap. Sedo's cross-listing ban appears to be well enforced since no cross-listing in any other marketplace was found. Furthermore, each domain that was cross-listed appeared at most in two marketplaces at the same time. Generally, cross-listing is a way to promote auction and sale listings and attract more attention from potential buyers. For private sellers on Namecheap, GoDaddy and SnapNames, cross-listing their domain in other marketplaces is the only possibility to auction off their domain. Listing domains in an auction and another format is not problematic as marketplaces allow sale-type listings to be cancelled [13], [14].

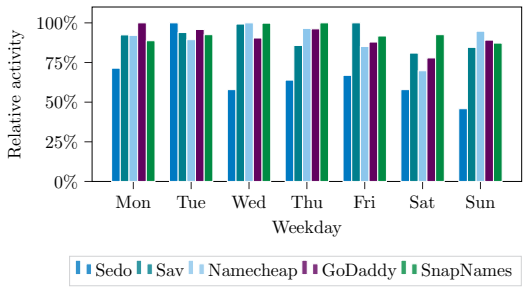


Figure 2: Relative activity per weekday in marketplaces: No significant peaks or troughs at any provider.

Next, the activity in each marketplace is examined. Whenever a bid count for a domain listing increases from one day to the next, the count for that weekday is incremented. Figure 2 contains the combined activities of the five providers. Each bar represents the number of bids relative to the respective highest bid count of the week per seller. The more prominent fluctuations of Sedo are explained by the small available dataset. The absolute difference between minimum and maximum is 72 bids. Apart from a small dip around Saturdays for all remaining providers, the distributions don't have any significant peaks or troughs in activity which suggests that people bid on domains on all weekdays, including weekends. Intuitively, this makes sense since an auction listing can start and end on any day of the week. To better understand the bidding behavior, Section 4.2 will delve into when bids come in for individual auctions.

4.2. Auctions

All five providers state in their marketplace terms that bids cannot be retracted once submitted. As such, a query for the highest bid amount per domain listing was performed to get a number that should at least provide a bottom line for the final auction price. The same is done for the number of bids each auction receives. The results are shown in Figure 3. Outliers from rare occasions in

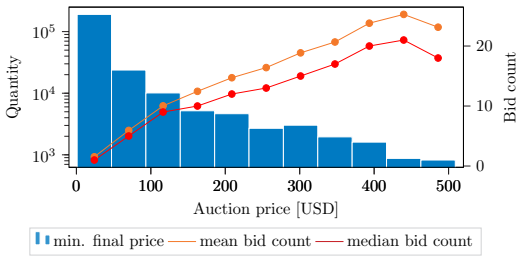


Figure 3: Minimal final auction price (on a logarithmic scale) and mean/median bids per bin.

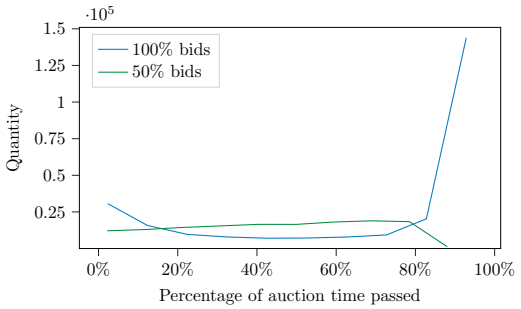


Figure 4: Percentage of days to 50%/100% bids: Most final bids come in the last 10-20% of an auction.

which domains sell at very high prices would impair the readability of the figure and are, therefore, discounted for using the 95th percentile. Instead, these special cases need to be analyzed separately in future work. Approximately 73% of all domains sell for likely less than 50 USD and 83% for likely less than 100 USD. In line with the low average sales price, almost 77% of auctions from sold domains likely end with less than five bids. The mean and median bid counts for each bin are marked by the orange and red line, respectively. As prices increase, both mean and median bid counts increase as well while staying relatively close. Their trends suggest that auctions ending with high prices usually receive a higher number of bids. To find out whether high bid counts result from more bidders participating or a small number of buyers submitting more bids, more information from the marketplaces is needed. Solely Sav provides data on how many individual bidders partake in an auction, thus, our analysis is very limited in this regard. Out of all successful auctions on Sav, about 87% have one or two bidders per auction. At least for the marketplace of Sav, this distribution matches the findings of Roth and Ockenfels [15].

To examine when each auction received its likely winning bid, the total auction duration and the highest number of bids for each auction listing were determined first, then, after what percentage of the total duration that bid count was reached. With the same method, the percentage of time passed until registering 50% of the final bids was obtained. The line graph for auctions that have just received their final bid (i.e., 100% of their bids) in Figure 4 displays a steep incline after 80% of auction time, indicating that a large number of auctions receive their likely winning bid in the final 10-20% of listing duration. Combined with the graph representing the time it takes to receive half of the respective final bids, one can infer that bidding activity appears to increase in the last 10-20% of

 doi: 10.2313/NET-2025-05-1_14

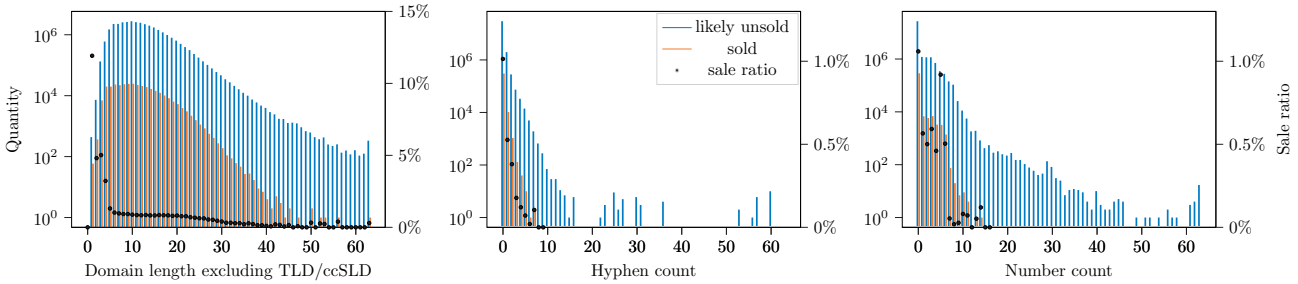


Figure 5: Comparison of lengths, hyphens and numbers between sold and likely unsold domains on a logarithmic scale. Sale ratios were calculated per attribute value and confirm a preference for shorter domains with fewer non-letter characters.

auction time. This observation is consistent with the late-bidding behavior described in [15]. Late-bidding is more prominent in hard-close auctions but it exists to a lesser degree in marketplaces that automatically extend auction timers.

However, one should bear in mind that the data here does not have the same granularity as [15]. Consequently, 10% of an auction duration can be an entire day. While up to half of the bids are submitted late for the majority of sold domains, this does not necessarily mean that interested parties are engaged in a bidding war in the last hour of an auction.

4.3. Properties of Domains Sold

In this final section of the analysis, the focus lies on the characteristics of domains and what sets apart those that have been bought from domains that seem less desirable. Taking advantage of the findings in [2], [7]–[10], the length of the domain name without TLD/country-code SLD (e.g. `co.uk`), the number of numerical characters and the number of hyphens are picked as attributes to analyze. For each property, the dataset is split into sold and unsold domains. Subsequently, the domain lengths, i.e., the number of characters until the first period, and the number of hyphens and numerical characters are determined. Lastly, the sale ratios for each respective attribute value were calculated. For instance, the sale ratio for the hyphen count of one is the ratio of sold domains with one hyphen to all listed domains with one hyphen. The results are summarized in Figure 5.

As described in Section 3, length is one of the most important factors in determining a domain's desirability. Short domains are easier to remember, type out and more rare [10]. To illustrate this, using letters from the English alphabet, there are 676 possible two-letter and 17576 three-letter SLDs. If the letters need to form a word or a nice-sounding combination, the number of possible SLDs is further reduced. Therefore, short domains are generally more valuable. This is reflected in our data as the sale ratios are visibly higher for short domains. In addition, sold domains have a slightly lower mean and median length: 10.87/10 vs 11.98/11. The steeper decline of sold domains as length increases can be interpreted as long domains being less in demand.

The number of hyphens and numerical characters in a domain name displays a similar negative correlation with desirability. Both graphs have a peak in total sales and sale ratios at zero hyphens and zero numbers, respectively. The hyphen count in the set of sold domains does not exceed seven and most sold domains have less than six digits. Hyphens and numbers can impede the memorability of domain names and even lead to misunderstandings [9]. For example, the number '4' being used symbolically for 'for', or not spelling out hyphens when communicating a domain name verbally.

Apart from a structural analysis, the English keywords in sold domains are examined. To find the words a domain is composed of, each domain name is first tokenized and then matched against a corpus of English words from the Python `NLTK` library. Keywords in which single letters have been replaced by numbers, e.g. '3' for the letter 'E' cannot be recognized. Figure 6 shows the most frequent English words that were identified. The size of a word is indicative of how often it occurred. The word cloud mostly contains keywords that are short and generic, yet descriptive enough to give potential visitors an idea of what the contents of a website might be. For example, '`insurance`' in a domain name makes it easy to guess what type of business that domain belongs to. Plugging the keywords into ahrefs, a tool that estimates search engine volumes, further reveals that 82% of these words return more than 100,000 search results each.

Figure 6: Frequent English keywords found in sold domains.

5. Conclusion and Future Work

This paper analyzes second-hand markets for domains by examining the activity in multiple marketplaces, the outcome of auctions, bidding behavior of their customers,

and characteristics of the domains sold with the help of [2], [7]–[10]. Generally, marketplaces display little fluctuation in the number of offered and sold domains. While the bidding activity remained fairly consistent throughout a week, most auctions of sold domains receive up to half of their bids fairly late in their listing duration. Lastly, we showed that the set of sold and unsold domains differ in their domain structure, i.e., SLD length, the number of hyphens and the number of numerical characters.
The analysis was in part limited by the rate at which the data was fetched. More frequent snapshots of marketplaces could reveal more insights into the late-bidding behavior and possibly yield more accurate bid counts and final prices.

References

[1] L. H. T. Inc., "L3 Harris Technologies Inc. Securities and Exchange Commission Filing," https://www.sec.gov/ix?doc=/Archives/edgar/data/202058/000020205824000168/hrs-20240927.htm, 2023, [Online; accessed 05-December-2024].

[2] Z.-g. Wu, G.-h. Zhu, R. Huang, and B. Xia, "Domain name valuation model based on semantic theory and content analysis," in *2009 Asia-Pacific Conference on Information Processing*, vol. 2, 2009, pp. 237–240.

[3] R. Rader, "Domain Name and Related Definitions," https://datatracker.ietf.org/doc/html/draft-ietf-provreg-dn-defn-00, Tech. Rep., 2001, [Online; accessed 05-December-2024].

[4] P. Mockapetris, "Domain names - concepts and facilities," https://www.rfc-editor.org/info/rfc1034, 1987, [Online; accessed 05-December-2024].

[5] ICANN, "EPP Status Codes," https://www.icann.org/resources/pages/epp-status-codes-2014-06-16-en, [Online; accessed 05-December-2024].

[6] DENIC eG, "Redemption Grace Period for .de Domains," https://www.denic.de/domains/de-domains/loeschung/redemption-grace-period, 2013, [Online; accessed 05-December-2024].

[7] S. Dieterle and R. Bergmann, "A Hybrid CBR-ANN Approach to the Appraisal of Internet Domain Names," in *Case-Based Reasoning Research and Development*, L. Lamontagne and E. Plaza, Eds. Springer International Publishing, 2014, pp. 95–109.

[8] Z. Bikadi, S. Ahangama, and E. Hazai, "Prediction of Domain Values: high throughput screening of domain names using support vector machines," 07 2017.

[9] R. Moro-Visconti, *The Valuation of Digital Intangibles Technology, Marketing, and the Metaverse*. Palgrave Macmillan, 2022.

[10] J.-H. Tang, M.-C. Hsu, and T.-Y. Hu, "A General Domain Name Appraisal Model," *Journal of Internet Technology*, vol. 15, pp. 427–431, 2014.

[11] T. Lindenthal, "Valuable Words: The Price Dynamics of Internet Domain Names," *Journal of the American Society for Information Science and Technology*, vol. 65, no. 5, pp. 869–881, 2014.

[12] Verisign, "The Domain Name Industry Brief - quarterly report q3 2024," Verisign, Tech. Rep., 2024.

[13] Sav, "Domain Name Marketplace Agreement," https://www.sav.com/terms/marketplace, [Online; accessed 05-December-2024].

[14] Namecheap, "Domain Market Agreement," https://www.namecheap.com/legal/domains/marketplace-agreement/, [Online; accessed 05-December-2024].

[15] A. Ockenfels and A. E. Roth, "Late and multiple bidding in second price Internet auctions: theory and evidence concerning different rules for ending an auction," *Games and Economic Behavior*, vol. 55, no. 2, pp. 297–320, 2006.

Evaluating Profile-Guided Optimization for DPDK

Lando Jahn, Stefan Lachnit*, Sebastian Gallenmüller*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: lando.jahn@tum.de, lachnit@net.in.tum.de, gallenmu@net.in.tum.de

***Abstract*—This paper evaluates the performance impact of compiler-provided profile-guided optimization (PGO) on a data plane packet processing application that utilizes the Data Plane Development Kit (DPDK).**

We explain the compilation and application process for using PGO and show differences in the setup and performance compared to the regular (no-PGO) DPDK application. In our case, we gain slight but consistent performance benefits from applying PGO. Throughput increases by 0.06%, and latency is decreased by 0.36%.

When applying PGO to the entire application (not just DPDK), we also find that PGO may not always improve performance, as that application is sometimes slower than the one that had no PGO applied.

We conclude that PGO can be a valid optimization technique for packet processing applications when keeping limitations in mind and running performance tests to ensure a positive impact.

***Index Terms*—data plane, DPDK, PGO, packet processing throughput, packet processing latency**

1. Introduction

The "Data Plane Development Kit" (DPDK) [1] is an open-source framework for userspace packet processing. Data plane (as defined by Bifulco and Rétvári in [2]) refers to the part of a router that performs the routing decisions given a policy (which originates from the control plane). For an IP router that is determining where an IP packet is sent. DPDK is implemented in userspace so it avoids overhead-expensive system-calls, which can improve the performance of applications using it [3]. Since processor state data has to be saved/invalidated and restored later (described by Gerhorst et al. in [4]), system calls can take thousands of CPU cycles to execute, as was shown by Soares and Stumm in [5].

When using DPDK, packets skip the processing done by the standard network stack. Copy operations between user and kernel memory are also omitted, as explained by Gallenmüller et al. in [6]. This can improve performance.

A further source of performance improvement can be profile-guided optimization (PGO). Kumar describes how PGO works in [7]: profiling is used to collect data on how a program behaves during runtime. That is then employed to optimize future executions of the program by including the gained knowledge on the code behavior during optimization/compilation.

The goal of this paper is to evaluate whether PGO can have a performance benefit on a DPDK-based application.

2. Related Work

Miano et al. introduce "Morpheus" in [8], which optimizes data plane code (e.g. such that uses DPDK) using a form of PGO (in combination with other techniques). During evaluation (in comparison to the baseline), Morpheus improves the throughput of a DPDK application by up to 469% in some cases, while not affecting it in others. Latency is affected in a strongly positive way (decreasing it by up to 76%) in some cases while in other cases there is almost no effect.

Wintermeyer et al. introduce "P²GO" in [9], a system to optimize P4 data plane programs using traffic traces (representing real-world usage). Resource allocation is reduced to a realistically necessary (instead of theoretically possible) level, leading to increased efficiency.

3. Profile-Guided Optimization Background

PGO is often implemented in a two-stage process [7]: first, the program is compiled with special instructions/-function calls that generate profiling information, such as how often a line of code is executed [10]. This is then run, and the profiling data is collected.

In the second step, the profiling results are taken as optimization input when compiling the code a second time, taking into account the commonly observed behavior of the program. That allows the compiler to optimize the code specifically to what is expected to be needed during runtime. That second version is then used as the final program since it has been optimized and does not contain the profile-generation overhead anymore.

An example of an optimization that could be performed based on profiling data is branch prediction. GCC, for instance, has a mechanism to influence a processor's branch prediction mechanism (accessible using the `__builtin_expect` function, see [11]). When using PGO, we can collect statistics on whether a jump instruction is taken or not during the execution of the first version of the program. We can then use that information to influence the branch prediction to take the branch that is likely to be correct based on real-world runtime data.

What code could look like if one were to perform that manually is shown in Figure 1 using an example.

 doi: 10.2313/NET-2025-05-1_15

```
if (x < 0) {
    // branch true
} else {
    // branch false
}
```

Profiling shows else branch likely to be taken ↓

```
if (__builtin_expect(x < 0, 0)) {
    // branch true
} else {
    // branch false
}
```

Figure 1: Before and after a branch prediction hint is manually applied, assuming profiling has shown the condition to likely be untrue, i.e., the else branch is most likely taken.

One caveat has to be kept in mind: The data/environment that the profile-generating program is executed on should be quite similar to what the final program will mostly encounter. If that is not the case, profile-guided optimizations could decrease performance since they provide false hints to the compiler about the runtime data. This will then likely lead to it generating unfitting optimizations [7]. Going back to the previous example, we can imagine an input that causes the jump instruction to be taken at a rate of 90%. The compiler generates code including static branch-prediction information indicating to the CPU that the branch is expected to be taken. When the actual data then differs, for example by making the program take the branch at a rate of only 20%, the CPU will still predict the branch to be taken, even though it is not taken in most cases. That leads to the processor performing unnecessary operations in the wrong branch and having to go back once the actual result of the if statement's expression is known. The CPU's pipeline will therefore be empty and have to be filled again, decreasing instruction throughput and therefore performance.

4. Test Setup

This section describes how reproducibility is ensured, the PGO is applied, and the tests are executed.

4.1. Ensuring Reproducibility

Since we want the results to be reproducible and as independent of environmental influences as possible, we use the TUM Chair of Network Architectures and Services' testbed infrastructure with the plain orchestration system (pos) introduced by Gallenmüller et al. in [12]. This system allows us to use test nodes connected with 10GigBaseLR (R fiber over 1310 nm optics) to evaluate the influence PGO has on performance metrics. The nodes are reset and equipped with an OS image for each set of runs executed, ensuring reproducibility. There are different test nodes. We use a "device under test" (dut) (where the program to be tested is run) and a peer that the dut communicates with. For some tests, we also use a capture device connected with an optical splitter to accurately determine latency. pos coordinates and synchronizes the test execution and distributes the program/data among the test nodes.

All utilized nodes are running a Debian "bookworm" image and are equipped with an Intel Xeon D-1518 quad-core processor running at 2.2 GHz combined with 32 GiB of DDR4 memory. Intel X552 10 GbE SFP+ network interface controllers (NICs) are used and code is compiled using GCC 12.2.0.

We use two test setups: One for measuring throughput and one for measuring latency. Both are based on MoonGen (introduced by Emmerich et al in [13]), which is a packet generator that uses libmoon as a library. libmoon relies on DPDK.

When running the regular setup (without PGO), the test nodes are first rebooted and the code (scripts and MoonGen/libmoon/DPDK) is copied/cloned to them. It is then compiled right on the nodes before pos initiates the actual tests, which are repeated multiple times to achieve more accurate results.

4.2. Applying PGO

Since PGO requires a modified, more complicated two-stage compilation/execution process, we need to modify the setup and execution phases of the existing codebase. The PGO code evaluated in this paper can be found in [14].

4.2.1. Compiling and Executing DPDK to Generate Profile Data. The first step of the two-stage PGO compilation process calls for compiling the program such that it generates profiling information while running. We do that in the compilation/setup phase the device-under-test node goes through.

The main part of the program that should be compiled with PGO is DPDK. In the MoonGen version we use, DPDK is compiled from a build script inside the aforementioned libmoon using Meson, which is a build system (see [15]). We therefore need to modify the build script such that it adds the necessary flags to the `meson build` command. We use the Unix sed stream editor with a fitting regular expression to achieve that. sed looks for the `meson build` command inside the build script and replaces it with a Meson command that sets the C/C++ compiler/linker flags (`-Dc_args`, `-Dcpp_args`, `-Dc_link_args`, `-Dcpp_link_args`) to the GCC `"-fprofile-generate=<profile-dir>"` flag (see [16]). If concurrency is present in the part of the code that uses PGO, we also add `"-fprofile-correction"` to account for profile data that may not be program-flow consistent (see [17]). Compilation in the second stage (applying the profile) might fail otherwise. That already takes care of compiling DPDK itself to generate profile data.

Since the previously built DPDK code is linked to in the libmoon build stage (which uses CMake, another build system, see [18]), we need to add the library necessary to generate the profiling information (`libgcov.a`, the GNU (test) coverage library, see [10]) to the libraries CMake links against when building MoonGen/libmoon. We therefore add according `target_link_libraries` commands

 doi: 10.2313/NET-2025-05-1_15

to the CMakeLists.txt files present in the MoonGen codebase on the node before calling the build script.

In the test setup execution phase, profiling data will be written to the directory specified by `<profile-dir>`. Each DPDK source file that was used during program execution will have a profiling file (`.gcda`) associated with it inside that directory. It stores profiling information about the code inside the according source file to use when applying PGO for it. This folder will get copied back to the main orchestration node (the node pos was launched from) since it would not be available for future use on the dut after rebooting and loading another image.

4.2.2. Compiling and Executing DPDK to Use Profile Data. The second stage (application of profile data to optimize the code) looks quite similar from our perspective. We need to apply the previously generated profile data. That requires us to copy the profiling data generated in the runs of the first stage to the dut node before setting it up. Also, the `meson build` has to happen with the GCC flag `"-fprofile-use=<profile-dir>"` (see [17]) instead of `"-fprofile-generate=<profile-dir>"`, which will make GCC apply the profile data generated in the profiling runs to compile specifically optimized code. `"-fprofile-correction"` may also need to be applied in some cases as mentioned before.

The compiler will complain that for some files, no profiling information is available. That is expected since DPDK is built as a library, so the entire DPDK codebase is compiled. However, only certain parts of the code (e.g. some functions) are used by MoonGen, so others have never been executed and have therefore not generated any profiling data. We confirm the assumption that PGO is still being applied correctly by trying the second stage compilation (applying profiling data) with an empty directory as the `<profile-dir>`. We can see considerably more warnings about missing profile information. The difference between this and the regular compilation with a filled `<profile-dir>` has to be the set of files for which profile data was successfully found and applied in the regular setup. The performance results of this stage's execution phase are the ones we are interested in for our comparison since the program had PGO applied to it.

4.2.3. Adding Other Parts of MoonGen. We cannot only apply PGO to the DPDK part of MoonGen, but also to the MoonGen code itself. To achieve that, we add the flags as described in subsubsection 4.2.1 and subsubsection 4.2.2 to the various CMake/-Makefile invocations in the MoonGen/libmoon build script. Again, we need to use two different flags (`"-fprofile-generate"` and `"-fprofile-use"`) for the two stages. PGO for those parts of MoonGen did not work without `"-fprofile-correction"`, so concurrency is always present there.

By looking at the generated files inside the directory passed to GCC, we can see that only a small set of files was added to the PGO step compared to the large amount from DPDK. The vast majority of optimizations, therefore, have likely already been applied when PGO was only used with DPDK.

5. Test Results

We perform throughput and latency tests, each with three different cases we can compare: no PGO used, PGO only applied to DPDK, and PGO applied to the entirety of MoonGen (as described in subsection 4.2). The application on the dut forwards packets between two ports without modifications.

Throughout the tests, we observe performance results with very small margins between the different optimization levels (i.e., if and where PGO was applied). To ensure significance, we run tests multiple times, while also alternating the order of the optimization levels to mitigate effects such as hardware warming up. We find the results to always have the same trends regardless of the order or the set of runs. We therefore assume the results to be valid and not based on fluctuations.

5.1. Throughput Tests

For the throughput test setup, results are taken from the device under test. It measures how many packets, and following from that, bytes are sent/received per second.

The exact throughput numbers are taken from the loadgen_runX.stdout files, where X is the number of the run. We average the throughput over all runs of a set to balance out inaccuracies.

Figure 2 shows the throughput performance results. We can see that the program which had PGO applied to only DPDK performs slightly better than the program which does not use any PGO. Both the RX (receive), and the TX (transmit) throughputs are about 0.06% higher for the PGO program. When PGO is applied to the whole program, it performs just slightly worse for both RX (not visible in Figure 2 due to rounding) and TX throughputs than the no-PGO version. This result is surprising, since when applying PGO to the entire program, we also apply it to the DPDK component present in MoonGen. Under the assumption that the DPDK component with PGO performs the same in both scenarios, these results indicate that the non-DPDK part of MoonGen does not only not profit from PGO, but is harmed by it performance-wise.

5.2. Latency Tests

For the latency test setup, results are taken from the capture device which is connected to the other two nodes using an optical splitter. That allows it to measure precisely when a packet is received and when it is answered. It can then calculate accurate latencies from that.

The latency numbers are taken from the summary of the capture_runX.stout files, where X is the number of the test run. The values over all these files of a test set are again averaged to balance out fluctuations.

Figure 3 shows the latency performance results. Again, the program that had PGO only applied to DPDK performs best, achieving the lowest latency on average. It is roughly 0.36% lower than that of the no-PGO program. Applying PGO also to the rest of MoonGen proves counterproductive for performance, increasing latency compared to the DPDK-only PGO program by roughly 0.10%. While its performance was below that of the no-PGO program in the throughput tests, it is now slightly faster (decreasing

latency by 0.26%), but still slower than the version that had PGO only applied to DPDK.

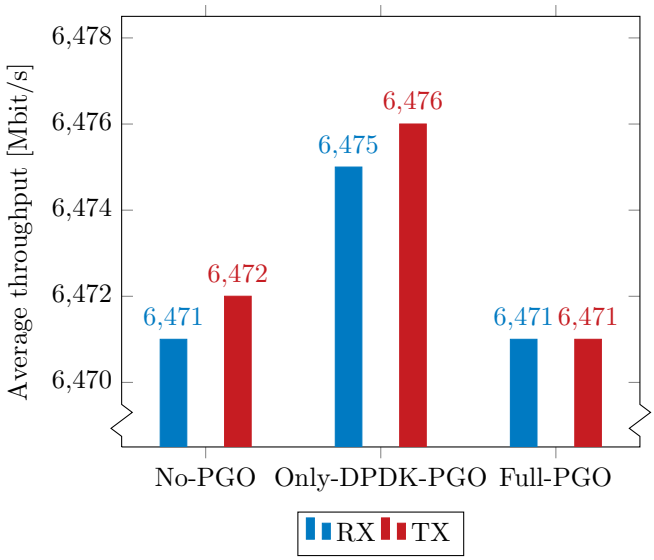


Figure 2: Performance results of the MoonGen DPDK applications testing throughput. Results are divided into RX (receive) and TX (transmit) throughput for each version of the program. We compare performance results without PGO (No-PGO), with PGO only applied to the DPDK component of MoonGen (Only-DPDK-PGO), and with PGO applied to the entire program (Full-PGO).

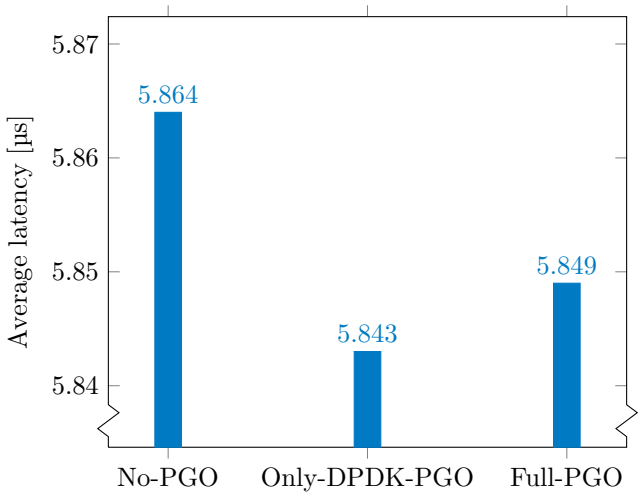


Figure 3: Performance results of the MoonGen DPDK applications testing latency. We compare performance results without PGO (No-PGO), with PGO only applied to the DPDK component of MoonGen (Only-DPDK-PGO), and with PGO applied to the entire program (Full-PGO).

5.3. Interpretation of the Performance Results

The performance results show a slight performance benefit (increase in throughput and decrease in latency) when applying PGO to DPDK only. Applying it to the rest of the program proves to be counter-productive in our tests. We do not have an explanation for this behavior, though we can form some assumptions.

While the DPDK-only PGO latency tests worked without the `"-fprofile-correction"`-flag, that was not the case for the whole-program PGO tests. This implies that concurrency is present in the non-DPDK part of MoonGen, since this flag is used to mitigate profiling files corrupted by concurrent execution of profile-generating program code [17].

We can also see that the performance benefit from the no-PGO to the DPDK-only PGO program is considerably larger in the latency tests compared to the throughput tests (by about a factor of six). When taking into account that all throughput PGO runs (DPDK-only and whole-program) needed the `"-fprofile-correction"`-flag (implying that concurrency is also present in the DPDK part here, likely through different usage of the DPDK library), it could be inferred that concurrency in profiled program parts might be hindering performance when using PGO.

5.4. Comparison with Previous Work

When comparing our results to the ones obtained on a DPDK application using Morpheus (as described in section 2), we achieve a drastically lower speedup. However, Morpheus' approach is quite different from ours, so comparability is lacking. While we restrict ourselves to regular PGO (as supported by modern compilers such as GCC through gcov-profiling, see [10]), Morpheus combines several techniques, including static code analysis [8].

While our approach is limited to profiling the CPU domain, Morpheus' approach does not have this limitation. It can access domain-specific information relevant to packet-processing applications, such as match-action table access patterns [8].

6. Conclusion and Future Work

We showed that regular compiler profile-guided optimization can have a positive impact on both the throughput and latency performance of a DPDK application.

The usage of PGO is only moderately complicated, requiring an extra stage for compiling with the profile-generating instructions/function calls. Once that is done and profiling has been generated, it is as simple as applying the generated profiling information while compiling. The generated code can then be used just like it would be without PGO while potentially providing improved performance.

Also to be kept in mind is that PGO does not always have a non-negative impact on performance, as seen with the Full-PGO tests in section 5. This shows performance benefits should not just be assumed to exist, but actively verified for a worthwhile PGO application.

In the future, it would be interesting to evaluate different PGO approaches, such as AutoFDO introduced by Chen et al. in [19]. Evaluation could be done based on ease of use and performance.

Additionally, it could be beneficial to evaluate our applications using PGO in a more real-world environment, where the traffic the profiling was generated on may differ from the traffic that is present. This could demonstrate what influence the level of similarity of the profiling to the application data has on performance.

References

[1] DPDK Project, "DPDK," accessed 14.12.2024. [Online]. Available: https://www.dpdk.org/

[2] R. Bifulco and G. Rétvári, "A survey on the programmable data plane: Abstractions, architectures, and open problems," in *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, 2018, pp. 1–7.

[3] AvidThink, Converge! Network Digest, and The Linux Foundation, "Myth-busting DPDK in 2020," accessed 14.12.2024. [Online]. Available: https://nextgeninfra.io/dpdk-myth-busting-2020/

[4] L. Gerhorst, B. Herzog, S. Reif, W. Schröder-Preikschat, and T. Hönig, "Anycall: Fast and flexible system-call aggregation," in *Proceedings of the 11th Workshop on Programming Languages and Operating Systems*, ser. PLOS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1–8. [Online]. Available: https://doi.org/10.1145/3477113.3487267

[5] L. Soares and M. Stumm, "FlexSC: Flexible system call scheduling with Exception-Less system calls," in *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*. Vancouver, BC: USENIX Association, Oct. 2010. [Online]. Available: https://www.usenix.org/legacy/event/osdi10/tech/full_papers/Soares.pdf

[6] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle, "Comparison of Frameworks for High-Performance Packet IO," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2015)*, Oakland, CA, USA, May 2015.

[7] N. Kumar and International Business Machines Corporation, "Profile-guided optimization (PGO) using GCC on IBM AIX," accessed 15.12.2024. [Online]. Available: https://developer.ibm.com/articles/gcc-profile-guided-optimization-to-accelerate-aix-applications/

[8] S. Miano, A. Sanaee, F. Risso, G. Rétvári, and G. Antichi, "Domain specific run time optimization for software data planes," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1148–1164. [Online]. Available: https://doi.org/10.1145/3503222.3507769

[9] P. Wintermeyer, M. Apostolaki, A. Dietmüller, and L. Vanbever, "P2go: P4 profile-guided optimizations," in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, ser. HotNets '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 146–152. [Online]. Available: https://doi.org/10.1145/3422604.3425941

[10] Free Software Foundation, Inc., "10 gcov—a Test Coverage Program," accessed 16.12.2024. [Online]. Available: https://gcc.gnu.org/onlinedocs/gcc/Gcov.html

[11] ——, "6.64 Other Built-in Functions Provided by GCC," accessed 16.12.2024. [Online]. Available: https://gcc.gnu.org/onlinedocs/gcc/Other-Builtins.html

[12] S. Gallenmüller, D. Scholz, H. Stubbe, and G. Carle, "The pos framework: a methodology and toolchain for reproducible network experiments," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 259–266. [Online]. Available: https://doi.org/10.1145/3485983.3494841

[13] 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, p. 275–287. [Online]. Available: https://doi.org/10.1145/2815675.2815692

[14] S. Lachnit, "DPDK-IOMMU-Effects," accessed 20.12.2024. [Online]. Available: https://gitlab.lrz.de/sl/misc-stuff/dpdk-iommu-effects/-/tree/315d9edc1f11ce4db74aae4b1625814fe81be2cc

[15] J. Pakkanen, "The Meson Build system," accessed 16.12.2024. [Online]. Available: https://mesonbuild.com/

[16] Free Software Foundation, Inc., "3.12 Program Instrumentation Options," accessed 16.12.2024. [Online]. Available: https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

[17] ——, "3.11 Options That Control Optimization," accessed 16.12.2024. [Online]. Available: https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

[18] Kitware, Inc., "CMake," accessed 16.12.2024. [Online]. Available: https://cmake.org/

[19] D. Chen, D. X. Li, and T. Moseley, "Autofdo: Automatic feedback-directed optimization for warehouse-scale applications," in *CGO 2016 Proceedings of the 2016 International Symposium on Code Generation and Optimization*, New York, NY, USA, 2016, pp. 12–23.

Categorization of TLS Scanners

Youssef Jemal, Tim Betzer*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: youssef.jemal@tum.de, betzer@net.in.tum.de

***Abstract*—The Transport Layer Security (TLS) protocol is a cornerstone of modern web security. In the TLS ecosystem, the configurations of servers are mostly concealed from clients, as servers only react to the clients' proposals during the handshake process. Scanning and fingerprinting approaches address this gap by performing several handshakes with the server in order to extract as much information as possible about its TLS configuration. This information can be helpful in improving the network security by discovering misconfigurations and identifying malicious servers. This paper classifies existing TLS scanners into three categories. We introduce every category and discuss its characteristics while highlighting its advantages and drawbacks. We also introduce a local testbed to compare the performance of various scanners based on their ability to distinguish between different TLS configurations.**

***Index Terms*—Active scanning, Fingerprinting, TLS, SSL**

1. Introduction

Throughout the last years, Transport Layer Security (TLS) has become a widely used security protocol and the standard for encrypted communication over the Internet [1]. It guarantees the integrity and confidentiality of the data as well as the authentication of the involved parties. The TLS protocol begins with a handshake where the client and the server negotiate a common cryptographic base. In the handshake, the client shares all their capabilities with the server. The latter, however, only chooses from the client's proposals according to its internal configuration. As a result, the server's TLS capabilities remain unclear for external parties.
A possible way to uncover this information is by passively listening to the server's TLS communications. Based on the content of the captured data packets, we attempt to reconstruct the server's TLS configuration.This approach does not generate any additional traffic and does not strain either the target server or the network. It is, however, inherently inefficient with protocols that implement encryption mechanisms since a third-party listener has no access to the session's cryptographic keys. For example, in TLS 1.3, the server already encrypts several fields in the *Server Hello* message, which makes them inaccessible for analysis by passive monitoring tools [2]. This is where active scanning tools become relevant. The idea is to craft and transmit several *Client Hello* messages and then observe the server's responses to attempt to reconstruct its TLS configuration. Such scans are beneficial not only because they provide us with a detailed overview of the server's configuration, but also because they are powerful tools for finding vulnerabilities and misconfigurations. Furthermore, scanning approaches can help detect malicious Command & Control (C&C) servers as they usually have the same, or similar, configurations [3].
This paper categorizes existing TLS scanners into three types: Elementary TLS Scanners such as TUM goscanner [4] or zgrab2 [5]. In their basic mode, these scanners only initiate one TLS handshake and return the server's response. They are practical for Internet-wide measurements [5] and can provide us with invaluable insights into the TLS ecosystem. The second type are server debugging tools that are able to reconstruct a detailed and comprehensive representation of the server's internal configuration, such as DissecTLS [6], SSLyze [7], and testssl.sh [8].
The third and last type are fingerprinting approaches. The most relevant representatives are JARM [9] and Active TLS Fingerprinting (ATSF) [10]. These approaches only send a small fixed number of *Client Hello* messages and generate a unique fingerprint for every TLS configuration, which is primarily used to differentiate and compare servers.
The remainder of this paper is organized as follows: We start by explaining the methodology of the TLS protocol in section 2. Section 3 summarizes the most important TLS configuration parameters from a scanner's perspective. Section 4 analyzes different TLS scanning approaches and discusses their advantages and limitations. In section 5, we compare the performance of TLS scanners across different categories in a local environment before we conclude in section 6.

2. Methodology

A basic understanding of the TLS protocol, especially the TLS handshake, is required to understand the scanners' work system. This paper only focuses on TLS 1.2 and TLS 1.3, as all previous versions are deprecated [11]. They are also the most relevant and widely used versions within the TLS ecosystem [6]. TLS 1.2, standardized in 2008 [12], has been the backbone of secure Internet communication for over a decade. Its successor, TLS 1.3, was standardized in 2018 [13] and introduced several significant improvements. The faster handshake process is one of the most important enhancements. TLS 1.3 only takes 1 Round-Trip Time (RTT) to complete the handshake instead of 2 RTTs in TLS 1.2. The newest version is also inherently more secure as it only supports AEAD cipher suites [13] that simultaneously provide privacy and authenticity [14].

 doi: 10.2313/NET-2025-05-1_16

The following figure captures the most important steps of the TLS 1.3 handshake:

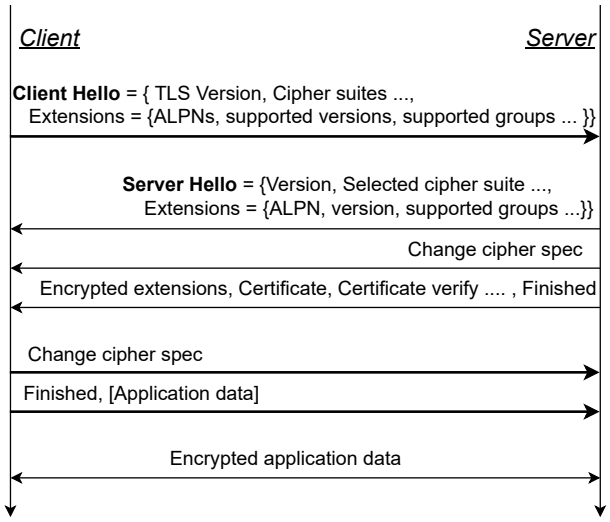


Figure 1: TLS 1.3 Handshake [15]

2.1. Client Hello

The client initiates the handshake by sending a *Client Hello* message to the server [13]. This message specifies the client's supported TLS versions and suggests several cipher suites for the server to choose from. The *Client Hello* message can potentially contain more than ten extensions [13] with the majority of them not relevant for our work. This paper only focuses on the most important extensions from a scanning perspective. As part of the handshake, the client also generates multiple key pairs (public and private) based on the proposed key exchange algorithms (in the *supported groups* extension) and includes the public keys in the *key share* extension of the CH message [13]. These keys are necessary for generating the symmetric encryption key.

2.2. Server Hello

In response to the *Client Hello* message, the server replies with a *Server Hello* message [13]. This message selects a cipher suite from those proposed by the client and specifies the server's extensions. After generating its key pair and receiving the client's public key, the server can calculate the symmetric encryption key. It also includes its public key in the *key share* extension to allow the client to calculate the same cryptographic material. Furthermore, in order to authenticate itself, the server sends its *Certificate* and a *Server Certificate Verify* message to prove ownership of the private key associated with the certificate. To ensure the integrity of the handshake, the server applies a hash function on all the previously exchanged data and sends the result in the *Finished* message (encrypted). The client also performs the same operation and compares the output with server's hash. If both hashes are equal, then it means that the handshake messages were transmitted correctly. To conclude the handshake, the client sends a *Change Cipher Spec* message followed by a *Finished* message. From this point, the client and the server can start exchanging application data in an encrypted and secure manner. In earlier TLS versions, the *Change Cipher Spec* record was used as an indication that all subsequent records will be encrypted [12]. To avoid misbehavior of middleboxes, implementations try to make the TLS 1.3 handshake similar to TLS 1.2 [13]. This includes the transmission of the *Change Cipher Spec* record, which does not serve any function in TLS 1.3.

3. TLS Configuration parameters

It would be impractical and costly for scanners to extract all the server's TLS properties since it would require a lot of time and resources. Depending on the information it wants to gather, each scanner prepares a specific strategy to extract this information [6]. The table below outlines some of the TLS configuration parameters that can be extracted by scanning tools [6].

TABLE 1: The typically extracted TLS configuration properties

Parameters	Representation
Supported Versions	set
Cipher Suites Supported Groups ALPNs	priority list/set[1]
Deflate compression	bool

[1] A set in case the server uses the client preferences.

The Supported Versions field lists all the TLS versions supported by the server. These versions include TLS 1.0, TLS 1.1, TLS 1.2, and TLS 1.3. Although TLS 1.0 and TLS 1.1 were formally deprecated in RFC 8996 due to security concerns [11], they continue to be accepted by servers within the TLS ecosystem [6]. One of the most important properties of a TLS server is its supported cipher suites. A cipher suite in TLS 1.3 specifies the AEAD (Authenticated Encryption with Associated Data) cipher mode that will be used for bulk encryption and a hash algorithm. This is different from previous TLS versions, where the cipher suite also contained information about the key exchange algorithm. In TLS 1.3, the key exchange algorithm is negotiated through extensions as discussed in the previous section. A server's supported cipher suites are defined in relation to a specific TLS version. For example, TLS 1.3 only defines and supports 5 cipher suites [13], meaning that any server using this version should only accept a subset of these 5 cipher suites [13].

The supported groups parameter specifies the cryptographic groups that the server supports for the key exchange protocol used for the generation of the symmetric encryption key.

The Application-Layer Protocol Negotiation (ALPN) extension optimizes the communication. In the *Client Hello* message, the client provides a list of the supported application protocols, and the server selects at most one protocol based on its capabilities. The server then includes

 doi: 10.2313/NET-2025-05-1_16

it in the ALPN extension of the *Server Hello* message [16]. This extension enables the negotiation of the application layer protocol during the TLS handshake. It, therefore, reduces the overhead that would have otherwise occurred if the negotiation were to be completed after the handshake.
In versions prior to 1.3, the TLS protocol included support for compression methods, with the most used being the *deflate compression* method. Clients and servers would negotiate a commonly supported compression method through the *compression methods* field. However, TLS 1.3 obsoleted this feature entirely due to known security vulnerabilities, such as the CRIME attack [17]. TLS 1.3, however, still implements the *compression methods* field in the handshake messages, with its value always set to 0 (indicating no compression). This is done to maintain backward compatibility with previous versions.

4. Capabilities of TLS Scanners

This section explores the three categories of TLS scanners: Elementary scanners, fingerprinting approaches, and server debugging tools. In the following subsections, we introduce each type and provide examples of different implementation approaches.

4.1. Elementary Scanners

Elementary scanners initiate a single handshake with the server and return the server's response. This approach provides very limited information about the server's configuration, which makes such tools impractical for scanning in their base mode. They are utilized as foundations for more complex implementations, where their elementarity and scalability can be leveraged as part of a deeper analysis. A prominent example is zgrab2, which can perform a TLS handshake over HTTP with the entire IPv4 address space in less than 6 hours and 20 minutes [5]. This tool is highly effective for conducting Internet-wide surveys. *Censys.io* [5] is a search engine powered by zgrab2 that automates Internet-wide scanning. It offers a REST API and a web interface for querying an up-to-date database of the public address space gathered through continuous scanning with zgrab2. The search interface supports advanced search features, including "full-text searches, regular expressions, and numeric range queries" [18]. For example, a command such as `services.tls.versions.tls_version = "TLSv1_3" and location.country_code = "DE"` returns all IPv4 hosts in Germany that support TLS 1.3, which are currently around 3.7M hosts. Durumeric *et al.* [5] argue that this approach democratizes the internet-wide scanning process by making the scanning of the TLS ecosystem accessible to researchers and users in general without concerns about legal permissions or network infrastructure.
Another example is TUM goscanner [4] which serves as a foundation for the implementation of DissecTLS.

4.2. TLS Debugging Tools

Scanners of this type provide a detailed overview of the server's TLS configuration. This is typically achieved by sending a large number of *Client Hello*s to exhaustively test all possible configurations. As there are over 370 ciphers that can be used across different TLS versions, the number of *Client Hello*s required is primarily determined by the *Cipher Suites* parameter [19]. SSLyze [7] is a scanning tool that implements a "naive" algorithm. It performs a full cipher suite scan by conducting one TLS handshake for each cipher suite. This results in approximately 543 transmitted *Client Hello* messages [19]. A main advantage of this stateless approach is that it allows for parallelization of requests, trading the high generated traffic for a speedup in the scan execution [19]. testssl.sh [8], while more efficient and optimized than SSLyze [6], also remains impractical for big-scale measurements due to the high number of sent requests. Such tools should only be used on a small scale, where we're only interested in fast and precise results and where the scanning costs can be neglected. The most prominent debugging tool that combines the lightweight nature of fingerprinting approaches with the precision and completeness of debugging tools is DissecTLS. Integrated as a feature of TUM goscanner [4], DissecTLS divides the scanning process into multiple subtasks, each responsible for collecting information about specific parameters [6]. Each task maintains a state and dynamically crafts the next *Client Hello* based on this state. This approach reduces redundant handshakes that do not yield new information.

4.3. Fingerprinting Approaches

Fingerprinting approaches, unlike the previously discussed categories, do not produce a human-readable representation of the server's configuration. Instead, they aim to minimize the number of sent *Client Hello*s while extracting as much information as possible. The collected data is then stored in a concise format called a "fingerprint", which is primarily used for comparing servers [3]. The reduced traffic footprint and the lightweight output make these methods highly scalable and well-suited for large-scale scanning. The most significant use case for these tools is the identification of Command and Control (C&C) servers, which are a fleet of computers that have the same or similar configurations and are used to transmit malicious commands and steal data. By comparing the similarity between a fingerprint of an unknown server and that of a known C&C server, fingerprinting tools provide a high probability indication on whether the server is a C&C server. Althouse *et al.* [3] argue that their fingerprinting tool JARM can identify "most, if not all Cobalt Strike C2 servers" on the IPv4 address space on port 443.
JARM [9] sends 10 *Client Hello*s specifically designed to extract as much information as possible about the server's TLS configuration. It then uses the received *Server Hello*s to produce a 62-character fingerprint with the following structure [3]:

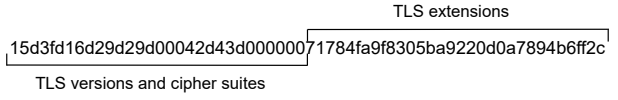


Figure 2: JARM Output Example

Each three characters of the first 30 provide information about the TLS version and cipher suite chosen by the

 doi: 10.2313/NET-2025-05-1_16

server as a response to each of the ten *Client Hello*s. A "000" indicates that the server refused the connection [3]. The remaining 32 bits are constructed using a truncated SHA256 hash on the TLS extensions sent by the server in the *Server Hello*.
An even more advanced fingerprinting tool is Active TLS Fingerprinting (ATSF). Sosnowski *et al.* [10] suggest that this tool is superior to JARM in identifying C2 Servers and generally better at distinguishing server configurations [10]. Similar to JARM, ATSF sends ten distinct *Client Hello* messages. However, it provides a more precise fingerprint by leveraging additional TLS handshake messages. While JARM only deals with the extensions present in the *Server Hello* message (Plus the ALPN extension), ATSF also incorporates extensions from *Encrypted Extensions*, *Certificate Request*, *Hello Retry Request* and *Certificate* TLS messages [10]. ATSF, however, produces a longer output than JARM since it employs a different technique for generating fingerprints. In fact, for every TLS handshake, it produces an independent fingerprint and then concatenates all these outputs to generate the final server's fingerprint [10]. This is an example of an elementary handshake fingerprint:

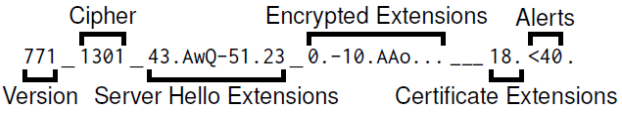


Figure 3: ATSF handshake fingerprint example

5. Comparison of Scanners in a Local Testbed

In our local testbed, we compared the TLS scanners based on two important criteria: their capability to correctly distinguish between different TLS configurations, and the amount of traffic they produce throughout the scanning process. The main test parameter for this experiment is the *Cipher suites* parameter. We selected this parameter because it allows for a wide range of possible configurations, matching our experiment's needs. We selected 5 TLS 1.2 cipher suites, resulting in 325 unique configurations. We can compute this number by calculating the number of all permutations of all the possible combinations of the selected cipher suites. The use of permutation was necessary because server preferences were enabled throughout the experiment (except when scanning the *Preferences* property). This means that the server keeps an internal priority list of the supported cipher suites and always picks the highest-priority cipher suite available. For each configuration, we deployed a Docker container running an Nginx Version 1.27.3 server. We ran each scanner against all 325 servers and counted the unique scan outputs generated. This value reflects the number of configurations successfully identified by the scanner. For ATSF and JARM, we were able to use the output directly. For the other scanners, we used a Python script that extracts the scanned cipher suites from the scan results, which we can then use as a basis for the differentiation. This measure was necessary to avoid unstable results, as some parameters, such as *scan time*, can vary even for the same configuration. We also extracted other simple parameters, such as *Preferences*, and *Session Tickets* which can be either en- or disabled. To assess the scanning costs, We used tcpdump to capture the number of *Client Hello*s sent by the client during each scan, and then computed the average number across all scans.
The following table summarizes the results of the experiment.

TABLE 2: Identified number of Nginx configurations for each scanner.

Test Case	ATSF	JARM	SSLyze	testssl.sh	DissecTLS	Total
Cipher Suites	22	17	31	325	325	325
Preferences	2	2	1	2	2	2
Session Tickets	2	2	2	2	2	2
Average CHs	10	10	447	128.1	11	-

As shown in the table, only dissecTLS and testssl.sh were able to completely identify all the servers' cipher suites configurations. ATSF and JARM, however, were only able to differentiate around 20 unique configurations. This result is expected, as these fingerprinting tools only send a fixed number of *Client Hello*s. Their lower level of precision is offset by significantly reduced scanning costs, making them ideal tools for large-scale deployments. SSLyze had a relatively poor performance, as it was only able to differentiate 31 unique configurations. The reason behind this is that SSLyze does not consider the order of the cipher suites [6], meaning it could only differentiate between the different combinations of the 5 cipher suites, which amount to 31 configurations. On top of that, it was by far the most costly scanner in terms of generated traffic, with an average of 447 *Client Hello*s. While testssl.sh was able to distinguish all configurations, its high average scanning cost of 128 *Client Hello*s makes it unsuitable for large-scale use cases. DissecTLS is the best-performing scanner in this experiment. It effectively combines the low cost typically associated with fingerprinting approaches while maintaining the accuracy and precision of TLS debugging tools. The average of approximately 11 sent *Client Hello*s, along with the successful identification of all the TLS configurations, clearly demonstrates this.

6. Conclusion and Future Work

In this paper, we introduced the concept of TLS scanning. We presented three different scanning categories: Elementary scanners, fingerprinting techniques, and debugging tools. We explained each category's methodology and introduced its most relevant representatives. Additionally, we discussed each type's advantages, limitations, and use cases. We then compared the performance of different scanners in a controlled environment by evaluating their ability to identify different TLS configurations while monitoring the scanning costs. The conclusion was that DissecTLS is the most efficient scanner, providing the precision of debugging tools while generating minimal traffic, typically a feature of fingerprinting tools. It should be mentioned, however, that the experiment was conducted in an artificial environment. The results could be slightly

 doi: 10.2313/NET-2025-05-1_16

different from reality. As part of future work, we could scan actual servers within the TLS ecosystem and then compare the scanners' performance. This would provide more realistic and precise results.

References

[1] R. Holz, J. Hiller, J. Amann, A. Razaghpanah,T. Jost, N. Vallina-Rodriguez, O. Hohlfeld, "Tracking the deployment of TLS 1.3 on the Web: A story of experimentation and centralization," *ACM SIGCOMM Computer Communication Review, Volume 50, Issue 3*, 2020.

[2] R. Holz, J. Amann, A. Razaghpanah, and N. Vallina-Rodriguez, "The era of tls 1.3: Measuring deployment and use with active and passive methods," 2019. [Online]. Available: https://arxiv.org/abs/1907.12762

[3] J. Althouse, A. Smart, RJ Nunnally, M. Brady, "Easily identify malicious servers on the internet with jarm." [Online]. Available: https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a/

[4] O. Gasser, M. Sosnowski, P. Sattler, J. Zirngibl. (2022). [Online]. Available: https://github.com/tumi8/goscanner

[5] Z. Durumeric, D. Adrian, A. Mririan, M. Bailey, J. Alex Haldermann, "A search engine backed by internet-wide scanning," *CCS '15: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[6] M. Sosnowski, J. Zirngibl, P. Sattler, and G. Carle, "Dissectls: A scalable active scanner for tls server configurations, capabilities, and tls fingerprinting," in *Passive and Active Measurement*, A. Brunstrom, M. Flores, and M. Fiore, Eds. Cham: Springer Nature Switzerland, 2023, pp. 110–126.

[7] A. Diquet. [Online]. Available: https://github.com/nabla-c0d3/sslyze

[8] Dr. Wetter. Testing tls/ssl encryption. [Online]. Available: https://testssl.sh/

[9] J. Althouse, A. Smart, RJ Nunnally, M. Brady. [Online]. Available: https://github.com/salesforce/jarm

[10] M. Sosnowski, J. Zirngibl, P. Sattler, G. Carle, C. Grohnfeldt, M. Russo, D. Sgandurra, "Active TLS Stack Fingerprinting: Characterizing TLS Server Deployments at Scale," 2022. [Online]. Available: {https://arxiv.org/abs/2206.13230v1}

[11] K. Moriarty, S. Farrell. (2021) Deprecating tls 1.0 and tls 1.1. [Online]. Available: https://www.rfc-editor.org/info/rfc8996

[12] I. E. T. Force. (2008) The transport layer security (tls) protocol version 1.2. [Online]. Available: https://www.rfc-editor.org/info/rfc5246

[13] ——. (2018) Rfc 8446: The transport layer security (tls) protocol version 1.3. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8446#page-60

[14] P. Rogaway, "Authenticated-encryption with associated-data," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 98–107. [Online]. Available: https://doi.org/10.1145/586110.586125

[15] M. Driscoll. The illustrated tls 1.3 connection. [Online]. Available: https://tls13.xargs.org/

[16] I. E. T. Force. (2014) Transport layer security (tls) application-layer protocol negotiation extension. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7301#page-3

[17] P. Sirohi, A. Agarwal, and S. Tyagi, "A comprehensive study on security attacks on ssl/tls protocol," in *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, 2016, pp. 893–898.

[18] [Online]. Available: https://search.censys.io/

[19] W. Mayer and M. Schmiedecker, "Turning active tls scanning to eleven," in *ICT Systems Security and Privacy Protection*, S. De Capitani di Vimercati and F. Martinelli, Eds. Cham: Springer International Publishing, 2017, pp. 3–16.

Current State of BBR Congestion Control

Miyu Kitamura, Benedikt Jaeger*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: ge23cag@mytum.de, jaeger@net.in.tum.de

***Abstract*—Widely deployed, loss-based, congestion control and avoidance mechanisms, such as CUBIC and RENO, have served well for many years. Until recently, packet loss may have been a good indicator for congestion, but with increasing wireless networks in public venues, as well as ever-growing buffer sizes leading to bigger delays, packet loss has become an unreliable indicator of congestion. In this paper, we present the Bottleneck-Bandwidth and Round-Trip (BBR) algorithm and its evolution from version 1 to version 3. Originally developed by Google in 2017 [1], BBR uses a model-based approach to regulate congestion, making it better suited to modern network conditions.**

***Index Terms*—BBR, Network Path Modeling, Congestion Control, TCP**

1. Introduction

When Jacobson et al. set out to analyze the reason for the "first of [...] a series of congestion collapses" [2] they developed a new way to prevent such an event from happening again. It was then that packet loss was decided to be used as the main indicator for congestion in a network [1]. They proposed to use a mechanism called the congestion window (**cwnd**) in order to start sending packets at a slow rate, increasing it on every successful transmission indicated by a received acknowledgement (**ACK**). This algorithm is now known as TCP Tahoe [3]. Many other congestion control algorithms, such as TCP Reno and TCP CUBIC, the latter of which has become the default algorithm in Linux as stated by Sangtae Ha et al [4], were developed over the years. Lately however, the rise of wireless network in public venues and ever-growing buffer sizes have led to packet loss not being a reliable indicator of congestion anymore [1]. This is where the BBR algorithm developed by Google comes into play. The main contribution of this paper is to give an overview of the current state of BBR congestion control and compare the different versions that have been developed since. In the following sections, we first provide an overview of conventional congestion control methods and discuss the limitations of relying solely on packet loss. BBR's fundamental concepts and goals are then introduced. We explore how each successive version refines the algorithm's design, model parameters, and fairness. Finally we discuss the current challenges, achievements and future directions for BBR.

2. TCP Congestion Control

Traditional congestion control algorithms, dating back to TCP Tahoe and Reno, rely on packet loss as a primary indicator of congestion and use the cwnd as primary mechanism to regulate the sending rate. This cwnd describes the maximum not yet acknowledged amount of data in bytes that can be in flight at any given time as described by Rasool Al-Saadi et al [5]. The sender continually increases its cwnd until loss is detected, then reduces it aggressively. The window is then steadily increased again, starting the process anew. Loss itself is detected via duplicate ACKs triggered by out-of-order arrivals. Instead of reacting to the first duplicate ACK the algorithm waits for three or more duplicates (a process called *fast retransmit*) [5] to confirm that out-of-order arrival is a result of congestion. By how much the cwnd is reduced and in what manner it is restored thereafter differs between algorithms. Loss-based algorithms therefore require a low loss environment to ever efficiently utilize the connection for a prolonged time. Even if the cwnd ever reaches its maximum, loss-based algorithms will continue growing it, leading to an unavoidable loss event and cwnd reduction. Gomez et al. state that "using Reno in a 10 Gbps link with a 100 ms propagation delay needs more than an hour to fully utilize the bandwidth. Avoiding this issue demands a loss rate lower than 0.00000002%" [6]. One important aspect to notice here is that by detecting loss on duplicate acknowledgments it takes at least one round-trip time (**RTT**) to detect the congestion and make adjustments which is why longer buffers make it difficult to adapt quickly due to the delay they inherently introduce by filling the network bottleneck's buffer.

2.1. CUBIC

One of the most widely used algorithms is TCP CUBIC, which was developed by Sangtae Ha, Injong Rhee, and Lisong Xu in 2008 [4]. Instead of growing the cwnd linearly after loss like TCP Reno does, Sangtae Ha et al. propose to instead use a cubic function to restore the cwnd, making the recovery of the cwnd much faster. The cwnd growth after loss can be split into three phases: slowing growth that grows the cwnd very quickly at first, plateau and increasing growth. Figure 1 displays the various phases that CUBIC goes through. Additionally, the cwnd is only reset by 30% after a loss as compared to Reno which reduces it by 50%. This process makes CUBIC and similar loss-based algorithms such as TCP Reno predictable in terms of their sending behaviour.

 doi: 10.2313/NET-2025-05-1_17

However, CUBIC still does not solve the inherent problem of loss-based algorithms. These algorithms remain stuck in a loop: they probe for the maximum speed, then reduce it again after a loss. This often shows as a sawtooth-like pattern, although in CUBIC it is less pronounced thanks to the cubic function. Since packet loss in itself is only a binary indicator of congestion, it does not provide any information about how strongly the network is congested, which means that the algorithms have only the way of reducing the cwnd by a fixed factor.

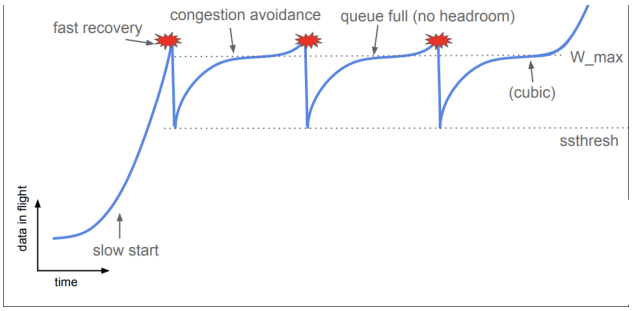


Figure 1: Function of data in flight over time in CUBIC [7].

The question why TCP CUBIC—or, more broadly, loss-based congestion control approaches—may no longer be sufficient, and what led to the development of BBR, arises. Loss-based algorithms and their way of growing the cwnd makes them inherently suffer from the following problems:

- With wireless networks becoming more and more common, packet loss can also be caused by interference or other factors unrelated to congestion.
- Network nodes with deep buffers can store a lot of data before they start dropping packet. Long queueing of packet leading to high RTT make it hard to adapt to congestion swiftly and can lead to a lot of data being sent at an excessive rate.
- Network nodes with shallow buffers might simply be overwhelmed by a short-lived burst of data and drop packets despite there being no congestion. Furthermore even just a slight overshoot of the maximum bandwidth can lead to packet loss and an adjustment of the cwnd by much more than would actually be required.

To solve the stated problems a new approach is needed where the BBR algorithm comes into play.

3. BBR

BBR was originally developed in 2017 and has since evolved over three versions. This section gives an overview of what BBR aims to solve as well as its implementation details and the differences between the versions.

3.0.1. Kleinrock's Optimal Operating Point. BBR tries to operate at a point that is close to the optimal operating point as described by Kleinrock [8]. Figure 2 shows the optimal operating point as a function of the bottleneck bandwidth and the RTT.

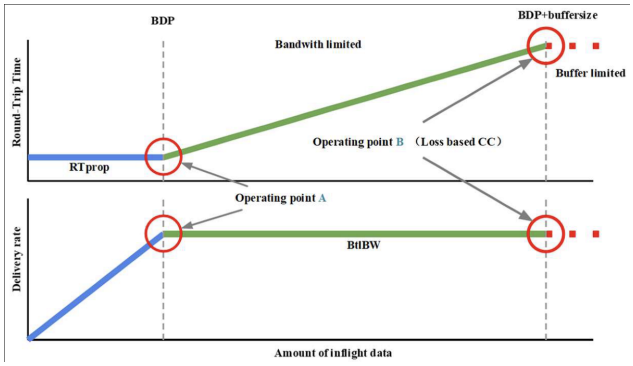


Figure 2: Impact of inflight data volume on RTT and delivery rate [9].

The two graphs display how the RTT and the delivery rate change in correlation to the amount of data in flight. It can be seen that, as the amount of data in flight increases, there is a limit to the increase in delivery rate. Furthermore, the round-trip time continues to rise even long after the delivery rate stopped increasing. This can be explained by the fact that, while the data volume in flight is below a certain threshold, there are no queues yet because the network is under-utilized and delivery-rate can still be increased by sending more packets. As the threshold is exceeded, however, the data in flight will exceed the bottleneck's capacity, which causes it to be queued in the nodes buffer resulting in a RTT increase. Since the bottleneck is already sending as fast as it can, the delivery rate will not increase anymore as newly arriving packets are either queued or dropped entirely. While loss-based algorithms tend to operate at point B as shown in Figure 2, BBR tries to operate at the point where the delivery rate is maximized and the RTT is minimized. This is where the bottleneck is fully utilized excessive forming of queues, Kleinrock's optimal operating point [1].

3.1. Goals

BBR departs from the loss-based approach and instead keeps an updated model of the network path to smoothly and continuously adapt its sending rate to the current network conditions. By doing so, it aims to be proactive in its approach rather than reactive, as loss-based algorithms are. Keeping continuous track of the network path solves the issue of only having packet loss as binary indicator of congestion. The goals of BBR can be summarized as follows:

- **Maximizing Throughput at Minimal Delay:** BBR aims to fully utilize the bottleneck link by sending at a rate close to the measured bottleneck bandwidth, while keeping the in-flight data near the bandwidth-delay product (**BDP**). It avoids persistent queue buildup (and the resultant bufferbloat) by doing so and thus maintains low latency for network traffic [1].
- **Proactive, Model-Based Control:** Instead of waiting for packet loss, BBR continuously monitors both bottleneck bandwidth and minimum RTT to build a near real-time model of the path. It adjusts its sending rate according to that model so

that it operates near Kleinrock's optimal operating point.

- **Improved Robustness to random loss:** By decoupling congestion control from packet loss, BBR is inherently more tolerant of random losses and does not over-correct when just a slight congestion, which could be solved by just a small reduction of the sending rate, occurs.

3.2. BBR State Machine Phases

BBR operates by creating a network path model using the RTT and the bandwidth with which it can calculate the BDP.

$$\text{BDP} = \text{Max. Bandwidth} \cdot \text{Min. RTT}$$

In order to find the minimal RTT one has to send at a rate that avoids the formation of queues so that no delay is introduced by the network. On the other hand, in order to find the maximum bandwidth one has to send at a rate that makes queues form as those are an indicator of congestion and therefore a nodes limit. These two measurements are crucial for BBR to operate at Kleinrock's optimal operating point and can only be measured individually as they are exclusive to each other. BBR does this by utilizing an internal state machine, see Figure 3, that switches between probing for exactly these two measurements [7]. During each of these phases BBR maintains a set of parameters that influence the sending rate [10]. Some of the most important parameters are:

- **pacing_rate**: Main parameter that controls the spacing between packets. It is updated on each received ACK and aims to match the bottlenecks rate instead of sending packets in bursts. In a perfect scenario, the pacing rate would be equal to the maximum bandwidth, but the distinction between pacing rate and sending rate is deliberate. The sending rate, i.e. actual throughput, can however fall below the pacing rate, for example if the sender is application-limited or the cwnd limit has been reached.
- **pacing_gain**: Multiplier for the pacing rate. Used to increase or decrease the rate for probing in various phases.
- **cwnd**: Limits the maximum inflight data volume. Bound by inflight_hi and inflight_lo.
- **cwnd_gain**: Multiplier for the cwnd. Used to increase or decrease the cwnd for probing in various phases.
- **inflight_hi**: New since BBRv2 inflight_hi is a long-term upper bound on inflight data based on past loss events. The congestion window is limited to this value.
- **inflight_lo**: New since BBRv2 inflight_lo is a short-term upper bound on inflight data in the probing current cycle.

3.2.1. Startup Phase. When establishing a new connection, BBR will start in the startup phase. This phase aims to estimate the bottleneck bandwidth very quickly by setting the pacing_gain and cwnd_gain to high values and therefore allowing the cwnd and pacing_rate to effectively double every round. Generally the maximum bandwidth is found in around $\mathcal{O}(\log_2(\text{BDP}))$ RTTs in version 3 [10].

This phase is exited as soon as packet loss has reached a certain threshold or the maximum bandwidth has been found as indicated by a plateau. In context of BBR a plateau is reached when the delivery rate has increased by less than 25% over the last three RTTs.

Life cycle phases	Property	BBRv1	BBRv2	BBRv3
Startup	cwnd_gain	$2/\ln 2$ (~ 2.89)	$2/\ln 2$ (~ 2.89)	2.00
	pacing_gain	$2/\ln 2$ (~ 2.89)	$2/\ln 2$ (~ 2.89)	2.77
	Max. cwnd	3xBDP	-	-
	inflight_hi	-	max.cwnd	max(est. BDP, last cwnd)
	Exit send.	rate <25% for 3 consec. RTTs	loss/ECN rate $\geq$ thresh. (8)	loss/ECN rate $\geq$ thresh. (6)
Drain	pacing_gain	-	0.35	-
	Exit	-	cwnd $\leq$ 1xBDP	-
ProbeBW	Phases	8 fixed gain cycles	{Cruise, Refill, Up, Down}	{Cruise, Refill, Up, Down}
	-	Cycle = RTprop	cwnd limits = {inflight_hi, inflight_lo}	-
	cwnd_gain	-	cwnd_gain(Up)=2.0	cwnd_gain(Up)=2.25
	pacing_gain	[1.25, 0.75, 1, 1, 1, 1, 1, 1]	pacing_gain(Down)=0.75	pacing_gain(Down)=0.90
	Exit	cwnd $\geq$ (pacing_gain · BDP) or loss	loss/ECN rate $\geq$ thresh.	loss/ECN rate $\geq$ thresh.
ProbeRTT	Frequency	10s	5s	5s
	cwnd	4	BDP/2	-
	Duration	200 ms + RTprop	-	-

TABLE 1: Evolution of BBR parameters over various versions [11].

This phase is exited as soon as packet loss has reached a certain threshold or the maximum bandwidth has been found as indicated by a plateau. In context of BBR a plateau is reached when the delivery rate has increased by less than 25% over the last three RTTs.

3.2.2. Drain Phase. During the startup, phase a queue of around 1 BDP [10] has been built up at the bottleneck. In order to remove this queue, the drain phase reduces the pacing gain which in turn reduces the sending rate. While any value below 0.5 should be able to drain the queue within ≤ 1 RTTs, BBRv3 uses a value of 0.35 [10]. Once the volume of data in-flight has been reduced to <= 1 BDP the drain phase is exited. If inflight_hi was set during startup, indicating that there was packet loss, the drain phase will empty the queue even further to provide some headroom for better coexistence with other flows.

3.2.3. Probe Bandwidth Phase. The probe bandwidth phase is a long-lived phase in which BBR operates most of the time. While originally this phase was only called ProbeBW it has been split into four sub-phases in BBRv2 between which the algorithm cycles: ProbeBW_DOWN, ProbeBW_CRUISE, ProbeBW_REFILL and ProbeBW_UP [12].

- **ProbeBW_DOWN**: Aims to drain the volume in flight to below 1 BDP with potentially additional headroom that can be used by other flows.
- **ProbeBW_CRUISE**: Sending rate is adjusted to the maximal available bandwidth with some margin **BBRPacingMarginPercent** [10]. If packet loss occurs bw_lo and inflight_lo are adjusted to reduce the sending rate as necessary. The phase is exited after a certain amount of packets depending on the environment to be more fair towards coexisting Reno and CUBIC flows [10].
- **ProbeBW_REFILL**: Aims to "refill the pipe" [10] to prepare for the next phase. Necessary to not underestimate the path by causing packet loss with a sudden burst of data.
- **ProbeBW_UP**: Probes for changes in maximum bandwidth by sending with an increased rate. It is exited after a delivery rate increase plateau has been found or packet loss exceeds a threshold.

3.2.4. Probe RTT Phase. ProbeRTT is entered after at most 5 seconds have passed since the last ProbeRTT phase and can be switched into from any other phase. The goal of this phase is to measure the minimal RTT. Because existing queues would introduce a delay and therefore

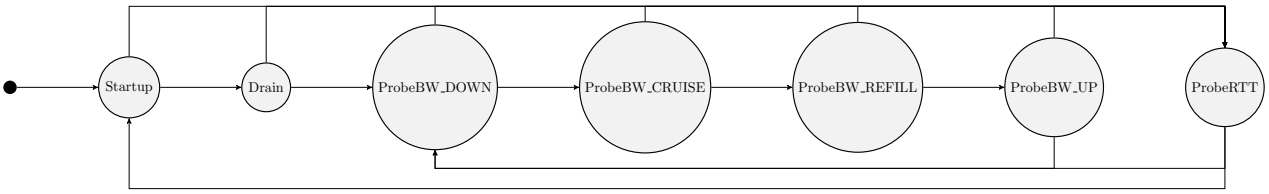


Figure 3: State transitition diagram for BBRv3 [10].

skew the measured minimal RTT they must be drained by sending a volume of data that is lower than the BDP. This is done by halving the cwnd_gain to lower inflight data volume before taking measurements [10].

3.3. Evolution of BBR

The most notable changes between versions are the incorporation of packet loss when determining the bottleneck bandwidth [11], [13] and the breakdown of the ProbeBW phase into four sub-phases in BBRv2 [12]. Piotrowska found in a study that BBRv1 caused excessive packet loss in some scenarios [12] which likely led to this decision. To further limit the inflight data volume the parameters inflight_hi and inflight_lo have been introduced. The former is used in BBR's long term path model while the latter is used in the short term model to limit the cwnd [10]. The long term model is supposed to be more stable and keep track of long term safe data rates and volumes, while the short term model is adjusted more often to adapt to current network conditions [10]. BBRv3's changes were much less radical than the ones introduced in v2. BBRv3 is mostly BBRv3 with the inclusion of fixes for bugs that affected fairness towards other flows [12], [14]. Furthermore parameters have been adjusted, pacing_gain during startup has been been lowered to 2.77 and cwnd_gain was drastically reduced to 2.00. A full overview of parameter changes can be found in Table 1.

3.4. Fairness

BBR is not the only congestion control algorithm in use and therefore has to compete with others for bandwidth. While only a couple of congestion control algorithms have been outlined in this paper, there are many more in use for various scenarios. All concurrent internet flows using various algorithms have to compete for their share of available bandwidth and therefore it is important that none takes away most of it while others are left with the bare minimum. This section briefly summarizes research about BBR's fairness towards itself as well as other congestion control algorithms.

3.4.1. Fairness towards BBR. Zeynali et al. have evaluted fairness between multiple BBR flows in various scenarios such as flows starting at the same point in time and flows being started at different times [11]. They have found that while for two same version BBR flows starting at the same time the fairness as described by Jain's Fairness Index [15] has indeed improved a little bit, it has worsened extremely for staggered flows. While two BBRv1 and BBRv2 flows being started with a gap of 15 s of each other converged to a similar bandwidth rather quickly, it took almost 5 minutes for the same to happen among two BBRv3 flows for a buffer with 16*BDP size. In total, while BBRv1 displays many more retransmissions due to its complete ignorance of packet loss as a signal for congestion, it ends up being the fairest when fighting for bandwidth among itself.

3.4.2. Fairness towards other congestion control algorithms. Gomez et al. have evaluated fairness between various versions of BBR among themselves as well as CUBIC [6]. They observed various scenarios such as two flows competing with each other without any loss, 100 flows (split up in 50 of each version) competing with each other and 100 flows (split up in 50 of each version) without loss, as well as the same scenarios with a loss of 0.025%. For 100 flows and a loss of 0.025% they have found that BBRv3 and CUBIC have similar throughput for higher buffer sizes. For smaller buffer sizes BBRv3 reaches a higher throughput than CUBIC. This effect is even worse for BBRv2 and CUBIC, where BBRv2 takes almost all the bandwidth in shallow buffer settings. All in all their results show that better resource sharing and therefore fairness is achieved with bigger buffer sizes the more flows compete against each other. However, even for shallow buffers Piotrowska et al. found that BBRv3 does indeed enhance "fairness by 12%" towards CUBIC [12]. On the other hand Zeynali et al. found that even multiple CUBIC flows cannot compete against a single BBR flow in terms of Jain's Fairness Index and "end up competing between themselves for the bandwidth leftover[...]" [13], [15].

4. Conclusion and future work

In this paper, we presented the past and current state of Google's BBR algorithm. BBR, according to Google, shows much promise in terms of operating at near BDP and providing high bandwidth. They state that "Playbacks using BBR show significant improvement in all of YouTube's quality-of-experience metrics" [1] and "BBR reduces median RTT by 53 percent on average globally and by more than 80 percent in the developing world." [1]. BBRv3 has become the default congestion control algorithm for traffic in Google's services [13] and now accounts for at least more than 40% of the internet's total traffic volume according to Mishra et al [16]. It is consistently developed and improved upon and - given its parametrized nature and usage of a state machine - it is easy to imagine that more states could be added in the future to further improve the algorithm or better adapt to specific scenarios. Although BBRv3 brings

notable improvements over earlier versions, research has revealed that it can still be highly unfair towards other BBR connections. Therefore it is critical that further refinements are needed to enhance fairness not only when BBRv3 coexists with other congestion control algorithms, but especially when several BBRv3 flows share the same bottleneck, especially if BBRv3 is to become the default congestion control algorithm of the future. As of March 2024 however, BBRv3 was not yet included in Linux's mainline TCP, for which a submission was planned as soon as possible [17]. Looking at the latest meetings of the Congestion Control Working Group (**ccwg**) it becomes clear that development of BBR is in full force with one of the most recent topics that are being discussed being making improvements to BBR for real-time connections [18].

References

[1] 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, 2017.

[2] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM computer communication review*, vol. 18, no. 4, pp. 314–329, 1988.

[3] J. Sharma and A. K. Garg, "Analysis of tahoe: A tcp variant," *International Journal of Engineering and Advanced Technology (IJEAT) ISSN*, pp. 2249–8958.

[4] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[5] R. Al-Saadi, G. Armitage, J. But, and P. Branch, "A survey of delay-based and hybrid tcp congestion control algorithms," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3609–3638, 2019.

[6] J. Gomez, E. F. Kfoury, J. Crichigno, and G. Srivastava, "Evaluating tcp bbrv3 performance in wired broadband networks," *Computer Communications*, vol. 222, pp. 198–208, 2024.

[7] N. Cardwell, Y. Cheng, K. Yang, D. Morley, S. Hassas, P. Jha, and Y. Seung, "Bbr congestion control: Fundamentals and updates," 2023.

[8] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in *ICC 1979; International Conference on Communications, Volume 3*, vol. 3, 1979, pp. 43–1.

[9] S. Yang, Y. Tang, W. Pan, H. Wang, D. Rong, and Z. Zhang, "Optimization of bbr congestion control algorithm based on pacing gain model," *Sensors*, vol. 23, no. 9, p. 4431, 2023.

[10] N. Cardwell, I. Swett, and J. Beshay, "BBR Congestion Control," Internet Engineering Task Force, Internet-Draft draft-ietf-ccwg-bbr-01, Oct. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-ccwg-bbr/01/

[11] D. Zeynali, E. N. Weyulu, S. Fathalli, B. Chandrasekaran, and A. Feldmann, "Promises and potential of bbrv3," in *International Conference on Passive and Active Network Measurement*. Springer, 2024, pp. 249–272.

[12] A. Piotrowska, "Performance evaluation of tcp bbrv3 in networks with multiple round trip times," *Applied Sciences*, vol. 14, no. 12, p. 5053, 2024.

[13] D. Zeynali, E. N. Weyulu, S. Fathalli, B. Chandrasekaran, and A. Feldmann, "Bbrv3 in the public internet: a boon or a bane?" in *Proceedings of the 2024 Applied Networking Research Workshop*, 2024, pp. 97–99.

[14] N. Cardwell, Y. Cheng, K. Yang, D. Morley, S. Hassas, P. Jha, Y. Seung, V. Jacobson, I. Swett, B. Wu *et al.*, "Bbrv3: algorithm bug fixes and public internet deployment," *Presentation in CCWG at IETF*, vol. 117, 2023.

[15] R. K. Jain, D.-M. W. Chiu, W. R. Hawe *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, vol. 21, p. 1, 1984.

[16] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The great internet tcp congestion control census," in *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*, 2020, pp. 59–60.

[17] N. Cardwell, Y. Cheng, K. Yang, D. Morley, S. Hassas Yeganeh, P. Jha, Y. Seung, and V. Jacobson, "Bbrv3: Algorithm overview and google's public internet deployment," IETF 119 Meeting Materials, March 2024, presented at IETF 119, CCWG meeting, Brisbane. [Online]. Available: https://datatracker.ietf.org/meeting/119/materials/slides-119-ccwg-bbrv3-overview-and-google-deployment-00

[18] C. Huitema, S. Nandakumar, and C. Jennings, "Bbr improvements for real-time connections," IETF 120 Meeting Materials, July 2024, presented at IETF 120, CCWG meeting, Vancouver. [Online]. Available: https://datatracker.ietf.org/meeting/120/materials/slides-120-ccwg-bbr-improvements-for-real-time-connections-00.pdf

Reliable Broadcast Networking Stacks

Mariya Koeva, Filip Rezabek*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: mariya.koeva@tum.de, frezabek@net.in.tum.de

***Abstract*—This paper investigates the design, analysis, and potential of robust and reliable broadcast protocols in distributed systems, addressing challenges like fault tolerance, latency, and communication efficiency in both stand-alone and simulation-based frameworks. Existing broadcast protocols are examined, with a focus on evaluating their scalability and resilience under failure conditions. We explore the limitations of traditional transport protocols like TCP and introduce modern alternatives such as QUIC, which offer enhanced performance in high-latency, high-concurrency environments. QUIC's built-in encryption, faster connection setup, and improved multiplexing make it a promising alternative for reliable broadcast communication. The study provides a comparative analysis of TCP and QUIC, highlighting their strengths, weaknesses, and use cases. Additionally, it examines the role of advanced cryptographic techniques like threshold cryptography and Distributed Key Generation (DKG) and discusses open-source implementations, for instance, libp2p. The findings offer valuable insights for future research in the deployment of the QUIC protocol in reliable broadcasts.**

***Index Terms*—reliable broadcast, distributed systems, broadcast protocols, fault tolerance, communication efficiency, scalability, TCP, QUIC, TLS, consensus algorithm, threshold cryptography, distributed key generation (DKG), libp2p**

1. Introduction

Reliable broadcast is a fundamental primitive in distributed systems, ensuring that messages are consistently delivered to all intended recipients, even under adverse conditions such as network failures or malicious attacks [1]. These protocols underpin critical applications like blockchain networks, distributed databases, and fault-tolerant systems. Three primary requirements must be met for reliability: **validity, agreement and integrity** [2]. Validity ensures that all correct nodes eventually deliver a message if the sender is correct; agreement guarantees that all correct nodes deliver the same message; and integrity prevents tampering or duplication of messages. These requirements were first formalized in 1983 [3] and remain foundational in distributed systems research.

Traditional transport protocols like TCP have been widely adopted due to their robustness and well-understood reliability mechanisms. Combined with TLS (Transport Layer Security) for encryption, TCP has long been the standard for secure communication. However, these solutions face challenges in scalability and efficiency, particularly in high-latency environments or under heavy network congestion. To address these limitations, modern alternatives like QUIC [4] have emerged. QUIC incorporates built-in encryption, multiplexing, and faster connection setups, presenting a promising alternative for reliable communication, including broadcast scenarios.

This paper explores the current state of reliable broadcast solutions, analyzing existing algorithms and used protocols with a focus on evaluating the feasibility of QUIC as a substitute for TCP. This is achieved through a comparative analysis of these transport protocols, their advantages and disadvantages, and use cases involving advanced cryptographic techniques such as threshold cryptography and Distributed Key Generation (DKG). Furthermore, we investigate open-source implementations, including libp2p, and discuss the possible future developments in reliable broadcasting.

2. Background, Related Work and Existing Broadcast Solutions

Ensuring reliable broadcast in distributed systems is a fundamental challenge, especially in environments where communication may be unreliable, and nodes may fail. To address this, various consensus mechanisms and synchronization models have been developed to guarantee message delivery and agreement among nodes. This section explores the principles behind reliable broadcast and consensus, highlighting key algorithms, and examines the role of different synchronization models in achieving reliable communication.

2.1. Consensus Algorithms

Consensus algorithms are fundamental to ensuring consistency in distributed systems, especially when dealing with unreliable communication and potential faults. These algorithms enable a set of processes or nodes to agree on a single value, even in the presence of failures, ensuring reliable broadcast of messages. A few notable consensus algorithms are widely used to achieve this reliability:

2.1.1. Paxos. Paxos is a foundational consensus algorithm used in partially synchronous to synchronous systems. Paxos ensures that a majority of nodes reach agreement on a single value despite faults (e.g, node crashes). It operates under the assumption that at least one correct process is not faulty, and it guarantees safety and liveness. Safety in this context means that no two nodes will

 doi: 10.2313/NET-2025-05-1_18

decide on different values and liveness means that a decision will eventually be made under specific conditions. While Paxos is a breakthrough in consensus protocols, its implementation is viewed as complex and inefficient in terms of performance due to frequent communication and coordination overheads [5].

2.1.2. Raft. Raft was designed as a more understandable alternative to Paxos, focusing on clarity and ease of implementation. Raft divides the consensus process into three key components: leader election, log replication, and safety. A leader is elected to manage the consensus process, and the leader's log is replicated across follower nodes. Raft ensures that all changes to the system are coordinated through the leader, simplifying the process and improving efficiency in comparison to Paxos. [6].

2.1.3. Byzantine Fault Tolerant (BFT) Algorithms. In more adversarial environments, where nodes may behave arbitrarily, BFT algorithms are used. For example, Practical Byzantine Fault Tolerance (PBFT) is designed to tolerate up to one-third of the nodes being compromised by a Byzantine adversary. The algorithm works by having nodes exchange messages in multiple rounds to achieve agreement on a transaction, even in the presence of faulty or malicious participants. [7].

2.2. Sybil resistance approaches

While Consensus algorithms determine how nodes agree on a single state or value in the system, sybil resistance determines who gets to participate in the network and ensures that participation is fair, preventing adversaries from gaining excessive influence. A Sybil attack occurs when an adversary controls a large number of nodes, allowing them to disrupt the network by censoring messages, invalidating transactions, or influencing consensus outcomes. In the context of reliable broadcast, Sybil resistance ensures that an adversary cannot gain disproportionate control over message dissemination.

Decentralized systems implement Sybil-resistant mechanisms to counterfeit sybil attacks by imposing economic or computational barriers to prevent adversaries from cheaply creating multiple identities. Proof of Stake (PoS): Forces participants to stake cryptocurrency as collateral, ensuring economic penalties for dishonest behavior.

Sybil resistance mechanisms often incorporate consensus techniques to validate participation. For example, PoS uses consensus among staked participants to finalize blocks while preventing Sybil attacks by requiring economic commitment. Sybil resistance is a prerequisite for secure consensus in decentralized networks. Without it, consensus protocols become vulnerable to 51% attacks (in PoW) or stake concentration attacks (in PoS), where an attacker gains the majority influence and disrupts agreement.

Moreover, the efficiency and scalability of consensus mechanisms are influenced by the choice of Sybil resistance methods. PoW offers strong Sybil resistance but suffers from high energy consumption, whereas PoS reduces energy usage but introduces new challenges, such as stake centralization risks. Some systems combine multiple mechanisms, such as PoS with reputation-based Sybil resistance, to balance security, efficiency, and decentralization.

2.2.1. Proof of Work (PoW). In this approach, miners compete to solve complex mathematical puzzles using computational power. The first to succeed earns the right to create a new block and receive rewards. PoW ensures security by making mining computationally expensive, deterring Sybil attacks and fraudulent transactions. An attacker would need to control more than 51% of the network's total computational power to alter transactions, which is highly costly and impractical. However, PoW's high energy consumption raises questions regarding its scalability and environmental impact. Despite this, it remains widely used in networks like Bitcoin, Litecoin, and Monero, where strong security and decentralization are prioritized [8].

2.2.2. Proof of Stake (PoS). PoS offers a more energy-efficient alternative by selecting validators based on the number of coins they have staked as collateral rather than computational work. This system ensures security by making dishonest behavior costly, as malicious validators risk losing their staked assets [9]. Unlike PoW, PoS significantly reduces energy consumption while maintaining Sybil resistance, as attackers would need to control a majority of the staked assets to manipulate the network. It also supports faster transaction finality, making it more scalable for high-volume applications. PoS is widely adopted in blockchain platforms like Ethereum 2.0.

2.3. Synchronization Models in Reliable Broadcasting

In the context of reliable broadcasting, the synchronization model of a system plays a crucial role in determining the guarantees provided by the underlying communication protocols. These synchronous, partially synchronous, and asynchronous models define the expectations regarding timing and network delays, and they influence the design and robustness of reliable broadcast mechanisms.

Synchronous systems operate under strict timing assumptions, where a known finite upper bound (Δ) ensures that messages are delivered within a specified timeframe [10]. This model simplifies protocol design by providing predictable communication, making it suitable for applications requiring high reliability and determinism. Examples include high-frequency trading systems and real-time control networks. However, the reliance on accurate Δ value presents challenges: a conservatively large Δ may degrade performance due to long timeouts, while a small Δ risks safety violations in real-world conditions.

The partially synchronous model bridges the gap between synchrony and complete asynchrony. It assumes an unknown finite upper bound (Δ) on message delays, which holds only after a Global Stabilization Time (GST) [11]. Before GST, the system behaves asynchronously. This model is highly applicable in real-world distributed systems where networks experience transient disruptions but eventually stabilize.

 doi: 10.2313/NET-2025-05-1_18

Asynchronous systems place no bounds on message delivery times, making them highly flexible but challenging for achieving consensus. The lack of timing guarantees means that protocols must ensure eventual message delivery and consensus without relying on temporal assumptions. This model is essential for environments with unpredictable delays, such as highly decentralized peer-to-peer networks.

3. Comparative Analysis: TCP vs. QUIC for Reliable Broadcast

Reliable broadcast depends heavily on the underlying transport protocol, which influences factors such as latency, security, and message delivery efficiency. This section explores the strengths and weaknesses of TCP and QUIC protocols and compares them.

3.1. TCP as the traditional transport layer

TCP (Transmission Control Protocol) [12] is the cornerstone of reliable data transmission in modern networking. It offers reliability through mechanisms such as retransmissions, in-order delivery, and congestion control. These guarantees make TCP essential for applications like file transfers, web browsing, and distributed databases like Google Spanner. TCP's acknowledgment system ensures data integrity, while its flow control mechanisms [13] aim to prevent network congestion.

3.1.1. Challenges of TCP. TCP suffers from inherent latency due to its three-way handshake and sequential acknowledgment system. The Round-Trip Time (RTT) for a TCP connection establishment is at least one RTT for the handshake, and additional RTTs are incurred for data transfer, especially with larger datasets. Moreover, the sequential acknowledgment of packets introduces head-of-line blocking, where a single lost packet can delay the entire stream, a significant drawback for time-sensitive applications [14].

The inefficiency of TCP becomes more pronounced in large-scale systems, particularly where high throughput and low-latency communication are required, such as in real-time media streaming or large distributed systems.

3.1.2. Deployment in Real-World Broadcast Systems. TCP is integral to traditional, reliability-sensitive systems like distributed databases (e.g., Apache Cassandra, MongoDB) where data consistency and integrity are critical. The established ecosystem around TCP and different versions of TLS guarantees secure and reliable connections in transactional systems. However, the latency and lack of multiplexing are limitations in dynamic, distributed environments, where faster connection establishment and handling of multiple streams are required [15].

3.2. QUIC as an Alternative

QUIC is a modern transport protocol designed by Google to address some of the TCP limitations, especially in terms of connection setup time, security, and multiplexing. Key Features of QUIC are:

Built-in Encryption: QUIC integrates Transport Layer Security (TLS) 1.3 directly into the protocol, eliminating the need for a separate security layer as required by protocols like HTTP/2 over TLS.

Reduced handshake: This built-in encryption also reduces connection setup time by consolidating the transport(previously TCP) and security(TLS) handshakes into a single phase. QUIC establishes secure connections more efficiently compared to TCP, as the TLS handshake occurs alongside the initial connection establishment, reducing latency significantly [4].

Fast Connection Establishment: QUIC also supports 0-RTT (zero round-trip time) connection establishment, allowing data to be sent in the initial packet even before the handshake is complete. This is particularly beneficial in environments where latency is a concern, such as high-frequency trading or live-streaming applications. [4]

Multiplexing: QUIC improves the efficiency of multiplexing by allowing multiple independent data streams within a single connection [4]. Unlike TCP, where head-of-line blocking can occur, QUIC allows independent streams to proceed without interference. This is particularly important in modern applications where multiple data types are transmitted simultaneously.

3.2.1. Potential Advantages in Reliable Broadcasting. QUIC's design introduces new possibilities for use in reliable broadcasting scenarios, particularly in distributed systems where rapid and fault-tolerant message delivery is crucial.

QUIC's low-latency connection setup and inherent support for multiplexing make it suitable for distributed systems requiring rapid message dissemination. This, combined with its resistance to head-of-line blocking, ensures that broadcast messages can be delivered quickly and efficiently, even in environments with unreliable network conditions, where network changes are a possibility. This is due to the fact that QUIC relies on a Connection ID to identify connections, and not on the IP Addresses and Sockets of the participants.

3.2.2. Limitations and Open Questions. Despite its many advantages, QUIC presents certain challenges that may affect its adoption in all contexts.

One significant challenge to QUIC's adoption is its limited compatibility with legacy systems, particularly those that rely on TCP-based communication stacks. QUIC requires modern infrastructure and support for UDP (User Datagram Protocol), making it less suitable for environments where legacy protocols are deeply integrated.

4. Analysis and open-source implementations

In this section we explain existing implementation approaches and make assumptions about possible enhancements through future work.

4.1. Implementations focusing on TCP-based solutions

One notable example where TCP outperforms QUIC in reliable broadcast scenarios is its role in PBFT consensus. A study evaluating the performance of QUIC in

PBFT-based blockchain networks found that TCP achieves better execution times due to its optimized message replication and congestion control mechanisms [16]. PBFT relies on frequent and structured message exchanges across multiple communication rounds, requiring consistent, in-order delivery and efficient retransmission of lost packets. TCP, with its persistent connections and built-in reliability mechanisms, efficiently handles this high message volume, ensuring that consensus operations proceed with minimal delays. In contrast, QUIC must implement reliability and congestion control at the transport layer, leading to additional processing overhead. QUIC reduces connection establishment latency and supports multiplexing, but the study showed that these benefits do not improve PBFT execution times. QUIC's congestion control mechanisms, such as BBR and New Reno, don't consistently outperform TCP due to their handling of packet loss and congestion feedback in high-frequency message replication scenarios. The simulation results indicate that TCP remains the preferred choice for PBFT and similar consensus protocols, where timely, ordered message delivery and efficient network congestion handling are crucial. However, the study suggests that QUIC may become more viable with further optimizations in congestion control for large-scale distributed systems. Protocols designed for partial synchrony, such as Paxos or Raft [17], can be implemented over TCP/TLS, especially when message delivery times are uncertain but the system can eventually stabilize. These protocols ensure that once a system reaches GST, reliable communication can be achieved, with TCP/TLS offering secure and reliable transport. However, when low-latency and high-throughput are required, QUIC may be favored.

4.2. QUIC use cases

Asynchronous Byzantine Fault Tolerant (ABFT) algorithm is an example of an asynchronous protocol that can be implemented over QUIC in cases where low latency is critical, especially in decentralized or mobile networks. QUIC offers distinct advantages over TCP in asynchronous systems due to its reduced handshake overhead, which allows for faster message exchanges. However, those are mainly assumptions that haven't been consolidated by by research.

In asynchronous environments, TCP can still be used, but it is typically less efficient compared to QUIC due to the higher latency introduced by TCP's handshake and slower connection re-establishment after packet loss. TCP and TLS are more suitable for environments with moderate to low network instability, while QUIC performs better in volatile networks(e.g, mobile environments or global-scale decentralized systems).

4.2.1. Potential of QUIC for Reliable Broadcast in Decentralized Transactions. QUIC has shown promise in decentralized transaction systems, particularly in peer-to-peer Bitcoin transactions and payment channels. Studies highlight that QUIC's low-latency handshake, built-in encryption, and multiplexing make it well-suited for fast and secure financial transactions in blockchain networks. QUIC Bitcoin Transactions leverage QUIC's Connection ID mechanism to enable direct, unpublished transaction exchanges between peers, reducing reliance on intermediaries and mitigating security risks such as man-in-the-middle (MITM) attacks [18]. Additionally, QUIC Bitcoin Channels introduce efficient payment channels that maintain state even during network switches, allowing seamless transactions between users and machines. These capabilities suggest that QUIC could be a valuable alternative to TCP for high-speed, trustless financial exchanges, particularly in use cases requiring lightweight, adaptable, and encrypted communication channels in decentralized networks.

4.2.2. Threshold Cryptography and Distributed Key Generation (DKG). Threshold cryptography, particularly Distributed Key Generation (DKG), plays a fundamental role in decentralized security mechanisms. As outlined in Das and Ren's work [19], DKG ensures that signing keys are securely distributed among multiple participants rather than being held by a single entity. Their scheme employs adaptively secure BLS threshold signatures, which maintain security even if an adversary selects corrupted nodes dynamically. The protocol minimizes overhead compared to prior DKG approaches, ensuring efficient key management in distributed environments. This decentralized key-sharing mechanism is particularly relevant for secure broadcast systems, where maintaining integrity and fault tolerance in adversarial conditions is crucial.

QUIC, on the other hand, integrates cryptographic handshakes with transport-layer encryption, minimizing round-trip times and ensuring secure data transmission [4]. While QUIC itself does not inherently implement DKG, its built-in encryption and low-latency communication may offer advantages in scenarios where distributed key agreement protocols like DKG are deployed. Specifically, QUIC's multiplexed streams and rapid reconnection capabilities could potentially reduce the overhead associated with key exchange and cryptographic signing in decentralized networks. However, no existing research—including that of Das and Ren—has explicitly evaluated QUIC's interaction with DKG. Further studies would be required to determine whether QUIC's performance benefits align with the security guarantees of threshold cryptographic protocols.

4.2.3. DAG-Based Consensus Mechanisms:. QUIC's low-latency and efficient communication protocols align well with Directed Acyclic Graph (DAG) structures used in systems like IOTA [20]. DAGs enable parallel transaction processing without the bottleneck of sequential blocks typical of traditional blockchain systems. QUIC's quick message propagation and connection setup are beneficial for the high-throughput and low-latency requirements of DAG-based consensus, where real-time message delivery is essential for scalability and network efficiency.

4.2.4. libp2p. In libp2p, reliable broadcast is essential for efficient message delivery across peers. TCP and QUIC—play key roles in this functionality.

TCP's head-of-line blocking can be a significant drawback in real-time applications. Additionally, TCP's reliance on middleboxes for header inspection can create challenges for deploying new protocol features [21]. QUIC avoids head-of-line blocking which makes it ideal

for real-time and high-concurrency environments. WebTransport, built on QUIC, offers an alternative to WebSockets by enabling bidirectional communication over stream multiplexing [22]. Unlike WebSockets, which use a single connection, WebTransport allows multiple streams to operate in parallel, improving performance. It also enables browsers to connect to libp2p nodes securely using self-signed certificates, addressing WebSocket's limitations in peer-to-peer networks. A standard WebSocket connection is conducted through 6 RTTs:

- 1 RTT for TCP handshake.
- 1 RTT for TLS 1.3 handshake.
- 1 RTT for WebSocket upgrade.
- 1 RTT for multistream security negotiation (Noise or TLS 1.3).
- 1 RTT for security handshake (Noise or TLS 1.3).
- 1 RTT for multistream muxer negotiation (mplex or yamux).

In comparison, WebTransport only requires 3 RTTs:

- 1 RTT for QUIC handshake.
- 1 RTT for WebTransport handshake.
- 1 RTT for libp2p handshake; one for multistream and one for authentication(with a Noise handshake) [22].

5. Conclusion and future work

In conclusion, while the exploration of TCP and QUIC for reliable broadcast in distributed systems has provided valuable insights, it is clear that further research is needed, particularly regarding QUIC's evolving capabilities and potential for broader adoption. QUIC's inherent advantages in latency reduction, multiplexing, and built-in encryption make it an increasingly promising candidate for high-performance applications, especially in real-time communication, streaming, and large-scale decentralized systems. QUIC is still undergoing development and its full potential in diverse distributed environments remains an area for continued investigation. QUIC continues to mature and gain support across the industry. In my opinion it is expected that QUIC's deployment will expand, offering new opportunities for optimizing reliable broadcasting in environments where speed, scalability, and fault tolerance are critical. Future research should focus on refining QUIC's interoperability with legacy systems, exploring hybrid solutions that combine the strengths of both TCP and QUIC, and further integrating advanced cryptographic techniques to enhance the security and efficiency of reliable broadcast protocols. As the adoption of QUIC accelerates, it is anticipated that it will play an increasingly integral role in shaping the next generation of communication protocols, driving innovation in distributed systems and beyond.

References

[1] H. Zhu, F. Bao, and R. H. Deng, "Robust and reliable broadcast protocols in the stand-alone and simulation-based frameworks," *2008 IEEE International Conference on Communications*, pp. 1635–1641, 2008.

[2] A. Kshemkalyani and M. Singhal, "Chapter 14: Consensus and Agreement," pp. 510–566, 2012.

[3] D. Dolev and H. R. Strong, "Authenticated algorithms for byzantine agreement," https://www2.imm.dtu.dk/courses/02220/2015/L12/DolevStrong83.pdf, 1983.

[4] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[5] "Paxos consensus algorithm," 2024. [Online]. Available: https://www.geeksforgeeks.org/paxos-consensus-algorithm/

[6] S. Aggarwa, "Raft and paxos : Consensus algorithms for distributed systems," Aug. 2023. [Online]. Available: https://medium.com/@mani.saksham12/raft-and-paxos-consensus-algorithms-for-distributed-systems-138cd7c2d35a

[7] R. Behnke, "What is practical byzantine fault tolerance in blockchain?" Oct. 2023. [Online]. Available: https://www.halborn.com/blog/post/what-is-practical-byzantine-fault-tolerance-in-blockchain

[8] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[9] "Understanding sybil attacks and consensus mechanisms in blockchain," 2023. [Online]. Available: https://www.nervos.org/knowledge-base/sybil_attacks_consensus_mechanisms_(explainCKBot)

[10] I. Abraham, "Synchrony, asynchrony and partial synchrony," Jun. 2019. [Online]. Available: https://decentralizedthoughts.github.io/2019-06-01-2019-5-31-models/

[11] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the Association for Computing Machinery*, vol. 35, no. 2, pp. 288–323, Apr. 1988. [Online]. Available: https://groups.csail.mit.edu/tds/papers/Lynch/jacm88.pdf

[12] "Transmission Control Protocol," RFC 793, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc793

[13] E. Blanton, D. V. Paxson, and M. Allman, "TCP Congestion Control," RFC 5681, Sep. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5681

[14] A. Varshney, "Head-of-line (hol) blocking in http/1 and http/2," Jun. 2021. [Online]. Available: https://engineering.cred.club/head-of-line-hol-blocking-in-http-1-and-http-2-50b24e9e3372

[15] "Latency in distributed system," Aug. 2024. [Online]. Available: https://www.geeksforgeeks.org/latency-in-distributed-system/

[16] T. L. Habibi and R. F. Sari, "Performance evaluation of quic protocol in message replication overhead in pbft consensus using ns-3," *International Journal of Electrical, Computer, and Biomedical Engineering*, 2023.

[17] P. Sapkota, "Consensus algorithms: Paxos and raft," Sep. 2023. [Online]. Available: https://pragyasapkota.medium.com/consensus-algorithms-paxos-and-raft-1b3af91bed80

[18] A. Pagani, "Quic bitcoin: Fast and secure peer-to-peer payments and payment channels," 10 2022, pp. 578–584.

[19] S. Das and L. Ren, "Adaptively secure BLS threshold signatures from DDH and co-CDH," Cryptology ePrint Archive, Paper 2023/1553, 2023. [Online]. Available: https://eprint.iacr.org/2023/1553

[20] "An obvious choice: Why dags over blockchains?" 2023. [Online]. Available: https://blog.iota.org/dags-over-blockchains-iota20/

[21] "Quic." [Online]. Available: https://docs.libp2p.io/concepts/transports/quic/

[22] "Webtransport." [Online]. Available: https://docs.libp2p.io/concepts/transports/webtransport/

An Architectural Analysis of Cloudflare's QUIC Implementation quiche

Sebastian Langner, Daniel Petri*, Marcel Kempf*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: sebastian.langner@tum.de, petriroc@net.in.tum.de, kempfm@net.in.tum.de

***Abstract*—Technical implementations of the QUIC protocol differ significantly despite its in-depth `RFC 9000` specification. The design choice of an appropriate QUIC library is crucial for the available functionality and performance of a launching project. This paper analyzes the application and internal architecture of Cloudflare's QUIC implementation quiche using the C4 model, its relation to the host application, and quiche-specific features not explicitly stated in the `RFC`. Our findings are subsequently added to the QUIC Explorer, a tool for developers that facilitates this decision.**

***Index Terms*—QUIC, QUIC explorer, quiche, C4 model, architecture, software, congestion control, stream prioritization, BoringSSL**

1. Introduction

Since the Internet became increasingly important in our daily lives, TCP has been one of the most used protocols for data transmission, but it faces challenges regarding throughput, latency, and security. In 2016, the first Internet-Draft of Quick UDP Internet Connections (QUIC) was published by Google's developer team on the IETF forum [1]. The main goal of QUIC was to merge the advantages of TCP and UDP in one protocol with a focus on security, reliability and performance. In 2021, after several iterations and improvements, the IETF published the `RFC 9000` that defines the QUIC protocol [2]. Cloudflare had already released version 0.1.0 of their QUIC implementation called quiche on GitHub in October 2019 [3]. Since then, the library has been continuously developed and improved.

According to Alessandro Ghedini, one of quiche's leading developers, a major design goal was providing most of QUIC's functionality to the host application via a minimal and intuitive Application Programming Interface (API). However, quiche's application areas should not be restricted by any assumptions taken during the development process [3].

In this paper, we will analyze quiche's architecture and internal relations from an abstract point of view. Mainly, two abstraction levels (container & component) of the C4 model by Simon Brown will be considered [4]. We will deliberately not analyze and discuss the code level, as it goes beyond our scope and could be outdated shortly due to quick development. In Section 2, the communication between quiche, its environment, and inner elements will be analyzed. Section 3 discusses features of quiche not strictly defined in the `RFC 9000`. Furthermore, to enable easy access to our analysis results, we add them to the `QUIC Explorer` [5]. This information pool is designed for developer teams to gain a comprehensive overview of the different capabilities of different QUIC implementations and faciliate their design choices.

2. C4 Model-Based Analysis

To strike a balance between an abstract view of quiche's integration into a software project and a detailed analysis of its internal structure, we chose the C4 model to visualize and describe the architectural elements of the library. Unlike the prevalent Unified Modeling Language (UML), the C4 model is less formal and more lightweight, which makes it easier to understand, especially for people without a deep knowledge of software architecture. Nevertheless, it can represent a system as complex as the architecture of quiche in a simple but precise way [6]. In total, the C4 model consists of four abstraction levels: system context, container, component, and code. Each level focuses on a certain degree of abstraction to address different stakeholders [7, p. 920].

Since we want to focus on a structural analysis of quiche, we start in Section 2.1 at the container level to point out the interaction of the library with the software environment. Afterwards, we will dive deeper into the component level (Section 2.2) to analyze the library's architecture. To clarify the terms *container* and *component* in the context of the C4 model, we consider a component as a cohesive set of functionalities that is not separately deployable. It functions as a facade with a well-defined interface implemented by its underlying elements (e.g., structs, classes, instances) one level deeper. In contrast, a container is composed of several logically separate components that operate as a single deployable unit [4].

2.1. Container Level

quiche is designed as a multifunctional user-space library that enables rapid, iterative development and the possibility to be integrated into various services of different purposes [8]. Cloudflare developed quiche for their application in their Content Delivery Network (CDN) backbone, where it needs to be highly performant and reliable [3], [9]. At the same time, since its initial release, quiche has been an open-source project to be co-developed by the community and integrated into a wide range of applications outside of Cloudflare. As an example, Google integrated quiche in Android for secure and fast DNS

 doi: 10.2313/NET-2025-05-1_19

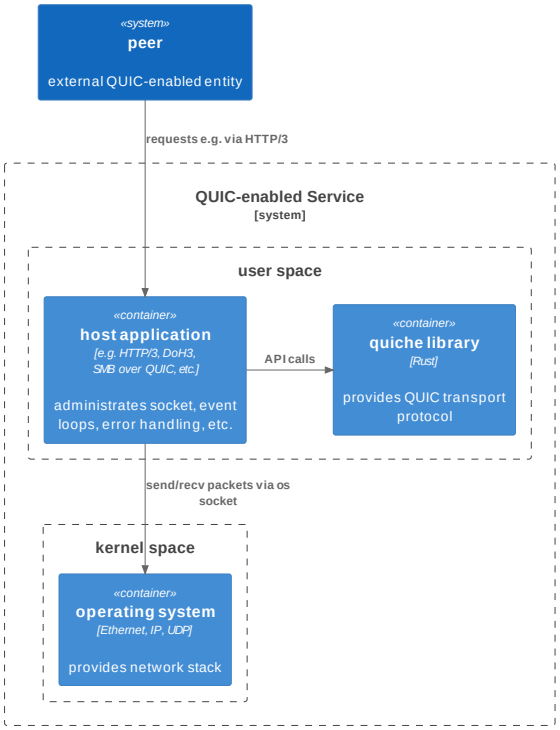


Figure 1: Software Architecture of a QUIC-enabled Service at the Container Level

resolution via DNS-over-HTTP/3 (DoH3) [10]. This accentuates the flexibility and adaptability of quiche for different architectures and use cases.

Figure 1 illustrates the container level of an arbitrary QUIC-enabled service that uses quiche as its network communication library. The core element is the host application, which implements the main logic and functionality of the software. Since quiche is designed as a low-level, rather passive library, the host application is responsible for the entire network processing. quiche itself implements the entire logic and algorithmic of QUIC according to the `RFC 9000` [2] and administrates the current connection state but does not proceed autonomously. As shown, quiche never interacts directly with the OS' network stack. It instead serves as a simultaneous interpreter and decision maker for the QUIC communication. The enforcement of the library's ruling is part of the host application's accountability based on the provided metadata. To accomplish connection establishment (handshake), sending/receiving data, handling packet recovery, etc. quiche offers a well-defined API. Nevertheless, every action must be triggered, and every active part, like I/O events, polling mechanisms, timeout handling, etc. must be implemented by the host application.

2.2. Component Level

To understand how quiche was designed and works internally, we will go one level deeper into the C4 model and analyze its structure on the component level. Figure 2 clarifies the general calling behaviour. For reasons of intelligibility, not all information flows are shown but are described in this Section.

The core element of quiche is the `Connection` component. It provides the central part of the API for the host application and processes or delegates, respectively, incoming communication (control and data) to the corresponding subcomponents. Moreover, this component manages the entire internal communication and the flow of information within the library. The connection state and connection-specific properties (e.g. TLS handshake parameters, transport statistics, IDs, timers) are stored, updated and accessible via this component.

Before a connection can be established, the host application must create an instance of the `Config` component. It stores global, non-connection-specific (but often device- or application-specific) properties and settings (e.g. security certificates and settings, supported protocols and algorithms, initial and maximum values). Therefore, the same `Config` instance can initialize multiple connections with the same settings. Especially the server side that can connect to multiple clients benefits from this feature.

Keeping Round-Trip Time (RTT) low and ensuring a high level of security, QUIC combines the session and TLS 1.3 handshake into one step. This part is managed by the `Crypto & TLS` component, which relies on an external library (`BoringSSL` or `OpenSSL`) for Transport Layer Security (TLS) and `ring` for cryptographic primitives and hash functionality. Section 3.3 takes a closer look at their integration.

To send the first payload data to the communication partner, which is already possible during the handshake procedure, the `Packet/Frame` component is responsible for creating, manipulating and verifying packets and frames. Outgoing data is assembled into packets, encoded and encrypted using `ring` whereby the Authenticated Encryption with Associated Data (AEAD) parameters (algorithm and secrets) are provided by the `Config`. For the incoming packets, the procedure is reversed, but with the addition that the integrity is verified as well.

Among other things, QUIC was designed to solve the Head-of-Line blocking problem faced by TCP. Therefore, the `Stream` component is responsible for multiplexing all streams within a connection. It manages the current states and flow control of each stream individually. Additionally, it implements stream prioritization and scheduling based on the decisions made by the host application and provided by the `Config`. The quiche-specific implementation of this feature is illustrated in Section 3.2.

To provide a connection-oriented and lossless data transmission on top of UDP, the `CC & Recovery` component implements the Congestion Control (CC) logic and recovery mechanisms. It keeps track of the current transmission state (congestion window (CWND), RTT, pacing and loss rate, . . .), updates its properties and sends ACK frames for received packets. If packets are lost or run into a timeout, this component triggers retransmissions. Section 3.1 discusses the possibilities of the natively implemented CC algorithms and how newly developed algorithms can be added.

 doi: 10.2313/NET-2025-05-1_19

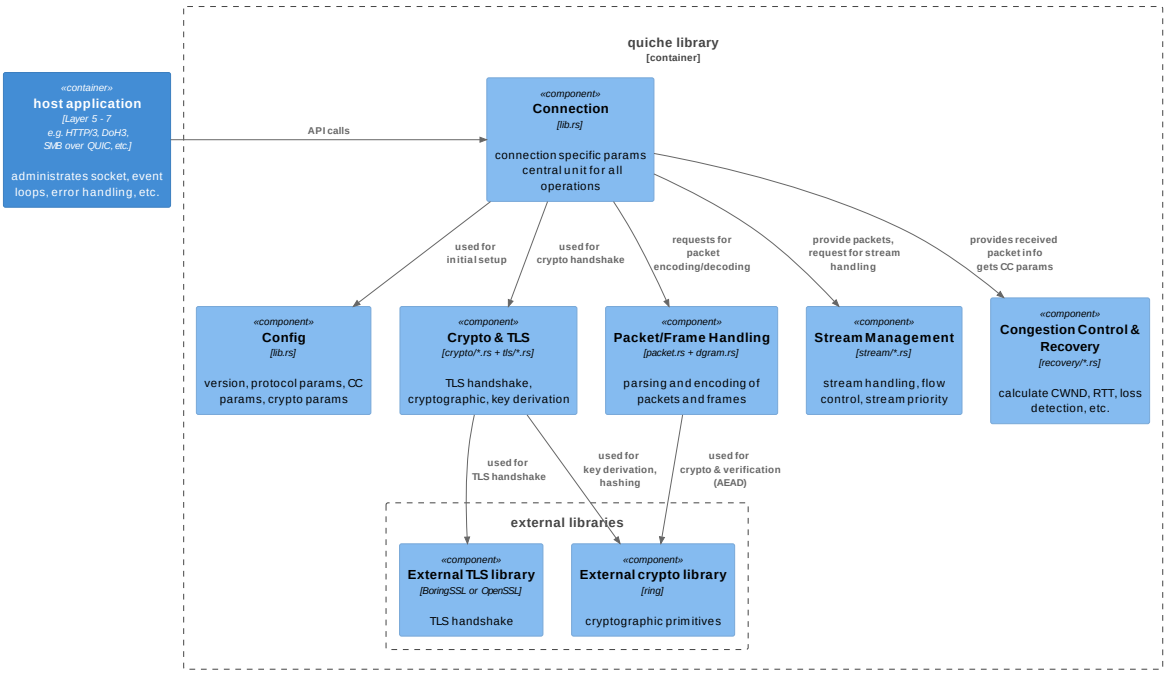


Figure 2: Internal Architecture of quiche at the Component Level

3. Special Features

The RFC 9000 is a comprehensive document that specifies QUIC. Fundamental mechanisms such as the connection establishment (including the handshake process) or the state and error handling are defined with a high level of detail. The line format in particular (e.g. layout of packets, frame types, ID ranges) is standardised with a high level of detail in the RFC. On the other hand, some parts define boundary conditions or optional features instead and leave room for user-specific implementations [2]. In this section, we will focus on some quiche-specific implementations that are not explicitly specified in the RFC 9000.

3.1. Congestion Control

In RFC 9000, CC is defined as a mandatory feature of QUIC, but no exact implementation or algorithm is prescribed. It only provides TCP-based guidelines. Cloudflare implemented a couple of different CC algorithms in quiche to provide a wide range of options for different network conditions. Two of them, Reno and Cubic, are well-known and widely used in TCP and were implemented with a few minor adjustments to fit the QUIC environment. One reason these “old” algorithms are still available is their well-known behaviour and the fact that they are easy to implement. In this way, Cloudflare intends to boost the deployment of QUIC, and of course also of quiche, by providing a familiar environment for developers [11]. To improve their performance HyStart++ was implemented according to RFC 9406. This feature enhances the initial slow start phase by gracefully transitioning into the congestion avoidance phase when early congestion is detected. In contrast to the loss-based algorithms, the library also provides two delay-based CC algorithms: Bottleneck Bandwidth and Round-trip propagation time (BBR) and BBRv2 [12].

A major advantage of quiche is the interchangeability of its CC algorithms. It offers a simple API to switch between them or to implement custom ones. As a real-world example, hosts and servers can dynamically switch between the algorithms based on the current network conditions. For high-capacity and relatively stable links, more aggressive algorithms like BBRv2 can be effective, while on mobile or high-latency networks — where conditions are more variable — more conservative algorithms like Cubic generally provide more stable and predictable performance.

3.2. Stream Prioritization and Scheduling

According to the RFC 9000, a QUIC implementation should provide the possibility for the application to prioritize streams relative to each other [2]. quiche satisfies these specifications with its stream prioritization and scheduling feature. Every stream carries a `Stream Priority Key`, which contains three parameters: `urgency`, `incremental` and `id`. The `id` orchestrates all stream priorities in a Red-Black Tree. It is ordered by the `urgency` (0 to 255, default: 127), where a lower value indicates a higher priority. The `incremental` boolean indicates if the data can be sent in a round-robin fashion or processed online in pieces. If the `urgency` of two streams is the same, non-incremental streams are prioritized over incremental ones. quiche itself only implements the scheduling logic and provides an API to set and update the stream priorities. The decision-making process is the host application’s responsibility, which communicates the listed parameters to quiche and thus triggers the scheduling [12].

3.3. Integration of Crypto and TLS

As mentioned, QUIC's security is based on the TLS 1.3 protocol. Therefore, quiche relies on the external library `BoringSSL`, a fork of `OpenSSL` developed and maintained by Google. Cloudflare migrated their Secure Sockets Layer (SSL) connection termination stack already in 2017 to `BoringSSL` to reduce maintenance [13]. It offers a dedicated API that can be used by QUIC implementations [3], but as shown in Figure 2, it is not directly applicable since `BoringSSL` is a C library and quiche is written in Rust. Therefore, the `Crypto & TLS` component uses Rust's Foreign Function Interface to interact with the external library and provides the necessary functionality to the inside of quiche. Nevertheless, the usage of `OpenSSL` is still natively available in quiche.

Even though `BoringSSL` offers the entire cryptographic primitives that are mandatory for QUIC, quiche employs another external library for this purpose: `ring`. As `ring` uses in parts the same implementation of cryptographic primitives as `BoringSSL`, this does not noticeably affect the performance of quiche [3]. This design choice was, moreover, security-driven, as this library is written in Rust, so it can guarantee memory safety and prevent undefined behaviour by default [14].

A slight anomaly that contrasts with the general communication structure within quiche is the direct interaction between `Packet/Frame` and `ring`. Furthermore, QUIC uses common symmetric AEAD (AES-128-GCM, AES-256-GCM, ChaCha20-Poly1305) for payload encryption and integrity protection, but the headers are protected (masked) with a different set of keys [15].

4. Conclusion

In order to meet all specifications of the extensive `RFC 9000`, Cloudflare's developer team inevitably had to choose a well-considered architecture for quiche. At the container level, we examined the relation between the host application and the library. It offers a simple but well-defined API for interaction and, therefore, defines a precise division of tasks and responsibilities concerning the host application. One level deeper, at the component level, we recognized the same design principle among the components of quiche. Nearly every information flow was initialised by the central `Connection` component. This demonstrates a low coupling within and a high cohesion of the library to the outside world.

However, from a methodological point of view, the analysis based on the source code was very challenging because some central files are built up of thousands of lines of code (e.g. `lib.rs`: > 17.000). In combination with better documentation, especially of helper functions, this is a starting point for structural improvement.

A further example of a good design choice is the integration of `BoringSSL` and `ring`. These libraries are extensively tested and used under real-world conditions and satisfy all requirements (security, performance, reliability) of QUIC. In addition, the facade-like design of all components, but `Crypto & TLS` in particular, indicates high abstraction and encapsulation.

According to Google, the utilization of `BoringSSL` for third parties is expressly not recommended, as the stability of API, for example, is not guaranteed [16]. Even if Cloudflare's internal migration to `BoringSSL` was reasonable, and quiche relies only on a snapshot of it, this potential risk should be treated with caution, as changes and incompatibilities can occur in the future.

Nevertheless, the quiche library is a well-designed and well-structured QUIC implementation that can be further developed to improve the overall structure and maintainability.

References

[1] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, Internet-Draft draft-hamilton-quic-transport-protocol-00, Jul. 2016, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-hamilton-quic-transport-protocol/00/

[2] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021, accessed: 2024-12-15. [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[3] A. Ghedini, "Enjoy a slice of QUIC, and Rust!" Jan. 2019, accessed: 2024-12-15. [Online]. Available: https://blog.cloudflare.com/enjoy-a-slice-of-quic-and-rust/

[4] S. Brown, "C4 Model," Aug. 2017, accessed: 2024-12-13. [Online]. Available: https://c4model.com/

[5] M. Kempf, "QUIC Explorer," https://quic-explorer.net, accessed: 2024-12-16 (Commit e06efd2). [Online]. Available: https://quic-explorer.net

[6] S. Brown, "Visualising software architecture with the C4 model - Simon Brown, Agile on the Beach 2019," https://www.youtube.com/watch?v=x2-rSnhpw0g, 2019, accessed: 2024-12-16.

[7] A. Vázquez-Ingelmo, A. García-Holgado, and F. J. García-Peñalvo, "C4 model in a Software Engineering subject to ease the comprehension of UML and the software," in *2020 IEEE Global Engineering Education Conference (EDUCON)*, 2020.

[8] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 183–196. [Online]. Available: https://doi.org/10.1145/3098822.3098842

[9] L. Pardue, "QUIC Version 1 is Live on Cloudflare," May 2021, accessed: 2024-12-16. [Online]. Available: https://blog.cloudflare.com/quic-version-1-is-live-on-cloudflare/

[10] M. Maurer and M. Yu, "DNS-over-HTTP/3 in Android," Jul. 2022, accessed: 2024-12-15. [Online]. Available: https://security.googleblog.com/2022/07/dns-over-http3-in-android.html

[11] J. Choi, "CUBIC and HyStart++ Support in quiche," https://blog.cloudflare.com/cubic-and-hystart-support-in-quiche/, May 2020, accessed: 2024-12-15.

[12] Cloudflare, "quiche repository," 2024, accessed: 2024-12-16. [Online]. Available: https://github.com/cloudflare/quiche

[13] A. Ghedini, "Make SSL boring again," Dec. 2017, accessed: 2024-12-18. [Online]. Available: https://blog.cloudflare.com/make-ssl-boring-again/

[14] J. Blandy, J. Orendorff, and L. F. Tindall, *Programming Rust*, 2nd ed. "O'Reilly Media, Inc.", 2021.

[15] R. Seal, "Looking into QUIC Packets in Your Network," Jul. 2021, accessed: 2024-12-19. [Online]. Available: https://www.keysight.com/blogs/en/tech/nwvs/2021/07/17/looking-into-quic-packets-in-your-network

[16] Google, "BoringSSL Repository," 2024, accessed: 2024-12-18. [Online]. Available: https://github.com/google/boringssl

Feature Selection for ML in Self-Managing Networks

Youssef Mebazaa, Johannes Späth*, Max Helm*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: youssef.mebazaa@tum.de, spaethj@net.in.tum.de, helm@net.in.tum.de

***Abstract*—Despite its ever-increasing complexity, the management of networks, for example those operated by Internet Service Providers, is still heavily relying on human intervention that need to monitor the infrastructure and react accordingly to any unforseen changes. This task is becoming harder as networks nowadays generate large amounts of telemetric data that require human intervention to analyse it.**

Fortunately, self-managing networks can automate tasks such as monitoring, troubleshooting and optimizing network performance while reducing human intervention. These self-managing networks rely on machine learning (ML) which has emerged as a powerful tool for automating such tasks. However, feeding a large amount of unprocessed data into the ML model will greatly reduce its accuracy. For ML models to work effectively, they require preprocessing steps like feature selection (FS) to identify the most relevant features and metrics of our data.

This paper presents different approaches for feature selection in machine learning as well as idetify those that can be applied to self-managing networks.

***Index Terms*—feature selection, self-managing networks, network telemetry**

1. Introduction

Modern networks are becoming increasingly complex with diverse devices, high traffic and new protocols. However, as they grow in scale and complexity, their management also becomes harder and more complex. While these tasks are still heavily relying on human intervention, it is clear that we will some day reach a point where it is not feasible anymore and will simply become unsustainable. To address this challenge, self-managing networks (SMN) have emerged. These networks can automate tasks such as monitoring, anomaly detection, traffic optimization, resource allocation... [1]. These SMNs rely on machine learning models that must interpret the vast amount of telemetric data generated by network devices and act accordingly.

Machine learning models must process and interpret this data efficiently and deliver accurate predictions. The sheer volume of the data that is fed into the model makes this task tedious as it usually consists of a very high number of features that may have bad repercussions on our models predictions. To overcome this challenge, feature selection comes into play. It is commonly used in ML models as it can improve their performance and speed by identifying the most relevant data attributes or features of a dataset and eliminating those it deems unnecessary. By reducing the dimensionality of the data, feature selection not only improves ML model performance but also minimizes computational overhead and enhances scalability.

In this paper, we provide a comprehensive overview of existing feature selection methods that are used in ML. This will be done in Section 2. In Section 3, we will briefly introduce self-managing networks as well as the datasets that can be used by ML models. Finally, in Section 4 we apply the feature selection methods introduced in Section 2 to SMNs and summarize advantages, disadvantages and scalability of these methods.

2. Feature Selection Methods

Feature selection aims mainly at identifying and selecting a subset of relevant features for use in model construction. This process is a critical aspect of machine learning as it helps reduce the dimensionality of data with many features. Feature selection is therefore the removal of features that are redundant and increase computational overhead while decreasing the accuracy for no gain [1].

It is also a good solution for the curse of dimensionality. According to Bishop [2], that phenomenon occurs in high-dimensional datasets where the number of configurations grows exponentially as the dimensionality of the dataset increases.

Feature selection has the following goals according to Guyon and Elisseeff [3]:

- Remove useless or insignificant data.
- Enhance the model's performance by reducing overfitting.
- Improve its accuracy.
- Reduce training time.

Each method, given a set of features, has its own criteria and will output a subset of features that should contain only the most relevant ones.

This section presents feature selection methods that are used in machine learning in general. According to Girish and Ferat [4] , most feature selection methods can be classified into three types :

- Filter Methods
- Wrapper Methods
- Embedded Methods

 doi: 10.2313/NET-2025-05-1_20

2.1. Filter Methods

Filtering is a preprocessing step that is applied before the learning phase and is therefore independent of the learning algorithm and has no bias toward the models. The main idea behind filtering is to assign a rank or a weight to each feature by using some criteria, sorting them the features according to their score, and finally selecting the features with the highest scores while discarding those under a certain threshold [4]. However, we still have to determine how exactly we can assign a rank to each feature.

According to Law et al. [5], a feature can be considered irrelevant if it is conditionally independent of the class labels given other features. What this implies is that a feature that has no influence on the class labels can be considered irrelevant, as it does not contain useful information about the data. Such features should be discarded. On the other hand, features which contain information about the labels should be kept. Many filter methods use statistical functions to determine the degree of dependency of a feature with its output or label. In the following, we can observe two filter methods:

- **Chi-square statistic** According to Ray et al. [6] Chi-square statistic can be used to compute the relation of each individual feature with the outcome. It can be calculated as follows

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

  Where:

  - O_i: The observed frequency in category i.
  - E_i: The expected frequency in category i, calculated under the null hypothesis.
  - n: The total number of categories.

  A larger χ^2 value indicates a greater dependency between the feature and the output.
- **Mutual information** Mutual information is based on a concept called entropy which is a measure of uncertainty or unpredictability of the variable. It measures "the amount of information that one random variable has about another variable" [7]. The mutual information between two variables X and Y can be defined as follows :

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) \log \left(\frac{P(x,y)}{P(x)P(y)} \right)$$

  Where :

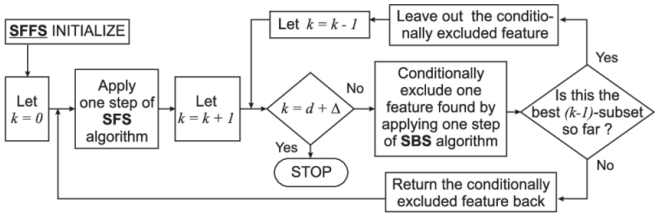
  - $P(X)$ and $P(Y)$ are the probabilites of X and Y respectively
  - $P(X,Y)$ is the joint probability of X and Y

2.2. Wrapper Methods

While filter methods evaluate the relevance of a feature, wrapper methods evaluate the relevance of a subset of features. The criterion for choosing the right subset is the performance of the learning algorithm. The latter is used as a black box (i.e no knowledge of the algorithm is required). Wrapper-based techniques use an iterative approach, choosing a different subset of features in each iteration and fitting it in a model to determine the most optimal subset, i.e the subset that outputs the highest performance or accuracy will be chosen [4].

This also means that the algorithm has an influence on the results and is tightly connected to it unlike filter methods. However, for N features, there are 2^N subsets. Naively evaluating all possible subsets is therefore impossible since it is an NP-hard problem [8].

One of the most basic algorithms is called Sequential forward Selection (SFS). The algorithm starts with an empty set of features and iteratively adds the one that gives the highest value for some objective function until the number of wanted features is reached. Pudil et al. [9] present an algorithm called Sequential Forward Floating Selection (SFFS) which utilises SFS. Just like SFS, it iteratively chooses the best feature. However, before starting the next iteration it makes sure that the current subset is the most optimal one of its size. Figure 1a showcases how this algorithm works.



(a) How the SFFS algorithm works [10]

Another version of it called Sequential Backward Selection exists, in which the algorithm starts with the full set of features and iteratively removes the one that gives the lowest value. [4]

2.3. Embedded Methods

Both methods presented above share the similarity that they are a preprocessing step applied to the data before performing the training. Embedded methods on the other hand do not separate the feature selection part from the training and share the benefits of both of these methods [1]. They are therefore more efficient in terms of computational costs in comparison to wrapper methods since they do not need to perform classification on different subsets. Commonly used techniques include

- **Regularization-Based Methods** such as Lasso or L1 Regression which adds an L1 penalty term to the loss function during training: $L1 = \sum_{i=1}^{n} |w_i|$ where w_i is the i-th feature [11].
  Features with coefficients reduced to exactly zero are effectively removed from the model.
- **Decision Trees** which inherently rank features based on their importance during training. These trees split data using features that maximize impurity, for example using Gini index or entropy [2].

2.4. Other Methods

There are also other feature selection methods that cannot be classified into the three types described above

 doi: 10.2313/NET-2025-05-1_20

since they differ from our definition of FS. They are also called feature extraction methods [1] and the main difference is that instead of choosing a subset of the features, feature extraction methods create a completely new set of features that is smaller than the original one, while also preserving its most important characteristics [12].

2.4.1. Autoencoder FS.

Han et al. [13] propose an Autoencoder based feature selection (AEFS) method. Autoencoders are a type of neural networks (NN) whose main goal is to receive an input and encode it into a compressed representation and are then able to reconstruct or decode it into a state that should resemble the original input [14].

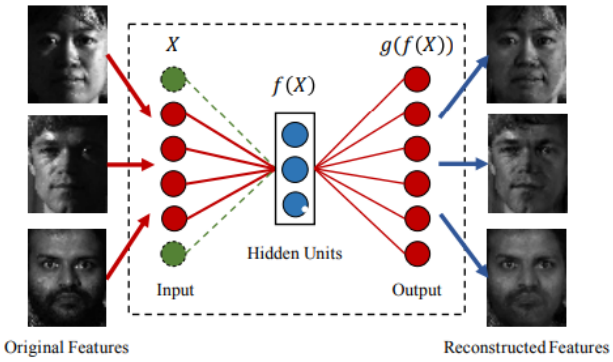
AEFS works by squeezing or encoding the input feature space into a lower dimensional one. By decoding the data we are left with only the most relevant features. This is done by minimizing the equation

$$J(\Theta) = \frac{1}{2m}\|X - g(f(X))\|_F^2 + \alpha\|W^{(1)}\|_{2,1} + \frac{\beta}{2}\sum_{i=1}^{2}\|W^{(i)}\|_F^2$$

where

- X denotes the (unlabeled) data matrix and m the number of samples.
- f and g are the encoder and decoder functions respectively.
- $\|W^{(1)}\|_{2,1}$ is a regularization on $W^{(1)}$, which is the weight matrix in the input layer.
- α is the trade-off parameter of the reconstruction loss and the regularization term.
- $\sum_{i=1}^{2}\|W^{(i)}\|_F^2$ denotes the weight decay regularization.
- β is the penalty parameter.

The figure 2a showcases how AEFS can be used to only keep relevant features.



(a) AEFS applied to faces. On the left we can see the original ones with all their features. On the right, we can see the reconstructed features that represent the most important ones [13].

2.4.2. PCA.

According to Jolliffe [15], principal component analysis (PCA) is a technique used for reducing the dimensionality of such datasets, where a lot of features are correlated. It also increases interpretability while at the same time minimizing information loss. PCA works by creating new uncorrelated variables that successively maximize variance. For example, for an input data of dimension d, it is reshaped into a new coordinate system, which also has d dimensions and is ordered in such a way that the first ones, also called principal components, contain most of the variance of the input features. It can also be applied to feature extraction by only keeping the first few principal components which contain most of the information needed.

3. Applying FS to SMNs

In this section we explain what exactly network telemetry is. We also define SMNs, as well as their roles and tasks. We will do so in subsections 3.1 and 3.2, respectively.

3.1. Network telemetry

RFC9232 [16] defines network telemetry as being "information that can be extracted from networks". It can then be analyzed and used by humans or even ML. It contains information about network configuration, logs, event records as well as statistics.

Network telemetry serves as the backbone for self-managing networks since it provides the essential data that needs to be analyzed and studied by the ML model, which will then take action accordingly.

There are various methods or frameworks to extract telemetry data from a network. An example is YANG: "YANG is a data modeling language used to model configuration data, state data, Remote Procedure Calls, and notifications for network management protocols" [17].


```
"admin_status": "up",
"description": null,
"enabled": true,
"ifindex": 1,
"ipv4": {
    "address": [
        {
            "ip": "127.0.0.1",
            "origin": "static",
            "prefix-length": 8
        }
    ],
    "enabled": true,
    "forwarding": true,
    "mtu": 65536,
    "neighbor": []
},
"ipv6": {
    "address": [
        {
            "ip": "::1",
            "origin": "static",
            "prefix-length": 128
        }
    ],
    "autoconf": {
        "create-global-address": false,
        "create-temporary-addresses": false,
        "temporary-preferred-lifetime": 0,
        "temporary-valid-lifetime": 0
    },
    "dup-addr-detect-transmits": false,
    "enabled": true,
    "forwarding": true,
    "mtu": 65536,
    "neighbor": []
},
"link-up-down-trap-enable": false,
"name": "lo",
"oper-status": "unknown",
"phys-address": "00:00:00:00:00:00",
"statistics": {
    "in-crc_errors": 0,
    "in-discarded": 0,
    "in-errors": 0,
    "in-fifo_errors": 0,
    "in-frame_errors": 0,
    "in-length_errors": 0,
    "in-missed_errors": 0,
    "in-multicast-pkts": 17872,
    "in-octets": 18781974,
    "in-packets": 17872,
    "out-aborted_errors": 0,
    "out-carrier_changes": 0,
    "out-carrier_errors": 0,
    "out-discarded": 0,
    "out-errors": 0,
    "out-fifo_errors": 0,
    "out-octets": 18781974,
    "out-packets": 17872,
    "out-window_errors": 0
},
"type": "loopback"
```


(a) Example of a data in YANG

As we can observe in the figure 3a the data contains a lot of features and not all of them are relevant. The irrelevant ones have to be filtered using the filter selection methods described in section 2.

3.2. Self-Managing Networks

According to Behringer et al. [18], self-managing networks, also referred to as autonomic networks, are systems capable of performing key management functions autonomously and are able to adapt to a changing environment on their own. This autonomy is comprised of the following properties:

- **Self-Configuration**: They should be able to automatically configure themselves with minimal human intervention.
- **Self-Healing**: Detection and mitigation of failures or anomalies without external guidance.
- **Self-Optimizing**: They automatically determine ways to optimize their behavior against a set of well-defined goals.
- **Self-Protection**: They should identify potential attacks and implement security measures to defend against them.

4. Comparison of FS methods

In this section, we compare the different feature selection methods we presented in Section 2 as well as summarize the advantages and disadvantages of each method. Since we also know the roles of self-managing networks, we can also identify which methods can be used for which tasks.

This comparison is mainly based on how applicable they are to supervised or unsupervised learning.

4.1. Supervised Methods

Supervised learning is a category in ML where the class labels of the data are known [2].

4.1.1. Filter methods.

Filter methods, especially the two methods we explained, are mainly used for classification [12] since the data needs to be labeled.

- **Advantages**: The main advantage of filter methods is that they classify features based on their relative performace and have no bias toward the classification algorithm used [4]. They also do not require much computing.
- **Disadvantages**: Since these methods are independent of the learning algorithm, the selected features can be not optimal for it [19].
- **Application**: Filter methods should be used as a preprocessing step and can therefore be combined with other methods such as embedded methods [8]. They can be applied in many tasks such as self-healing tasks [20] and traffic classification [21].

4.1.2. Wrapper methods.

According to Bommert et al. [8] "Wrapper methods consider subsets of the set of all features. For each of the subsets, a supervised learning (e.g. classification) model is fitted". This makes them also supervised methods.

- **Advantages**: They are more precise than filter methods since the features are studied.
- **Disadvantages**: Since there are a lot of subsets to consider, they require a lot of computing which makes them slower than filter methods. The result is also biased toward the learning algorithm [12] and they rely on the relative accuracy of the classifier which makes them prone to overfitting [4].
- **Application**: It is not a good idea to use wrapper methods since they "are computationally infeasible for high-dimensional data sets" [8]. They can, however, be used in combination with filter methods to enhance their performace.

4.1.3. Embedded methods.

In the case of embedded methods, the feature selection part is embedded in the training of the model. The model therefore determines if it is supervised or unsupervised.

- **Advantages**: They share the advantages of both filter and wrapper methods since the features interact with the learning algorithm like wrapper methods. However they are much more efficient since they do not need to evaluate all subsets [19].
- **Disadvantages**: Just like wrapper methods, the result may be dependant on the algorithm used.
- **Application**: Applications include Traffic classification [22], Misuse Detection [23], Load balancing [24].

4.2. Unsupervised Methods

In contrast to supervised learning, there are no class labels used in unsupervised learning.

4.2.1. AEFS.

AEFS is an unsupervised algorithm since the data used is not labelled and autoencoders are mainly used in unsupervised learning [14].

- **Advantages**: Just like embedded methods, AEFS is efficient and has great accuracy and performance [13]. It has also been proven to be better than PCA since it can capture non-linear dependencies among the features [25].
- **Disadvantages**: Autoencoders are prone to overfitting [14].
- **Application**: Anomaly detection [25].

4.2.2. PCA.

PCA is an unsupervised algorithm since it is a clustering algorithm [15].

- **Advantages**: Also very efficient in reducing the dimensionality of the data while only preserving the most important features.
- **Disadvantages**: Can only capture linear dependencies among the features [25].
- **Application**: Anomaly detection [25].

5. Conclusion

In this paper we explained what feature selection is. We also presented different methods and ways to do it. Then we discussed network telemetry and the roles and

tasks of self-managing networks. Finally we compared the different feature selection methods, evaluating their advantages, drawbacks and also applying them to self-managing networks.

References

[1] Boutaba, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed., ser. Information Science and Statistics. Springer New York, NY, 2006, hardcover published: 17 August 2006, Softcover published: 23 August 2016.

[3] A. E. Isabelle Guyon, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research 3 (2003)*, pp. 1157–1182, 2003.

[4] F. S. Girish Chandrashekar, "A survey on feature selection methods."

[5] A. K. J. Martin H.C. Law, Mario A.T. Figueiredo, "Simultaneous Feature Selection and Clustering Using Mixture Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1154 – 1166.

[6] S. Ray, K. Alshouiliy, A. Roy, A. AlGhamdi, and D. P. Agrawal, "Chi-squared based feature selection for stroke prediction using azureml," in *2020 Intermountain Engineering, Technology and Computing (IETC)*, 2020, pp. 1–6.

[7] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural Computing and Applications*, vol. 24, pp. 175–186, 2014. [Online]. Available: https://doi.org/10.1007/s00521-013-1368-0

[8] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Journal of Statistical Software*, 2020.

[9] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, no. 11, pp. 1119–1125, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167865594901279

[10] P. Somol, J. Novovicova, and P. Pudil, *Efficient Feature Subset Selection and Subset Size Optimization*, 02 2010, vol. 56.

[11] J. Tang, S. Alelyani, and H. Liu, *Feature selection for classification: A review*. CRC Press, Jan. 2014, pp. 37–64.

[12] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1200–1205.

[13] K. Han, Y. Wang, C. Zhang, C. Li, and C. Xu, "Autoencoder inspired unsupervised feature selection," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2941–2945.

[14] D. Bank, N. Koenigstein, and R. Giryes, *Autoencoders*. Cham: Springer International Publishing, 2023, pp. 353–374. [Online]. Available: https://doi.org/10.1007/978-3-031-24628-9_16

[15] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed., ser. Springer Series in Statistics. Springer New York, NY, 2002. [Online]. Available: https://doi.org/10.1007/b98835

[16] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang, "Network Telemetry Framework," RFC 9232, May 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9232

[17] M. Björklund, "The YANG 1.1 Data Modeling Language," RFC 7950, Aug. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7950

[18] M. H. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. E. Carpenter, S. Jiang, and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals," RFC 7575, Jun. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7575

[19] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, Dec. 2017. [Online]. Available: https://doi.org/10.1145/3136625

[20] D. Palacios, I. de-la Bandera, A. Gómez-Andrades, L. Flores, and R. Barco, "Automatic feature selection technique for next generation self-organizing networks," *IEEE Communications Letters*, vol. 22, no. 6, pp. 1272–1275, 2018.

[21] T. T. T. Nguyen and G. Armitage, "Grenville, a.: A survey of techniques for internet traffic classification using machine learning," vol. 10, pp. 56 – 76, 12 2008.

[22] A. Azab, M. Khasawneh, S. Alrabaee, K.-K. R. Choo, and M. Sarsour, "Network traffic classification: Techniques, datasets, and challenges," *Digital Communications and Networks*, vol. 10, no. 3, pp. 676–692, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864822001845

[23] A. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," vol. 18, pp. 1–1, 01 2015.

[24] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," no. 4, pp. 2392–2431, 2017.

[25] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *2018 Wireless Telecommunications Symposium (WTS)*, 2018, pp. 1–5.

Review of commercial VPN provider claims

Anton Scheitler, Lion Steger*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: anton.scheitler@tum.de, stegerl@net.in.tum.de

***Abstract*—Commercial virtual private networks (VPNs) have gained immense popularity because of their claims to provide secure, fast and obfuscated connectivity. This is especially important, as the number of data breaches is on the rise and user's seek to protect themselves [1]. It is the goal of this paper to review the claims made by large commercial VPN providers and determine whether they are correct or not. To this extent, four of the most popular VPN providers and their protocols are evaluated in terms of security, obfuscation capabilities, speed and trustworthyness. Overall, NordVPN offers great security and speed in their service and only suffers from the fact that their custom VPN protcol is not open-sourced. ExpressVPN shares this problem and also logs identifiable information. Surfshark offers a slightly slower VPN experience and shares user data with advertisers. Hide.me uses secure and open-sourced protocols by default and logs very little and non-identifiable data. Their service, however, is also the slowest. None of the VPN providers are able to obfuscate VPN traffic and make VPN usage invisible.**

***Index Terms*—networks, VPN, OpenVPN, IKEv2/IPSec, WireGuard, NordLynx, Lightway**

1. Introduction

The rising number of data breaches and censorship in countries around the world leads to a growing interest in VPN services [1]. VPN providers claim that, with their services, users are able to use the Internet more securely and circumnavigate governmental censorship and geoblocking, while suffering minimal latencies. However, these claims can be incorrect or misleading and might lead consumers to make poor buying decisions. This paper seeks to address this issue by analyzing different providers and their protocols to determine if their claims of security, obfuscation capabilities and speed hold up to reality. The criterium for security will be that all of the protocols used by a provider are either open-sourced or audited regularly and haven't had major security vulnerabilities in the past. To compare speed, the latencies of the connections of the providers are measured. Finally, a VPN provider is considered to offer decent obfuscation, if it can hide the fact that their customers are using their service.

2. Background

A private network is a network that is isolated from other networks. Communication within a private network cannot leak to the outside. This can be achieved by leasing physical private communication lines and connecting hosts with it. A VPN (Virtual Private Network) is a private network, built on top of a public network. In a VPN, hosts are blocked off from a public network and can only be connected to over secure tunnels [2, section 8.6.3]. A tunnel is a special connection between hosts. When a data packet passes through a tunnel, it is encrypted and encapsulated. This is useful, when communication needs to be secure or the tunnel ends at a location other than the final destination of the packet. These tunnels do not have to be formed via private communication lines but instead over the public Internet. This allows any host on the Internet with the necessary credentials to connect to a VPN. [2, section 8.6.1] In order to connect to a VPN, a user first needs to authenticate themselves, often followed by a negotiation of a cipher suite with the VPN server. The client then agrees to a tunneling protocol and exchanges secrets with the server. [3] A tunneling protocol determines how data is encrypted and encapsulated before being sent over a tunnel. It is the backbone of a VPN.

Commercial VPN providers offer a VPN which consists of a set of proxy servers, that their customers can connect to. Thanks to the properties of a VPN, customers can therefore communicate over an encrypted connection, while obfuscating their IP address and physical location.

A single VPN provider may support a number of tunneling protocols which is why it is important to understand them in order to be able to compare providers.

2.1. OpenVPN

OpenVPN is one of the most popular tunnel protocols used by VPN providers. It uses the widespread SSL/TLS mechanisms to authenticate hosts, exchange cryptographic secrects between them and encrypt messages. It uses the OpenSSL library to implement this. In addition, it runs on all major operating systems, including Windows, macOS, Linux, Android, iOS and even OpenBSD [4]. Packets traveling through an OpenVPN tunnel can be encapsulated in TCP, as well as UDP packets [5]. OpenVPN can also be configured to establish connections via the port 443. This is the same port used for HTTPS, which makes it harder for ISPs to use firewalls to block VPN traffic [6, section 8.2.3]. Additionaly, OpenVPN is open-sourced which reduces the risk of unpatched vulnerabilities and backdoors

 doi: 10.2313/NET-2025-05-1_21

2.2. IKEv2/IPSec

This protocol is a combination of two different mechansims. The first is IPSec, which is used for encryption. The second is the the Internet Key Exchange Version 2 (IKEv2), which is used for authentication and the exchange of secrets [6, section 3.1]. Once keys have been generated and exchanged, IPSec is used to encapsulate and encrypt packets. IPSec offers different encapsulation mechanisms, however, for IKEv2/IPSec, the Encapsulation Security Payload (ESP) is used. This works by first encrypting the original message and wrapping it with an ESP header and trailer. The resulting message is wrapped inside another IP packet [6, section 4.1]. This approach of wrapping an entire packet within another IP packet is called the tunnel mode of IPSec. The resulting packet has two IP addresses. The "inner" IP address is that of the original message and the "outer" address is that of the message after encapsulation. One benefit of this protocol is that it supports MOBIKE, which can handle changes in the outer IP address of a device while still preserving the connection to a VPN [6, section 3.9]. This makes IKEv2/IPSec especially well suited for VPN usage on mobile devices and laptops.

2.3. WireGuard

WireGuard is a new and open-sourced VPN Protocol. It uses public keys instead of SSL certificates for authentication and the Noise Protocol Framework, which is based on Diffie-Hellman, for key exchanges [7]. As opposed to the previous protocols, WireGuard does not work with a suite of cryptographic ciphers and instead handles all encryption using the stream cipher ChaCha20-Poly1305. The WireGuard protocol does not specify how to dynamically assign IP addresses to clients connecting to a server. Instead, a naive implementation of WireGuard would simply store the static IP addresses of those clients. All in all, WireGuard is a fast and secure protocol but has some anonymity concerns that come with storing static IP addresses [6, section 8.3]

2.4. SSTP

The SSTP protocol is a closed-source VPN protocol developed by Microsoft. It is similar to OpenVPN in that it uses SSL/TLS for authentication, key exchanges and encryption. SSTP can be configured to use TCP, as well as UDP for encapsulation. SSTP connections can also be set up over port 443, achieving some level of obfuscation, as described in 2.1. Overall SSTP servers are easier to setup than OpenVPN servers. However, the protocol is only supported by Windows [6, section 8.2.1].

2.5. L2TP

The Layer 2 Transport Protocol (L2TP) uses IKE for authentication and key exchange and IPSec for encryption and encapsulation. L2TP is an older VPN protocol and can be configured with IKEv1 which leads to the use of a weak group PSK. Even if an implementation of L2TP is configured correctly, it adds layers of unnecessary encapsulation which increases network issues like packet fragmentation. Also, L2TP does not support AEAD algorithms which leads to an increased CPU usage [6, section 8.5.2].

TABLE 1: VPN Provider Protocol Support

	NordVPN	ExpressVPN	Surfshark	Hide.me
OpenVPN	✓	✓	✓	✓
IKEv2/IPSec	✓	✓	✓	✓
WireGuard			✓	✓
SSTP			✓	✓
L2TP/IPSec			✓	
NordLynx	✓			
Lightway		✓		

[12] [13] [14] [15] [16]

2.6. NordLynx

NordLynx is a VPN protocol built on top of WireGuard and created by the VPN provider NordVPN. It addresses the anonymity issues of WireGuard by constructing a layer of double NATs around a WireGuard server. The first NAT assigns the same IP address to every user, making them indistinguishable to the server. The second NAT assigns a unique address to a user from a pool of IP addresses. This obfuscates traffic. While the NordLynx protocol is not open-sourced, its foundation WireGuard is. This makes it more transparent than completely closed-source protocols, such as SSTP [8].

2.7. Lightway

Lightway is a protocol created by the provider ExpressVPN. Similar to NordLynx, it seeks to address the anonimity issues of WireGuard. Different from NordLynx however, it has no association with WireGuard and is instead built from the ground up. Lightway utilizes SSL/TLS for authentication, key exchange and encryption [9]. A collection of components of the Lightway protocol is also open-sourced under the name lightway-core. However, the protocol itself is not. To assure users of its security, ExpressVPN has also issued independent audits of its protocol [10].

3. Analysis

In order to evaluate VPN providers based on the protocols they offer one needs an overview over which protocol is offered by which provider. In this paper, the focus will be on four VPN providers in total, namely NordVPN, ExpressVPN, Surfshark and Hide.me. The first three were chosen as they are among the largest commercial VPN providers. Hide.me is an another interesting provider as it is free and has been operating with a long and positive track record [11]. An overview over what protocols are supported by them and what their default protocols are, is provided in table 1. As shown by the overview, the most popular protocols such as OpenVPN and IKEv2/IPSec are offered by every provider. However, there are some protocols that are only supported by a single provider. This is especially the case for the custom protocols developed by a provider. The difference between these protocols will be a deciding factor in the evaluation of providers.

 doi: 10.2313/NET-2025-05-1_21

4. Design

In this paper, providers will be evaluated based on four criteria.

The first is security. For many consumers, the main reason of using a VPN is for the additional layer of security provided by it. How secure a protocol, and by extension its provider is, is determined by the security of the key exchange mechanisms and encryption ciphers that they use. Offering outdated protocols to customers can pose a security risk.

The second criterium is obfuscation. Many VPN users suffer from government censorship and use VPNs to work around them. Since VPNs are also deemed illegal in many countries they want to obsucre their traffic as much as possible and avoid their VPN usage being detected.

The third criterium is speed. This is a deciding factor for VPN users when picking a provider. How fast a VPN connection is, is determined by the protocol used but also by the density of a provider's network of VPN servers.

Lastly, customers value transparency in VPN providers. They want to be sure that their VPN provider has their privacy and security interests at heart. Providers can do this by using open-sourced protocols, running frequent and independent audits and avoiding logging user data whenever possible.

5. Findings

The following sections outline the findings of the reasearch into provider claims regarding security, speed, obfuscation and trustworthyness.

5.1. Security

As explained in 2.1, OpenVPN is an SSL-VPN which offers every cipher supported by SSL/TLS. This means that it has access to very secure encryption algorithms such as AES-256, but it also means that it can be misconfigured. In the past there have been instances of OpenVPN implementations using the outdated and insecure hashing algorithm MD5 [17]. However, as long as it is configured properly, OpenVPN is widely considered secure.

IKEv2/IPSec is a secure protocol and all implementations adhere to strong cryptographic standards [18].

WireGuard uses ChaCha20-Poly1305, which does not have any known significant security problems [19, section 4]. Despite this, the encryption algorithm is not approved by NIST [6, section 8.3].

L2TP is considered deprecated by NIST and can be misconfigured quite easily. This is why it is suggested that L2TP implementations should be migrated to IKEv2/IPSec [6, section 8.5.2].

NordLynx uses the same encryption as WireGuard, since it is built on top of it [8].

Lightway can use any cipher, wolfSSL provides, including AES-256 [12]. Independent audits have also confirmed that Lightway is secure [10].

While SSTP offers the encryption and integrity algorithms of SSL/TLS [6, section 8.2.1], it also had severe vulnerabilities in the recent past which allowed for remote code execution [20].

5.2. Obfuscation

While OpenVPN can be configured to use port 443 to form connections, it is still vulnerable to fingerprinting, meaning that OpenVPN traffic can be identified and blocked with a very low false-negative rate [21].

IKEv2/IPSec services can be blocked easily by restricting acces to the ports it uses, namely UDP ports 500 and 4500 [6, section 3.1]

Like with anonomity, WireGuard leaves traffic obfuscation up to the VPN providers that implement it [22]. In this regard, NordVPN, Surfshark and Hide.me all offer obfuscated VPN servers, which they claim make OpenVPN traffic invisible [23] [24] [25]. This claim, however, is false as it has been shown that all of these obfuscated VPN services suffer from insufficient obfuscation over the length of packets [21, section 8]. This allows for the identification of VPN traffic, rendering it anything but invisible.

5.3. Speed

Figure 1: Comparison of OpenVPN UDP speeds across VPN providers

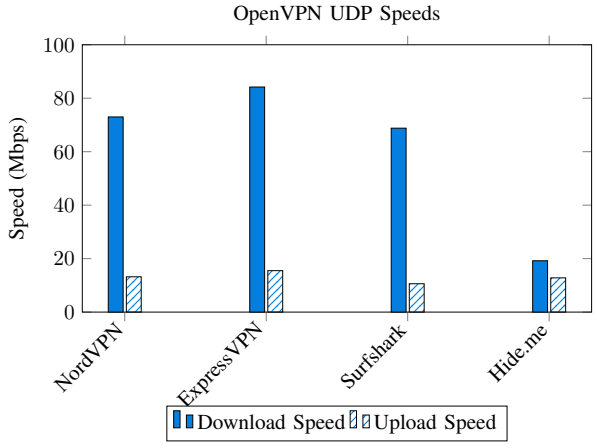
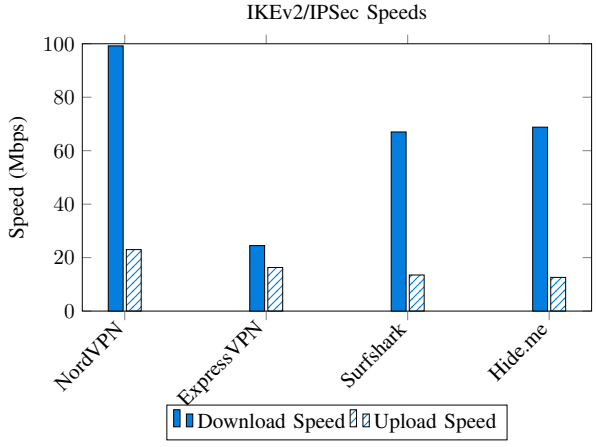
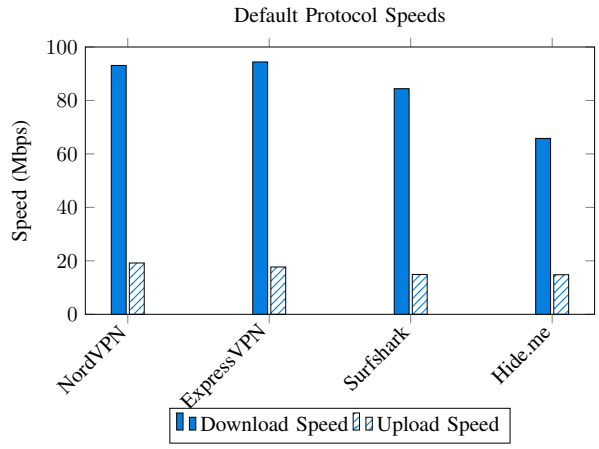


Figure 2: Comparison of IKEv2/IPSec speeds across VPN providers



All providers make claims about the speed of their connections. ExpressVPN and Surfshark both reference third party reviews that give top ratings to their

Figure 3: Comparison of default protocol speeds across VPN providers



speed [26] [27]. NordVPN claims that it is among the fastest VPN providers on the market [28] and Hide.me boldly states that they are the fastest VPN ever seen [29]. To compare the performance of the different protocols on offer by each provider, speed tests were conducted for every provider and their protocols. The protocols that were compared are OpenVPN, IKEv2 and the default protocols for each provider, namely NordLynx, Lightway and WireGuard. These protocols were chosen because they are supported by every provider. For each protocol and provider, a connection was established to the provider's best choice for a server in the United States. Then, speedtest.net was used to determine the the download and upload speed of the connection. The US was chosen as all providers have a high server density there [30] [31] [32] [33]. The result of these speed tests are shown in figures 1, 2 and 3.

5.4. Transparancy and Trustworthyness

As mentioned in 2, OpenVPN and WireGuard are both open-sourced, which makes them trustworthy protcols.

Similarly, IKEv2/IPSec is defined via an RFC standard [34], for which open-source implementations exist.

SSTP on the other hand is a closed-source protocol, which still showed severe vulnerabilities in the past as shown in 5.1. This and the fact that Microsoft has collaborated with governmental institutions, such as the NSA, in the past raises trust issues [35].

While NordLynx itself is not open-sourced, its foundation is, meaning that this protocol offers an acceptable amount of trustworthyness.

The other VPN-provider-made protcol, ExpressVPN is open-sourced in some form, via the lightway-core repository which contains certain components of the protocol. However the documentation of this repository is anything but in-depth and quite incomplete with lots of sections marked as "Coming Soon" [9]. ExpressVPN advertises that they run security audits on their software, including Lightway, [12], however, the last audit was two years ago [10]. Also, reviews online praising Lightway are financed directly by the parent company of ExpressVPN, namely Kape Technologies [36]. While this does not have any impact on the actual security of the protcol it at least raises a few eyebrows.

A trustworthy VPN provider should log as little user data as possible. In this regard, NordVPN stores only usernames and timestamps of their customers connections in order to determine how many concurrent users are active. This information is deleted 15 minutes after the session terminates [37].

ExpressVPN stores more information, including the days, on which a user has established a successful connection to which VPN server location from which country. They also log how much data has been transferred by a given user [38].

Surfshark stores metrics, such as how much data has been transferred by a user and the number of times they have used Surfshark's services. In addition, Surfshark collects data, including their users' mobile device id, the browsers they used and what network was used to access the VPN. They use this data in collaboration with advertisers to provide tailored ads to their customers [39].

Hide.me in comparison stores only very little data. Namely a user's, randomly generated, username and internally assigned IP address. This is only done for troubleshooting purposes and their logs are cleared every few hours. They also log traffic metrics of users in order to bill them properly [40].

6. Evaluation

In terms of security, all of the providers offer secure protocols, such as WireGuard, NordLynx or Lightway as their default. The most insecure protocol on offer by any provider is L2TP/IPSec, which Surfshark still supports. However in order to use L2TP, Surfshark users need to really go out of their way, as it is buried in options and configurations. They also make it clear in their online resources that they strongly advise against it's use [15].

In terms of VPN traffic obfuscation, ExpressVPN is the only provider which does not make wrong claims about obfuscated VPN servers that make traffic invisible. NordVPN [23], Surfshark [24] and Hide.me [25] all make these claims, which gives their customers a false sense of security [21, section 8].

The speed measurements make it clear that NordVPN and ExpressVPN are the fastest VPN providers. Therefore they are the most attractive provider for consumers who value faster connections. Surfshark falls slightly behind in terms of speed and Hide.me is by far the slowest provider among them.

In terms of transparency, NordVPN and ExpressVPN underperform as their custom protocols, NordLynx and Lightway, are both closed-source. Though NordLynx fares a little better as it is based off of WireGuard. Surfshark and Hide.me on the other hand only offer open-sourced protocol as their defaults. The logging policies of ExpressVPN and Surfshark are quite intrusive. ExpressVPN is capable of determining that a given user has accessed their services. This puts customers at risk that live in countries where VPN usage is illegal. Surfshark uses the data they log to collaborate with advertisers which should raise red flags for consumers who seek out VPNs to enhance their privacy online.

7. Related work

Other works have already evaluated VPN providers based on different criteria, such as speed, security, server locations and confidentiality [41] [42]. This paper is different from these evaluations because it focuses on the protocols offered by the providers instead of their general characteristics. It also performs measurements of all the available protcols instead of just using a provider's default. There are also papers which have shown that vpn providers make false claims [21, section 8]. Those are, however, often focused on certain aspects, such as the lack of obfuscation in a particular protocol. This paper instead offers a broad examination of several characteristics and puts them in relation to one another.

8. Conclusion and future work

Even though the set of observed providers is quite small with just four providers, it nevertheless showed that false and misleading claims are not uncommon in this industry. Many providers state that they are able to completely obfuscate VPN traffic or that they log zero information that can trace users back to them. Every provider that has been examined here is guitly of at least one of those claims. In addition, if a provider offers a custom protocol, it is advertised heavily and unrealistic claims about it are made, such that it is the "most secure" protocol in existence [12]. Also, all of the providers mentioned in this work offer at least one proprietary tunneling protocol. In future work, the set of examined providers could be expanded to include smaller providers that have a greater focus on transparency and trust.

References

[1] "Global number of breached accounts," https://www.statista.com/statistics/1307426/number-of-data-breaches-worldwide/, 2024.

[2] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*. Prentice Hall, 2011.

[3] "How Does a VPN Work?" https://www.paloaltonetworks.com/cyberpedia/how-does-a-vpn-work, 2024.

[4] M. Feilner and N. Graf, *Beginning OpenVPN 2.0.9*. Packt Publishing, 2009.

[5] "OpenVPN Protocol," https://openvpn.net/community-resources/openvpn-protocol/, 2024.

[6] E. Barker, Q. Dang, S. Frankel, K. Scarfone, and P. Wouters, "Guide to IPsec VPNs." NIST, 2020.

[7] J. A. Donenfeld, "Wireguard: Next generation kernel network tunnel." in *NDSS*, 2017, pp. 1–12.

[8] E. Juodytė, "NordLynx protocol – the solution for a fast and secure VPN connection," https://nordvpn.com/blog/nordlynx-protocol-wireguard, Jan 2022.

[9] ExpressVPN, "Lightway Core," https://lightway.com/docs/lightway-core/1.0.0/intro.html, 2024.

[10] "Pentest-Report ExpressVPN Lightway 10.-11.2022," https://cure53.de/pentest-report_expressvpn-lightway.pdf, 2022.

[11] "VPN reviews," https://www.top10vpn.com/, 2025.

[12] "Welche Arten von VPN-Protokollen gibt es?" https://www.expressvpn.com/de/what-is-vpn/protocols, 2024.

[13] "VPN protocols: L2TP/IPsec," https://www.expressvpn.com/what-is-vpn/protocols/l2tp, 2024.

[14] "hide.me VPN Protocols Explained," https://hide.me/en/knowledgebase/hide-me-vpn-protocols-explained, Dec 2023.

[15] "What protocols can I use with Surfshark?" https://support.surfshark.com/hc/en-us/articles/360010324739-What-protocols-can-I-use-with-Surfshark, Apr 2024.

[16] "Next-generation VPN encryption," https://nordvpn.com/features/next-generation-encryption/, 2024.

[17] "OpenVPN Security Advisories," https://openvpn.net/security-advisories/, 2024.

[18] J. Park, T. Ahn, and J. Ryou, "Efficient Analysis Method for IKEv2/IPsec Traffic Visibility: Applications on Mobile Platforms," in *2024 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, 2024, pp. 1–4.

[19] "ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)," https://datatracker.ietf.org/doc/html/rfc7905, 2016.

[20] "Windows Secure Socket Tunneling Protocol (SSTP) Remote Code Execution Vulnerability," https://msrc.microsoft.com/update-guide/vulnerability/CVE-2023-21535, 2023.

[21] D. Xue, R. Ramesh, A. Jain, M. Kallitsis, J. A. Halderman, J. R. Crandall, and R. Ensafi, "OpenVPN is Open to VPN Fingerprinting," *Commun. ACM*, Jun. 2024. [Online]. Available: https://doi.org/10.1145/3618117

[22] "Known Limitations," https://www.wireguard.com/known-limitations/, 2022.

[23] "Enable or disable Obfuscated servers," https://support.nordvpn.com/hc/en-us/articles/19615332252561-Enable-or-disable-Obfuscated-servers, 2024.

[24] "Use obfuscated servers for extra privacy," https://surfshark.com/features/obfuscated-servers, 2024.

[25] "VPN Obfuscation Methods: Hide That You Are Using VPN," https://hide.me/en/blog/vpn-obfuscation-methods/, 2024.

[26] "The fastest VPN gets even faster," https://www.expressvpn.com/de/lightway, 2025.

[27] "Get a high-speed VPN," https://surfshark.com/features/fast-vpn, 2025.

[28] "One of the fastest VPN providers on the market," https://nordvpn.com/features/high-speed-vpn/, 2025.

[29] "What makes Hide.me VPN stand out," https://hide.me/en/, 2025.

[30] "NordVPN server locations," https://nordvpn.com/servers/, 2025.

[31] "Full list of ExpressVPN server locations," https://www.expressvpn.com/vpn-server, 2025.

[32] "VPN server list," https://surfshark.com/servers, 2025.

[33] "hide.me's Worldwide VPN Locations," https://hide.me/en/network, 2025.

[34] C. Kaufman, P. E. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)," RFC 7296, Oct. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7296

[35] "Microsoft handed the NSA access to encrypted messages," https://www.theguardian.com/world/2013/jul/11/microsoft-nsa-collaboration-user-data, 2013.

[36] K. Hassel, "ExpressVPN Lightway Protocol Review and Comparison 2024," https://www.vpnmentor.com/blog/lightway-protocol-review/, 2024.

[37] "NordVPN Privacy Notice," https://my.nordaccount.com/legal/privacy-policy/nordvpn/, 2023.

[38] "ExpressVPN Privacy Policy," https://www.expressvpn.com/privacy-policy, 2024.

[39] "Surfshark Privacy Policy," https://surfshark.com/privacy, 2024.

[40] "Privacy Policy," https://hide.me/en/privacy, 2023.

[41] M. T. Khan, J. DeBlasio, G. M. Voelker, A. C. Snoeren, C. Kanich, and N. Vallina-Rodriguez, "An Empirical Analysis of the Commercial VPN Ecosystem," in *Proceedings of the Internet Measurement Conference 2018 (IMC '18)*. ACM, 2018. [Online]. Available: https://doi.org/10.1145/3278532.3278570

[42] O. Turki, "A comparison of VPN providers focusing on their security and speed." Turku University of Applied Science, 2024.

The State of DNS Delegation: Technical Challenges and Solution Approaches

Karoline Spohn, Patrick Sattler, Christian Dietze*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: karoline.spohn@tum.de, sattler@net.in.tum.de, diec@in.tum.de

***Abstract*—The Domain Name System (DNS) is a fundamental component of the Internet. Even so, DNS and DNS delegation face several challenges: nameservers have no way of signaling which protocols they support, there are complexities in domain control outsourcing, parent and child nameservers have inconsistent record sets and lame delegations occur frequently. This paper explores these challenges and presents two solutions proposed by the *DELEG* working group, which was brought to life by the IETF. The first solution places SVCB records under *_deleg* labels at zone apexes, offering simpler deployment but potentially doubling DNS traffic and undermining DNS's consistent naming structure. The second solution introduces a new DELEG record type, which maintains DNS's hierarchical integrity but requires more extensive infrastructure changes. Both solutions effectively address capability signaling and operator outsourcing through their use of SVCB-style records, but neither addresses the persistent issues of parent-child inconsistencies or lame delegations. Our analysis concludes that while the DELEG record approach offers better architectural alignment with DNS principles, significant work remains to address the full spectrum of DNS delegation challenges.**

***Index Terms*—dns, domain name system, dns delegation, deleg**

1. Introduction

The Domain Name System (DNS) has been a cornerstone of the Internet since its inception, allowing users to refer to web pages by their domains instead of their IP addresses [1]. As the Internet continues to grow and evolve, with billions of users and complex applications, the reliability and efficiency of DNS operations become even more critical. Although DNS has proven to be a scalable mechanism, certain challenges persist, in particular in relation to DNS delegation as well as nameserver operations. Operator outsourcing is associated with unnecessary overhead, which has become an increasingly prevalent problem as the relationships between domain owners, operators and registrars have become more complex. Despite the invention of additional protocols to enhance security, there is no standardized mechanism for nameservers to signal which protocols they support [2]. Furthermore, there are many cases in which inconsistencies between records in parent and child zones occur (section 3.3), and lame delegations (section 3.4) take place frequently.

These problems have significant practical implications for the Internet. In particular, they lead to additional traffic, slower query resolution, more overhead for the operators and even security weaknesses.

To combat some of these issues, the Internet Engineering Task Force (IETF) established the *DNS Delegation and Operation* (DELEG) working group.

This paper describes the functionality of the Domain Name System, analyzes present challenges, examines solutions proposed by the DELEG working group, and discusses remaining issues.

2. Background

Using domain names instead of numerical IP addresses yields many benefits: Domain names are easier to remember, less cumbersome to type out, and they do not change whenever a server is moved. To allow users to do just that, DNS was introduced: A hierarchical system functioning as the Internet's address book.

2.1. Domain Namespace

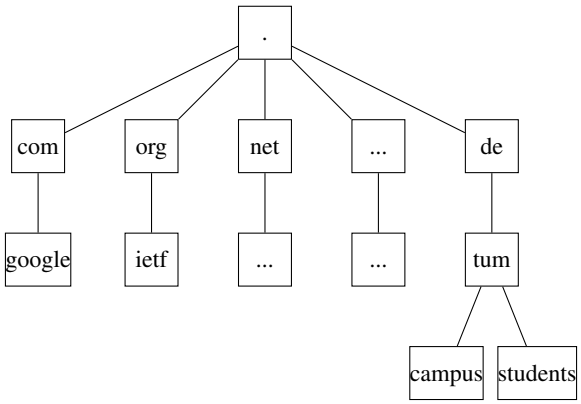


Figure 1: Hierarchical Namespace Structure

As seen in Figure 1, the domain namespace comprises a tree, where each node is a string representing a part of a domain name, called a *label*. At the root of the namespace, there is a ".". The first layer below the root contains top-level-domains (TLDs), such as the generic domains `.com`, `.net` and `.org` as well as the country-specific domains, e.g. `.us` and `.de`. Below each TLD, there are second-level domains, sold by different registrars. For instance, below `.com`, there is `google.com` and below `.de`, there is `tum.de`. Domain owners can create additional subdomains beneath their second-level domains. The owner of `tum.de` can create subdomains like `campus.tum.de`,

 doi: 10.2313/NET-2025-05-1_22

`students.tum.de` or even `accounts.students.tum.de`. Every sequence of labels is a domain-name, but a domain name is only called a *fully qualified domain name* (FDQN) if it is a continuous sequence of labels starting anywhere in the tree and ending at the root. Searching for a domain that is not fully qualified, such as `campus.tum`, yields no results. The "." at the end of an FDQN is left out, since it is redundant [1].

A domain can have associated resource records, consisting of fields for the domain name, the time to live, as well as the class, type, and value. The type specifies the kind of information held by the resource record. For instance, records with a *start of a zone of authority* (SOA) type define authoritative information about a DNS zone, whereas *A* and *AAAA* records map domain names to IPv4 and IPv6 addresses respectively. CNAME records are used to provide an alias for a domain. For instance, the CNAME record `www.example.de. IN CNAME www.tum.de.` aliases `www.example.de` to `www.tum.de`. When someone searches for `www.example.de`, they will be directed to `www.tum.de` [1].

2.2. Nameservers

Nameservers are responsible for answering DNS queries by providing authoritative information about their zones. The DNS namespace is split into multiple disjunct zones, which are managed by one or more nameservers. Each zone has a primary nameserver, and there can additionally be secondary nameservers, which replicate the data from the primary nameserver to create redundancy [1].

2.2.1. DNS Delegation. Parent nameservers can delegate authority over part of their namespace to child-nameservers in a mechanism called *DNS Delegation*. This is implemented through the use of Name Server records (NS records), where the authoritative nameservers for a certain zone are specified [3]. The border point between two zones is called a *zone cut* and the topmost node of a zone is calles *zone apex*.

The parent zone can enhance security by using delegation signer records (DS records) at a zone cut, specifying that the zone below is signed with a key stated in the record. Both the parent and the child zone have their own key pair. The child zone signs its own records with its own private key and shares its public key with the parent zone. The DS record contains a hash of the child zone's public key, and it is signed with the parent zone's private key. If a resolver trusts the parent zone's key, and can therefore verify the parent's records, it can also validate the child zone's key and their records. A chain of trust is created. These mechanisms are called *DNS Security Extensions* (DNSSEC) [4].

2.2.2. Resolver. DNS resolvers act as intermediary servers that accept client queries and resolve them. When a client queries the address `campus.tum.de`, the query resolution roughly works as follows:

1) A query is sent to a preconfigured resolver, which contacts one of the root servers.
2) The root server responds with NS records pointing to the authoritative nameservers for the `.de` TLD.
3) The resolver queries one of these nameservers, which responds with NS records for the authoritative nameservers managing `tum.de`.
4) The resolver again queries one of the given servers, and receives NS records for the authoritative nameservers managing `campus.tum.de`.
5) Lastly, the resolver queries one of these servers, retrieves A or AAAA records for `campus.tum.de` and returns them to the client.

This process is depicted in Figure 2. Note that caching mechanisms exist, but they are out of scope for this paper [1].

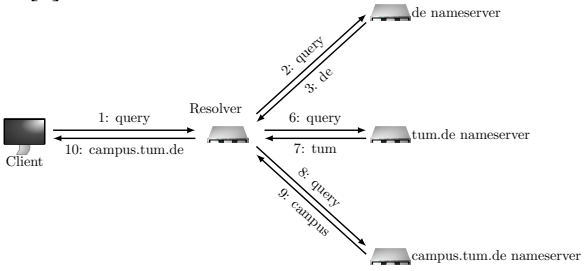


Figure 2: Query Resolution Graphic [1, Figure 7.6]

3. Issues with DNS Delegation

The DNS delegation mechanism has remained fundamentally unchanged for the last four decades. As such, it has become increasingly misaligned with modern requirements. Multiple issues have arisen in relation to DNS and DNS delegation, four of which will be described in the following paragraphs.

3.1. Capability Signaling

Various protocols have been developed, aimed at enhancing the security or performance of DNS . For instance, Encrypted Client Hello (ECH) encrypts client hello parameters like the Server Name Indication (SNI), thereby preventing observers from seeing which domain a client is attempting to connect to during the initial TLS handshake [5]. DNS-over-TLS [6], DNS-over-HTTPS [7] and DNS-over-QUIC [8] prevent man-in-the-middle attacks through query encryption. DNS-over-QUIC additionally prevents head-of-line blocking issues, which occur when packets in a network queue are delayed because the first packet in the queue is being processed or has been lost, even if the subsequent packets could be processed independently. However, nameservers have no way of communicating to resolvers which protocols they support [2]. Instead, resolvers have to spend additional time and resources to find out which protocols are supported, for instance by using dummy-queries [6]. This results in additional traffic as well as slower resolution.

3.2. Outsourcing Challenges

Domain owners frequently delegate control of their domains to web-hosting providers or DNS service providers, commonly referred to as operators.. While the DNS

protocol supports basic authority delegation through NS records, it lacks mechanisms for service providers to update nameserver configurations independently [2]. For instance, to change DNSSEC keys, the updated keys must be submitted to the registrar, which manages domain registrations, so they can be updated in the parent zone's records (see section 2.2.1). While an operator can create new keys and DS records, they cannot directly submit the records to the registrar, since they do not own the domain. Instead, they have to contact their customer and ask them to submit the records to the registrar, and then wait for the customer to do so. This three-way communication is suboptimal since it requires action from the customer, who may make mistakes or be slow to respond. It also creates additional overhead to the operator, in particular, if they are managing multiple domains [2].

3.3. Inconsistencies Between Parent and Child Nameservers

For a given domain, the sets of NS records (NSset) maintained by parent and child server should be identical, meaning both should agree on which nameservers are authoritative for that domain. However, Sommese et al. found that across the `.com`, `.org` and the `.net` zones, there were inconsistencies between the NSsets of parent- and child-nameservers for approximately 8% of the domains [9].

Four different types of inconsistencies in the NSsets were found: The parent and child nameservers' sets are disjunct, the parent set is a subset of the child set, the child set is a subset of the parent set or the sets have a non-empty intersection, but neither set is a subset of the other set. These inconsistencies result in additional latency, unresponsive nameservers, improper load balancing, added fragility and a higher risk of human error [9].

Disjoint sets may also cause lame delegations, which will be expanded on in section 3.4 [9].

To prevent these inconsistencies, operators should verify that parent and child are consistent, and resolvers should explicitly query child NS records when possible [9]. In addition, *Child-to-Parent Synchronization* records (CSYNC records) can be used, which provide an automated mechanism for signaling when delegation records in the parent zone should be updated to match the records in the child zone [10].

3.4. Lame Delegations

A lame delegation occurs when a DNS nameserver is listed in a domain's NS records, despite not being the authoritative nameserver for that domain. The DNS queries are sent to servers that lack information about the given domain, or even to hosts that do not run a nameserver [11].

In some cases, lame delegations only lead to additional latency, but the correct result is returned, since the query times out and is redirected to nameservers with correct configurations. Additionally, lame delegations add more traffic to other nameservers. Akiwate et al. [3] found that around 12% of the requests to GoDaddy's nameservers queried domains outside their authoritative zones. Lastly, lame delegations can be a security risk. When a domain has an NS record for a nameserver that is not registered, an attacker can register it and control the DNS resolution for all queries going through it. To prevent lame delegations, domain owners are advised to monitor the configurations of their own domains [3].

4. The DELEG Working Group

The DELEG working group was brought to life by the Internet Engineering Task Force (IETF) to address current issues of DNS. They have set the goal to address the challenges related to operator outsourcing as well as capability signaling (see sections 3.2 and 3.1 respectively). parent-child inconsistencies and lame delegations (see sections 3.3 and 3.4 respectively) are not objects of their work. Note that issues related to capability signaling are not directly related to DNS delegation, but they are still addressed by the DELEG Working Group.

4.1. Technical Foundations

To combat issues with capability signaling and outsourcing, two different solutions were proposed by the DELEG working group, both of which rely on SVCB records. These records consist of a `SvcPriority` field, a `TargetName` field, and a `ServiceParams` field. Depending on the value of the `SvcPriority`, a record is either in `ServiceMode` or in `AliasMode` [12].

Records in `ServiceMode` associate connection configuration parameters with service endpoints. The `TargetName` specifies the service endpoint (or `.` to indicate the owner name itself), and the `ServiceParams` field contains connection parameters such as supported protocols, IP hints, ports, and other configuration details that clients may use to reach the service. Potentially, this reduces the number of needed DNS queries [12].

Records in `AliasMode` serve a similar purpose to CNAME records; they provide an aliasing function where the `TargetName` field specifies another name whose SVCB records should be looked up for the actual service configuration. A key advantage over CNAME records is that SVCB records in `AliasMode` can be placed at the zone apex, which is not possible with CNAME records [12].

4.2. Proposed Solutions

The first solution that will be described is called *incremental deleg* and it places SVCB records under label in the zone apex. The second solution uses *DELEG* resource records, which are based on SVCB records, with only minor differences.

4.2.1. Commonalities of Incremental DELEG and DELEG Records. In both proposed solutions, SVCB records or DELEG records in `ServiceMode` are used to address challenges related to capability signaling. Through the `ServiceParams`, support for DoT, DoH, and DoQ can be signaled, but not for Encrypted Client-Hello. However, since `ServiceParams` were designed to be extensible, this could be added in the future [2], [13].

To solve operator outsourcing issues, using the respective records in `AliasMode` has been proposed. This offers

two advantages for operators. For one, the `AliasMode` functionality enables operators to have multiple customer domains point to a single service configuration [2], [13].

For another, dealing with DS records can be facilitated, because the operator no longer has to communicate changes in the DS records to the parent zone of the client's domain. Instead, the operator's zone is aliased under the respective records, and the operator simply needs to update the DS record for this zone, over which they have control. The operator can use DS records to establish a chain of trust with their own parent zone without involving the client's parent zone. This way, security is provided through DS records without the overhead of contacting the customer whenever a DNSSEC key change happens. Additional specification is still needed to define how exactly this mechanism will work [2].

Within a resource record set maintained by a nameserver, records in AliasMode and ServiceMode can be stored, whereby both problems can be solved simultaneously. While both proposed solutions leverage SVCB records for these capabilities, they differ significantly in how they integrate them into the DNS infrastructure.

4.2.2. Incremental DELEG. The first proposed solution, described by Homburg et al. [13], is storing delegation information under a special "_deleg"-label in the parent zone. For instance, for the domain `example.com`, the delegation information is stored at `example._deleg.com` in the parent zone. NS records for `example.com` and the new delegation information under the *_deleg* label are stored in the `.com` zone [13].

The query resolution process in incremental DELEG varies, based on whether a query is for the zone apex or not. Queries for the zone apex are resolved the same way traditional queries are resolved, since there is no delegation to a subdomain. For all other queries, two parallel queries are sent out by the resolver, as shown in Fig. 3: A legacy query following the conventional DNS resolution process, and an SVCB query directed at the *_deleg* label to obtain new delegation information. Three possible outcomes can result from the process. If an SVCB resource record set is found, the `TargetName` specified in that record is used as the nameserver. If the name of the domain does not exist or there is no *_deleg* record, *NXDOMAIN* is returned and the resolver falls back to using legacy queries. If the response is received but contains no data, *NOERROR* is returned [13].

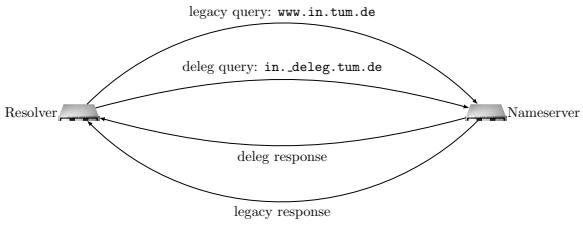


Figure 3: Parallel Queries

Incremental DELEG is compatible with existing DNS infrastructure because the delegation information stays in the delegation zone. This prevents the need for changes of the zone authority rules and most existing DNS software can remain unchanged [13].

4.2.3. DELEG Record. The second proposed solution, described by April et al. [2], is to implement a new DELEG resource record type, based on the SVCB format. DELEG records appear in the authority section of a DNS response, alongside NS and DS records. Like DS records, DELEG records only appear at zone cuts.

A key feature that distinguishes DELEG records from SVCB records is that parent zones can return DELEG records in `ServiceMode` and in `AliasMode`, which differs from standard SVCB specification. This allows operators to directly specify authoritative nameserver information through `ServiceMode` or manage multiple delegations through indirection by using `AliasMode` at the delegation point [2].

DELEG-aware resolvers should prioritize information in DELEG records over the information in NS and Glue records [2].

April et al. offer the following example of a query for the domain `www.example.com`. The resolver queries the root server about the given domain, and receives a response containing an NS record and a DELEG record. The NS record points to `ns1.example.com`, whereas the DELEG record points to `config1.example.com` with an IPv6 address hint. The resolver uses the information in the DELEG record to query `config1.example.com` and receives an A or AAAA record containing the IP-address of `www.example.com` [2].

4.3. Comparison

The deployment of incremental DELEG offers potential benefits in terms of simplicity, since it does not require a new record type. However, incremental DELEG has been criticized for other reasons. In particular, placing delegation control of a domain under a special *_deleg* label undermines DNS's consistent naming structures since parent zones no longer have direct control over their child domains. Furthermore, _deleg label may also make resolution more complex and traffic could potentially be doubled due to the parallel queries in incremental DELEG [2].

Meanwhile, DELEG records preserve DNS's consistent naming structure, but require introducing a new record type into the DNS protocol.

Both proposed solutions fulfill the DELEG working group's goal of simplifying operator outsourcing and capability signaling. However, as of February 2025, it has not been decided, which solution will be chosen.

5. Conclusion

This paper presents four major problems relating to DNS and DNS Delegation: A lack of a mechanism for the nameserver to signal supported protocols, complexities relating to operator outsourcing, inconsistencies between NS records in parent and child nameservers, and lame delegations. The DELEG working group addresses the first two problems. Two different approaches are proposed, both of which leverage SVCB records. The first proposed approach, *incremental DELEG*, places an SVCB record under a *_deleg* label in the zone apex. The second approach is introducing a new DELEG record, similar to SVCB records. Although incremental DELEG allows for

simple and smooth deployment, it leads to significantly more traffic and undermines the consistent naming structure in DNS. The other two issues have not been addressed by the DELEG working group and could be subject to future work.

References

[1] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. USA: Prentice Hall Press, 2010.

[2] T. April, P. Špaček, R. Weber, and D. C. Lawrence, "Extensible Delegation for DNS," Internet Engineering Task Force, Internet-Draft draft-wesplaap-deleg-01, Oct. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-wesplaap-deleg/01/

[3] G. Akiwate, M. Jonker, R. Sommese, I. Foster, G. M. Voelker, S. Savage, and K. Claffy, "Unresolved issues: Prevalence, persistence, and perils of lame delegations," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 281–294. [Online]. Available: https://doi.org/10.1145/3419394.3423623

[4] Ólafur Guðmundsson, "Delegation Signer (DS) Resource Record (RR)," RFC 3658, Dec. 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3658

[5] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, "TLS Encrypted Client Hello," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-esni-22, Sep. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-esni/22/

[6] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. E. Hoffman, "Specification for DNS over Transport Layer Security (TLS)," RFC 7858, May 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7858

[7] P. E. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," RFC 8484, Oct. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8484

[8] C. Huitema, S. Dickinson, and A. Mankin, "DNS over Dedicated QUIC Connections," RFC 9250, May 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9250

[9] R. Sommese, G. C. M. Moura, M. Jonker, R. van Rijswijk-Deij, A. Dainotti, K. C. Claffy, and A. Sperotto, "When Parents and Children Disagree: Diving into DNS Delegation Inconsistency," in *Passive and Active Measurement*, A. Sperotto, A. Dainotti, and B. Stiller, Eds. Cham: Springer International Publishing, 2020, pp. 175–189.

[10] W. Hardaker, "Child-to-Parent Synchronization in DNS," RFC 7477, Mar. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7477

[11] A. Romao, "Tools for DNS Debugging," RFC 1713, Nov. 1994. [Online]. Available: https://www.rfc-editor.org/info/rfc1713

[12] B. M. Schwartz, M. Bishop, and E. Nygren, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)," RFC 9460, Nov. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9460

[13] P. Homburg, J. van Zutphen, and W. Toorop, "Incrementally Deployable Extensible Delegation for DNS," Internet Engineering Task Force, Internet-Draft draft-homburg-deleg-incremental-deleg-00, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-homburg-deleg-incremental-deleg/00/

ISBN 978-3-937201-82-5
DOI 10.2313/NET-2025-05-1
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)