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)

Summer Semester 2025

March 6, 2025 – August 24, 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)

Summer Semester 2025

Munich, March 6, 2025 – August 24, 2025

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

Network Architectures
and Services
NET 2025-11-3

Proceedings of the Seminar
Innovative Internet Technologies and Mobile Communications (IITM)
Summer Semester 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 SS 25
Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)
Munich, Germany, March 6, 2025 – August 24, 2025
ISBN: 978-3-937201-85-6

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2025-11-3
Innovative Internet Technologies and Mobile Communications (IITM) NET 2025-11-3
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 Summer Semester 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 Julian Forster with the paper *Energy Consumption Reports Using Jupyter Notebooks* and Leonard Auer with the paper *QWACs in Automated Environments* .

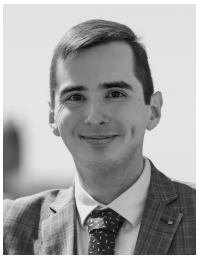
We hope that you appreciate the contributions of these seminars. If you are interested in further information about our work, please visit our homepage `https://net.in.tum.de`.

Munich, November 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

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

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

Kilian Holzinger (holzinger@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

Michael Oberrauch (oberrauc@net.in.tum.de)
Technical University of Munich

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

Manuel Simon (simonm@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/ss25/seminars/`

Contents

Blockseminar

Seminar

Energy efficiency of DPDK

Konstantin Fedorov, Stefan Lachnit*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: fedo@cit.tum.de, lachnit@net.in.tum.de

***Abstract*—This paper discusses methods for improving the energy efficiency of the Data Plane Development Kit (DPDK), a high-performance software developed by Intel for accelerated processing of network packets. The problems with its energy efficiency and approaches to their solution are also considered. Using DPDK makes it possible to achieve low latency, but the constant polling mechanism uses the CPU completely, which leads to inefficient use of resources. To reduce the load, we looked at various power management methods, as well as the associated implementation difficulties. We have reviewed dynamic voltage and frequency scaling (DVFS), low-power idle states (LPI), adaptive polling, and the application-level thread sleep method. The concept of increasing the flexibility of service cores, which allows for more efficient allocation of tasks, was also considered. In addition, this document discusses energy efficiency issues in the field of network function virtualization (NFV) as well as in 5G networks, where a mechanism such as hardware offload was also considered, which, together with the use of micro-sleep methods, can significantly reduce energy consumption.**

***Index Terms*—user-mode sleep states, NFV, PMD, EAL, DVFS, LPI, adaptive polling**

1. Introduction

The Data Plane Development Kit (DPDK) is a software product originally developed by Intel designed to speed up network packet processing. DPDK was initially created for telecommunications infrastructure, but today it is used in almost all areas, because it can provide high throughput and low latency for data transmission. DPDK is widely used in data centers, cloud systems, 5G networks and other areas where high-speed processing is required. The special feature of DPDK is that it operates in user-space, so it can directly interact with network equipment bypassing the kernel. Instead of the traditional method of processing packets through interrupts, DPDK uses an active polling mechanism called busy polling, in which processor cores constantly check for new data. However, busy polling always keeps threads active, even when there is little or no incoming data. Therefore, even under low network activity, the processor cores remain fully utilized, which leads to overheating of the CPU and reduces its lifespan. In data centers, network traffic usually has the characteristics of a "tidal effect ", meaning the amount of data fluctuates, with periods of high traffic, then low traffic. This behavior causes DPDK usage to result in significant energy losses. [1]

1.1. Important components of DPDK

1.1.1. Environment Abstraction Layer(EAL). EAL is an abstraction layer that provide a universal interface This enables to hide the differences between hardware platforms and OS from applications. [2] EAL is responsible for the DPDK initialization and startup process, allocation and management of low-level resources (memory, timers, etc.). In addition, DPDK provides mechanisms for binding cores to applications. [3]

1.1.2. Poll Mode Driver(PMD). PMD is a network interface driver that runs in user space. It can directly access the Network Interface Card's(NIC's) RX/TX queues. Standard driver implementations use an interrupt mechanism to process packets. This is inefficient because the system is forced to suspend the task at the time of interruption and switch the context, which spends system resources on interrupt maintenance and scheduling instead of completing the task itself. In addition, the system may be overloaded, thus the number of packets that it can process will be limited. By using the polling mode driver, the disadvantages of the interrupt method can be minimized. PMD runs in user space, but it has direct read and write access to the NIC port. This makes it possible to read packets from the network card without interrupting other processes. PMD constantly checks the network interface for incoming data, regardless of whether any packets have actually been received. This way much more packets can be processed, however packet processing applications will always use the CPU resources at full capacity, even if this is not necessary. [3]

1.2. Approaches To Speed Up Data Packet Processing

1.2.1. Zero copy. Many applications are intermediaries through which data is copied. Zero copy is a method that allows you to transfer data bypassing the application, copying data from a file on disk directly to a socket. This way it is possible to reduce the number of context switches between user space and the kernel. Using this method, the kernel copies data from a file on disk to a socket, avoiding copying through the application's memory. Zero copying allows you to transfer data about 65% faster than with the traditional approach. [4]

1.2.2. Busy polling. Busy polling is a feature designed for cases where low latency is necessary. Instead of an interrupt mechanism in which the processor waits for a

 doi: 10.2313/NET-2025-11-3_01

signal from the network card, DPDK uses this feature to continuously poll for new data. One of the main advantages of this approach is to avoid the overhead of system calls between user space and the kernel.

This allows DPDK to eliminate delays associated with interrupt processing, however, CPU cores will always be fully utilized regardless of the amount of work, which leads to excessive power consumption and reduced processor lifespan. [5]

1.2.3. Batch processing. The batch processing method allows processing multiple packets in one polling cycle, optimizing resource usage. It increases throughput and improves routing efficiency. [6] It could amortize the overhead of memory management, improve cache locality and prediction accuracy, which together significantly improves data processing rates and reduces CPU waiting time. [1] It also improves processor utilization by applying the same operations to a group of packets. [6]

2. Evaluation

This section will explore approaches to improve the Energy efficiency of DPDK. DPDK-based systems use various power management techniques that reduce energy consumption and increase energy efficiency. Some of these approaches will be explored below.

2.1. Dynamic Voltage and Frequency Scaling (DVFS)

DVFS is a technology that can be used to adjust the frequency and voltage of the processor depending on the current workload. This helps reduce temperature and power consumption. DVFS prevents system overheating, which avoids hardware damage. Decreasing the frequency increases energy efficiency but also limits the number of instructions executed per unit of time, accordingly reducing performance. [7] Simultaneously reducing both voltage and frequency can be applied for energy savings but also decreases the overall power with which the system operates. [8]

2.2. Low-Power Idle (LPI)

LPI reduces power consumption by disabling CPU subcomponents when there are no active tasks to process. When the system receives new data or requires actions, LPI reactivates previously disabled subcomponents. Because of LPI the CPU can switch to power-saving states (C-states) [9] during idle, thereby significantly reducing power consumption. [7] The use of standard implementations of such methods as LPI and DVFS has the following disadvantages:

1) CPU usage is always at its maximum, so it is impossible to adjust the frequency correctly depending on the load. [1]
2) It has been found that changing the processor frequency affects the packet transmission delay, but the optimal frequency that provides a balance between minimizing latency and power consumption is unknown. [1]
3) Switching to low-power states can take from tens to more than 100 microseconds, and besides, the CPU's exit from this state requires an interrupt, which leads to additional delays. [1]
4) Decreasing the CPU frequency may negatively affect the performance of other applications. [7]

2.2.1. C-states. Modern processors have the ability to transition into power-saving C-states. In the C0 state, the processor executes instructions, while in all other states, it switches to idle mode. The higher the C-state number, the more processor subcomponents will be disabled to save power. Intel processors provide special instructions for entering power-saving states. These commands help to reduce energy consumption by transitioning into deeper C-states. Standard instructions such as MONITOR/MWAIT and PAUSE [10] allow the core to switch to C1 and C6 states, and UMONITOR/UMWAIT and TPAUSE instructions [10] enable the core to enter C0 substates: C0.1 and C0.2. [11] Similarly, AMD processors provide similar MONITORX and MWAITX instructions for entering and exiting these low-power states. [12] The advantage of the C0.1 and C0.2 states is that their exit latency is significantly lower compared to, for example, the C1 state: C1 requires 2.2 more time to exit than C0.1 and about 1.8 more time than C0.2. Another advantage of these new instructions is that they do not require the intervention of the operating system and can be executed at the user-space level. In addition, the energy consumption in these substates is lower compared to the C0 state. Both substates consume about 30% of polling power, and C0.2 state is approximately 15% more power-efficient than C0.1. TPAUSE provides even faster exit delay compared to UMWAIT. TPAUSE is well suited for scenarios in which data arrives at fixed intervals, whereas UMWAIT will be better for situations in which a change in memory location is expected. Tests of the DPDK application with 16 cores showed that without traffic, the system consumes approximately the same amount of power when using C1 or C0.1/C0.2 and it was possible to save about 22% of energy. In a scenario that simulated real conditions at the lowest traffic rate, energy consumption was reduced by 4%. [11]

2.3. Adaptive Polling

Adaptive Polling is a method that allows applications to dynamically adjust the polling frequency of a network interface depending on the current network activity. Its main goal is to minimize the load on the processor during low network activity and maintain high performance during peak load periods. An increase in the polling interval leads to a decrease in the number of CPU cycles used to process a network packet. This increases the efficiency of applications by up to 60% and reduces the power consumption of the cores responsible for data transmission, and also reduces system power consumption by 6.5 watts with a low network load. Experimental data shows, that with a load of 10 Gbit/s, processor cores spend only 50% of the time processing packets, which indicates high optimization. In addition, energy savings reach 20 kWh per year due to lower energy consumption. Using dynamic polling frequency allows you to significantly reduce the

 doi: 10.2313/NET-2025-11-3_01

frequency of surveys. At 1 Gbit/s, the polling frequency is reduced by 34–87 million polls per second, which corresponds to a 97–99.94% reduction. At 10 Gbit/s, the reduction reaches 12–65 million polls per second, which is 90.82–99.28% lower than the standard values. When testing IMIX (Internet MIX) traffic, it was found that processing efficiency is increased by 50%, and the processor load is reduced to 16% of the total operating time. [13]

2.4. Application-Level Thread Sleep Method

The application-level thread sleep method allows a thread to enter a short waiting state if no packet is received after polling. This way, during periods of low activity, processor cycles can be freed up. Functions such as usleep and nanosleep in Linux, or special functions in DPDK (e.g., rte_pause() or rte_delay_us_block()) allow precise control of thread pausing, reducing the waiting time. However, the difficulty of this method lies in the fact that it is hard to determine the optimal "sleep" duration. This is due to the fact, that the duration of the sleep depends on various factors, such as packet size, frequency and overall traffic load. To solve this problem, a dynamic algorithm was developed that evaluates the packet size and transmission rate in real time and also determines the appropriate sleep duration. A modified Kalman filter prediction algorithm was applied to estimate the packet size and transmission rate. This allows for accurate prediction of idle time and reduces the number of empty polling cycles. Experimental results show that using this approach reduces CPU load by more than 80% with a slight decrease in transmission performance. For example, with a packet size of 512 bytes, the transmission speed drops by about 4.3%, and with 768 bytes by about 4.7%. Thus, this model can be applied to various network devices, such as 5G network packet processors, which will significantly reduce power consumption. [1]

2.5. Service Cores

DPDK provides a service cores mechanism that enables dynamic task distribution among CPU cores. This functionality is integrated into EAL, which provides an API that allows applications to manage service cores during runtime.

The DPDK Service core library is a software abstraction that allows you to hide the features of specific hardware or software. An important advantage of service core is that applications can distribute tasks between cores depending on the platform and environment. This way, you can compensate for the differences between the capabilities of different platforms. [14]

The developer can assign a task to a service core, and it will be able to independently determine whether the required hardware is present. If there is no such hardware, then the task will be executed via software. Each service runs on a dedicated service core, although the same core can serve multiple services. They are scheduled in a simple round-robin run-to-completion. If there are too many services on a single core, the processing latency of some services will significantly increase. In the standard DPDK implementation, the distribution of new services across service cores is done manually by the developer. This is a difficult task, and incorrect service assignment can reduce the throughput of individual cores and decrease their efficiency. For example, it is critical for packet processing services to maintain reliable throughput during packet transmission and reception. This creates the need for a way to migrate services between cores so that each can get enough CPU cycles without compromising other services. Using a load balancer allows each service core to theoretically run at about 100% of the CPU load, excluding the overhead incurred when switching between services. There are two strategies for implementing a load balancer: a static load balancer and a dynamic load balancer. [3]

- **Static load balancing.** assumes that the assignment services to cores is predefined (at the compilation stage or earlier). In the current DPDK implementation, developers themselves determine which cores will be used for service tasks and manually distribute services between them. There is a way to automate this process: When running, the DPDK application can scan which services are available and distribute them evenly across the cores.
- **Dynamic load balancing.** offers dynamic assignment of services to cores based on a continuous assessment of which core is best suited to execute a service. If there is a more suitable core, the service should be migrated there. Dynamic load balancing is more appropriate for DPDK, as CPU resource usage is constantly varying. When the load of the CPU decreases, the load balancer can migrate all services to fewer number of cores, and the remaining cores can either be disabled or put into idle state to save energy. A predefined activation threshold of the load balancer (for example, 70% of CPU load) ensures that the cores operate efficiently, preventing excessive peak frequency usage and freeing resources when load decreases. When fewer cores are active, the processor can redistribute energy and increase their clock frequency, which speeds up the performance for tasks requiring high performance in a single thread. [3]

3. Scopes of application

3.1. Energy Efficiency in Virtualized Network Functions (NFV)

Network Function Virtualization (NFV) allows the replacement of specialized hardware with virtual network functions (VNF) running on standard servers. This approach significantly reduces capital and operating costs, simplifying the scaling and maintenance of network infrastructure.

However, with all the advantages of NFV, energy efficiency remains an important issue, especially when using resource-intensive accelerators like DPDK, which often lead to energy overruns.

As part of NFV's energy efficiency research, various software platforms for high-speed network traffic processing compared, such as DPDK-OVS (Open vSwitch

 doi: 10.2313/NET-2025-11-3_01

with DPDK support), Click Modular Router, and Netmap VALE. DPDK demonstrates consistently high power consumption (about 138 watts) regardless of the load level, as it uses PMD, which makes processor cores work at 100% even in the absence of traffic. When processing virtual I/O (for example, when transferring packets via QEMU to virtual machines), DPDK-OVS turns out to be less efficient than other solutions. In addition, when running virtual network functions such as IDS/IPS Snort or Bro, the platform demonstrates lower energy efficiency compared to alternatives, especially when working with large packets and low or variable load. Although DPDK-OVS provides the highest performance and minimal latency, Netmap VALE shows greater potential in environments where power consumption is critical. [15]

3.2. 5G Energy Efficiency and User-plane Functions (UPF)

A network packet is the main component of a network. Micro-sleeps can be applied to all types of packet processing, but they are especially useful in the User Plane Function (UPF). UPF is one of the functions of the network, and it is a separate VNF function. Its main role is to forward packets to and from the Internet. Packets are usually processed by the kernel, but this approach has many disadvantages. System calls and copying packets to and from the kernel space lead to high overhead, which makes it difficult to scale and increase the packet transfer rate. Therefore, the DPDK network interface is used to process packets bypassing the kernel.

However, DPDK has the above-described problems that need to be solved: busy waiting consumes more energy than necessary and always loads the core at 100%, regardless of the load. In addition, DPDK doesn't provide information to make a decision on reducing power consumption. Several improvements have been made to resolve these issues. One of the methods is to switch to an idle state using the UMONITOR or UMWAIT instructions described earlier. This allows server processors to enter these states while waiting for network events. These instructions allow you to reduce power consumption even during very short idle periods than before, and can also be more efficient at higher data transfer speeds. When using user-mode sleep states, you can see the CPU thread load using the measurement functionality.

There is also a Hardware offload method that allows you to transfer processing of processor threads to the NIC. Accordingly, the processor core will be freed up and more CPU cores will have the opportunity to switch to microsleep. HW offload also offers other potential benefits with improved throughput and latency. Laboratory experiments have shown that the processor's power consumption has decreased from 190W to 61W and at maximum load from 190W to 145W. Even under the worst conditions, power-saving methods remain effective. The utilization of the technologies described above has reduced energy consumption by 68%. And with the maximum traffic load, power consumption was reduced by 24%. [12]

4. Conclusion

In this article, various approaches to improving DPDK efficiency were discussed, and special attention was paid to methods for reducing DPDK energy consumption. In particular, mechanisms such as zero copy, busy polling and batch processing were considered, which can significantly increase the speed of packet processing. We have considered methods such as DVFS, LPI and specifically C-states, adaptive polling, as well as the thread sleep method. We have proven that all these methods have had a significant effect on reducing energy consumption. For example, adaptive polling can increase application efficiency by up to 60% and reduce power consumption by up to 6.5 watts, while data processing efficiency can be increased by up to 50%. LPI provides energy savings of 15-30% using special C-states. It was also experimentally found out that the thread sleep method can reduce the load by more than 80% while slightly losing performance. In addition, the service cores method was considered, which allows you to redistribute the load and thereby reduce energy consumption. Also, we reviewed the application of DPDK in various fields such as 5G networks and NFV, and approaches for efficient energy consumption in these areas. The main contribution of this work is a detailed overview of the mechanisms for improving DPDK efficiency.

References

[1] Q. C. Mingjie Wu and J. Wang, "Toward low CPU usage and efficient DPDK communication in a cluster," 2021.

[2] *Improving the flexibility of DPDK Service Cores*, https://doc.dpdk.org/guides/prog_guide/env_abstraction_layer.html.

[3] M. J. Denis Blazevic, "Improving the flexibility of dpdk service cores," 2019.

[4] S. K. Palaniappan and P. B. Nagaraja, "Efficient data transfer through zero copy," 2008.

[5] *power-man*, https://doc.dpdk.org/guides-19.02/prog_guide/power_man.html.

[6] P. Okelmann, F. G. Leonardo Linguaglossa, and G. C. Paul Emmerich, "Adaptive batching for fast packet processing in software routers using machine learning," 2021.

[7] X. Li, T. Z. Wenxue Cheng, and B. Y. Fengyuan Ren, "Towards power efficient high performance packet i/o," 2020.

[8] R. B. Dongsheng Ma, "Enabling power-efficient dvfs operations on silicon," 2010.

[9] "Low-power idle states," https://edc.intel.com/content/www/us/en/design/ipla/software-development-platforms/client/platforms/alder-lake-desktop/12th-generation-intel-core-processors-datasheet-volume-1-of-2/009/low-power-idle-states/.

[10] *Intel® 64 and IA-32 Architectures Software Developer's Manual*, https://cdrdv2.intel.com/v1/dl/getContent/671110.

[11] D. Hunt, R. Pattan, P. Shah, R. Sexton, and C. MacNamara, "Power Management – User Wait Instructions Power Saving for DPDK PMD Polling Workloads," 2023.

[12] P. H. Leif Johansson and R. Skog, "Energy-efficient packet processing in 5g mobile systems," 2022.

[13] H. G. Trifonov, "Traffic-aware adaptive polling mechanism for high performance packet processing," 2017.

[14] *Service cores*, https://doc.dpdk.org/guides/prog_guide/service_cores.html.

[15] Z. Xu, F. Liu, T. Wang, and H. Xu, "Demystifying the Energy Efficiency of Network Function Virtualization," 2016.

Overview of Network Telescopes

Niklas Feurstein, Tim Betzer*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: niklas.feurstein@tum.de, betzer@net.in.tum.de

***Abstract*—Network telescopes, also known as darknets, are tools that aid in identifying and characterizing events that happen on the Internet. Without a clear-cut classification of the different network telescopes, it is difficult to keep track of them. Therefore, we present an overview of the different telescopes. In this paper, we classify those tools into two main groups. In addition to highlighting the main differences between passive and reactive network telescopes, we list the largest active telescopes and describe their properties.**

***Index Terms*—Network telescopes, Darknets, UCSD, Orion, NICTER**

1. Introduction

Internet security is becoming more and more important. Every day, thousands of machines are infected with malware. Malicious actors continuously try to discover unpatched vulnerabilities in online hosts. This not only puts companies and governments at risk of cybersecurity incidents but also endangers ordinary citizens. One thing is certain: The damages victims of hacking attacks suffer are not to be underestimated.

1.1. Tools for Security

To achieve their goals, hackers rely on a variety of programs and strategies. One of the first steps for cyber criminals is identifying a suitable victim. Instead of doing this manually, many hackers make use of automated programs called scanners. After a potential victim has been found, its computer can be attacked. [1]

In order to adequately defend against cybercrime, attacks should be detected as early as possible. Therefore, it is crucial that security researchers stay on top of the latest developments in the cybersecurity sphere.

To identify ongoing attacks, researchers and analysts need lots of data. Network telescopes provide this data by monitoring a part of the unused IP address space [2]. Packages sent to those regions are frequently related to malicious activities [3].

By leveraging this data, the latest exploits can be discovered and mitigated. Depending on their goals and objectives, researchers and companies make use of network telescopes. To make better use of those tools, organizations like IBM developed their own darknet. This allows them to have a tailor-made network telescope for their use case. [4]

1.2. Telescope Categorization

A categorization scheme is required to keep track of the multitude of telescopes.

One main difference between telescopes is how they react to incoming packets. Based on this characteristic, they have been grouped into two categories.

Passive telescopes store the data of the incoming package, but they do not send a response. They simply ignore the sender. Reactive telescopes, on the other hand, actively respond to incoming packages in real-time. This allows them to gain insight into more types of attacks. [5]

Chapter 2 presents the background information needed to understand network telescopes and their benefits. Chapter 3 explains how the telescopes work in general. In Chapter 4 and 5, we showcase the two main types of telescopes and list what we think are the largest projects in this field.

2. Background

To fully understand how network telescopes work, we need to look at the basics of networking using IPv4 and IPv6. Recognizing the benefit of darknets also requires us to examine the different types of attacks that cybercriminals use.

2.1. IPv4 Networking

Devices require an Internet Protocol version 4 (IPv4) address to communicate over the Internet. An IPv4 address identifies computers. It is 32 bits long. Therefore, the address space is limited to $2^{32} = 4\,294\,967\,296$ addresses. The 32 bits of an IP address are made up of a network identifier and a host identifier. [6]

One popular way of writing IPv4 addresses is the CIDR notation. In CIDR notation, the "/" character indicates how many bits are used for the network identifier. /x indicates that x bits are used to encode the network part of the address. The remaining 32-x bits identify the host. Such a network contains 2^{32-x} hosts. Using the CIDR notation is especially useful when it comes to network telescopes.

Assume a host sends a packet to a random IP address. We want to calculate the probability that the randomly chosen IP address is within a specific targeted address range. This probability is described by the ratio of targeted IP addresses to all the available IP addresses [2]. Using the CIDR notation, we can quickly compute this with the

 doi: 10.2313/NET-2025-11-3_02

following formula: $p_x = \frac{1}{2^x}$ So for a /10 network the probability is $p_{10} = \frac{1}{2^{10}} = \frac{1}{1024}$.

As mentioned, the number of IPv4 addresses is limited. Yet more and more people own a digital device and use it to connect to the Internet. The growing demand for IPv4 addresses can only be met using workarounds like Network Address Translation and dynamic IPv4 address assignment. [6]

Besides the very limited number of available addresses, there are other problems with IPv4. One of those problems is that IPv4 packages are not authenticated when they are transmitted. [6]

2.2. IPv6 Networking

The Internet Protocol version 6 (IPv6) is the successor to IPv4. In contrast to the previous version, the address length in IPv6 is much larger and totals 128 bits. So the available address space of 2^{128} addresses is 2^{96} times larger than IPv4. This larger address space prevents IP address exhaustion and ensures that hosts benefit from better network performance. [6]

The shift from IPv4 to IPv6 has already begun [6]. According to Fachkha and Debbabi [4], this far-reaching change impacts the attack and defense capabilities of hackers and security analysts. The larger address space and sparse distribution of addresses make it infeasible to scan all possible addresses [7]. This makes it harder for attackers to find victims. This is due to the fact that the number of devices with IP addresses assigned is only a fraction of the total IPv6 addresses. Fachkha and Debbabi [4] conclude that hackers need to scan more addresses to find one that is currently in use. In this case, the attackers also need to expend more computational and financial resources.

On the flip side, network telescopes will have trouble monitoring huge segments of the unused IP address space, as this would require more computation power. They are only able to monitor a fraction of the available data because packets from denial-of-service attacks, worms, and network scans are distributed among the entire IP address range. The smaller share of packets they can inspect is due to the lower percentage of addresses that they can monitor. [4]

2.3. Scanners

Scanners serve a variety of purposes. Amongst other reasons, they can be used by researchers to check if an IP address is currently in use or which services are running on a device. Those scanners can also be utilized by network administrators or security analysts for network maintenance or to provide a security assessment. [8]

However, they can also be used by malicious actors. In this case, scanners are often used to check the ports of a system for vulnerable applications. They not only focus on specific operating system exploits but also scan for vulnerable services such as SSL, SMTPS, and so on. [5]

Bakar and Kijsirikul [9] state that traditional scanners establish TCP connections to various ports of an IP address. They describe that using a particular payload, traditional scanners can check for vulnerabilities. Yet according to Li et al. [8], this type of scanner does not scale well. He and his team found that lots of processing power needs to be used upfront, without the scanner knowing if there is a machine behind the targeted IP address.

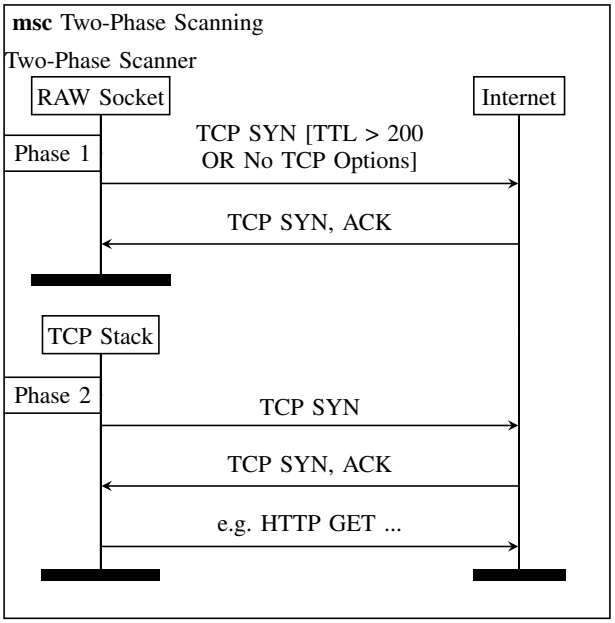


Figure 1: Two-phase scanning adapted from [5].

In order to prevent this waste of resources, the scanning process is split into two phases, as depicted in Figure 1. In the first phase, the scanner enumerates targets that need to be checked in more detail. This is achieved by sending an initial request to a host and waiting for a reply. If the host reacts to the original message, it is alive. [5]

In the second phase, the scanner will conduct a more in-depth scan of the targets that it has discovered. This approach reduces the scanning complexity and addresses the scalability issues. Two-phase scanners do not need to maintain a state, as they can make use of SYN cookies to encode information. [5]

Stateless scanners often use handcrafted packets to increase their scan speed. These irregular packages typically have larger TTL values, no TCP Options, or both. [5]

2.4. Malicious Activities

Hackers use a variety of strategies and actions to achieve their goals. They make use of network reconnaissance, spread malware, and try to disable computer systems. In this chapter, we only cover the activities that are related to network telescopes.

2.4.1. Denial-of-Service. A denial-of-service (DoS) attack describes an act by which an attacker sends a multitude of requests to a computer to make it unavailable to its normal users. One single device can hardly send enough requests to achieve this. Therefore, attackers rely on a large set of hosts to launch a coordinated attack on a victim. [4] [10]

In order to prevent the target from easily blocking such requests, the malicious actors spoof (forge) their IP

 doi: 10.2313/NET-2025-11-3_02

address. The spoofed IP address is typically chosen at random and included in the TCP package as the source address. When the victim receives a request, it sends a reply to the faked source address. As the attacker used a random IP address, the victim's response will be randomly distributed across the entire IP address space. Some of those packages will be directed to random addresses monitored by a network telescope. The addresses monitored by the telescope are globally routed but unutilized. This means that besides the telescope, no active hosts or services are assigned to them. For this reason, those unsolicited responses can indicate that the sender is a victim of a DoS attack. In order to classify such traffic as a denial-of-service attack, the number of packets, the attack duration, and the packets received per second need to be taken into account. [10]

2.4.2. Internet Worms. One more use case for network telescopes is detecting computer worms [4]. Viruses and trojans require human interaction to spread. According to Li et al. [11], worms can automatically infect other devices in a network. The researchers identified two classes of computer worms. Classical worms do not take measures to evade worm detectors. The more advanced evasive worms continuously evolve. Li et al. outline that this allows the evasive worms to bypass existing detectors.

Network telescopes make it possible to collect large amounts of data so that the characteristics and spread of computer worms can be better understood [12].

2.4.3. Network Scans. One of the most important steps in carrying out an attack on a network is reconnaissance [4]. Using network scans, malicious actors can identify vulnerable machines. If such attacks are detected early, security experts can effectively mitigate them. [3]

Network scanning accounts for most of the data received by the telescopes. TCP packets comprise most of the network telescope traffic. This is because the TCP protocol allows for a variety of scanning techniques. [4]

3. How do Network Telescopes work?

Network telescopes are one of the many tools in the arsenal of researchers and security analysts. In contrast to defending and protecting a particular company or set of devices, network telescopes are used to monitor large-scale events occurring across the Internet [3]. The intelligence gained from network telescopes can help protect all Internet users [4].

To accomplish their task, the darknets monitor globally routed and unutilized IP address space that has been assigned to them [2]. An address is *unutilized* if there are no devices assigned or services running on it besides the network telescope. As none of those IP addresses are allocated to a normal host, there is no legitimate traffic to those address ranges. The unrequested packages are typically referred to as Internet Background Radiation (IBR) [3].

This data can be analyzed to gain insight into various types of malicious events, which have been described in Section 2.4. Network telescopes have proven to be an invaluable tool for gaining information on the spread of malware, network reconnaissance, denial-of-service, misconfigured devices, and software bugs [2] [12] [13] [14] [15]. Security analysts and researchers can leverage this data to mitigate the damage of attacks currently in progress and to prevent similar future attacks [3].

Depending on the number of IP addresses a network telescope is monitoring, it can also detect rare network events and provide more context. Like astronomical telescopes that provide a better resolution, the larger its aperture, the resolution of a network telescope increases the more unutilized addresses it is monitoring. [2]

In practice, a network telescope is only useful if its resolution is large enough to witness the type of events we want to detect with high enough probability. Small telescopes with very few monitored addresses will only rarely detect important events. Therefore, analyzing them by classifying the events and counting them becomes unreliable. To get meaningful results, we require a lot of data. [2]

4. Types Of Telescopes

In order to better understand network telescopes, it is essential to classify them. Broadly speaking, there are two main categories of network telescopes.

4.1. Passive Telescopes

As the name indicates, passive telescopes simply capture the packages that have been directed to one of their monitored IP addresses. This rather simple approach allows them to surveil large blocks of the IP address space using minimal resources and processing power. They can be utilized to gain information on malware spread, DoS attacks, and misconfigured devices. Passive telescopes are also suited for detecting and identifying network reconnaissance. They can be used to monitor traditional scanners and also provide basic insight into the initial phase of stateless two-phase scanners. However, since passive telescopes do not respond to any of the received packages, they are not able to see the second phase of such a scanning effort. [5]

4.2. Reactive Telescopes

Most network telescopes are passive measurement instruments. Due to this, they can not fully detect all types of attacks. Reactive telescopes were invented to address the shortcomings of this approach. Reactive network telescopes respond to TCP SYN packages in real-time. By answering the original packages, a reactive network telescope continues the interaction and gains more insight as it receives more packages from the adversary. Therefore, this telescope category offers the additional benefit of fully detecting two-phase scanners. One example of a reactive network telescope is Spoki. [5]

5. Network Telescope Projects

Over the years, many network telescopes have been created. They were used for very specific purposes, and many of them have changed over the years. The aim of this section is to present a selection of large and influential network telescopes. The findings of this section are summarized in Table 1.

TABLE 1: Comparison of different telescopes

	Telescope			
Property	**Spoki**	**UCSD**	**Orion**	**NICTER**
Type	Reactive	Passive	Passive	Passive
Size	not actively deployed could handle /8 IPv4 prefixes	∼12,500,000	∼500,000	∼300,000
Use-Cases	Monitor two-phase scanners	Monitor DoS attacks, Internet worms and networks scans	Track botnets, monitor DoS attacks and network scans	Analyze network attacks

5.1. Spoki

Spoki is a reactive network telescope developed by the researchers Hiesgen et al. [5]. One of the aims of their newly developed telescope is to gain more insight into two-phase scanners. To achieve this, they deployed Spoki to four /24 IP prefixes. The research team demonstrated that their newly developed scanner can handle one million packages per second.

Spoki replies to the SYN packages that are sent to the unutilized IP space it monitors. If the original request is sent by a regular scanner, a host infected by a worm, or a misconfigured device, we do not gain any more information than from a passive telescope. In case the originator of the SYN package is a two-phase scanner, the scanner ignores the telescope's response at first. However, after a short delay, the scanner sends a regular SYN package, thereby starting phase two. Spoki also completes the second handshake, resulting in the stateless scanner sending its payload. The reactive telescope then stores the payload that the scanner sent and resets the connection. [5]

5.2. UCSD Telescope

This passive telescope is run and maintained by the University of California, San Diego (UCSD). Currently, the project makes use of a globally routed /9 and /10 network [15]. Those IP ranges have been allocated to "Amateur Radio Digital Communications" (ARDC) and are typically referred to as 44Net [16].

Before 2019, the network spanned more than 16 million routable IPs and made up $\frac{1}{256}$ of all IPv4 addresses. Nowadays, the network telescope contains approximately 12.5 million addresses and makes up roughly $\frac{1}{340}$ of all IPv4 addresses. [17]

A very small fraction of the IPs within this range are utilized by ARDC to educate their members on digital radio communication and to conduct experiments [15]. The UCSD telescope simply filters out legitimate traffic to the utilized addresses and focuses on those that are not assigned to an active host [15]. In 2018, the UCSD telescope received an average of 3.6 TB of network data per day [18].

The UCSD telescope has already been used in the past to detect events like DoS attacks, Internet worms, and network scans [15].

On July 19, 2001, a computer worm called Code-Red infected multiple hundred thousand machines. It caused economic damage exceeding $2.6 billion. Normally, analyzing the spread of such worms is very challenging. Using the UCSD network telescope, Moore et al. managed to detect more than 359,000 hosts infected with Code-RedI v2. To conclusively identify a machine as infected by the worm, it had to send two probes to IP addresses monitored by the network telescope. [12]

The UCSD telescope was also used by Dainotti et al. [19] to analyze the network scans conducted by the Sality botnet.

The researchers Gao et al. [14] developed the analytical framework DarkSim. DarkSim makes use of the UCSD telescope data to identify traffic patterns that need to be investigated further. The project was used to detect a change in scanning behavior after the disclosure of vulnerabilities related to Microsoft products. They also managed to gain insight into the systems conducting those scans.

5.3. Orion Network Telescope

The Orion Network Telescope is operated by the Merit Network, an independent nonprofit corporation run by universities in Michigan. The telescope is accessible to researchers. Like the UCSD telescope, it is entirely passive. One difference between those two darknets is that the Orion telescope makes use of /24 networks. By combining 1856 of those /24 networks, they created a network telescope that can monitor roughly 500,000 IP addresses. [20]

By aggregating all those networks, this telescope effectively tracks a /13 address block. On a typical day, the telescope run by Merit Network captures ∼100GB of compressed network data consisting of roughly 3 billion packages. [21]

All the traffic that reaches the Orion Network Telescope is saved in the PCAP file format [20]. The stored data includes the origin IP, targeted port, timestamp, and other information. In the past, the Orion Network telescope had coverage of roughly 75% of a /8 address block [22].

The Orion Network Telescope is used for tracking botnets, detecting scanners, and gaining insight into DoS attacks [20]. Near the end of 2016, the Mirai malware popped up. This malware infected IoT devices, which it then used to conduct DDoS attacks. Antonakakis et al. [23] made use of the Orion Network Telescope to retrospectively analyze how the botnet emerged. They also managed to provide a history of the botnet's DDoS victims. They found that during the 7-month timeframe starting from July 18, 2016, the Orion Network Telescope received roughly 1.6 billion packages per day.

5.4. NICTER

The Network Incident Analysis Center for Tactical Emergency Response (NICTER) project aims to analyze and understand ongoing network attacks [24]. This large-scale passive network telescope is run by the Japanese Research Institute NICT [24]. The project was launched in 2005 and monitored approximately 16,000 addresses. In the following years, the number of captured IP ranges constantly increased. Currently, the size of the network telescope hovers around 300,000 IP addresses. Using the captured data, NICTER publishes a detailed report every year. According to NICTER's yearly report, they observed roughly 1.9 billion packets per day in the year 2024. [25]

NICTER's darknet observations have been used to gain insight into botnets like Mirai and Hajime [26] [27].

6. Conclusion

In this paper, we examined multiple different network telescopes. All the telescopes were either classified as passive or as reactive. Chapter 4 also highlighted the key differences between those two categories. The main difference is how the telescope reacts to incoming packages. This behavior decides how well the network telescope can monitor tools used for network reconnaissance.

References

[1] W. Mazurczyk and L. Caviglione, "Cyber reconnaissance techniques," *Commun. ACM*, vol. 64, no. 3, pp. 86–95, Feb. 2021. [Online]. Available: https://doi.org/10.1145/3418293

[2] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Network Telescopes: Technical Report," Cooperative Association for Internet Data Analysis (CAIDA), Tech. Rep., July 2004.

[3] M. Kallitsis, R. Prajapati, V. Honavar, D. Wu, and J. Yen, "Detecting and interpreting changes in scanning behavior in large network telescopes," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3611–3625, 2022.

[4] C. Fachkha and M. Debbabi, "Darknet as a Source of Cyber Intelligence: Survey, Taxonomy, and Characterization," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1197–1227, 2016.

[5] R. Hiesgen, M. Nawrocki, A. King, A. Dainotti, T. C. Schmidt, and M. Wählisch, "Spoki: Unveiling a new wave of scanners through a reactive network telescope," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 431–448. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/hiesgen

[6] O. Babatunde and O. Al-Debagy, "A comparative review of internet protocol version 4 (ipv4) and internet protocol version 6 (ipv6)," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 13, no. 1, 2014.

[7] Y. Fang, L. Zhang, L. Li, C. Sun, Y. Guo, H. Zhang, B. Lin, J. Wang, and W. Xia, "An ipv6 address fast scanning method based on local domain name association," *Scientific Reports*, vol. 15, no. 11524, Apr 2025.

[8] G. Li, M. Zhang, C. Guo, H. Bao, M. Xu, H. Hu, and F. Li, "IMap: Fast and scalable In-Network scanning with programmable switches," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association, Apr. 2022, pp. 667–681. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/li-guanyu

[9] R. Abu Bakar and B. Kijsirikul, "Enhancing network visibility and security with advanced port scanning techniques," *Sensors*, vol. 23, no. 17, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/17/7541

[10] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring internet denial-of-service activity," *ACM Transactions on Computer Systems*, vol. 24, no. 2, pp. 115–139, May 2006. [Online]. Available: https://doi.org/10.1145/1132026.1132027

[11] J. Li, D. Sisodia, and S. Stafford, "On the detection of smart, self-propagating internet worms," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 3051–3063, 2023.

[12] D. Moore, C. Shannon, and J. Brown, "Code-Red: a case study on the spread and victims of an Internet worm," in *Internet Measurement Workshop (IMW)*, November 2002, pp. 273–284.

[13] K. Benson, A. Dainotti, k. claffy, A. Snoeren, and M. Kallitsis, "Leveraging Internet Background Radiation for Opportunistic Network Analysis," in *ACM Internet Measurement Conference (IMC)*, October 2015.

[14] M. Gao, R. Mok, E. Carisimo, k. claffy, E. Li, and S. Kulkarni, "DarkSim: A Similarity-Based Time Series Analytic Framework for Darknet Traffic," in *Proceedings of the 2024 ACM on Internet Measurement Conference*, November 2024.

[15] C. for Applied Internet Data Analysis, "The UCSD Network Telescope," https://www.caida.org/projects/network_telescope/, 2025, [Online; accessed 25-May-2025].

[16] A. R. D. Communications, "AMPRNet Wiki," https://wiki.ampr.org/wiki/Main_Page, 2024, [Online; accessed 25-May-2025].

[17] A. Camargo, L. Bertholdo, and L. Granville, "Less is more? exploring the impact of scaled-down network telescopes on security and research," in *Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2024, pp. 1050–1063. [Online]. Available: https://sol.sbc.org.br/index.php/sbrc/article/view/29854

[18] A. Mangino, M. S. Pour, and E. Bou-Harb, "Internet-scale insecurity of consumer internet of things: An empirical measurements perspective," *ACM Trans. Manage. Inf. Syst.*, vol. 11, no. 4, Oct. 2020. [Online]. Available: https://doi.org/10.1145/3394504

[19] A. Dainotti, A. King, K. Claffy, F. Papale, and A. Pescapé, "Analysis of a "/0" Stealth Scan From a Botnet," *IEEE/ACM Transactions on Networking*, vol. 23, no. 2, pp. 341–354, 2015.

[20] M. Network, "Orion Network Telescope," https://www.merit.edu/research/projects/orion-network-telescope/, [Online; accessed 25-May-2025].

[21] R. Prajapati, V. Honavar, D. Wu, J. Yen, and M. Kallitsis, "Shedding light into the darknet: scanning characterization and detection of temporal changes," 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, pp. 469–470. [Online]. Available: https://doi.org/10.1145/3485983.3493347

[22] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir, "Taming the 800 pound gorilla: The rise and decline of ntp ddos attacks," in *Proceedings of the 2014 Internet Measurement Conference*, ser. IMC '14, 2014, pp. 435–448. [Online]. Available: https://doi.org/10.1145/2663716.2663717

[23] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[24] NICTERWEB, "What is the NICTER Project?" https://www.nicter.jp/en/project, [Online; accessed 25-May-2025].

[25] N. I. of Information and C. Technology, "NICTER Observation Report 2024," National Institute of Information and Communications Technology (NICT), Tech. Rep., February 2025.

[26] S. Pham Anh and Y. Nakamura, "A baseline investigation into the evolution and prevalence of mirai and hajime utilizing a network telescope," *IEEE Access*, vol. 12, pp. 103 789–103 809, 2024.

[27] T. Kasama, "Long-term darknet analysis in nicter," *Journal of the National Institute of Information and Communications Technology*, vol. 63, no. 2, pp. 25–31, 2016.

Energy Consumption Reports Using Jupyter Notebooks

Julian Forster, Kilian Holzinger*, Sebastian Gallenmüller*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: j.forster@tum.de, holzinger@net.in.tum.de

***Abstract*—Reproducibility is a key aspect of scientific research, ensuring that experimental results can be independently verified. However, achieving reproducibility remains a challenge due to the additional effort required for documentation and automation. The plain orchestration service (pos) was developed to address these issues by enforcing reproducibility in network experiments. This paper presents an approach to enhance pos with an automated energy consumption reporting system based on Jupyter Notebooks. By integrating energy data collection through Prometheus and Grafana, we enable detailed insights into power usage during experiments. The proposed solution focuses on flexibility and adaptability, allowing it to be used across different environments. We discuss the architecture, implementation, and a case study demonstrating the practical application of the system.**

***Index Terms*—plain orchestration service (pos), Network Experiment, Energy consumption, automated reports**

1. Introduction

Having independent reproducible experimental results is crucial for any scientific research [1]. It makes results more trustworthy because it allows other researchers to recreate and verify the results. However, it is not that widespread to create reproducible experiments in the science community. The main problem is the increased effort researchers have to put into their experiments to make them reproducible, and it comes with some technical limitations [2] like privacy reasons or custom-built lab equipment resources.

To achieve that, the whole experiment process has to be automated (to prevent human errors) and documented. The documentation includes the hardware and environment, scripts and parameters used, and the actual results from the experiment. Depending on the experiment, more insights into the actual behavior of the hardware can be beneficial. This includes, for example, the energy consumption of a node during the experiment.

To encourage the creation of reproducible experiments, the ACM, which is the world's largest scientific and educational computing society, dedicated to advancing computing as a science and profession, introduced badges, which are awards for papers that make their experiments reproducible [3].

pos was created to assist researchers in achieving reproducibility with almost no additional effort, it is a methodology and a testbed where network experiments can be tested on [4]. The main advantage of pos is, that reproducibility was a main design consideration and, therefore, is enforced to run the experiments. At the end of an experiment, all the scripts, parameters, and results are saved and optionally published.

To gain deeper insights into the experiments, we want to create a dashboard that can collect the information from an experiment and give valuable insights into the energy consumption during the different runs. Although the dashboard is already created and works with pos [5], the goal is to make it flexible and adaptable to work in different environments and backends.

There are already some projects and tools supporting the creation of reproducible experiments. Some of them are OMF [6], which is a testbed controller, or SNDZoo [7] or WalT [8]. However, all of them only allow the creation of reproducible experiments but do not enforce them, which means that an experiment can be run without the proper documentation for other researchers. In the project [9], they tested the capability to do reproducible network experiments using container-based emulation. Adding a layer of abstraction always interferes with the results of an experiment, especially on low latency network experiments. These are the reasons why we want to focus on a solution, which works especially for pos, which is built for direct hardware access, but makes it adaptable for other projects and tools as well.

The remainder of this paper is structured as follows: Sec. 2 gives more background on the project, Sec. 3 talks about the approach, Sec. 4 describes a short case study, Sec. 5 talks about the future work, and Sec. 6 concludes the paper. Appendix 1 shows the output of the jupyter energy evaluation dashboard (all the required dependencies and code can be found in the repository [10]).

2. Background

This section focuses on the architecture and technology stack the testbed uses to run and evaluate reproducible experiments. First, we discuss pos and how it is built, and then the other technologies used to retrieve the energy data required for the jupyter-based experiment energy assessment dashboard.

2.1. Architecture of pos

pos was created with the following requirements for reproducible experiments in mind [4]:

 doi: 10.2313/NET-2025-11-3_03

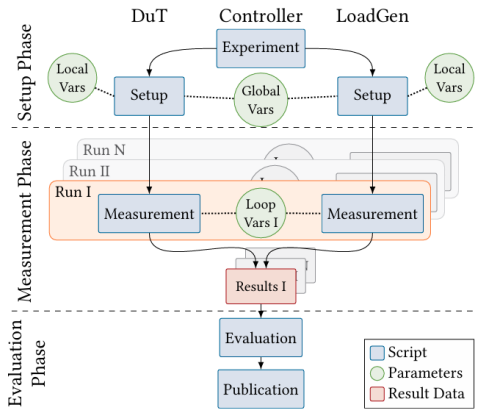


Figure 1: Architecture of pos [4]

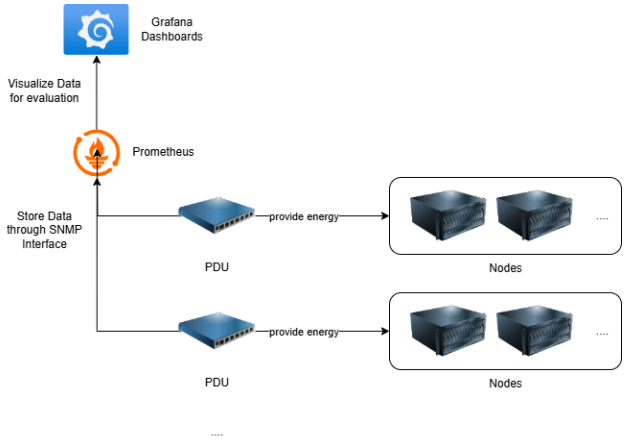


Figure 2: Architecture and Data Collection of the Testbed (diagram selfmade)

- *Heterogeneity (R1)* support for wide range of different devices
- *Isolation (R2)* the experiment nodes from non-experiment related devices
- *Recoverability (R3)* reverts devices into working mode even after a crash
- *Automation (R4)* avoids errors of misconfiguration
- *Publishability (R5)* document everything automatically

With these requirements in mind, pos architecture is built to enforce these with as little effort as possible. To achieve this, pos offers a flexible and well-defined workflow as shown in Fig. 1.

In the setup phase of the experiment, one or more nodes are allocated and booted with a predefined live image to ensure that, per each run, the experiment starts from the same point. After the boot, a basic setup script runs to make some configuration like installing network measurment software etc. After the setup phase, the actual measurement phase begins, where the nodes are monitored, and the results are stored. One experiment can involve multiple different runs with different parameters. pos separates the scripts and the according parameters into two files to ensure easy adaptation to different environments without needing to change the script. After all runs, the evaluation phase starts. Custom scripts can aggregate and further process the results and optionally publish them. It is currently used, for example, for TSN-based network experiments [11].

2.2. Energy Data Collection & Visualization

In the following, we discuss how the energy data is collected from the testbed and later evaluated and visualized in the current form.

As shown in Fig. 2, each node of the testbed gets its power through a power distribution unit (PDU), which can log the power consumption of each power outlet. In the current testbed of the chair, we have 4 PDUs connected to the nodes. A local instance of Prometheus, an open-source monitoring and alerting toolkit [12], collects and stores power consumption data via the SNMP interface. The stored metrics are then visualized using Grafana, a widely-used open-source platform for monitoring and observability [13]. This setup enables detailed visualization

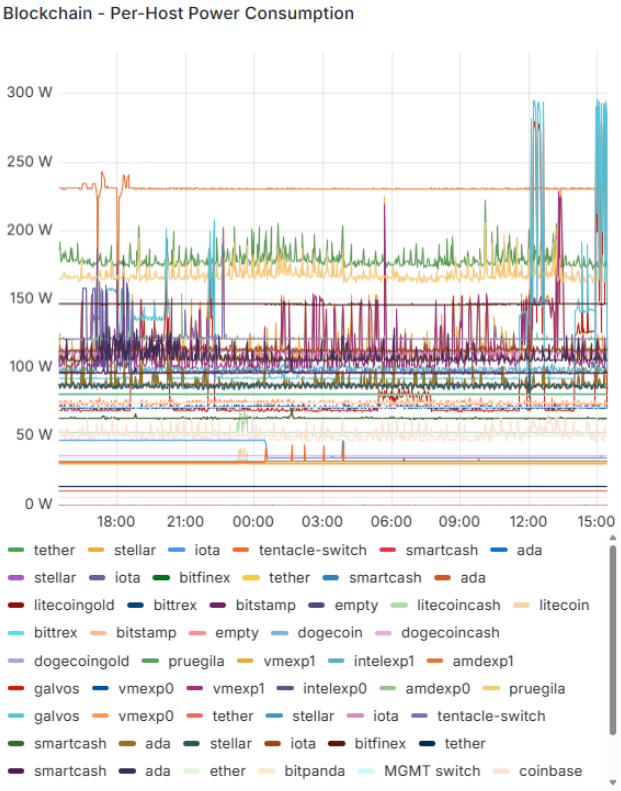


Figure 3: Per-Host Power Consumption in Grafana [14]

of per-node power consumption statistics, as illustrated in Fig. 3.

During each run, pos also save the energy consumption data and all the other experiment-related data in an RO-Crate format. So it is possible to retrieve the data through experiment results (only the used nodes and only for the specific time range) and through Grafana.

2.3. Jupyer Based Experiment Energy Assessment

In the current jupyter notebook [5], the output folder, which is in RO-Crate format, displays important data about the experiment and plots energy data for different

runs. These include current, voltage, power consumption, and aggregated power consumption. In specific, the notebook shows the following information:

- *Creator information* extracted from the RO-Crate metadata
- *Node information & topology visualization* extracted from the RO-Crate as well.
- *Power consumption over time* shows the power consumption each node has per run
- *Cumulative energy consumption*
- *Current and voltage trends*
- *Energy Consumption Rate Over Time* the rate at which energy is consumed over time (mW/s)

The dashboard currently uses two data sources to create the tables and plots: the *RO-Crate metadata file* and the *energy data folder* in which the data is stored in CSV format per each node and run with the following structure: current_mA, voltage_V, power_active_W, and energy_counter_Wh.

3. Approach

With our approach we want to make the dashboard more universally adaptable, as currently the dashboard can only work with the RO-Crate format. To do so, we made it possible for the dashboard to work with the Grafana API to get the required information.

3.1. Design

Grafana has no information about the experiments and related runs or used nodes. This is why in the first prototype, we used the pos python API to retrieve the reserved nodes as well as the date and time slots for each experiment. However, pos currently does not support querying entries from the past, which is why this is not a feasible solution. So for the calendar data, we used an exported CSV dump from the pos database. This is just a temporary solution until the pos API is further developed to get the data through the API directly.

We had multiple choices on how to implement the flexibility for the dashboard. One option was to utilize an object-oriented approach to make the dashboard extensible and open for multiple different interfaces and data sources. An alternative approach was to create different scripts that work as preprocessors, which collect the data from the sources and store them in a similar CSV energy format which then can be read by the dashboard.

The Grafana API currently only supports power consumption data. As already written, the pos API is currently too limited for practical use, which is why we had to switch to the CSV dump of the pos database. This shows that every testbed environment is unique, which makes it quite difficult with an object-oriented approach as this requires some similar processes to work correctly. So we decided to go with the preprocessor approach.

3.2. Implementation

For that, we created a new script, which queries the 4 PDUs of the pos testbed via the Grafana API, aggregates them together, and cleans up the data. Next, we take the information from the pos calendar CSV database dump and choose one experiment. The data is then exported to CSV file(s), which then can be read by the main energy consumption dashboard notebook as shown in Fig. 4. The "other data sources" in Fig. 4 show, that we can add any other data sources later, which work with the dashboard as well.

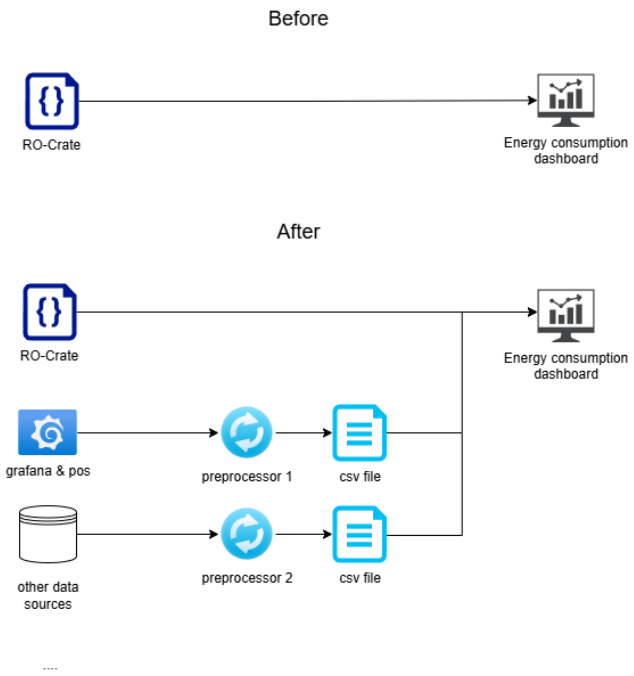


Figure 4: Energy Consumption Dashboard Architecture Before & After

Because only the RO-Crate Metadata file contains creator and node information, we can not show this data, when we use grafana and pos as the datasource. So we changed the dashboard script to be able to visualize the information with and without the RO-Crate Metadata file and with the CSV energy files.

Because we only get the power consumption data from Grafana, we also had to adjust the dashboard to be able to work with more or less data, depending on what is provided in the CSV file.

4. Case Study

In this section, we do a short example go through the project to show how the process works. First, we have to choose one experiment run. Because the web portal does not show any IDs for one experiment, we have to look that up in the CSV. For this example, we chose the entry with the id 7807, which uses in total three nodes (bitcoin, bitcoincash, bitcoingold) and it ran on 07.03.2025 between 12 am and 6 pm. Next, we query the data from Grafana in this time range. When we plot that data, we get the energy consumption of all the nodes. After some cleanup, we get the following diagram as shown in Fig. 5.

Now, we only have to filter for the nodes we are interested in and export that data to a CSV file. After we run the process, we can check the results by running

 doi: 10.2313/NET-2025-11-3_03

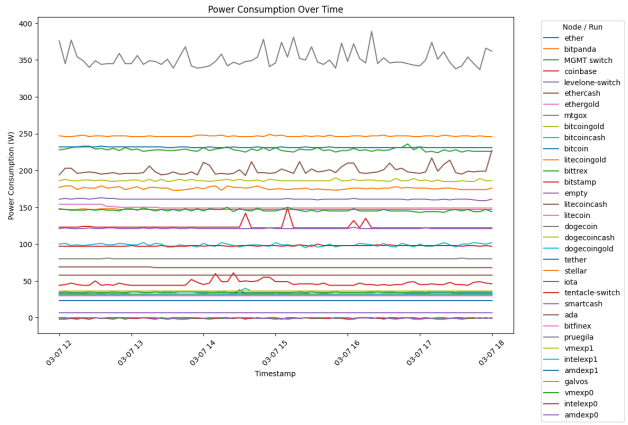


Figure 5: Power Consumption over Time for all Nodes

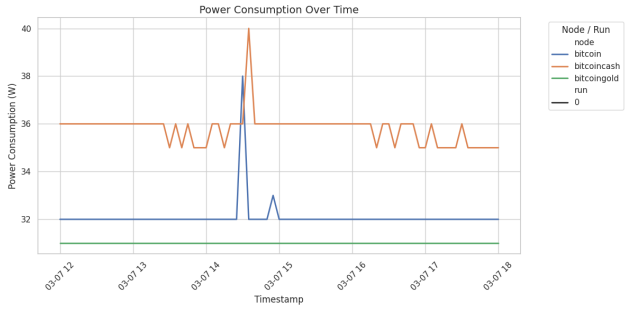


Figure 6: Example Power Consumption over Time for Three Nodes [5]

the dashboard script. We then get the energy consumption over time for the nodes as shown in Fig. 6. A full output of the energy consumption dashboard can be found in Appendix 1.

Because Grafana currently only supports power consumption, we had to set all the other columns to 0. Therefore the other diagrams are currently not working. As soon as we have the option to query more data, we can use all the other diagrams in the dashboard.

5. Conclusion

In this paper, we introduced an approach to improve the monitoring of energy consumption for network experiments conducted with pos [4]. Using Prometheus, Grafana, and Jupyter Notebooks, we developed a system that enables automated and reproducible power usage reporting. The implemented solution provides valuable information on trends in energy consumption and can be extended to other testbeds by adapting data sources.

References

[1] C. S. Collberg and T. A. Proebsting, "Repeatability in computer systems research," *Communications of the ACM*, vol. 59, no. 3, pp. 62–69, 2016.

[2] N. Zilberman, "An artifact evaluation of ndp," *Computer Communication Review*, vol. 50, no. 2, pp. 32–36, 2020.

[3] ACM. (2020) Artifact review and badging version 1.1. Last accessed: 2022-05-06. [Online]. Available: https://www.acm.org/publications/policies/artifact-review-and-badging-current

[4] S. Gallenmüller*, D. Scholz*, H. Stubbe, and G. Carle, "The pos Framework: A Methodology and Toolchain for Reproducible Network Experiments," in *The 17th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '21)*, Munich, Germany (Virtual Event), Dec. 2021.

[5] K. Warmuth. (2025) Greendigit evaluation repository. Chair of Network Architectures and Services, School of Computation, Information and Technology, Technical University of Munich, Germany. GitLab repository, last accessed: January 7, 2026. [Online]. Available: https://gitlab.lrz.de/GreenDIGIT/evaluation

[6] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "Omf: A control and management framework for networking testbeds," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4, 2010.

[7] M. Peuster, S. Schneider, and H. Karl, "The softwarised network data zoo," in *15th International Conference on Network and Service Management, CNSM 2019*. Halifax, NS, Canada: IEEE, Oct. 21-25 2019, pp. 1–5.

[8] P. Brunisholz, E. Dublé, F. Rousseau, and A. Duda, "Walt: A reproducible testbed for reproducible network experiments," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2016, pp. 146–151.

[9] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, 2012, pp. 253–264.

[10] J. Forster. (2025) Seminar: Energy consumption reports using jupyter notebook. Chair of Network Architectures and Services, School of Computation, Information and Technology, Technical University of Munich, Germany. GitLab repository, last accessed: January 7, 2026. [Online]. Available: https://gitlab.lrz.de/netintum/teaching/iitm/repos/2025ss-bs/u112

[11] M. Bosk, F. Rezabek, K. Holzinger, A. G. Marino, A. A. Kane, F. Fons, J. Ott, and G. Carle, "Methodology and infrastructure for tsn-based reproducible network experiments," *IEEE Access*, vol. 10, pp. 109 203–109 239, 2022.

[12] Prometheus Authors, "Prometheus: Monitoring system & time series database," https://prometheus.io, 2024, accessed: 2025-05-11.

[13] Grafana Labs, "Grafana: The open observability platform," https://grafana.com, 2024, accessed: 2025-05-11.

[14] Gude Dashboard, "Grafana dashboard of energy consumption per node," https://catalepsy.net.in.tum.de/d/8cJTjb1nk/gude?orgId=1&from=now-24h&to=now&timezone=browser, 2025, accessed: 2025-04-05.

Appendix 1: Output of the Jupyter Energy Evaluation Dashboard From The Run

EDITOR NOTE: The appendix is omitted in the proceedings. The full version including appendix is available at: https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2025-11-3/NET-2025-11-3_03.pdf

Firewalling with eBPF: A Performance Comparison of XDP-based Solutions

Sebastian Fritsch, Manuel Simon*, Sebastian Gallenmüller*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: s.fritsch@tum.de, gallenmnu@net.in.tum.de, simonm@net.in.tum.de

***Abstract*—Firewall capabilities on Linux are traditionally provided by the Netfilter framework. With the introduction of eBPF in kernel version 3.18 and XDP in kernel version 4.8, new ways of implementing firewalls on Linux have emerged.**

This paper compares three eBPF-based firewall implementations (`pcn-iptables`, `XDP-Firewall`, and `xdp-filter`), highlighting their architecture, capabilities and use cases. Those firewalls are evaluated in terms of latency and throughput within themselves and against a traditional firewall implementation. Our measurements show, that eBPF and XDP can be used to build firewall systems, that significantly outperform traditional firewalls, especially with large rulesets. However, we show that the performance of eBPF-based firewalls is not per se superior to traditional solutions, but is strongly dependent on how effectively eBPF features like hash tables are used.

***Index Terms*—XDP, eBPF, DDoS, Firewall**

1. Introduction

Web services have become central to everyday Internet usage, making their availability crucial for users and businesses. This increased dependence makes them attractive targets for various cyberattacks, which aim to overwhelm services, causing downtime and disruptions.

Traditionally, firewalls have been used as a primary defense mechanism to filter and block such harmful network traffic. Common firewall technologies, such as iptables and nftables in Linux, operate effectively but can become performance bottlenecks on large rule sets caused by complex attacks or high traffice volume. This limitation motivates the search for more efficient solutions.

In recent years, the introduction of technologies like the extended Berkeley Packet Filter (eBPF) framework and the eXpress Data Path (XDP) in the Linux kernel has opened new possibilities for firewall implementations. These technologies enable network packets to be processed much earlier and more efficiently within the kernel, significantly enhancing performance.

In this paper, we explore and compare these newer eBPF-based firewall solutions and evaluate their latency, throughput and overall performance within themselves and against traditional firewall implementations.

2. Background

To fully understand these modern firewall technologies, it is helpful to revisit traditional firewall concepts and the current Linux networking stack.

2.1. Network Firewalls

Firewalls can be categorized according to their position within the ISO/OSI model into three main types:

- **Packet-filtering firewalls** operate at the network layer and filter packets based on header information such as source and destination IP addresses, ports, protocols and flags. They can be implemented in hardware or software and are commonly used to control network access [1].
- **Circuit-gateway firewalls** function at the session layer. They verify whether a session, like a TCP connection, is permitted based on defined security policies [1].
- **Application-layer firewalls** operate at the application layer, inspecting the actual content of packets to detect and block malicious traffic. They are often used to protect web applications, email servers, and similar services from attacks [1].

Packet-filtering firewalls are further categorized as either stateless or stateful. Stateless firewalls filter each packet individually according to predefined rules without tracking the state of network connections. Stateful firewalls, on the other hand, maintain a state table that records active connections, allowing them to make more informed filtering decisions based on connection states [1].

Linux firewall implementations like iptables or nftables can support both stateless and stateful firewall configurations.

Firewalls can also be categorized by their deployment method:

- **Network-based firewalls** are positioned at the perimeter of the network, protecting the entire network from external threats. They are usually hardware-based and control network-wide access for multiple devices [2].
- **Host-based firewalls** run on individual devices, protecting them from both internal and external threats. They can are usually software-based and only control access to the individual device itself [2].

2.2. eBPF and Networking

The Berkeley Packet Filter (BPF) was first introduced in 1992 as a way to filter network packets directly in the kernel of Unix BSD systems. It used a simple virtual machine that executed filtering instructions [3].

 doi: 10.2313/NET-2025-11-3_04

However, the original BPF had some significant limitations—especially its inability to execute unbounded loops or function calls. These issues were addressed with the introduction of extended BPF (eBPF) in 2014. Designed as a universal in-kernel virtual machine, eBPF enables the execution of more sophisticated programs and extends its applicability beyond packet filtering into other kernel domains [3].

A particularly useful feature of eBPF for firewall implementations is the ability to create hash tables and arrays in the kernel using the `BPF_MAP_TYPE_HASH` and `BPF_MAP_TYPE_ARRAY` map types. These maps can be updated dynamically from user space, enabling efficient lookups and flexible firewall rules [4].

In the Linux network stack, eBPF programs can hook into different points to process packets. The following subsections describe the most relevant hooks.

2.2.1. Netfilter. Netfilter is a Linux kernel framework providing hooks for packet filtering and manipulation. The original BPF could be used by Netfilter, allowing users to attach BPF programs to Netfilter hooks through the `xt_bpf` iptables module [5].

Starting with Linux kernel version 6.4 in 2023, direct support for eBPF was added to Netfilter. Now, eBPF programs of type `BPF_PROG_TYPE_NETFILTER` can be attached directly to Netfilter hooks [4].

2.2.2. XDP. The eXpress Data Path (XDP) enables running eBPF programs as early as possible in the Linux networking stack—before packets undergo extensive processing or memory allocation (such as constructing the `sk_buff` structure for representing a packet and its metadata) [6].

XDP currently works only on the receiving (RX) side. An XDP program can instruct the kernel to handle incoming packets with one of five actions: `XDP_DROP`, `XDP_PASS`, `XDP_TX`, `XDP_REDIRECT`, or `XDP_ABORTED`. In firewall scenarios, the most important actions are typically `XDP_DROP` (drop the packet), `XDP_PASS` (accept the packet) and `XDP_TX` (return the packet on the same interface) [6].

Because XDP programs are compiled directly into native machine code using Just-In-Time (JIT) compilation, they can achieve a high performance by handling up to 24 million packets per second on a single CPU core [6]. XDP supports three modes of operation:

- **XDP Generic**: The XDP program runs within the kernel networking stack, similar to the ingress hook of Traffic Control (TC) (see Section 2.2.3) [3].
- **XDP Native**: The XDP program runs directly within the network driver, improving efficiency [3].
- **XDP Offload**: The XDP program runs directly on a compatible programmable network interface card (NIC) [3].

2.2.3. TC. Traffic Control (TC) is another kernel framework for managing network traffic through shaping, scheduling, and policing. Unlike XDP, TC allows attaching eBPF programs to the incoming (RX) and outgoing (TX) paths, giving it more flexibility for network management and firewall applications [7].

However, since packets must first be processed and allocated memory by the kernel, TC generally has a lower performance compared to XDP.

There are two main types of eBPF programs that can attach to TC:

- `BPF_PROG_TYPE_SCHED_CLS`: Classifies packets based on specified criteria [8].
- `BPF_PROG_TYPE_SCHED_ACT`: Specifies what action to take on packets after classification [8].

3. Overview of XDP-based Firewalls

Since eBPF was introduced into the Linux kernel, several projects have started using it at different points in the networking stack to build firewalls and protect against Distributed Denial of Service (DDoS) attacks. The currently available projects can be used both as a host-based or network-based firewall. Because eBPF runs directly in the kernel, most eBPF-based firewalls operate on layers 3 and 4 of the OSI/ISO model. They additionally can be combined with an application-layer firewall that runs in userspace.

In this section, we look at some of the more notable eBPF-based firewall projects, comparing their architecture and key features. A summary of our findings can be found in Table 1.

3.1. pcn-iptables

Polycube[1] is an eBPF and XDP-based framework providing tools for creating network services like firewalls, bridges, and routers. One of its components, `pcn-iptables`, is designed as a drop-in replacement for Linux iptables, offering the same syntax and behavior but using eBPF internally [9].

Architecturally, `pcn-iptables` uses the XDP hook to filter incoming traffic and the TC hook to filter outgoing traffic. Because these hooks run earlier in the kernel compared to traditional Netfilter hooks, `pcn-iptables` needs to reimplement some Netfilter features like connection tracking, forwarding, and NAT [9].

Despite the added complexity, `pcn-iptables` performs similarily or better than traditional iptables, especially when handling large rule sets. Its performance remains more stable under heavy workloads, experiencing a lower slowdown as the number of rules increases [9].

3.2. XDP-Firewall

XDP-Firewall[2] is an open-source firewall built entirely using eBPF and XDP. Unlike `pcn-iptables`, it only filters incoming packets (on the RX path) using XDP. Users can dynamically update firewall rules at runtime by editing a configuration file, which the program parses and loads into an eBPF map. The eBPF program iterates over these rules sequentially to determine if packets should be dropped or allowed.

During compilation, users can choose which network protocols (such as TCP, UDP, or ICMP) the firewall should

1. https://github.com/polycube-network/polycube
2. https://github.com/gamemann/XDP-Firewall

 doi: 10.2313/NET-2025-11-3_04

support. Limiting protocol support helps improve performance, as the program only parses the relevant headers.

XDP-Firewall provides basic packet filtering for IPv4 and IPv6 but does not support stateful filtering or connection tracking.

3.3. xdp-filter

`xdp-filter`[3] is a lightweight packet-filtering tool maintained by the official XDP project. It can filter packets based on MAC addresses, IPv4 and IPv6 addresses, and TCP or UDP ports. For TCP and UDP port rules, it uses an efficient eBPF array map with 65,536 entries (one per port), where each entry specifies a `XDP_DROP` or `XDP_PASS` action. MAC and IP address filtering uses an eBPF hash map for quick lookups. Similar to XDP-Firewall, it also is a stateless firewall.

3.4. Proprietary Solutions

Besides open-source projects, several proprietary solutions also use eBPF to provide firewall functionality, DDoS protection, and load balancing in enterprise environments.

Cloudflare, for example, initially relied on iptables to mitigate DDoS attacks. However, due to poor performance when dealing with high traffic volumes and complex rulesets, they transitioned to an eBPF-based solution. Their current system, named L4Drop, provides XDP-based filtering for incoming traffic. Abstract rules generated by their DDoS detection system, Gatebot, are dynamically compiled into eBPF programs and deployed directly onto their edge servers [5].

Another notable example is Meta, which utilizes XDP for high-performance packet processing. They open-sourced an XDP-based load balancer called Katran[4], which forwards packets directly to backend servers using the `XDP_TX` action. Additionally, Meta has published research describing their own XDP-based firewall, which processes packets before they reach the load balancer. This approach reduces unnecessary load balancing by filtering out packets that would eventually be dropped anyway [10].

4. Comparison of XDP-based Firewalls

Several studies have compared the performance of eBPF-based firewalls with traditional solutions like iptables and nftables. In terms of latency, simple XDP firewalls typically outperform Netfilter-based firewalls by around four times, although they increase the probability of latency outliers [11]. JIT compilation further improves performance, doubling the speed but again increasing the probability of outliers [11]. Regarding throughput, XDP firewalls running in native mode are particularly effective when handling a large number of rules, because eBPF allows for optimized rule matching using hash tables [9].

In this section, we compare the latency and throughput of the previously described XDP/TC-based firewalls: `pcn-iptables`, `xdp-filter`, and `XDP-Firewall`.

4.1. Measurement Setup

For our measurements, we connected two hosts using a 10 Gigabit Ethernet (GbE) link. One host acted as the load generator, while the second host functioned as the Device under Test (DuT), running the firewall being evaluated.

Both hosts had the following hardware specifications:

- CPU: AMD Ryzen 7 7700X (8 cores, 3.5 GHz)
- Memory: 64 GB DDR5 RAM
- NIC: Intel 82599ES 10-Gigabit Ethernet

We used iperf3[5] on the load generator running Ubuntu 24.04 (Linux kernel version 6.8) to generate traffic. The DuT ran iperf3 in server mode with the firewall under test. For compatibility reasons, `pcn-iptables` required Ubuntu 20.04, while the other firewalls were tested using Ubuntu 24.04.

To measure the average CPU cycles consumed by each XDP firewall, we used `bpftool`[6]. Specifically, we executed the command `bpftool prog profile`, which inserts performance counters via the `fentry` and `fexit` probes in eBPF programs, allowing us to measure the exact CPU cycles and instructions per program invocation. As `iptables` cannot be measured by `bpftool`, the latency was only measured for the XDP-based firewalls.

4.2. Latency

To evaluate latency differences between the firewall implementations, we measured the average CPU cycles per invocation of the XDP programs under varying numbers of firewall rules. For simplicity, we used basic TCP port-blocking rules equivalent to the following iptables rule:

```
iptables -A INPUT -p tcp --dport 80 -j DROP
```

The results of these measurements are shown in Figure 1. Overall, `xdp-filter` had the lowest latency in this test scenario, followed by `pcn-iptables` and `XDP-Firewall`.

An interesting observation is that the latencies of `pcn-iptables` and `xdp-filter` remain relatively stable as the number of firewall rules increases. However, the latency of `XDP-Firewall` increases almost linearly. This

3. https://github.com/xdp-project/xdp-tools/tree/main/xdp-filter
4. https://github.com/facebookincubator/katran
5. https://iperf.fr/
6. https://github.com/libbpf/bpftool

	Kernel Hook	Open-Source	Egress	Stateful	IPv4 L4 Protocols	IPv6 L4 Protocols
XDP-Firewall	XDP	Yes	No	No	TCP, UDP, ICMP	-
xdp-filter	XDP	Yes	No	No	TCP, UDP	TCP, UDP
pcn-iptables	XDP & TC	Yes	Yes	Yes	All	All

TABLE 1: Feature comparison of eBPF-based firewalls

 doi: 10.2313/NET-2025-11-3_04

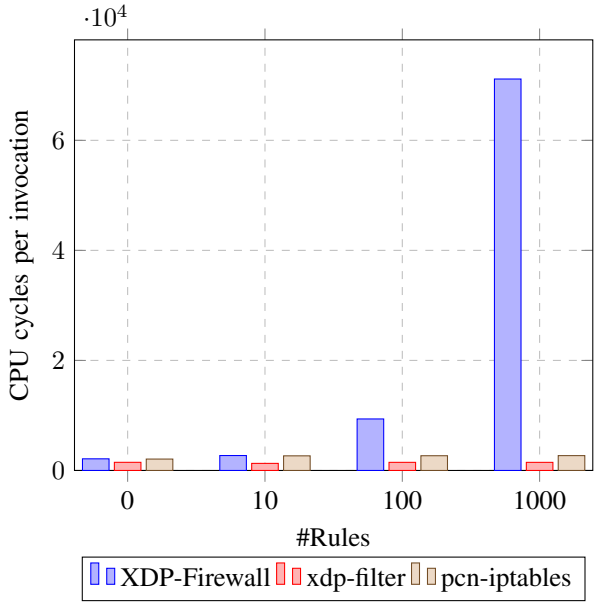


Figure 1: Latency of XDP-based firewalls

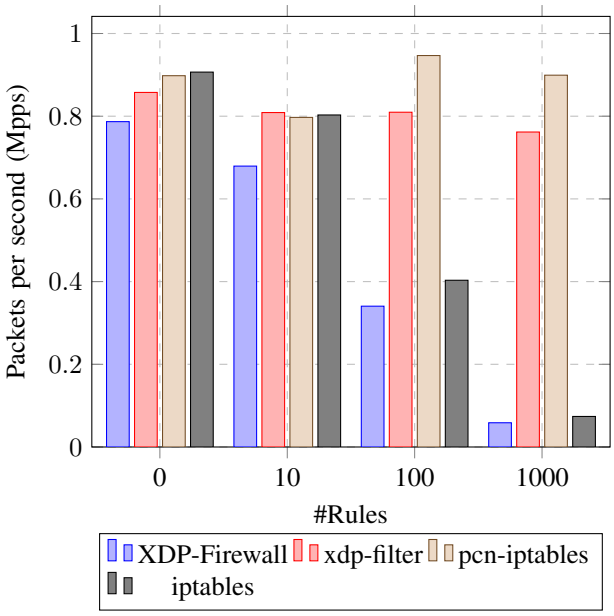


Figure 2: Throughput of XDP-based firewalls

happens because `XDP-Firewall` checks each rule sequentially in a loop until it finds a match or reaches the end.[7] In contrast, the other two implementations use an eBPF hash table, allowing them to quickly look up the appropriate rule after parsing the packet headers.

Since `pcn-iptables` is designed to handle more complex rulesets than `xdp-filter`, it employs the Linear Bit Vector Search (LBVS) algorithm to efficiently identify matching rules. This algorithm takes advantage of the fact that firewall rule sets are usually sparse, enabling it to significantly narrow down the search using a divide-and-conquer approach [12].

4.3. Throughput

Besides latency, we also measured the throughput of the different firewall implementations. Throughput tests were performed using iperf3 in UDP mode with a datagram size of 64 bytes, unlimited bandwidth, and a test duration of 30 seconds. The rules, that were employed on the firewalls were basic UDP port blocking rules equivalent to the following iptables rule:

```
iptables -A INPUT -p udp --dport 80 -j DROP
```

Figure 2 shows the results of these measurements. The throughput of `pcn-iptables` and `xdp-filter` remains relatively stable as the number of firewall rules increases, while the throughput of `XDP-Firewall` drops significantly.

Additionally, we measured the throughput of traditional iptables with the same ruleset and experimental setup. The throughput of iptables behaves similarly to `XDP-Firewall`, showing a linear decrease as more rules are added. This clearly illustrates the important relationship between the latency of individual XDP invocations and the overall throughput performance of firewall implementations.

7. https://github.com/gamemann/XDP-Firewall/blob/7dc351f3528ce00590ac16f6401f5a5be4c998d0/src/xdp/prog.c\#L363-L369

5. Conclusion and future work

In this paper, we have surveyed the current state of the art of eBPF-based firewalls, focusing specifically on XDP-based solutions. Our findings indicate that XDP-based firewalls are not inherently superior to traditional firewall approaches such as Netfilter. Instead, their effectiveness heavily depends on the specific implementation. However, when implemented efficiently and leveraging built-in eBPF features such as hash maps and JIT compilation, eBPF-based firewalls can clearly outperform traditional firewalls in terms of latency and throughput—particularly when handling large rulesets.

Despite the potential of eBPF-based firewalls, their adoption remains limited. This is likely due to the maturity and sufficient performance of traditional firewall tools like iptables, which already meet the requirements of many typical scenarios. Although projects like `pcn-iptables` aim to provide drop-in replacements for iptables using eBPF, they have yet to achieve widespread adoption and active maintenance.

Future work could involve developing more advanced XDP-based firewalls that support complex, stateful rulesets. Another promising direction could be exploring the implementation of a whole Layer 5 or even Layer 7 firewall capabilities entirely within eBPF.

References

[1] M. Mihalos, S. Nalmpantis, and K. Ovaliadis, "Design and implementation of firewall security policies using linux iptables," *Journal of Engineering Science & Technology Review*, vol. 12, no. 1, 2019.

[2] R. Alsaqour, A. Motmi, and M. Abdelhaq, "A systematic study of network firewall and its implementation," *International Journal of Computer Science & Network Security*, vol. 21, no. 4, pp. 199–208, 2021. [Online]. Available: https://koreascience.kr/article/JAKO202121055727021.pdf

[3] M. A. M. Vieira, M. S. Castanho, R. D. G. Pacífico, E. R. S. Santos, E. P. M. C. Júnior, and L. F. M. Vieira, "Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications," *ACM Comput. Surv.*, vol. 53, no. 1, Feb. 2020. [Online]. Available: https://doi.org/10.1145/3371038

[4] Reimerink, Dylan and Chen, Ian, "eBPF Docs - BPF_PROG_TYPE_NETFILTER," https://docs.ebpf.io/linux/program-type/BPF_PROG_TYPE_NETFILTER/, 2024, [Online; accessed 20-March-2025].

[5] G. Bertin, "Xdp in practice: integrating xdp into our ddos mitigation pipeline," in *Technical Conference on Linux Networking, Netdev*, vol. 2. The NetDev Society, 2017, pp. 1–5.

[6] T. Høiland-Jørgensen, J. D. Brouer, D. Borkmann, J. Fastabend, T. Herbert, D. Ahern, and D. Miller, "The express data path: Fast programmable packet processing in the operating system kernel," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 54–66. [Online]. Available: https://doi.org/10.1145/3281411.3281443

[7] M. Bertrone, S. Miano, F. Risso, and M. Tumolo, "Accelerating linux security with ebpf iptables," in *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 108–110. [Online]. Available: https://doi.org/10.1145/3234200.3234228

[8] Reimerink, Dylan and Chen, Ian, "eBPF Docs - BPF_PROG_TYPE_SCHED_CLS," https://docs.ebpf.io/linux/program-type/BPF_PROG_TYPE_SCHED_CLS/, 2024, [Online; accessed 20-March-2025].

[9] S. Miano, M. Bertrone, F. Risso, M. V. Bernal, Y. Lu, and J. Pi, "Securing linux with a faster and scalable iptables," *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 3, p. 2–17, Nov. 2019. [Online]. Available: https://doi.org/10.1145/3371927.3371929

[10] A. Deepak, R. Huang, and P. Mehra, "ebpf/xdp based firewall and packet filtering," in *Proceedings of the Linux Plumbers Conference*, 2018, pp. 1–5. [Online]. Available: http://oldvger.kernel.org/lpc_net2018_talks/ebpf-firewall-paper-LPC.pdf

[11] D. Scholz, D. Raumer, P. Emmerich, A. Kurtz, K. Lesiak, and G. Carle, "Performance implications of packet filtering with linux ebpf," in *2018 30th International Teletraffic Congress (ITC 30)*, vol. 01, 2018, pp. 209–217.

[12] M. Tumolo, "Towards a faster iptables in ebpf," Master's thesis, Politecnico di Torino, 2018. [Online]. Available: https://webthesis.biblio.polito.it/secure/8475/1/tesi.pdf

Congestion Control Schemes for Multipath QUIC

Julian Gassner, Daniel Petri*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: julian.gassner@tum.de, petriroc@net.in.tum.de

***Abstract*—The current IETF draft for MPQUIC employs a per path instance of QUIC's default congestion control algorithm for calculating path congestion windows. The lack of fairness between paths and the negative performance impact of competing standard single-path protocols are notable problems. Building upon the lessons learned from MPTCP, two novel approaches to MPQUIC congestion control called CC-OLIA and S2B2C mitigate these issues.**

***Index Terms*—MPQUIC, MPTCP, congestion control algorithm, OLIA, CC-OLIA, S2B2C**

1. Introduction

The Multipath Extension for QUIC (MPQUIC) [1] started development in October 2017 and is currently in its 13th IETF [2] draft iteration [1]. It seeks to standardize the use of multiple QUIC paths simultaneously within a connection to improve throughput and reliability [1]. However, to date, the draft still recommends the use of one instance per path of the standard QUIC congestion control algorithm as defined in RFC 9002 [3]. This causes several problems, most notably the lack of fairness between paths [1] and the negative performance impact on competing standard single-path protocols [4]. This paper looks at how TCP's Multipath Extension (MPTCP) [5] works and how it handles congestion control (CC). We discuss the basics of MPQUIC and analyze its default CC algorithm, and show two novel approaches to MPQUIC CC proposed by H. Wang et al. and Deng et al. named CC-OLIA and S2B2C that mitigate these issues.

2. Background

In this Chapter, we look at TCP's Multipath Extension [5] and its main CC algorithms LIA [6] and OLIA [7]. In addition, we analyze the functionality of the QUIC Multipath Extension [1]. We contextualize this to lay the groundwork for discussion of the multipath QUIC CC algorithms.

2.1. TCP's Multipath Extension

According to RFC 8684 [5], Multipath TCP (MPTCP) allows multiple paths to be used simultaneously between peers. This behavior contrasts standard TCP connections where only one path is used at a time. In this context, a path is defined by TCP's 4-tuple containing the source address, the source port, the destination address, and the destination port of the connection. Using multiple TCP paths has the advantage of making better use of available network resources. It also results in higher throughput and improved resilience to network failures. The connection multiplexing takes place entirely within the TCP layer, and MPTCP connections provide the same interfaces to the other layers as standard TCP connections. This makes it possible to utilize the benefits of MPTCP without the need to change intermediaries and the applications that work upon it. [5]

2.1.1. MPTCP Functionality. The multiple path connections are established using distinct network interfaces with different IP addresses on both hosts. From an external perspective, a MPTCP connection behaves and operates like any other TCP connection, the network layer is divided into multiple *subflows*. A sub-flow represents a stream of segments over a single path from a given TCP connection. The MPTCP extension creates, manages, and deletes these subflows, which all operate on a path identified by the 4-tuple. The number of currently active subflows can vary throughout the connection. [5]

Each sub-flow is an instance of a classic TCP flow with the addition of a new TCP option type. This option type has various subtypes used during the different states of a connection. The initial connection establishment for a MPTCP connection follows the same flow as normal TCP connections (SYN, SYN/ACK, ACK) and utilizes only a single path. However, each packet contains a MP_CAPABLE subtype option to declare that the sender of that packet wants to utilize MPTCP for the given connection, which MPTCP version (`v0` or `v1`) the sender is capable of running, and various flags that define the connection's used features. It also exchanges the keys of the connection partners used to authenticate later-created subflows. [5]

After the connection is established, hosts can create any number of new subflows using currently unused IP address pairs. Hosts can indicate available IP addresses on their side of the connection using the ADD_ADDR option subtype. As with the master connection, each new subflow starts with the standard TCP connection establishment procedure. However the subflows' packets do not contain the MP_CAPABLE subtype option but the MP_JOIN subtype option. It enables the flow establisher to inform the receiver which connection this subflow belongs to by sending a token generated from the receiver's key exchanged during the master connection establishment, which identifies the master connection. This happens in the SYN segment. The SYN, SYN/ACK, and ACK seg-

 doi: 10.2313/NET-2025-11-3_05

ments all contain the MP_JOIN option subtype and use this option subtype, besides the already mentioned purpose, for replay attack and integrity protection. Once the connection setup is finished, MPTCP is ready to transfer data. Data transfers over the different subflows happen by splitting the data to be sent and reassembling it on the receivers' side. This is done by using the DSS option subtype. It carries a Data Sequence Number (DSN), which enumerates the separated data used to indicate how to reassemble it. The DSS also contains a Data ACK field used to acknowledge given data parts by their DSN in their given subflow. This ACK is one of two available ACKs in MPTCP, with the other one being the standard TCP ACK used to acknowledge the segments in general and not a specific DSN. [5]

2.1.2. Congestion Control Algorithms for MPTCP. Several types of CC algorithms are used for MPTCP. RFC 8684 [5] suggests using the Coupled CC algorithm, also known as the Linked Increases Algorithm (LIA), which is defined in RFC 6356 [6]. The idea of LIA is to keep most of the original TCP CC algorithms, such as the slow start, fast recovery, and fast retransmit algorithms defined in RFC 5281 [8], and only change the congestion avoidance algorithm [6]. For this RFC 6356 [6] defines several variables similar to the variables defined in RFC 5281 [8] for single path connections:

- $cwnd_i$: Congestion windows of the i[th] subflow in bytes.
- $cwnd_{\text{total}}$: Sum of all congestion windows across all subflows in bytes.
- p_i: Loss rate of a subflow i.
- rtt_i: Round trip time of a subflow i.
- MSS_i: Maximum segment size of a subflow i in bytes.

Each time a segment ACK is received in a subflow during the congestion avoidance phase as defined in RFC 5281 [8], the variable $cwnd_i$ is updated using Formula 1, where b is the number of bytes acknowledged in the ACK and the aggressiveness α is defined in Formula 2 [6].

$$\Delta cwnd_i = \min\left(\frac{\alpha \cdot b \cdot MSS_i}{cwnd_{\text{total}}}, \frac{b \cdot MSS_i}{cwnd_i}\right) \tag{1}$$

$$\alpha = cwnd_{\text{total}} \cdot \frac{\max_i\left(\frac{cwnd_i}{rtt_i^2}\right)}{\left(\sum_i \frac{cwnd_i}{rtt_i}\right)^2} \tag{2}$$

Using this approach, LIA achieves two goals for multipath CC algorithms. The first goal, called "Improve Throughput" [6] states that the multipath approach performs at least as well as a single path approach does on its best path. This is done by introducing α in Formula 2 to ensure that the combined throughput of all subflows aligns with what one single TCP flow would achieve. Secondly, the multipath approach can only put the same load on any given resource, for example, an intermediate router, on any given path as a single path approach would ("Do no harm" [6]). LIA implements this in the calculation of the $cwnd_i$ increase in Formula 1 to ensure that the increase never exceeds a standard TCP flow congestion window increase. However, the RFC also points out a third desirable goal for multipath CC algorithms, which cannot be achieved by using LIA. It states that a multipath CC algorithm must be able to offload as much traffic as possible from congested paths with respect to goals one and two ("Balance congestion" [6]). Furthermore, LIA shows so-called *flappiness*, meaning that when all subflows exhibit the same congestion behavior, LIA tends to assign the entire total congestion window to one subflow while all other subflows have a congestion window of 0. [6]

To mitigate these issues, the Opportunistic Linked-Increases CC algorithm for MPTCP (OLIA) [7] is developed [7]. OLIA introduces 4 new relevant variables, alongside the already known variables from LIA [7]:

- l_r: $\max\{l_{1r}, l_{2r}\}$, where l_{1r} denotes the amount of acknowledged bytes between the last two packet loss events, and l_{2r} denotes the amount of acknowledged bytes since the last loss event.
- best_paths: Set of paths that maximize the ratio $\frac{l_r^2}{RTT_r}$.
- max_cwnd_paths: Subset of paths with the biggest $cwnd_i$ out of best_paths.
- collected_paths: All paths out of best paths which are not in max_cwnd_paths.
- all_paths: Set of all available paths.

Using this variables OLIA calculates the congestion window per flow during the congestion avoidance phase. Similar to LIA, the other TCP congestion algorithms remain untouched and α_r defines the aggressiveness in the $cwnd_i$ increases, however OLIA uses different α_r formulas depending on whether the path is in best_path, max_cwnd_paths or collected_paths as can be seen in Formula 4 [7]. The Formula used to calculate the increase of $cwnd_i$ can be seen in Formula 3 [7] where b is the number of acknowledged bytes. [7]

$$\Delta cwnd_i = \left(\frac{cwnd_i}{rtt_i^2} \Bigg/ \left(\sum_{p \in \text{all\_paths}} \frac{cwnd_i}{rtt_i}\right)^2 + \frac{\alpha_r}{w_r}\right) \cdot MSS_i \cdot \text{b} \tag{3}$$

$$\alpha_r = \begin{cases} \frac{1}{|\text{collected\_paths}|} & \text{if } r \in \text{collected\_paths}, \\ -\frac{1}{|\text{max\_w\_paths}|} & \text{if } r \in \text{max\_w\_paths} \\ & \wedge\ |\text{collected\_paths}| > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Using this methodology, OLIA can achieve the two goals already achieved by LIA, as well as goals 3, making this algorithm superior to LIA [7]. Furthermore, it fixes the flappiness of LIA [7]. However, as Walid et al. [9] find, OLIA sometimes does not react adequately when it comes to condition changes along the network paths under certain conditions.

2.2. Multipath Extension for QUIC

The Multipath Extension for QUIC [1] defines how the QUIC transport protocol can use multiple paths similar to MPTCP [1].

2.2.1. Functionality of MPQUIC. The Multipath Extension for QUIC follows the same idea as TCP's Multipath Extension. It takes an existing transport layer protocol and extends it to use multiple network paths simultaneously by adapting and extending already existing QUIC functionalities. Although QUIC already supports path migration, it does not have the capability to use multiple paths simultaneously. MPQUIC extends this capability to support concurrent path usage. [1]

In MPQUIC, paths are identified by a path ID. Similar to MPTCP, MPQUIC uses a new transport parameter to negotiate the use of MPQUIC in the initial client-server handshake. This transport parameter is called initial_max_path_id, and its presence signals that an endpoint is capable of using MPQUIC. The value of the parameter defines the maximum number of paths. The initiator sends it in the initial client crypto frame and the server also appends it to its initial crypto frame if it supports it. If the multipath handshake is successful, the endpoints start using PATH_ACK frames instead of ACK frames. [1]

PATH_ACK frames are an extension of QUIC's standard ACK frames, which are used to acknowledge packets in the context of a given path using the path IDs. The creation of a new path works by sending a PATH_NEW_CONNECTION_ID frame to a connection peer. It must contain a new connection ID and a linked new path ID used to identify the new path. The connection peer responds with its own PATH_NEW_CONNECTION_ID frame containing the same path ID and its own connection ID. [1]

After the path initiation, a path is validated with a PATH_CHALLENGE and PATH_RESPONSE frame sent by both connection partners to make sure that the path can be used. During the following transmission of data packets, the connection IDs are used to identify the path. Reassembling based on the stream offsets remains functionally the same as in normal QUIC operation, regardless of the paths used, as one stream can take multiple paths simultaneously. [1]

3. Congestion Control for MPQUIC

This Chapter discusses the default CC algorithm used by MPQUIC and present two novel approaches to MPQUIC CC proposed by H. Wang et al. [4] and Deng et al. [10] named CC-OLIA and S2B2C that improve various aspects of the default CC algorithm.

3.1. Default Congestion Control Algorithm

The Multipath Extension for QUIC specification draft [1] suggests utilizing a per-path instance of the CC algorithm used by the QUIC Transport protocol.

3.1.1. Functionality. MPQUIC maintains a congestion window per path, which limits how much data can be in flight at any given time. The slow start threshold known from TCP is initially set to infinity [11]. This threshold defines up to which congestion window size the slow start phase is used. As the slow start threshold is set to infinity, QUIC initially operates in a slow start phase. During the slow start phase, the congestion window, which is recommended to be set to 10 times the maximum datagram size, is increased by the bytes acknowledged in the PATH_ACK frames. This makes the congestion window grow exponentially. This behavior equals the behavior of TCP's slow start phase [11]. Once a loss occurs, MPQUIC enters a recovery phase. A loss is detected either by a missing acknowledgment or when a probe timeout (PTO) occurs. The PTO is started when a segment is sent on a specific path and is calculated as defined in Formula 5, where $smoothed\_rtt$ is the estimated RTT on the network path as defined in the MPQUIC specification, $4 \cdot rttvar$ is the variation of $srtt$, $gran$ specifies the time granularity and $maxdel$ is the maximum delay that can occur on the receiver side before sending the acknowledgment. [1] [3] [12]

$$\text{PTO} = \text{srtt} + \max\left(4 \cdot \text{var}, gran\right) + \text{max\_del} \quad (5)$$

After the recovery phase is entered the slow start threshold is set to the congestion window divided by 2. The congestion window must then be abruptly or slowly reduced to the threshold during the recovery phase. If further losses occur during the recovery phase no further reduction of the congestion window is performed. Once an acknowledgment from a packet sent during the recovery phase is received, the recovery phase is left, and the congestion avoidance phase is entered. During the congestion avoidance phase, an Additive Increase Multiplicative Decrease is used to increase the congestion window similar to TCP's congestion avoidance phase [11]. Should a packet loss, as defined before, occur during the congestion avoidance phase, the algorithm again enters the recovery phase. If only packet loss occurs during an implementation-specific timeframe, persistent congestion is assumed, and the congestion window is reset. [1] [3] [12]

3.1.2. Discussion. Referring back to the goals of multipath transport protocols defined in RFC 6356 by Raiciu et al. [6] and discussed by us in Section 2.1.2, one finds that this algorithm does not achieve the "Do no harm" goal as it actively competes with other paths for overlapping resources [4]. Wang et al. [4] find that this can significantly harm single-path applications. According to RFC 6356 [6] the issue can be fixed by adapting the principles of LIA to MPQUIC, however no concrete implementation is specified.

3.2. Modified Congestion Control Algorithms

Due to the reasons mentioned in Section 3.1.2, there is a need to modify the default MPQUIC CC mechanisms such that they fulfill the desirable goals for multipath CC algorithms.

3.2.1. Modified Congestion Control Algorithms using OLIA. H. Wang et al. [4] present a modified algorithm that adapts the ideas of OLIA to MPQUIC in the context of mobile networks called CC-OLIA. CC-OLIA is quite similar to OLIA [7], but instead of only modifying the congestion avoidance phase, they also modify the slow start phase according to a coupled slow start algorithm initially proposed by Y. Wang et al. [13] for MPTCP.

This slow start algorithm introduces an aggressiveness factor similar to what LIA and OLIA use to limit the total

congestion window of an entire MPTCP connection for MPTCP. This makes sure that the slow start phase, with its initial exponential growth across the paths, does not instantly congest shared parts of the network resulting in faster communication speeds for MPTCP. [13]

H. Wang et al. [4] adapt this slow start algorithm to work with the similar slow start mechanisms of each path of MPQUIC and use it as long as no packet loss occurs during the slow start phase. If packet loss occurs during the slow start phase, a multiplicative decrease mechanism to reduce the congestion window. This principle is also used by TCP in its slow start phase [11]. Afterward, the algorithm switches to the congestion avoidance phase where the congestion window of each path i is incremented according to $\Delta cwnd_i$ of OLIA as described in Section 2.1.2. Should a packet loss occur during the congestion avoidance phase CC-OLIA switches into the *Packet Loss Classification* algorithm. [4]

This algorithm tries to determine if the packet loss occurred randomly or was part of a congestion by checking how many events occurred in a defined interval. The $last\_cutback$ variable defines this interval. It stores the latest packet number used when the last decrease of the congestion window occurred. Should packet loss occur, the algorithm checks if the packet number of the lost packet is greater than $last\_cutback$, indicating whether the loss occurred randomly or was the start of a larger network degradation. If the former applies, the algorithm classifies the loss as Random Packet Loss (RPL) and leaves the congestion window unchanged but updates $last\_cutback$ to the current packet number used for new packets. [4]

Furthermore, the slowstart threshold is updated to the current congestion window. Should the packet number be smaller or equal to $last\_cutback$, the algorithm assumes continuous congestion, reduces the congestion window across all paths, and classifies the loss as Congestion Packet Loss (CPL). This behavior can also be seen in Formula 10 [4] where n denotes the amount of paths currently used and d represents the multiplicative decrease factor. All other variables are defined in Section 2.1.2. [4]

$$cwnd_i = \begin{cases} cwnd_i, & \text{if RPL,} \\ cwnd_i \left(\frac{n-1+d}{n}\right), & \text{if CPL.} \end{cases} \tag{10}$$

When evaluating CC-OLIA H. Wang et al. [4] find that they can reduce the transfer times of a 5 MB, 25 MB, and 35 MB file by 6.3%, 14.9%, and 16.2% respectively compared to an OLIA MPTCP implementation in the case of a shared bottleneck scenario. In the case of a non-shared bottleneck scenario they could reduce the transfer times for the same files by 18.7%, 29.8%, and 36.3%.

3.2.2. BBR based congestion control for MPQUIC. The Bottleneck Bandwidth and Round-trip propagation time algorithm (BBR) is a CC algorithm implemented for TCP and QUIC, which relies on measuring connection speed, round trip time, and packet loss to avoid congestions. It works by estimating a bandwidth-delay product (BDP) through probing to estimate the optimal throughput along the used network path, which does not overwhelm the bottleneck of that given path. [14]

BBR has four algorithmic phases: Startup, Drain, ProbeBW, and ProbeRTT. During the first phase, the Startup phase, BBR quickly increases the used network bandwidth until the bottleneck buffer on the network path fills up and packet loss or the delivery rate plateaus. This is used to find the maximum bandwidth bw and the minimum RTT RTT_{min}. The BDP is estimated by multiplying RTT_{min} with bw. The algorithm then moves on to the drain phase, where it quickly drains the intermediate buffers by ramping down the throughput until the data in flight equals the estimated BDP. BBR then periodically probes for more bandwidth and minimum RTT in order to adapt the BDP and the data in flight. [14]

While we could theoretically apply the principles of BBR to each individual path in MPQUIC, we would face the same problems as one would when using the default CC algorithm as defined in the MPQUIC draft [10]. Therefore, Deng et al. [10] developed a novel BBR-based CC algorithm called S2B2C, which utilizes the principles of BBR but ensures fairness. They do so by identifying MPQUIC paths that share a bottleneck and fairly adapt their pacing gain G during the ProbeBW phase. The pacing gain alongside other variables influences the rate at which data is sent and, therefore, the amount of data in flight [15]. Every time BBR tries to estimate the available bandwidth, it sets the pacing gain $\mathcal{G}$ to $\mathcal{G} \in [1.25, 0.75, 1, 1, 1, 1, 1, 1]$ in an 8-step process during the ProbeBW phase [10].

During the first step, the sending rate is set to 1.25 multiplied by bw. As this is done per path, Deng et al. assume that all paths S_1 where the RTT increases after this first step must share at least one bottleneck. The same goes for all paths S_2 during the second step of ProbeBW, where the pacing gain is set to 0.75. Therefore, $S_1 \cap S_2$ denotes all paths that share a bottleneck. Deng et al. then repeat this procedure for three entire 8-step runs to ensure that any other network noise does not influence the groupings. [10]

Afterwards, they do the same again for ProbeRTT as during ProbeRTT, the maximum amount of in-flight data is limited to 4 maximum segment sizes resulting in a significant increase in round trip times in other paths that share the same bottleneck. The set of these paths is called S_3. Calculating $S_1 \cap S_2 \cap S_3$ per path now yields a very probable set of paths that share the same bottleneck. For all paths $r \in \xi$ that share the same bottleneck G_r is set to

$$\mathcal{G}_r \in [1.25, 0.75, \alpha_r, \alpha_r, \alpha_r, \alpha_r, \alpha_r, \alpha_r]$$

where α_r is calculated according to Formula 6 [10] where S denotes the set of all available paths. During performance testing, Deng et al. find that this strategy ensures bottleneck fairness and balanced congestion. [10]

$$\alpha_r = 4 \left(\frac{bw_r \times \max_{r \in S}\{bw_r\}}{\sum_{r \in \xi} bw_r} - 1 \right) / 3 \tag{6}$$

4. Conclusion and Future Work

Over the course of this paper, we first looked into how MPTCP works and analyzed a non-exhaustive list of common CC algorithms used alongside MPTCP. After that, we dived into the core functionality of MPQUIC

and explained how its default CC mechanism works. We then discussed its shortcomings and introduced the novel adaptation CC-OLIA, which is based on MPTCP's OLIA CC algorithm, but has been adapted to work with MPQUIC. We also introduced S2B2C, a CC algorithm based on BBR, which uses an entirely different method to handle congestion compared to OLIA's more traditional approach.

Both algorithms show promising initial results and can overcome the shortcomings of MPQUIC's default CC algorithm. While this is a non-exhaustive list of CC algorithms for MPQUIC, we find that it represents the two most prominent groups of either BDP or LIA-based CC algorithms. An example of another implementation of a BDP-based MPQUIC CC algorithm is MACO [16], which was developed to provide congestion control for fast-moving MPQUIC-based satellite networks [16]. Regarding future work, S2B2C needs to be updated to support BBRv2 [17]. Furthermore, we find that there is a research gap when it comes to a CC algorithm for MPQUIC based on the Balanced Linked Adaptation CC Algorithm for MPTCP (BALIA) [9], a further development of OLIA, which overcomes its mentioned weaknesses [9].

References

[1] Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, "Multipath Extension for QUIC," Internet Engineering Task Force, Internet Draft draft-ietf-quic-multipath-13, Mar. 2025, num Pages: 40. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/13

[2] "IETF," Mar. 2025. [Online]. Available: https://www.ietf.org/

[3] J. Iyengar and I. Swett, "QUIC Loss Detection and Congestion Control," RFC 9002, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9002

[4] H. Wang, Y. Liu, Z. Li, Y. Zhang, W. Gong, T. Jiang, T. Bi, and J. Zhou, "Cc-olia: A dynamic congestion control algorithm for multipath quic in mobile networks," *Digital Communications and Networks*, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864824001640

[5] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses," Internet Engineering Task Force, Request for Comments RFC 8684, Mar. 2020, num Pages: 68. [Online]. Available: https://datatracker.ietf.org/doc/rfc8684

[6] C. Raiciu, M. J. Handley, and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols," Internet Engineering Task Force, Request for Comments RFC 6356, Oct. 2011, num Pages: 12. [Online]. Available: https://datatracker.ietf.org/doc/rfc6356

[7] R. Khalili, N. Gast, M. Popovic, and J.-Y. L. Boudec, "Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP," Internet Engineering Task Force, Internet Draft draft-khalili-mptcp-congestion-control-05, Jul. 2014, num Pages: 11. [Online]. Available: https://datatracker.ietf.org/doc/draft-khalili-mptcp-congestion-control-05

[8] E. Blanton, V. Paxson, and M. Allman, "TCP Congestion Control," Internet Engineering Task Force, Request for Comments RFC 5681, Sep. 2009, num Pages: 18. [Online]. Available: https://datatracker.ietf.org/doc/rfc5681

[9] A. Walid, Q. Peng, J. Hwang, and S. H. Low, "Balanced Linked Adaptation Congestion Control Algorithm for MPTCP," Internet Engineering Task Force, Internet Draft draft-walid-mptcp-congestion-control-04, Jan. 2016, num Pages: 11. [Online]. Available: https://datatracker.ietf.org/doc/draft-walid-mptcp-congestion-control-04

[10] Z. Deng, Y. Liu, J. Liu, A. Argyriou, and D. Liu, "Bbr-based and fairness-guaranteed congestion control and packet scheduling for mpquic over heterogeneous networks," *Computer Communications*, vol. 224, pp. 213–224, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366424002160

[11] E. Blanton, D. V. Paxson, and M. Allman, "TCP Congestion Control," RFC 5681, Sep. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5681

[12] 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

[13] Y. Wang, K. Xue, H. Yue, J. Han, Q. Xu, and P. Hong, "Coupled slow-start: Improving the efficiency and friendliness of mptcp's slow-start," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.

[14] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, p. 58–66, Jan. 2017. [Online]. Available: https://doi.org/10.1145/3009824

[15] N. Cardwell, Y. Cheng, S. H. Yeganeh, and V. Jacobson, "BBR Congestion Control," Internet Engineering Task Force, Internet-Draft draft-cardwell-iccrg-bbr-congestion-control-00, Jul. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-cardwell-iccrg-bbr-congestion-control/00/

[16] W. Yang, L. Cai, S. Shu, and J. Pan, "Mobility-aware congestion control for multipath quic in integrated terrestrial satellite networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11 620–11 634, 2024.

[17] N. Cardwell, I. Swett, and J. Beshay, "BBR Congestion Control," Internet Engineering Task Force, Internet-Draft draft-ietf-ccwg-bbr-02, Feb. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-ccwg-bbr/02/

Credit-Based Shaping As Defense Against DoS Attacks

Leonard Nolting, Florian Wiedner*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: leonard.nolting@tum.de, wiedner@net.in.tum.de

***Abstract*—Time-Sensitive Networking (TSN) is a frequently researched alternative for real-time Ethernet networks using time-synchronization, shaping, scheduling and more techniques to accommodate streams with different latency and bandwidth requirements on one network. The Credit-Based Shaper (CBS) awards credits to queues at a linear rate, limiting their bandwidth and controlling the burstiness of traffic.**

As Denial-of-Service (DoS) attacks remain relevant and TSN networks often are used in critical cyber-physical systems, the defense capabilities of TSN networks against DoS attacks need to be evaluated.

We explore how well CBS can protect TSN networks from DoS attacks by assessing the possible scenarios and categorizing attacks with a DoS taxonomy. We will find that CBS can by no means replace proper security mechanisms, but in certain scenarios can protect large parts of a TSN network from an attack, especially when the attacker can only send in the best-effort traffic class.

***Index Terms*—denial-of-service, time-sensitive networking, credit-based shaper, security**

1. Introduction

Best-effort traffic delivers "most packets, most of the time, mostly in order" [1]. It lacks determinism. Applications that require tighter guarantees by their network, such as industrial control, electrical grids or in-vehicle networks, traditionally created domain-specific solutions, for example EtherCAT, PROFINET or CAN [1]–[4].

Time-Sensitive Networking (TSN) is a set of standards that enable such guarantees for standard Ethernet by providing upper bounds on latency and decreasing packet loss. They are maintained by the IEEE 802.1 working group. Among others, TSN is "promising to replace existing protocols in mission-critical domains" [5], where correct timing not only affects performance but also safety and security.

The growing adoption of TSN, combined with its extreme susceptibility to traffic disruptions and the vulnerability of the domains it is employed in, raises questions about the protocol's security. The European Union Agency for Cybersecurity finds Denial-of-Service (DoS) attacks "ranked at the top during the reporting period for another year" [6], especially "on the critical infrastructures of countries" [7].

As TSN is still young and in the standardization phase, its security aspect has been researched significantly less than more established systems like Ethernet, IP or alternative real-time protocols. Several papers look specifically at IEEE 802.1Qci "Per-Stream Filtering and Policing" [8], [9]. The security of TSN in general is considered by Ergenç et al. [5], others focus on specific domains [2], [10]–[12]. Furthermore, building upon the Credit-Based Shaper (CBS), Meyer et al. propose Credit-Based Metering [13].

IEEE 802.1Q describes the Credit-Based Shaper (CBS), which limits the bandwidth of a traffic class by only awarding it a certain amount of credit over time. Using CBS shaping can potentially mitigate DoS attacks on TSN networks, but this has not yet been evaluated.

In this paper, we explore the strengths and limitations of using CBS without modifications to defend against varying types of DoS attacks.

We will first cover the underlying technological aspects, as well as DoS attack vectors on TSN networks. This will be followed by an evaluation and a clear compilation of the results, including suggestions for follow-up research.

2. Background

The paper combines several topics, which will be briefly summarized in the following subsections.

Time-Sensitive Networking

TSN is a feature offered by a network that simultaneously hosts regular best-effort traffic.

All TSN nodes synchronize their clocks on the network. TSN flows represent a contract between the network and the end hosts about bandwidth, latency, jitter and packet loss, and can be created and ended flexibly. It aims to eliminate congestion loss completely by controlling the traffic shape and schedule. Shaping limits bandwidth and smooths out traffic, whereas scheduling determines when packets are sent from a queue. TSN provides several algorithms for that. See [1] for a good in-depth introduction. In this paper, we will inspect the CBS.

Credit-Based Shaper

Traffic shaping creates gaps between packets [3]. This may seem counterintuitive to latency goals, but it gives other flows a chance to find a gap in a burst of packets, essentially skipping the queue which is full from the burst. For a minimal illustrative example, see [3].

One switch can have multiple CBS shaped queues per egress port. For each, a credit value is stored, starting at

 doi: 10.2313/NET-2025-11-3_06

zero. When a packet from that queue is sent, the credit value is reduced by the length of the packet. Now, credit replenishes at a set rate called `idleSlope` until it reaches zero. Only once it is back to zero, the queue may be eligible again, meaning it is ready to send.

If the queue is eligible but cannot send because the transmitter is currently occupied by another queue, this is the first time credits accumulate over zero. The queue can now send packets as long as its credit stays greater than or equal to zero, and it is nonempty. Once it empties but still has credit left, it is reset to zero.

Helpful visualizations and more detailed explanations can be found in [3].

Types of DoS attacks

A DoS attack is "an attempt to make a computer resource unavailable to its intended users." [14]. This general goal can be achieved in many ways, e.g. a physical attack on a facility. In order to limit the scope of this paper, we will only look at attacks performed through means of a network connection.

Over time, many attack and prevention mechanism categorizations for such DoS attacks have been published, such as by Karig and Lee [15], Fadlallah and Serhrouchni [16], Specht and Lee [17], Douligeris and Mitrokotsa [18] and Mirkovic and Reiher [19], some of which are more detailed than others. This paper is based on a later taxonomy by Ramanauskaite and Cenys [14] that reviews and combines the previous mentions into one.

DoS attacks vary significantly in nature and can be classified in multiple dimensions. Understanding this taxonomy is important for analyzing the effectiveness of CBS as a defense mechanism in TSN, which will be done in Section 4.

One main classification is based on the number of sources involved in the attack:

- **Single Source Attack**: DoS attack launched from a single machine
- **Distributed Denial-of-Service (DDoS) Attack**: coordinated DoS attack launched from multiple systems

Another dimension considers the vulnerability exploited:

- **Bug Exploitation Attack**: exploits software or hardware vulnerabilities in the victim's system to cause a denial-of-service
- **Resource Depletion Attack**: consumes a system's resources, making them unavailable for legitimate requests:
  - **Memory Depletion Attack**: fills up the system's memory
  - **CPU Work Depletion Attack**: overloads the system's CPU by requiring it to perform excessive processing
  - **Semantic Resource Depletion Attack**: exploits modified incoming packets to consume more resources
- **Bandwidth Exhaustion Attack**: floods the target with a large amount of data, consuming all available network bandwidth and preventing legitimate traffic from reaching the victim

These dimensions, along with a dimension distinguishing single nodes or the network being affected, are visualized in Figure 1.

In the context of Time-Sensitive Networks, the most relevant types of DoS attacks are resource depletion attacks and bandwidth exhaustion attacks. These attacks can directly impact the network's ability to deliver time-sensitive data, which can cause missed deadlines in real-time applications.

In the following sections of this paper, we will explore whether Credit-Based Shaping can be employed to counteract these attacks in Time-Sensitive Networks. First, we will look at the methodology used to analyze this topic.

3. Context

The purpose of this paper is to find out when CBS can be used to mitigate DoS attacks.

It is critical to understand why and how DoS attacks affect TSN networks, first. For that, we will now evaluate the relevance of DoS attacks for TSN networks and which scenarios that would affect. After that, we will cover existing security mechanisms in TSN. Finally, the evaluation and conclusion will follow in Sections 4 and 5.

Understanding DoS Applicability in TSN networks

TSN often operates in isolated networks and security benefits from that, as attackers need to gain access to the network first before they can start a DoS attack [1], [20]. One might wonder how DoS attacks are relevant to the typical network that employs TSN at all.

First, a device within the network could be infected over other means than a network connection, for example through a bad update. The infected node could then cause a DoS attack. This shows that isolation cannot be a full security guarantee and does not serve as reliable protection. Isolated networks therefore are not immune to DoS attacks and estimating their potential impact and countermeasures is still relevant.

Additionally, with the shift of TSN towards being used for routed networks, such as with DetNet, TSN networks are increasingly losing their isolation property as they are being connected to wide area networks. This increases the risk of attacks and lowers the barrier for potential attackers, especially from remote locations [1], [21].

Furthermore, the use-cases for Operational Technology (OT) networks and TSN significantly overlap [1], [4], [10], [22], and with the convergence of OT and IT networks come "cyber security challenges that are typically associated with only with IT infrastructures" [20], [21]

Due to the time-sensitive and cyber-physical nature of TSN networks, they "present potentially attractive targets for cyber attackers" [21].

An additional consideration is that real-time systems like TSN will often employ embedded microcontrollers with little resources, making them sensitive to even small attacks.

This provides context about why DoS attacks are relevant to TSN networks and under which circumstances they occur.

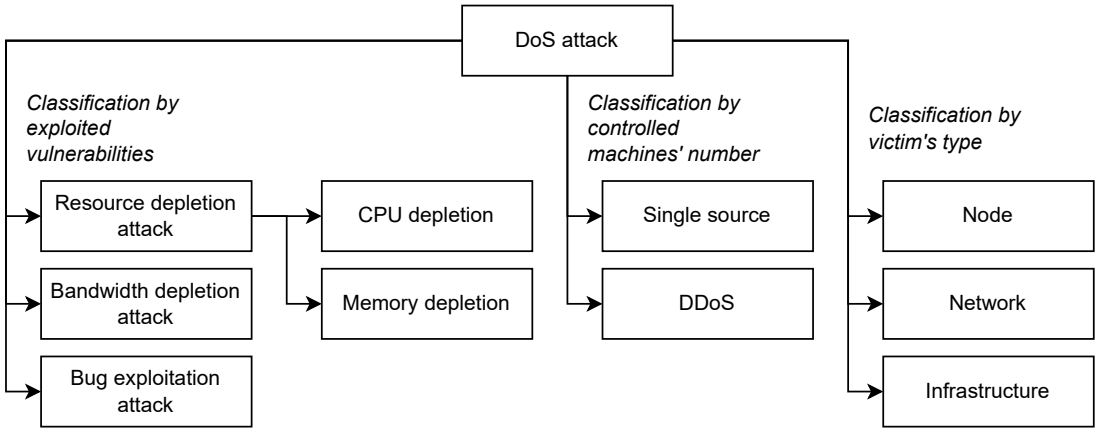


Figure 1: An adjusted version of the taxonomy suggested in [14].

Existing security mechanisms in TSN

TSN standards primarily focus on the key characteristics (performance, determinism) and practicality/ease-of-use. Security is not a core consideration and has to be actively added [5].

Existing network security paradigms, such as firewalls, traffic anomaly detection or authentication can be used with TSN, but may have to be adapted to the timing requirements by TSN [1], [5].

Additionally, the IEEE provides several proposed or standardized security mechanisms such as “Per-stream filtering and policing”, “Frame Replication and Elimination for Reliability” or “MACsec” [5], [23].

However, to isolate the effects that CBS has on security, we will consider a network implementation with CBS alone and no optional standards.

4. Evaluation

We will now assess the effect of CBS on security in TSN networks against DoS attacks based on the structure given by the DoS taxonomy discussed in Section 2.

In general, TSN and CBS greatly increase the complexity of a network, increasing the attack surface for semantic DoS attacks [19] / bug exploitation attacks [14]. Since CBS does not examine packet contents, it has no impact on the defense against this type of attack.

Attacker Inside the Network

As stated in Section 3, a DoS attack can occur in an isolated network from a node inside the network itself, if it has previously been infected.

This can lead to a special case, where if the infected node itself represents a service, its denial can be caused by it not sending any packets. In the DoS taxonomy, this is represented as “Denial of Node”. As CBS never goes into effect here for the lack of packets, it cannot stop this attack.

Otherwise, a DoS can only be achieved through sending many or malformed packets.

When a node sends too many packets through an existing flow, it will fill the queue of the switch it is connected to (for end devices the ingress switch). Since CBS does not allocate an individual queue to each flow, other flows sharing the same queue will be starved of buffer space. That is a Memory Depletion Attack, which can lead to significant congestion loss through buffer overflows. As TSN under normal operation eliminates congestion loss completely, the starved nodes might not know how to react to that, causing unpredictable errors [1]. For example, in combination with TSN Frame Replication and Elimination, a network can assume zero packet loss and might consider a node as faulty when its packets are not received. This leads to the perceived failure of entire nodes, virtually taking down entire machines.

Additionally, credit is only allocated per CBS-queue, thus other flows will be starved of credit as well, resulting in a perceived Bandwidth Depletion Attack [3].

CBS can only limit the starvation to the traffic class of the attacking flow and any lower priority classes, as that is the granularity of its queues and credits. Hence, higher priority traffic is protected from this type of attack, while lower priority traffic, such as best-effort, is not specially protected.

A sender might also send an amount of packets which overloads the classification algorithm of the receiving switch, causing a CPU work depletion attack, or more generally, if offloaded, a processing work depletion attack. This can cause the failure of the entire switch. Since shaping is performed after classification, the CBS shaper cannot stop that type of attack.

On the physical level, since Ethernet is a shared medium, an attacking node can deplete nodes also connected to its outgoing ports by sending excessive amounts of packets, provoking collisions. This bandwidth depletion attack also happens in front of shapers, hence CBS cannot prevent this.

So far, we assumed the attacking node only uses existing flows and their traffic classes. If it can create flows of arbitrary classes, with enough bandwidth, it can overload all queues of a port at the same time, which starves all other flows routed on that port of bandwidth and causes unexpected congestion loss. In this case, CBS cannot mitigate the DoS attack and the entire port fails.

If the attacker also sends to varying destination addresses which are routed on different egress ports of the

 doi: 10.2313/NET-2025-11-3_06

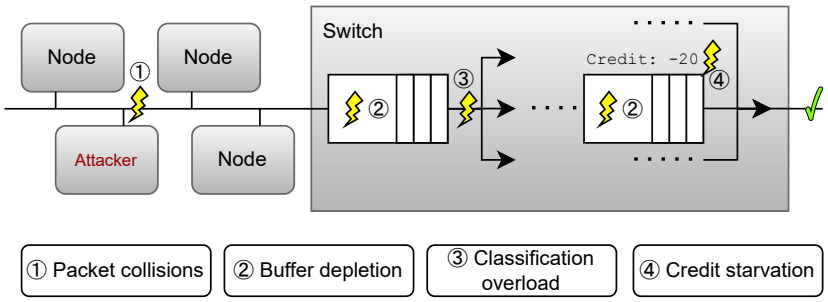


Figure 2: A simplified illustration of possible brute-force vulnerabilities with one attacking node inside the network.

same or different switches, it can multiply its effect across all egress port of all switches it is connected to.

As multiple nodes get involved, forming a DDoS attack, each node can attack in above fashion, denying the service of all switches that each attacker is connected to in the worst case.

Attacker Outside the Network

If the attacker is outside the TSN domain, it is most likely switched as best-effort IP traffic. Reference [1] states as an essential feature of TSN that the best-effort traffic class can employ all the usual tools for IP traffic, so the mitigation of DoS attacks can be handled by these. However, since credits are only given to CBS-queues and CBS traffic has higher priority than best-effort traffic, it is protected from bandwidth starvation even in the event of a DoS attack from best-effort traffic. If the outside machine is equipped with the necessary hardware and no security mechanisms are in place, it can also create CBS flows, essentially making it a part of the TSN domain. The above evaluation for an inside attacker applies.

After the Ingress Bridge

The CBS shaper forwards packets at a "rate such that, over a relatively short term, is equal to the total bandwidth allocated to the TSN flows using that queue" [1]. Bandwidth and burstiness are predictable and controlled, even in the event of a DoS attack. This leads to any DoS attack being contained to the connected ingress switches and all their nodes. It will not propagate further into the network, defending it against the attack.

Bandwidth and burstiness depend on the single parameter `idleSlope` that is passed to CBS-queues. As the network accounts for the queue behavior derived from the parameter, changes to it have no effect on the outcome [24].

Less resource-intensive pulsing attacks, with the goal of transmitting packets exactly when credits replenish, do not work, since CBS-queues are first-come, first-serve (FIFO) [3], [24].

A low and slow attack of creating many flows cannot specifically target the CBS, since it does not keep state or allocate computing resources for each flow.

As by design of CBS, it is also not possible to provoke one node to accumulate a very high credit number through blocking a port and subsequently a queue for a long time. CBS defines a `hiCredit` value, limiting the maximum amount of credits a queue can reach, which is taken into consideration for bandwidth and burstiness calculations [24].

5. Conclusion and Future Work

In this paper, we analyzed to which extent and with which constraints CBS can defend against DoS attacks. The core finding is that behind CBS shaped queues, the network is safe from brute-force attacks. Furthermore, CBS as a security mechanism works better the more restricted the access is that each node has to other ports and queues. However, the evaluation also clearly shows that CBS is by no means a complete tool against DoS attacks. Its most obvious shortcoming are semantic attacks, which it cannot detect and protect of and where it consequently cannot replace additional security tools, such as Intrusion Detection Systems. It is important to be aware it was not designed for security purposes and should not be advertised as a solution.

For more certainty about the positive results in this paper, future work needs to verify the results in both network simulators and real networks. In order to reduce the impact of DoS attacks, future work could explore introducing traffic class checks and destination checks at switches. Switches will check if the sender of an incoming packet may use that priority and send to the denoted node based on a network-wide policy. If the sender does not have that permission, it is considered infected and its traffic completely ignored (de-facto unplugged). If the ignoring mechanism is implemented properly, it can also protect against resource depletion of the classification algorithm. These checks would be especially effective against attacks from low-priority, less central nodes in a dense network graph.

It might also be interesting to investigate the ring and other network topologies and the combination and interaction of CBS with other mechanisms for TSN security, for example with Frame Replication and Elimination for path redundancy. If each node is connected to two switches, which both give access to the entire CBS-shaped network, single source attacks might have a reduced impact.

As it stands now, however, with TSN still being a newcomer to the industry and not enough data to draw sufficient conclusions about its security, future work should focus on dedicated security mechanisms to make reliable and universal guarantees.

 doi: 10.2313/NET-2025-11-3_06

References

[1] N. Finn, "Introduction to time-sensitive networking," *IEEE Communications Standards Magazine*, pp. 22–28, jun 2018.

[2] K. Zanbouri, M. Noor-A-Rahim, J. John, C. J. Sreenan, H. V. Poor, and D. Pesch, "A comprehensive survey of wireless time-sensitive networking (tsn): Architecture, technologies, applications, and open issues," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2024.

[3] J. Walrand, "A concise tutorial on traffic shaping and scheduling in time-sensitive networks," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1942–1953, 2023.

[4] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely survey of time-sensitive networking: Past and future directions," *IEEE Access*, vol. 9, pp. 142 506–142 527, 2021.

[5] D. Ergenç, C. Brülhart, J. Neumann, L. Krüger, and M. Fischer, "On the security of ieee 802.1 time-sensitive networking," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6.

[6] ENISA (European Union Agency for Cybersecurity), "Threat landscape." [Online]. Available: https://www.enisa.europa.eu/topics/cyber-threats/threat-landscape

[7] and European Union Agency for Cybersecurity, I. Lella, M. Theocharidou, E. Magonara, A. Malatras, R. Svetozarov Naydenov, C. Ciobanu, and G. Chatzichristos, *ENISA Threat Landscape 2024 – July 2023 To June 2024*, I. Lella, M. Theocharidou, E. Magonara, A. Malatras, R. Svetozarov Naydenov, C. Ciobanu, and G. Chatzichristos, Eds., 2024.

[8] A. Mahamid, "Time sensitive networking - 802.1qci," in *Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM), Winter Semester 2020/2021*, ser. Network Architectures and Services (NET), G. Carle, S. Günther, and B. Jaeger, Eds., vol. NET-2021-05-1. Munich, Germany: Chair of Network Architectures and Services, Department of Computer Science, Technical University of Munich, May 2021, pp. 9–12.

[9] R. Barton, M. Seewald, and J. Henry, "Management of ieee 802.1qci security policies for time sensitive networks (tsn)," Technical Disclosure Commons, October 2018. [Online]. Available: https://www.tdcommons.org/dpubs_series/1541

[10] F. Fischer and D. Merli, "Security considerations for ieee 802.1 time-sensitive networking in converged industrial networks," in *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, 2022, pp. 1–7.

[11] R. Sethi, A. Kadam, K. Prabhu, and N. Kota, "Security considerations to enable time-sensitive networking over 5g," *IEEE Open Journal of Vehicular Technology*, vol. 3, pp. 399–407, 2022.

[12] T. Häckel, P. Meyer, F. Korf, and T. C. Schmidt, "Secure time-sensitive software-defined networking in vehicles," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 35–51, 2023.

[13] P. Meyer, T. Häckel, F. Korf, and T. Schmidt, "Dos protection through credit based metering – simulation based evaluation for time-sensitive networking in cars," 08 2019.

[14] S. Ramanauskaite and A. Cenys, "Taxonomy of dos attacks and their countermeasures," *Central European Journal of Computer Science*, vol. 1, no. 3, pp. 355–366, 2011.

[15] D. Karig and R. Lee, "Remote denial of service attacks and countermeasures," Princeton University Department of Electrical Engineering, Tech. Rep. CEL2001-002, 2001.

[16] A. Fadlallah and A. Serhrouchni, "Denial of service attack and schemes analysis and taxonomy," in *IEEE SETIT 2005, International Conference on Sciences of Electronic, Technology of Information and Telecommunications*, 2005, 27–31 Mar. 2005, Tunisia.

[17] M. S. Specht and R. Lee, "Distributed denial of service: Taxonomies of attacks, tools, and countermeasures," in *17th International Conference on Parallel and Distributed Computing Systems*, 2004, pp. 543–550.

[18] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: Classification and state-of-the-art," *COMPUT NETW*, vol. 44, pp. 643–666, 2004.

[19] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, pp. 39–53, 2004.

[20] M. N. J. Glenn Murray and C. Valli, "The convergence of it and ot in critical infrastructure," in *The Proceedings of 15th Australian Information Security Management Conference*, C. Valli, Ed. Edith Cowan University, dec 2017, pp. 149–155.

[21] E. Grossman, T. Mizrahi, and A. J. Hacker, "Deterministic Networking (DetNet) Security Considerations," RFC 9055, Jun. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9055

[22] K. Stouffer, K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, S. Lightman, A. Hahn, S. Saravia, A. Sherule *et al.*, *Guide To Operational Technology (OT) Security*. US Department of Commerce, National Institute of Standards and Technology, 2023.

[23] F. Rezabek, M. Bosk, L. Seidlitz, J. Ott, and G. Carle, "Context matters: Lessons learned from emulated and simulated tsn environments," in *2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events, PerCom Workshops 2024*, ser. 2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events, PerCom Workshops 2024. Institute of Electrical and Electronics Engineers Inc., 2024, pp. 499–504, publisher Copyright: © 2024 IEEE.; 2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events, PerCom Workshops 2024 ; Conference date: 11-03-2024 Through 15-03-2024.

[24] *802.1Q-2022 - IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks*, IEEE Std. 802.1Q, Dec 2022. [Online]. Available: https://ieeexplore.ieee.org/servlet/opac?punumber=10004496

Governance of a Distributed Autonomous Organization

Keihan Pakseresht, Holger Kinkelin*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: keihan.pakseresht@tum.de , kinkelin@net.in.tum.de

***Abstract*—This paper explores and analyzes the governance of a Decentralized Autonomous Organization (DAO) in the context of managing a distributed research testbed. Since the absence of a central authority gives the members complete control over the network, governance plays a significant role in the decision-making processes in any DAO. Using a decentralized payment system as a case study, this paper analyzes applicable governance models and identifies key challenges in a DAO. By examining existing governance mechanisms and their suitability for our scenario, the paper aims to outline a strong governance methodology that fulfills the requirements of the DAO and facilitate its objectives.**

***Index Terms*—DAO, Governance**

1. Introduction

Blockchain technology has gained significant popularity in recent years, revolutionizing industries by enabling secure and transparent transactions between parties. Its decentralized nature eliminates the need for intermediaries, which is the initial idea for *Decentralized Autonomous Organizations (DAOs)*. DAOs have arisen as a promising governance model, using blockchain to enable collaborative decision-making without relying on classic hierarchical structures. They give their community complete control over their rules and operations.

Defining governance for each DAO is crucial, as it depends on the specific goals and objectives of the organization. Generally, governance refers to the rules and decision-making processes that manage a DAO. These decisions can vary from simple ones, such as adding or removing members, to more complex ones, such as making major changes to the organization or its structure.

Understanding how decisions are made within a DAO and how its token economy is managed is essential for long-term sustainability.

1.1. Goal

This paper aims to investigate governance in DAOs using a decentralized research testbed as a scenario. We will analyze the most common governance models and processes that are primarily used in DAOs and evaluate their applicability to our research testbed scenario. By the end of this paper, the reader should have a better understanding of governance in DAOs and be able to choose and design the most suitable solution for each specific organization.

1.2. Outline

The paper is structured as follows:

- Section 2 includes a brief overview of the DAO testbed scenario and relevant concepts and terms in DAOs.
- Section 3 analyses the governance models and process, discussing their advantages and disadvantages in the context of the research testbed.
- Section 4 discusses the suggested methodology for our use case.
- Section 5 concludes the paper with key takeaways and an overview of future work and studies.

By following this structure, the paper systematically builds an understanding of governance, ensuring practical applicability to the research testbed scenario.

2. Background and Terminology

To understand how decentralized autonomous organizations work and what governance challenges lie underneath, we need to define some terms and concepts used in the paper. It is worth mentioning that because the governance is the main focus of the paper, the terms are simplified. Further details can be found in [1] and [2].

2.1. DAOs and Relevant Concepts

Decentralized Autonomous Organization(DAO) is a transparent organization based on Blockchain technology. The organization is managed by the members, and depending on the rules defined in the *Smart contract*s, the members can vote on the decisions of the organization.

Smart contracts are self-executing contracts with the terms of the agreement between parties being directly written into lines of code. They play a crucial role in DAOs since there is no central authority to manage the organization, and the terms of a smart contract can be very different; one of the most commonly used methodologies are token-based contracts.

Tokens are digital assets used to represent ownership of a particular asset. They can be traded with other parties and can be exchanged for other assets. It is important to note that tokens are not always the same as cryptocurrencies. Depending on the use case, tokens can be used to represent any type of asset, such as stocks, paintings or even real estate.

With the power of smart contracts and tokens, *Decentralized Application*(dAPP) can provide a platform for

 doi: 10.2313/NET-2025-11-3_07

members to interact with the DAO. dApps are applications that run on decentralized networks. They follow the rules defined in the smart contracts and enable members to interact with the DAO, normally via predefined tokens.

In the managing DAOs, it is important to differentiate between *on-chain* and *off-chain* governance. On-chain governance refers to the governance processes that are proposed directly on the blockchain, and after voting, the decision is automatically executed on the blockchain. Off-chain governance, on the other hand, refers to the governance processes that are proposed outside the blockchain, for example, in forums or social media. After voting, the decision is executed manually by the members of the organization.

2.2. Relevant Concepts in Testbed DAO

The suggested testbed DAO in [3] is built on *Algorand* blockchain. Algorand is a blockchain platform that aims to provide a decentralized, secure, and scalable platform for developers to build applications. It uses a consensus mechanism called *Pure Proof of Stake*(PPOS), which allows for fast and secure transactions on the network [1]. This Blockchain technology offers the possibility to create and manage digital assets, called *Algorand Standard Assets*(ASA),which can represent any type of fungible (like cryptocurrencies) or non-fungible (like NFTs) assets. ASA can be traded efficiently and with low fees between members on the Algorand network [1]. The paper [3] suggests using ASA as a token to represent the resources in the testbed DAO.

2.3. Testbed DAO Scenario

The rapid advancement of scientific research has led to an increasing demand for computational resources. Many research institutions operate specialized testbeds equipped with specialized hardware, such as GPU clusters, 6G testing equipment, and other resources. However, researchers often require access to resources beyond their local infrastructure, necessitating a collaborative system for resource sharing across institutions.

The bachelor thesis "Trustless Resource Sharing between Scientific Testbeds" [3] proposes a DAO to manage sharing of computational resources. This system, built using Algorand's ASA token standard, enables compensation when one institution utilizes another's resources in exchange for tokens. The technical aspects of the DAO have been implemented, but the governance aspects remain unexplored.

Governance in the context of our scenario refers mainly to the decision-making processes for research site participation and management, including adding or removing sites and determining which decisions should be centralized or decentralized.

Since the testbed DAO is mainly based on collaboration between research sites, it is important to have a governance model that encourages participation and engagement from all members.

3. Analysis of DAO Governance

Selecting a suitable governance model for the DAO is essential. Different decisions need to be made by the DAO members, and these decisions can vary widely from minor to large. Although not all members have the same voting power, the governance model should still be fair and transparent to all members. At the same time, it should not allow any member to manipulate the voting process.

3.1. Challenges

Since every single decision and change needs to be made by members themselves, it is crucial to understand the challenges that might arise in a DAO.

Significant ownership concentration in DAOs can lead to governance challenges since only a few large token holders may have the power to influence decisions. This can lead to lower participation rates from smaller token holders, which can result in Growth challenges for the DAO in the long run [4].

Conflicts of interest between various stakeholders, particularly between large and small token holders, can also arise in DAOs. To address it an approach to governance can be the "one token, one vote" model. But this can lead to strategic voting and manipulation by large token holders since there is no guarantee that all the members will participate in the decision-making process.

Parties can buy votes and bribe other members to manipulate the DAOs in their favor.

Another challenge is the lack of anonymous voting; since all the votes are recorded on the blockchain(as a part of general rule in any blockchain system), this can lead to privacy concerns for the members of the DAO. Beyond the governance challenges.

DAOs also face technical challenges such as scalability, security, and code vulnerabilities, which are out of the scope of this paper.

3.2. Voting Process

Depending on the scale and type of the DAO, the voting process might differ from one DAO to another. In this section, we will discuss the general voting process in DAOs and the different stages that are involved in it. More information about the voting process can be found in [5]. Generally speaking, the voting process in DAOs can be divided into three main stages: Proposal process, voting, and post-voting implementation.

3.2.1. Proposal Process. Unlike corporations with annual meetings and management/shareholder proposals with specific requirements, DAOs typically have a continuous proposal submission process by members. This often involves two main steps: initial discussion and "temperature check" on governance forums, followed by a formal on-chain or off-chain proposal. Some DAOs require a minimum token quorum to create proposals.

3.2.2. Voting Procedure. DAOs specify quorum requirements in their protocol, often lower than in traditional corporations(where it is usually more than 50 percent) due to a lack of voter participation. Token holders' voting power is usually determined at the creation of the proposal. Many DAOs allow members to revote their tokens. It is important to note that tokens that are used for voting are not consumed in the voting process (however, they

 doi: 10.2313/NET-2025-11-3_07

might be staked, and as a result, they will not be accessible for a short amount of time),meaning that the tokens can be used for other purposes, such as trading after the voting process. Another key difference from traditional voting is the real-time transparency of individual token holders' votes in DAOs, Which means every member of the DAO is able to see the votes of the other members.

3.2.3. Post-Voting Implementation. : On-chain votes are automatically executed by smart contracts upon passing, often with a "timelock delay" for security. Off-chain votes (e.g. via Snapshot) typically require a trusted core team or multisig setup to implement the changes on-chain, introducing an element of centralization.

3.3. Voting Models and Strategies

In recent years, various governance models and strategies have been developed to address the challenges of DAO governance mentioned in the Analysis section. These models seek to improve decision-making processes, encourage active participation, and ensure the health of decentralized systems. Below, we explore some of the key voting models used in DAOs.

3.3.1. Token-based Quorum Voting. In token-based quorum system, voting power is directly proportional to the number of tokens held by a participant. Each token represents one vote, and decisions are made based on the majority of votes cast. Depending on the quorum requirement, a certain percentage of tokens must be in favor of the decision for it to be valid.

This approach is straightforward and easy to understand, making it accessible for participants [6]. Token-based quorum voting is the simplest and most basic voting model in DAOs. However, it can lead to centralization, as large token holders may dominate decision-making processes.

3.3.2. Quadratic Voting. Quadratic voting is a voting mechanism which allows parties to express not just their vote but also the intensity of their vote. Instead of a simple one-vote-per-person or one-token-one-vote system, the cost of casting additional votes on a proposal increases quadratically. This means that while individuals can vote multiple times, each additional vote becomes exponentially more expensive.

The primary goal of quadratic voting is to reduce the power of large token holders and promote more democratic decision-making. For example, if the cost of one vote is one token, then casting two votes costs four tokens, three votes cost nine tokens, and so on. This prevents wealth-based manipulation and ensures that only strongly supported proposals gain traction [5].

3.3.3. Conviction Voting. Conviction voting is a dynamic model where participants continuously allocate their votes (in this case, tokens) to proposals over time. Instead of a one-time vote, the voting power of a proposal increases as more participants commit tokens to it for a longer period. The longer tokens remain staked on a proposal, the more "conviction" it accumulates, making it more likely to be accepted.

This approach encourages sustained support and prevents short-term speculative voting. A proposal that receives steady, long-term backing is prioritized over one that receives sudden but fleeting support. This model aligns governance with the real commitment of stakeholders rather than short-term advantage seekers [5].

3.3.4. Tenure-Based Voting (Vote-Escrow). Tenure-based voting, also known as vote-escrow, ties voting power to the duration for which tokens are held or locked. Members who stake their tokens for a longer period gain greater voting influence, incentivizing long-term commitment and reducing short-term speculative behavior.

The key idea behind vote-escrow is to reward long-term investors and contributors over transient participants. For instance, a user locking tokens for one year might receive one vote per token, while a user locking tokens for four years could receive four votes per token. Contrary to conviction voting, where the age of the proposal matters, the age of the tokens plays the major role in vote-escrow [5].

3.3.5. Reputation-Based Voting. In reputation-based voting, a participant's voting power is determined by their contribution history, expertise, or past decisions within the DAO. Instead of relying solely on token holdings, this model assigns weight to votes based on the trust and reliability of the voter within the ecosystem.

The main objective of reputation-based voting is to ensure that governance decisions are influenced by experienced and credible participants rather than just wealthy token holders. For example, a researcher who has actively contributed to the DAO over several years may have more voting power than a newcomer with many tokens but no prior engagement.

This approach incentivizes responsible participation and helps maintain decision-making quality. Since trust between parties and the party's reputation is the measure of voting power, consistent communication is necessary [5].

3.3.6. Contribution-Based Voting. Contribution-based voting rewards participants with voting power based on their involvement in the DAO's ecosystem. Instead of token ownership being the primary determinant, voting rights are allocated based on contributions such as code development, governance participation, content creation, or research efforts.

This model encourages active engagement and rewards meaningful contributions to the organization. A developer who contributes to maintaining the testbed infrastructure receive additional voting power compared to a passive token holder. By tying governance power to active participation, this model ensures that decision-making remains aligned with the DAO's long-term success [5].

4. Discussion

As mentioned in [5], we need to differentiate between an internal and external governance model. Since there are more researchers from different institutions involved in the DAO, There needs to be an internal governance

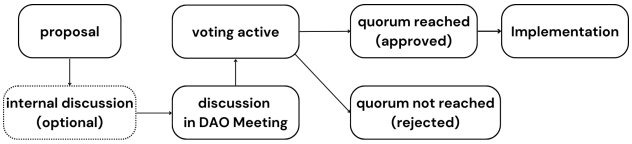


Figure 1: Proposal Process

system in each institution, which is responsible for token distribution and internal decisions.

The external governance model is responsible for the overall governance of the DAO, where each institution is an entity in the DAO and can vote on the decisions that affect the whole DAO. The paper [3] suggests that the internal governance model should be centralized, to simplify the governance of the DAO. This discussion focuses on the external governance model.

Most of the DAOs implement a hybrid governance system which consists of on-chain and off-chain governance models. Usually, the bigger the DAO gets, the more it leans towards off-chain governance; although our testbed DAO is not that big (since in our DAO, the only entities are the testbeds), the off-chain governance model is still recommended. Since essential decisions are made by the testbeds, such as adding or removing other members, it is important to have a model that is more flexible and allows discussions.

Figure 1 shows the suggested hybrid proposal process in our use case. The off-chain section of the process consists of two steps, internal and exteranl discussion. Upon creating a new proposal, it will be discussed fist internally in each organization (Since different organization have different internal governance model, this step is optional, but still recommended). Afterwards the external discussion takes place, where the representatives of the organizations discuss the Proposal. This debates can be done in the form of periodic meetings and, in urgent cases, in the form of emergency meetings. After the off-chain discussion, the on-chain voting phase will begin. Any proposal that reaches the required quorum will be finalized and executed on-chain via smart contracts.

Each voting model offers unique advantages and trade-offs, depending on the goals of the DAO. Token-based quorum voting is straightforward but can lead to centralization if a few large holders dominate decision-making. While quadratic voting enhances democratic participation, conviction voting prioritizes long-term commitment. Vote-escrow and reputation-based voting ensure governance stability by rewarding long-term stakeholders and trusted contributors, while contribution-based voting promotes engagement. A well-designed governance system may incorporate multiple strategies to balance fairness, decentralization, and efficiency within a DAO. Since in our use case, we aim to implement a governance model that encourages active participation and commitment, the Contribution-Based Voting model may be particularly suitable. By rewarding participants based on their contributions to the research testbed, we can incentivize ongoing engagement and ensure that governance decisions reflect the interests of dedicated stakeholders [7].

5. Conclusion

Selecting the right governance model for the DAO can be a very challenging task, and there is no one-size-fits-all solution. Depending on the scale, purpose, and type of the DAO, different governance models and strategies can be used. Two main focuses in the design of a governance model should be the proposal process and the voting process.

In the proposal process, the DAO should have a clear and structured process for submitting and discussing proposals. The transparency of the votes might seem concerning at first glance, but in the long term, it leads to open discussions and a more democratic decision-making process.

Unevenly distributed voting power among the members of the DAO, which is a characteristic of many DAOs, can seem like a challenge for the DAO governance at first. However, since each DAO has its unique characteristics and goals, leveraging the voting power of desired members can be an advantage for the DAO. In our use case, the members with the most contribution to the DAO have the voting power advantage.

Another aspect that should be considered is the tokenomics of the DAO. Future work should analyze the best tokenomics method for your use case and answer questions such as how tokens should be created, managed, and distributed. Token distribution is a very important aspect of our DAO, as there are always new testbeds joining and leaving the network, it also needs to be clear how to prevent inflation and deflation of the tokens.

References

[1] Algorand, “Algorand Developer Docs,” https://developer.algorand.org/docs.

[2] “Blockchain and Applications,6th International Congress,” 2025.

[3] L. Kleinheinz, “Trustless Resource Sharing between Scientific Testbeds,” 2024.

[4] L. T. Han J., Lee J., “DAO Governance,” 2023.

[5] T. L. Jungsuk Han, Jongsub Lee, “A review of DAO governance: Recent literature and emerging trends,” 2025.

[6] M. Schranz, “Governance and Voting Mechanism Models of DAOs,” https://blokk.studio/blog/governance-and-voting-mechanism-models-of-dao.

[7] C. B. Darcy W. E. Allen, “Blockchain Governance: What we can Learn from the Economics of Corporate Governance,” 2020.

 doi: 10.2313/NET-2025-11-3_07

Adding data visualization to pos-testbeds

Daniel Tarassenko, Kilian Holzinger*, Sebastian Gallenmüller*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: daniel.tarassenko@tum.de, holzinger@net.in.tum.de, gallenmu@net.in.tum.de

***Abstract*—The pos-testbeds provide a controlled and modular environment focusing on reproducible network experiments. At their core, the plain orchestrating service (pos) manages the allocation, configuration, execution, monitoring, and data collection of the experiments. However, pos lacks a user-friendly interface for accessing and visualizing the collected results. This paper presents a solution for this problem: a web-based application that can easily be integrated into the existing testbed infrastructure. Using it, the users can explore the testbeds' result directories in the browser and visualize the collected data on an organized dashboard. This approach supports data accessibility and reproducibility and enhances the overall user experience on pos-based testbeds.**

***Index Terms*—pos-testbeds, reproducibility, plain orchestrating service (pos), results visualization, web application**

1. Introduction

Reproducibility is a fundamental principle of scientific research. As described by Gallenmüller et al. [1], reproducibility enables researchers to verify and validate the results of their own and others' experiments, making it a key source of trust for scientific work.

In the context of computational research, ACM emphasizes [2] that a result is only fully established when it can be reproduced independently. They also define three categories to describe different levels of result validation:

- **Repeatability:** Using the same experimental setup, the same team obtains the same results. (As the weakest requirement for an experiment.)
- **Reproducibility:** Using the same setup and tools, a different team can independently reproduce the original results.
- **Replicability:** Using their own implementation and environment, a different team can reach similar results.

Despite the importance of reproducibility, it remains a challenging task for computer science testbeds. The systems are often complex and consist of many different components, tools, and services, which can significantly impact the results of an experiment, as described by Nussbaum in [3].

To address these challenges and enable reproducible experimentation in networking research the pos-testbeds were developed at Technical University of Munich. These testbeds provide a controlled and modular infrastructure for automated network experiments, which are reproducible by design [4]. With access to such a testbed, researchers from different institutions can run their experiments or verify and replicate the results of other researchers, using the same environment.

With the specially developed plain orchestrating service, the pos-testbeds are able to automate resource allocation, configuration, experiment execution, monitoring, and result collection [1]. While storing the results data in a predefined and structured way, pos lacks a built-in user-friendly interface for accessing and visualizing the collected results. Currently, the users are required to navigate the testbed filesystem manually and search through the output files generated by pos or run external scripts to visualize the data. As accessibility is a key factor for reproducibility, the absence of a user-friendly interface may be a barrier for researchers to effectively access and verify the results. ACM also requires the results to be accessible at least for the reviewers for granting an "Artifacts Evaluated" badge [2].

In this paper, we present an approach to improve the built-in accessibility and visibility of the vast amount of data generated by pos, contributing to better reproducibility and enhancing the user experience of the pos-testbeds. The solution consists of two integrated parts:

- **A dashboard generator**, which generates a static HTML page to organize and visualize data such as the used hardware nodes, energy data, executed scripts, and logs.
- **A lightweight server application**, which allows the user to browse the results directory of the testbed and select an experiment. The dashboard generator is then called for the selected experiment and generates a HTML page, which is displayed in the browser.

The paper is structured as follows: Section 2 provides an overview of the pos-testbeds and the plain orchestrating service itself. Then, Section 3 presents the design goals and structure of the visualization system. Section 4 describes the implementation details. Finally, Section 5 summarizes the results and highlights some possibilities for future improvements.

2. Background

The pos-testbeds are a research infrastructure developed by the Chair of Network Architectures and Services at the Technical University of Munich. They were designed to provide an environment for reproducible and automated network experiments [1]. Now these testbeds are even a part of large-scale European projects like

 doi: 10.2313/NET-2025-11-3_08

SLICES-RI [5] or GreenDIGIT [6], and used by hundreds of students and researchers from various institutions.

Pos ensures exclusive hardware usage by allowing only one experiment per hardware node to run at the same time [1]. In contrast to other platforms like Planet-Lab [3], which rely on container-based virtualization, the pos-testbeds offer bare-metal machines for each experiment. This approach eliminates distortions of the results caused by varying hardware loads at different times, which would make the results unreproducible and unreliable.

Furthermore, while offering many different OS images to choose from, the testbed nodes are live-booted [1]. In other words, the operating systems are not installed on the disks of the test nodes, but are only loaded into the main memory from the management node for each experiment run. This guarantees a clean and consistent software environment for each experiment and avoids any side effects from previous ones, enforcing repeatability.

According to the official pos documentation [7], each testbed consists of multiple test nodes and one central management node. These management nodes are hosting the plain orchestrating service (pos), which is therby the core component of the testbed infrastructure.

Running as a daemon on the management node, pos is responsible for:

- Managing the access among the researchers.
- Configuring and booting the test nodes with the selected OS image.
- Loading the experiment scripts on the test nodes.
- Synchronizing the test nodes and executing the experiment scripts.
- Collecting the logs and results from the test nodes.

For each finished experiment pos stores the results in a defined directory of the following structure:

- `/config` directory containing metadata files like:
  - `allocation.json` with the experiment ID, user name, loop variables, boot parameters, etc.
  - `NODE.json` with the hardware specifications for each node.
- `/energy` directory containing a CSV file per node with the energy data. The CSV files may contain different data sets, but in general, they always include this data for the time period of the experiment execution:
  - Current, Voltage, Power and total consumed energy
- `/setup` directory containing a PDF file with the topology for each node.
- One `/NODE` directory for each node containing files, like:
  - python bootstrap scripts
  - Log files like `status`, `stdout`, and `stderr` for each experiment command, named with the timestamp of their occurrence.

As described by Gallenmüller et al. [4], the entire output of the experiment script is recorded, including utility tools output, executed scripts, vars, device hardware, and topology information. All this information is stored in the result folder of the experiment and guarantees publishability (R5).

The approach of pos aligns with the principles of Open Science and FAIR data management. According to the FAIR principles [8], scientific artifacts should be Findable, Accessible, Interoperable, and Reusable. pos enforces structured data collection and centralized artifact storage, making experimental results easier to evaluate, compare, and reproduce. This also aligns with reproducibility guidelines such as the ACM Artifact Review and Badging standard [2], which emphasize transparency, automation, and accessibility as key enablers for trustworthy computational research.

3. Design

The goal of this solution was to provide the experiment results generated by pos in a user-friendly and accessible way. To achieve this, two main components were developed: a static dashboard generator and a dynamic experiment browser. These two components are not separate solutions or independent approaches, but rather two integrated parts of a single system and are designed to work together.

3.1. Dashboard Generator

As a standalone Python script, the dashboard generator takes a path to the desired pos result directory as an argument and generates a static HTML page to visualize the data stored in it. For the dashboard to be well structured and easy to navigate, it was divided into the following sections: (Screenshots for each section are provided in the appendix.)

Information. Figure 1 shows the information section containing the data taken from `/config/allocation.json`. It displays all key information about the experiment, such as ID, user name, creation date, modification date, used nodes, their global- and loop-variables, as well as their boot parameters. By parsing this file, the dashboard generator can provide a comprehensive and consistent overview of the experiment's configuration.

Nodes. As shown in Figure 2, this section provides hardware specifications for each of the used nodes. This data is taken from `/config/NODE.json` and includes information about the CPUs, RAM, networking cards, and all their interfaces, but also the storage, motherboard, and operating system image the experiment was run on.

Topology. The topology section displays images for each node, showing the network interfaces and their interconnections with other nodes. These images are taken from the testbeds topology directory, which can be defined as a program argument. An example is shown in Figure 3.

Timeline. As probably the most important part of the dashboard, the timeline displays all events for each node in chronological order. By selecting an event, the users are presented with all associated files, such as executed scripts, sent commands, standard output, standard error,

 doi: 10.2313/NET-2025-11-3_08

and status files, which allows them to analyze the experiment execution in detail. The timeline section is displayed in Figure 4.

Energy. The final section of the dashboard is the energy section, shown in Figure 5. It displays the energy consumption data for each node, found in the `/energy` directory. Initially, the data is stored in CSV files with varying columns, but in general, they contain the following datasets: Voltage, Current, Power, and Total consumed energy. The data is measured periodically with a sample rate of about 1 Hz for the duration of an experiment. In this section, users can turn on and off each data row to display only the data they are interested in.

While for the most part, the dashboard generator fulfills the main requirements, as a result visualizing tool, it produces only static HTML pages. This implies that pos would need to call the dashboard generator after every finished experiment to create a page that only contains the results of this specific experiment. As a consequence, the server would need to permanently store all generated HTML pages for each experiment, even if they are never visited after some time. This approach leads to redundant storage usage, as the information would be saved twice: as the original experiment result directory and as a static HTML page. In particular, it also would repeatedly store the identical web page layout structure, such as the header, navigation sidebar, CSS and JavaScript as part of the HTML files for each experiment, despite the fact they are always the same. Another drawback resulting from the static approach is that after making some changes to the dashboard, for example, adding a new section or changing the design, by editing some CSS, all previously generated HTML pages would be outdated. This would require the testbed to regenerate all HTML pages after every change.

3.2. Result Browser

To address these issues, a small flask web application was implemented as the second part of the solution. It is designed to run directly on the management node of each testbed and provide a user-friendly web interface for accessing the results of the testbed. Once configured with the path to the result directory containing all experiment results, it would allow the users of the testbed to browse through all these results from their browser. Figure 6 shows a demo results directory of the experiment browser. Here, the users can find and select the experiment they are interested in. The flask application will then call the dashboard generator to create a dashboard for the selected experiment and display it to the user. The benefit of this approach is, that the dashboard is constructed only in the main memory of the server, without being stored on the disk, therefore solving the problem of storing redundant data. Also, since the dashboard HTML pages are only created on demand, there are no extra operations for creating a dashboard after each experiment. Moreover, this allows the dashboard generator to be updated at any time, as the result browser will just call the updated version of the generator. So the users will get displayed the latest version of the dashboard on every experiment visit.

By replacing manual file navigation with an interactive web interface, the experiment browser improves accessibility. The dashboard follows established design principles such as logical grouping, progressive disclosure, and hierarchical layout, which reduce cognitive load and contribute to a more intuitive user experience [9]. Together, these two components significantly improve the usability of the pos-based testbeds and support reproducibility [10].

4. Implementation

4.1. Project structure

This section gives some insights into the details of the proposed solution's implementation. The project is structured as follows:

```
dashboardGenerator/
├── static/
│   └── images/
│       ├── tum-logo.svg
│       ├── folder.svg
│       └── ...
├── templates/
│   ├── base.html
│   ├── dashboard.html
│   ├── section_energy.html
│   ├── section_information.html
│   └── ...
├── generator.py
├── results_browser.py
└── README.md
```

4.2. Dashboard Generator

The dashboard generator is a standalone Python script (`generator.py`), which is the core of the visualization system. While it is not recommended to run the script directly for creating HTML pages, it may be used for testing purposes. The script takes three optional arguments: (`-i, --input`), (`-t, --topologies`) and (`-o, --output`). They define the paths to the experiment result directory, the topology images directory, and the output directory, respectively. The argument parsing is implemented using the `argparse` library. When executed, the script reads all files from the experiment directory and renders a static HTML page using Jinja2 templates.

4.2.1. Data Parsing. First of all, the dashboard generator parses all relevant files from the experiment directory. JSON files such as `allocation.json` and `NODE.json` are read using the built-in `json` module, while the energy data is processed using the built-in `csv` module. To improve the performance of the dashboard in the browser and reduce the file size of the generated HTML file, the energy data from the CSVs is downsampled to a default value of 200 data points per node. Since downsampling was not the main focus of this paper, a simple algorithm is currently used, that selects every N-th row and discards the rows in-between. This approach is very easy to implement, fast, and sufficient for demonstration purposes. In the future, a more complex algorithm like LTTB (Largest-Triangle-Three-Buckets) could be integrated to improve the downsampling quality. Unlike the N-th row approach, which could omit important values like peaks, LTTB selects the most significant points of the dataset, thereby preserving the visual characteristics of the original data.

 doi: 10.2313/NET-2025-11-3_08

All the parsed data is then stored in nested Python data structures, like lists of dictionaries or lists of JSON objects, which are then passed to the Jinja2 template engine for rendering the HTML page.

4.2.2. Template Rendering. The core part of the dashboard generator is the Jinja2 templating engine, which was chosen for its simple integration as a Python library, the Python-like syntax, good documentation and support for modular templates. It enables the creation of HTML templates with not only predefined static data, such as the layout elements but also dynamic placeholders, which are then filled in the rendering process on execution, as described in the Jinja2 documentation [11]. Furthermore, Jinja2 templates can contain variables, loops and conditions for displaying dynamic data, like tables or lists. An example of a Jinja2 loop is shown below:

```
<tbody>
    {% for node in nodes %}
    <tr>
        <td>{{ node.hostname }}</td>
        <td>
            <ul>
                {% for cpu in node.processor %}
                <li>{{ cpu.model }}</li>
                {% endfor %}
            </ul>
        </td>
        ...
        <td>{{ node.image }}</td>
    </tr>
    {% endfor %}
</tbody>
```

This code snippet would generate a table row for each node in the list of nodes while listing all CPU models within a table cell.

Another feature of Jinja2 is the support of template inheritance. Thus, a base template (`base.html`) defines the overall layout structure, like the page header. This base template is then extended by `browse.html` and `dashboard.html`, which are also templates.

For better code structure and easier maintainability, each section of the dashboard - `Information`, `Nodes`, `Topology`, `Timeline`, and `Energy` - is implemented in its own template file. Each of these section templates is then included in the `dashboard.html` template.

4.2.3. Layout and Interactivity. To ensure a clean and intuitive user interface, the Bootstrap framework [12] was used. Bootstrap provides many ready-to-use components for web applications, such as accordions, cards, buttons, and more. In particular, the accorions as collapsible boxes, are used to group the information, structure the dashboard and visually hide as much information as possible. Thus, the users are not overwhelmed by too much information at once and giving them control over how much detail they want to see.

The required Bootstrap CSS and JavaScript are currently included with CDN links in the `base.html` template. In the future, only the used assets could be compiled from Bootstraps SASS source files and stored locally on the testbed. This would avoid loading unused assets, reduce external dependencies and obsolete internet access.

In addition to Bootstrap, some simple JavaScript is used to hide all the dashboard sections (`Information`, `Nodes`, `Topology`, `Timeline`, and `Energy`) and only show the section selected on the sidebar. In combination with Bootstrap elements, this contributes to a clean structure and an interactive user experience.

4.3. Result Browser

The second part of the visualization system is a small web application built using Flask, a lightweight web framework for Python [13]. As it is intended to run directly on the management node of each pos-testbed, the path to the testbed base directory must be provided as an argument `-d, --directory`. Furthermore, the arguments (`-H, --host`) and (`-p, --port`) are used to specify the host and port of the web server. The application defines the following endpoints:

- `/browse/<path>` – Allows navigation through the results directory and displays its contents.
- `/experiment/<path>` – Initiates a dashboard generation and returns an HTML for a selected experiment.
- `/topology/<filename>` – Serves topology images on demand, as they are not stored in the static directory but on the testbed.

When listing a directory while browsing, the application checks for each subfolder if it is an experiment result directory. This is done by checking if the `config/`, `energy/`, and `setup/` directories are present. If so, the experiment directory is displayed with a link to the `/experiment/<path>` endpoint. When an experiment directory is clicked, a dashboard is generated using a direct function call to `generate_page()` from `generator.py`. The rendered HTML is delivered to the browser using a `Response` object without storing it on the disk of the hosting management node.

This on-demand generation of the dashboard avoids redundant file storage and enables the dashboard generator to be updated at any time, as already discussed in Section 3. In its current state, the application is designed for internal use only, as no authentication or encryption is implemented. This is because no HTTPS certificate was available during the development.

4.4. Performance Considerations

As the proposed visualization solution is designed for occasional, human-triggered interactions, rather than continuous or high-frequency data processing, the performance is not a critical factor for the system. Therefore a formal performance evaluation was left out of the scope of this paper. Nevertheless, some considerations are worth mentioning:

- The execution time of the dashboard generator depends on the size of the given experiment result directory. In particular, the number of used nodes and amount of recorded events for each node plays a significant role, as all these files must be parsed, saved in a data structure, and then rendered to an HTML page.
- JinJa2 extensively uses caching and avoids repeated compiling of the templates, making the rendering process nearly as efficient as executing a Python function [14].

- Since the dashboard generator is only invoked on demand, the system avoids redundant processing and remains idle most of the time.

5. Conclusion and Future Work

This paper presented a lightweight and modular solution to improve the accessibility and visibility of experiment results generated by the plain orchestrating service (pos) in reproducible testbed environments. The system consists of two parts: a static dashboard generator and a dynamic experiment browser, both implemented in Python. After integrating these tools directly into the testbed infrastructure, users can access information like experiment metadata, execution logs, hardware information, and energy consumption through an interactive and structured interface. This improves the user experience and supports reproducibility by making data more accessible and easier to interpret.

Future Work: Responsiveness improvements could enable usage on mobile devices. While security was not a focus in this prototype, future versions should include HTTPS and user authentication using login, to support secure and used-specific access to experiment results. Filter and sorting functions in the experiment browser would improve navigation, especially for users with many experiments. Integration of live result data streaming while experiment execution would allow real-time monitoring of experiments. Finally, exporting results in RO-Crate format [15] would enable standardized, machine-readable packaging of experiments and simplify publishing and sharing via FAIR-compliant repositories.

References

[1] S. Gallenmüller, D. Scholz, H. Stubbe, and G. Carle, "The pos framework: A methodology and toolchain for reproducible network experiments," *The 17th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '21)*, 2021.

[2] ACM, "Artifact review and badging version 1.1," https://www.acm.org/publications/policies/artifact-review-and-badging-current, 2020, online; Last access: 05.04.2025.

[3] L. Nussbaum, "Testbeds support for reproducible research," *Proceedings of Reproducibility '17*, 2017.

[4] S. Gallenmuller, D. Scholz, H. Stubbe, E. Hauser, and G. Carle, "Reproducible by design: Network experiments with pos," *Würzburg Workshop on Next-Generation Communication Networks (WueWoWas'22)*, 2022.

[5] S. R. Infrastructure, "Post-5g blueprint," https://doc.slices-ri.eu/BlueprintServices/beyond5G/beyond5G.html, 2024, online; Last access: 05.04.2025.

[6] C. of Network Architectures, I. Services, School of Computation, and T. U. o. M. Technology, "Greendigit," https://net.in.tum.de/projects/greendigit/, 2024, online; Last access: 05.04.2025.

[7] D. Scholz and S. Gallenmüller, "Welcome to the plain orchestrating service (pos)," https://i8-testbeds.pages.gitlab.lrz.de/pos/cli/#welcome-to-the-plain-orchestrating-service-pos, 2025, online; Last access: 04.04.2025; Note: Internal documentation, access restricted to registered users only.

[8] M. D. W. et al., "The fair guiding principles for scientific data management and stewardship," *Scientific Data*, 2016.

[9] UXPin, "Effective dashboard design principles for 2025," https://www.uxpin.com/studio/blog/dashboard-design-principles/, 2025, online; Last access: 10.05.2025.

[10] J. Leipzig, D. Nüst, C. T. Hoyt, K. Ram, and J. Greenberg, "The role of metadata in reproducible computational research," *Patterns*, 2021.

[11] Pallets, "Jinja — jinja documentation (3.1.x)," https://jinja.palletsprojects.com/en/stable/, 2025, online; Last access: 05.04.2025.

[12] T. B. Authors, "Get started with bootstrap," https://getbootstrap.com/docs/5.3/getting-started/introduction/, 2025, online; Last access: 05.04.2025.

[13] Pallets, "Welcome to flask — flask documentation (3.1.x)," https://flask.palletsprojects.com/en/stable/, 2025, online; Last access: 05.04.2025.

[14] Pallets, "Frequently asked questions," https://jinja.palletsprojects.com/en/stable/faq/, 2025, online; Last access: 10.05.2025.

[15] E. Hauser, S. Gallenmuller, and G. Carle, "Ro-crate for testbeds: Automated packaging of experimental results," *IFIP Networking Conference (IFIP Networking)*, 2024.

Appendix

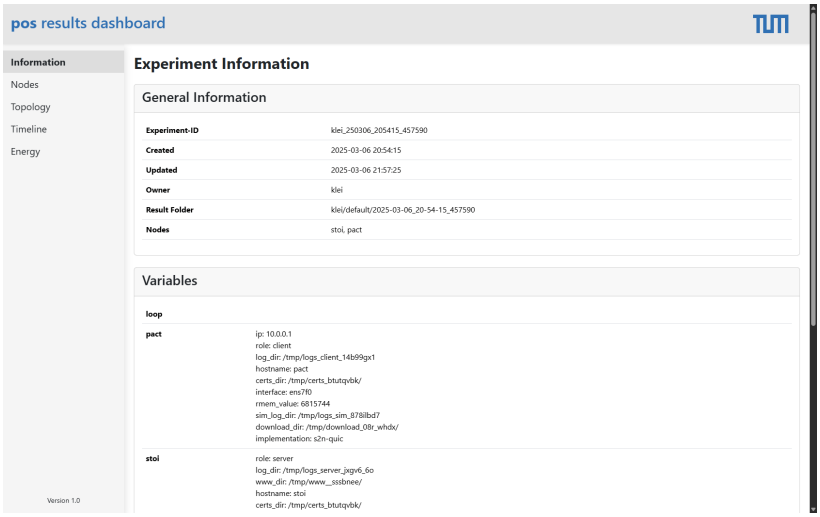


Figure 1: Informatin section

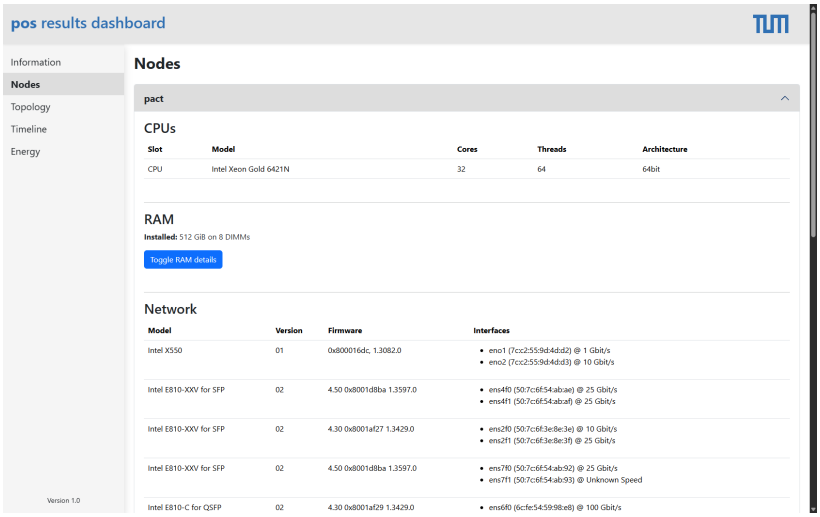


Figure 2: Nodes section

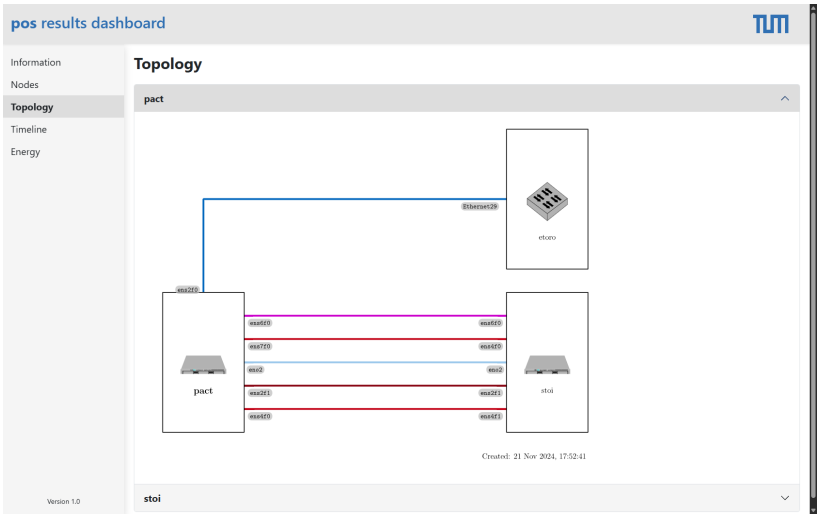


Figure 3: Topology section

 doi: 10.2313/NET-2025-11-3_08

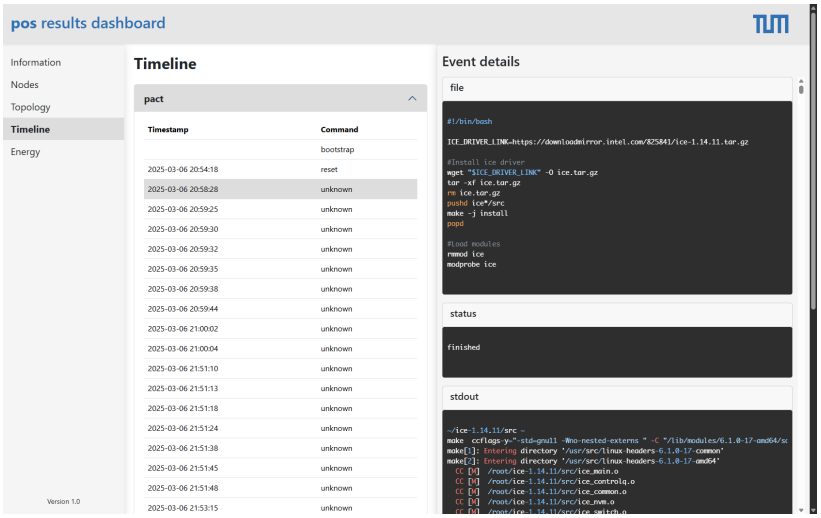


Figure 4: Timeline section

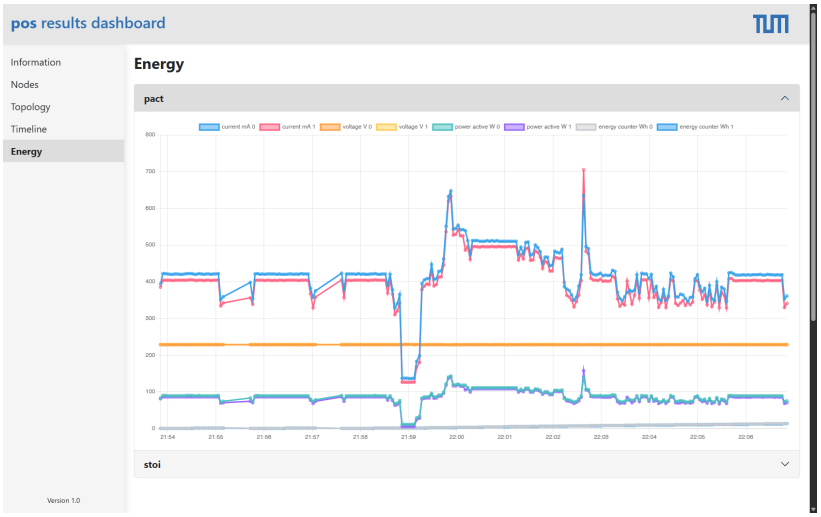


Figure 5: Energy section

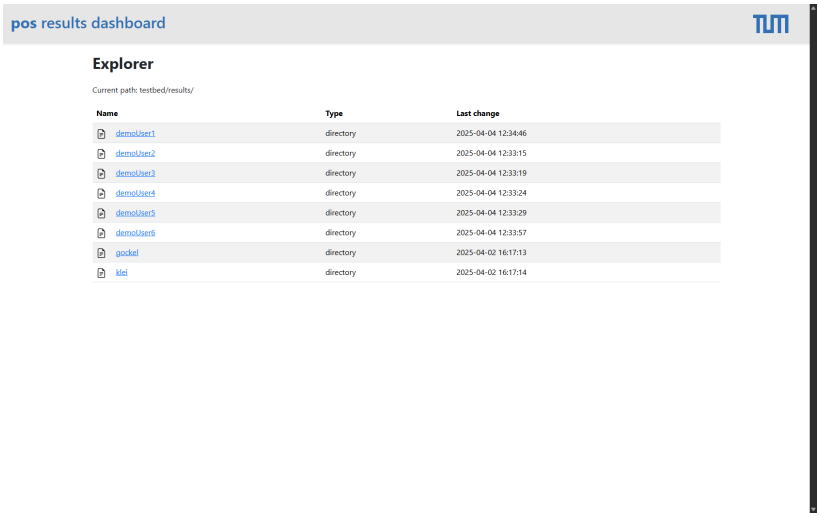


Figure 6: Experiment browser

Timeline of Host Monitoring Tools

Zeynep Öztürk, Tim Betzer*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: zeynep.oeztuerk@tum.de, betzer@net.in.tum.de

***Abstract*—Network monitoring is a fundamental component of ensuring the reliability, performance, and security of computer networks.**

This paper traces the evolution of host monitoring tools, from early packet-switching networks to modern observability platforms. It highlights the increasing complexity of network environments and the corresponding need for advanced monitoring solutions. The paper explores the integration of monitoring tools in cloud-native infrastructures, with particular attention to Kubernetes orchestration. Prometheus, a tool with a pull-based architecture, and Nagios, a widely-used legacy system, are examined as case studies to illustrate the trade-offs between traditional and modern approaches. The paper concludes by discussing current challenges in network observability, such as automation, real-time alerting, and the role of open-source tools in contemporary IT infrastructures.

***Index Terms*—network monitoring, host monitoring, SNMP, Prometheus, Nagios, Kubernetes, network management, network performance monitoring**

1. Introduction

Since the introduction of the first computer networks, the need for monitoring and managing them has been a critical aspect of ensuring performance and reliability. Network monitoring began in the 1960s and 1970s with simple packet-switching systems and basic command-line utilities. These tools enabled administrators to check device statuses and troubleshoot connectivity issues.

As networks expanded in size and complexity, it became clear that more sophisticated monitoring solutions were needed. Early tools, often developed in-house, were limited in functionality and required substantial manual effort. Despite their limitations, they laid the foundation for the development of more advanced systems.

The 1980s marked the emergence of commercial network monitoring tools, which were generally proprietary, expensive, and used by large organizations. These tools provided a more comprehensive view of network performance, helping administrators address bottlenecks and faults more effectively.

A significant milestone was the introduction of the Simple Network Management Protocol (SNMP) in the late 1980s. SNMP offered a standardized method to monitor and manage devices across different vendors, simplifying the deployment and maintenance of monitoring infrastructures. This standardization supported the rise of both commercial and open-source monitoring tools.

This paper provides a comprehensive overview of the evolution of host monitoring tools, beginning with traditional methods and continuing to modern, cloud-native observability platforms. It examines the fundamental Push vs. Pull data collection models and presents Prometheus and Nagios as case studies to explore the trade-offs between legacy and modern solutions. The discussion concludes with an analysis of current trends, such as real-time alerting, automation, and the growing importance of open-source tools in today's IT environments.

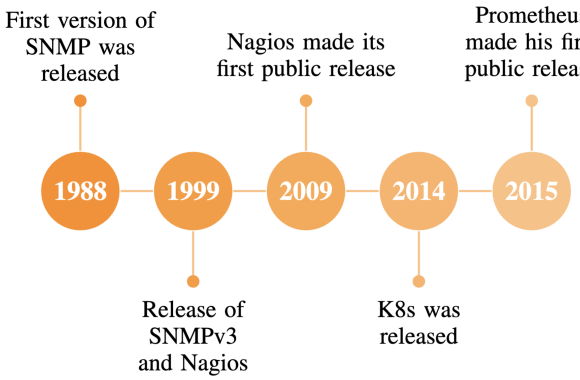


Figure 1: Timeline of host monitoring tools

2. Push vs. Pull

The general principle for collecting data in monitoring systems is divided into two primary models: the **Push Model** and the **Pull Model**.

2.1. Push Model

In the push model, monitored devices send their status information autonomously to the monitoring system. This typically occurs through agents or log forwarding and is well-suited for continuous management.

An example of a tool following this model is Checkmk, which uses agents to transmit data.

2.2. Opportunities and Challenges of the Push Model

One of the key advantages of the push model is the ability to detect problems quickly, as devices transmit alerts as soon as an issue arises. This is particularly beneficial in environments where real-time monitoring

 doi: 10.2313/NET-2025-11-3_09

is essential, such as financial institutions or healthcare systems. Additionally, the push model can help reduce overall network load since data is not polled constantly.

However, the push model presents certain drawbacks. It generally requires additional configuration effort, such as deploying and maintaining agents on the monitored devices, which can be time-consuming and operationally complex. Moreover, since data is transmitted continuously, this approach can lead to increased storage demands. This is especially problematic in environments with limited storage capacity or strict data retention policies.

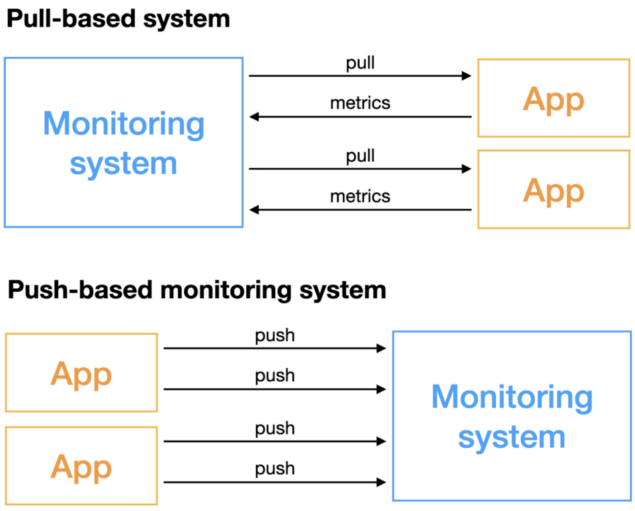


Figure 2: Push vs Pull
Source: [1]

2.3. Pull Model

In contrast, the pull model is based on the principle that the monitoring system initiates data collection by querying the monitored devices at regular intervals. This follows the request–response paradigm and is commonly implemented using protocols such as SNMP, WMI, or API-based interfaces. The pull model is particularly suitable for ad hoc monitoring and manual, on-demand analysis.

A practical example is Nagios, where the monitoring server polls the status of devices—such as servers or network hardware—at fixed intervals, typically every five minutes [2].

2.4. Opportunities and Challenges of the Pull Model

The pull model offers several advantages. It gives the central monitoring system control over the frequency of data collection, which can help balance network load and optimize system performance. Another benefit is the reduced complexity in deployment, as it does not require agents to be installed on the monitored devices.

On the downside, the pull model can introduce latency in problem detection. Since data is only collected at scheduled intervals, issues may go unnoticed until the next polling cycle. This delay can be problematic in environments that require real-time responsiveness, such as financial or healthcare sectors [3].

3. SNMP

The first attempts to create a unified monitoring tool date back to the 1960s and 1970s, when it became evident that the number of hosts in computer networks was growing rapidly [4]. During that time, monitoring systems were mostly proprietary, tailored to specific hardware and vendor environments.

The development of the **Simple Network Management Protocol** (SNMP) in the 1980s by a group of university researchers marked a turning point. It introduced a standardized framework for monitoring and managing devices across heterogeneous networks.

Although SNMP was originally intended as a temporary solution—"to fill the need for a network management tool while a more theoretically sound system was being developed by the IAB" [4]—it ultimately became the de facto standard, and the intended transition never occurred.

The first version of SNMP was released in 1988, with a design philosophy focused on simplicity and ease of implementation. In 1993, **SNMPv2** was introduced, bringing enhancements such as bulk data retrieval and improved error handling. However, it still relied on community strings for authentication, which were considered insecure. In response to these concerns, **SNMPv3** was released in 1999, offering robust security features such as authentication, encryption, and message integrity [4].

Today, SNMPv3 is the standard protocol for monitoring and managing network devices, including switches, routers, firewalls, printers, and servers.

3.1. Evaluation

One of the key factors behind SNMP's widespread adoption is its open nature. It is free software, not controlled by any specific vendor, allowing organizations to implement or build SNMP-based tools without incurring licensing costs or facing vendor lock-in. This has led to a broad ecosystem of SNMP-compatible devices and software solutions.

This openness is especially valuable in network management, where organizations often deploy heterogeneous environments with equipment from various vendors. SNMP provides a unified framework for managing such diverse devices, enabling centralized visibility and control [5].

Although SNMP has not demonstrated significant shortcomings in fault and performance management since its introduction, its continued prevalence can also be attributed to the complexity involved in migrating to alternative protocols or platforms.

Nevertheless, SNMP does have certain limitations. Being a pull-based protocol, it relies on the management system to periodically query devices for information. This approach can cause performance bottlenecks in large-scale environments, where thousands of devices may need to be polled regularly. Moreover, SNMP lacks robust support for complex data modeling and does not efficiently capture relationships between devices. These factors limit its scalability and adaptability in modern, dynamic network infrastructures, where topologies change frequently and systems must respond in real-time [4], [6], [7].

4. Kubernetes

Kubernetes, commonly known as K8s, is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications across clusters of machines.

4.1. History

The initial public commit of Kubernetes was made on GitHub on June 6, 2014. Shortly afterward, in September of the same year, Google officially announced Kubernetes as the open-source successor to its internal container management system, known as Borg.

From the beginning, several major technology companies, including Microsoft, RedHat, IBM, and Docker, joined the Kubernetes community, contributing to its rapid development and adoption. In 2015, with the release of Kubernetes version 1.0, Google donated the project to the newly established **Cloud Native Computing Foundation** (CNCF).

The CNCF, which operates under the umbrella of the nonprofit Linux Foundation, is a vendor-neutral organization that supports the development and adoption of cloud-native technologies. It serves as a collaborative home for open-source projects that are critical to the cloud-native ecosystem.

By 2016, Kubernetes became the first project to be officially hosted by the CNCF. In 2017, it achieved another milestone by becoming the first project accepted into the CNCF's Incubation Program. The following year, in 2018, Kubernetes was the first project to graduate from the CNCF, signaling its maturity and widespread adoption.

Today, Kubernetes is recognized as the second-largest open-source project in the world, surpassed only by the Linux kernel. It has become the default container orchestration solution used by approximately 71 percent of Fortune 100 companies. Kubernetes has played a pivotal role in shaping modern cloud-native architectures, setting a standard for cloud service providers and transforming how applications are developed, deployed, and managed in distributed environments [8], [9].

As of now, Kubernetes continues to be one of the fastest-growing and most actively maintained open-source projects in software history.

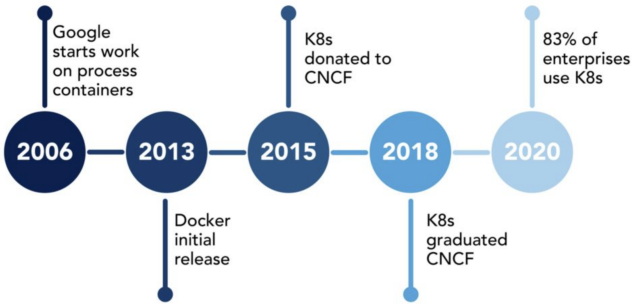


Figure 3: A brief history of Kubernetes
Source: [10]

4.2. Evaluation

Kubernetes offers numerous advantages that have made it the dominant platform in container orchestration. A key benefit is its ability to automatically scale applications based on real-time demand, improving resource efficiency and reducing operational costs. It ensures high availability of services through built-in features such as self-healing, load balancing, and automated failover, thereby enhancing system reliability and minimizing downtime.

Additionally, Kubernetes supports workload portability and seamless integration across hybrid and multi-cloud environments. This allows organizations to deploy and manage applications consistently across different infrastructures, which is particularly beneficial for enterprises with complex IT landscapes. Its robust networking capabilities also simplify service discovery, DNS management, and communication among distributed application components.

Despite its strengths, Kubernetes presents a range of challenges. Its architecture and operational model introduce significant complexity, particularly during initial deployment and cluster setup. The learning curve for administrators and developers new to Kubernetes can be steep, often requiring deep technical understanding and hands-on experience. Running Kubernetes in a production-grade environment demands continuous monitoring, fine-tuning, and specialized skills in areas such as networking, security, and observability.

Moreover, Kubernetes can be resource-intensive, which may not align well with the constraints of small-scale projects or low-powered edge computing environments. Debugging and incident resolution within Kubernetes clusters can also be complex due to the distributed nature of workloads and the limited visibility provided by default tools. In many cases, advanced observability solutions must be integrated to perform effective root cause analysis [8], [9], [11].

5. Prometheus

Started as an internal project at SoundCloud in 2012, **Prometheus** is now a widely used open-source monitoring and alerting toolkit. It focuses specifically on cloud-native environments, where it excels in collecting, storing, and analyzing metrics in real time.

After gaining increasing adoption within SoundCloud, and later from Docker and Boxever in 2014, Prometheus had its first public launch on January 26, 2015. Within one year of open-source development, Prometheus had grown into a thriving community of users and contributors. The project saw significant activity, with over 200 contributors, more than 2,300 pull requests, and over 4,800 stars on GitHub. Even large companies such as Google and CoreOS began using Prometheus in their infrastructure. "Google is now instrumenting its open-source container management system Kubernetes natively with Prometheus metrics," stated Volz [12].

On May 9, 2016, Prometheus joined the Cloud Native Computing Foundation (CNCF) as its second open-source project, following Kubernetes [5].

 doi: 10.2313/NET-2025-11-3_09

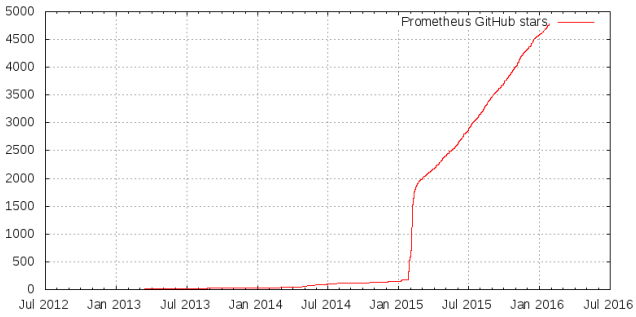


Figure 4: Peak after getting submitted to Hacker News
Source: [13]

5.1. Evaluation

Prometheus is a powerful monitoring and alerting toolkit, well-suited for cloud-native environments. Due to its pull-based architecture, it easily monitors dynamic systems, making it a popular choice for organizations using microservices and container orchestration platforms like Kubernetes. It is designed to be fast and efficient, with a time-series database that enables high-performance data storage and retrieval. It also supports flexible monitoring capabilities, allowing users to define custom metrics and alerts based on their needs. Additionally, it is highly scalable and can manage large volumes of data, making it suitable for organizations with complex infrastructures.

However, it does have some limitations. Prometheus does not provide native long-term storage for metrics, which can be a drawback for organizations that need to retain historical data for compliance or analytical purposes. It also focuses primarily on metrics and lacks built-in support for logs or distributed clustering. This can make it less suitable for organizations that require a more comprehensive monitoring solution that includes logs and distributed tracing [5], [11], [14], [15].

6. Nagios

Nagios is one of the oldest and most widely known monitoring tools for IT infrastructures. It enables the monitoring of servers, networks, applications, and services to detect and respond to issues proactively.

Nagios was developed by Ethan Galstad in 1999 as a side project called NetSaint. It was initially developed as an open-source solution for network monitoring, with a primary focus on hosts and services. Due to a trademark issue with the term “Saint” in 2001, the project’s name was changed to Nagios, an acronym for “Nagios Ain’t Gonna Insist On Sainthood“ [2]. In 2007, Nagios Enterprises was founded, and after two years of development, the first commercial edition of Nagios was released on December 31, 2009.

6.1. Evaluation

Thanks to its plugin-based architecture, Nagios can be extended to monitor a wide range of services and applications, such as the availability and performance of network devices, servers, and applications. It also supports monitoring system metrics including CPU usage, memory consumption, disk space, and network traffic. Widely adopted and well established, “Nagios was awarded ‘Best Monitoring Application’ in the 2013 Linux Journal Readers’ Choice Awards“ [16]—its third time receiving this title.

Despite its capabilities, Nagios is not ideal for large, dynamic environments where services and hosts change frequently. In such cases, the manual configuration requirements of Nagios can become cumbersome and time-consuming [2], [16].

7. Conclusion and future work

Since the beginning of the computer networks and the rising of the complexity of the network systems, the need for monitoring tools has increased. Since then SNMP has been the most widely used protocol for monitoring network devices and Kubernetes has become one of the most popular container orchestration platform. With the Prometheus monitoring system, it is possible to monitor the Kubernetes cluster and the applications running on it. Nagios is a widely used monitoring tool, which is used to monitor the host system.

Although commonly used host monitoring tools perform well for most modern systems, the continuous advancement of technology necessitates the development of newer and more sophisticated monitoring solutions. One emerging area of focus is the monitoring of **quantum computers**, which are gaining popularity and present unique challenges. Unlike classical systems that operate with bits representing either 0 or 1, quantum computers use qubits, which can exist in a superposition of states. This fundamental difference makes monitoring quantum systems significantly more complex than traditional computing environments.

Nevertheless, major companies such as IBM and Amazon are actively working on solutions in this domain. For instance, Qiskit, an open-source quantum computing framework developed by IBM, includes components that support basic monitoring of quantum systems.

As quantum computing technology continues to evolve, the demand for advanced monitoring tools will grow in order to ensure the performance, stability, and reliability of future quantum-based infrastructures [17].

References

[1] “Push vs pull architecture in cloud computing,” https://www.alibabacloud.com/blog/push-vs-pull-architecture-in-cloud-computing_595202, [Online; accessed 6-April-2025].

[2] Nagios, “The nagios story,” https://www.nagios.org/story/, [Online; accessed 30-March-2025].

[3] J.-P. Martin-Flatin, “Push vs. pull in web-based network management,” 1998.

[4] “Snmp history,” https://docs.lextudio.com/snmp/snmp-history, [Online; accessed 1-April-2025].

[5] “The prometheus monitoring system and time series database.” https://www.cncf.io/projects/prometheus/, [Online; accessed 31-March-2025].

[6] M. Fedor, M. Schoffstall, J. R. Davin, and D. J. D. Case, “A simple network management protocol (snmp),” May 1990.

[7] K. Brister, “High tech goes low key,” https://snmp.com/company/history.shtml, 1998, [Online; accessed 1-April-2025].

[8] “Kubernetes project journey report,” 2023, [Online; accessed 28-March-2025].

[9] S. Susnjara, “The history of kubernetes,” https://www.ibm.com/think/topics/kubernetes-history, 2023, [Online; accessed 28-March-2025].

[10] “Who made kubernetes?” https://www.opsramp.com/guides/why-kubernetes/who-made-kubernetes/, [Online; accessed 6-April-2025].

[11] N. S. an Elizabeth Bautista, “Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus,” IEEE, Ed., 2019.

[12] D. Harrington, B. Wijnen, and R. Presuhn, “An architecture for describing simple network management protocol (snmp) management frameworks,” *RFC 3411*, 2002.

[13] “One year of open prometheus development,” https://prometheus.io/blog/2016/01/26/one-year-of-open-prometheus-development/, [Online; accessed 6-April-2025].

[14] G. Darwesh, “Kubernetes monitoring with prometheus for security purposes,” in *Sammelband wissenschaftlicher Forschungsarbeiten auf Grundlage des internationalen Wettbewerbs, Technologische Innovationen und wissenschaftliche Entdeckungen*, A. A. Vorobeva, Ed.

[15] J. Turnbull, *Monitoring with Prometheus*, September 10, 2018.

[16] J. Gray, “Reader’s choice awards 2009,” *Linux Journal*, June 1, 2009.

[17] “Introduction to qiskit,” https://docs.quantum.ibm.com/guides, [Online; accessed 5-April-2025].

QWACs in Automated Environments

Leonard Auer, Stefan Genchev*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: auerle@in.tum.de, genchev@net.in.tum.de

***Abstract*—The European Union's eIDAS 2.0 regulation introduces Qualified Website Authentication Certificates (QWACs) to enhance web authentication through verified organizational identity. To address criticism regarding regulatory influence over the global root trust stores, the ETSI recently proposed a novel approach, 2-QWAC, which binds a QWAC to a standard TLS certificate via a digital signature. This binding process, however, introduces operational complexity, especially in environments where TLS certificates are frequently rotated using the ACME protocol. In this work, we survey common ACME implementations and analyze the feasibility of automating 2-QWAC binding renewal. We present a proof-of-concept automation that extends the ACME flow in a popular and open-source web server.**

***Index Terms*—qualified web authentication certificates, 2-qwac, eidas 2.0, acme, pki, automation**

1. Introduction

The European Union's eIDAS 2.0 regulation introduces Qualified Website Authentication Certificates (QWACs) to add verifiable organizational identities to web authentication [1]. Unlike the Extended Validation certificates, which have been removed from the user interfaces of major web browsers [2], QWACs are issued by Qualified Trust Service Providers (QTSPs) in the EU [3].

Initial criticism from browser vendors and academia focused on the regulatory requirement for browsers to accept TLS certificates from those QTSPs. This would have required including all EU-qualified Certificate Authorities (CAs) into global root stores. Browser vendors and academia raised two main concerns: First, this requirement would circumvent the existing restrictions and processes of global root stores, possibly introducing security risks. Second, government intervention in the web trust model would set a dangerous geopolitical precedent. [4], [5]

In response, the European Telecommunications Standards Institute (ETSI) revised the specification and introduced a novel approach called 2-QWAC. It does not provide backwards compatibility, but co-exists along the original approach now called 1-QWAC. This new framework decouples identity assertion from transport layer security by binding the QWAC to a separate, conventionally issued Transport Layer Security (TLS) certificate. Importantly, the QWAC is not used directly for the TLS handshake but is validated separately via a signed binding. While the specification still mandates visual indicators in web browsers when a website uses a valid QWAC, 2-QWAC provides a compromise between introducing website identity verification under eIDAS 2.0 and recognizing the autonomy of browser vendors since it allows for a separate QWAC root CA store instead of government-controlled additions to a browser's TLS root CA store. [3]

Major browser vendors have started efforts to implement QWACs [6], however, there do not yet exist solutions to integrate them with the TLS tool chains in modern web environments. With increasing TLS automation – encouraged by organizations like the National Institute of Standards and Technology [7] – via the Automatic Certificate Management Environment (ACME) protocol [8], the added complexity of the 2-QWAC binding presents a new challenge.

This paper investigates how such bindings can be automated and synchronized with TLS certificate issuance. We make the following main contributions:

- We provide an overview of popular ACME v2 implementations.
- We analyze integration opportunities for automating the QWAC bindings in a popular ACME v2 implementation.
- We implement a proof-of-concept binding renewal mechanism for a popular open-source web server.

2. Background

For analyzing opportunities for automating the 2-QWAC binding renewal, an understanding of the two main concepts involved is necessary. This chapter provides an overview of those – the 2-QWAC framework and the ACME protocol.

2.1. 2-QWACs

QWACs are X.509 certificates issued by Qualified Trust Service Providers. The certificates contain validated organizational identity attributes about a website's operator. [3] QTSPs are EU-qualified and -supervised CAs that conform to regularly audited legal requirements. Examples include the implementation of the NIS2 directive [9] and obligations to the identity verification process set by the eIDAS legal framework. [1]

The current ETSI specification [3] defines two co-existing approaches:

- *1-QWAC*: The QWAC is used as TLS certificate, combining the identity assertion and TLS handshake. In this approach, QWACs replace standard

 doi: 10.2313/NET-2025-11-3_10

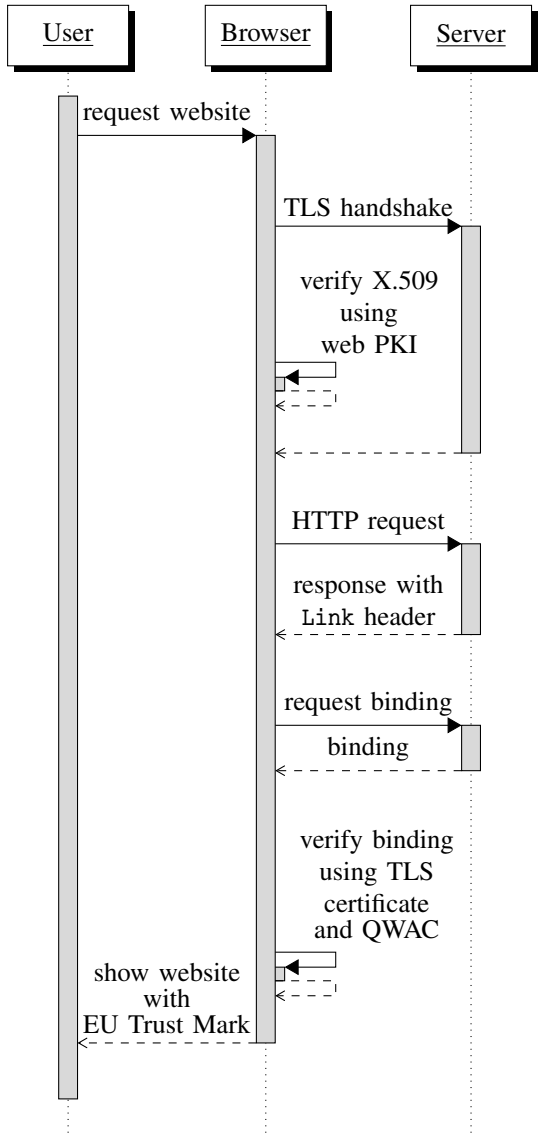


Figure 1: Sequence Diagram for a web request with valid 2-QWAC

TLS certificates. If the certificate received as part of the TLS handshake contains indicators that it is a QWAC, this approach is used.

- *2-QWAC*: The QWAC is digitally bound to a standard TLS certificate. Browsers validate the identity information independently of the TLS handshake.

The 2-QWAC binding is implemented through a JAdES (JSON Advanced Electronic Signature) structure signed using the QWAC private key. The structure references both the QWAC and the TLS certificate fingerprints. Website operators obtain QWACs and create bindings to their TLS certificates. [3]

Figure 1 shows the flow of a typical, successful 2-QWAC binding verification in web browsers. They first validate the TLS certificate using their root trust store as part of the usual TLS handshake. During a TLS session, the web server's response includes an HTTP `Link` header pointing to the binding. Web browsers then fetch the binding and validate it using the separate EU trust store. If validation succeeds, the browser is supposed to show the EU Trust Mark and the identity data of the QWAC. [3]

2.2. ACME

The Automatic Certificate Management Environment protocol standardizes automated certificate issuance. Manually obtaining a TLS certificate usually involves generating a Certificate Signing Request (CSR), sending it to a CA, proving domain ownership, and deploying the issued X.509 certificate on a web server. ACME allows web servers to automatically request TLS certificates, prove domain control, and deploy the certificates with minimal website operator input. [8]

The ACME specification defines the following flow for the automated process: [8]

1) *Account registration*: An ACME client – the web server – requests an account with an ACME server – the CA. The client sends optional contact information or other optional data along with a signature.
2) The client submits a *certificate order*.
3) *Proving domain ownership*: The ACME server sends the client a list of challenges to prove domain control. The client solves those challenges, for example, by creating a domain record containing a unique value (`dns-01`) or returning a unique value in an HTTP response (`http-01`). The ACME server validates the challenge results.
4) The client submits a *Certificate Signing Request* (CSR) to the ACME server.
5) *Deployment*: After the server issues the certificate, the client downloads and installs it.

3. ACME Client Survey

Automating the 2-QWAC binding requires awareness of when TLS certificates are issued or renewed. Thus, we aim to extend the ACME flow with logic to create the binding between the QWAC and the TLS certificate. Additionally, we need to extend the web server, for example, to serve the binding. Thus, an extensible ACME client with high integration with a web server is ideal for our use case.

In this chapter, we highlight three conceptually different, popular ACME v2 implementations. We evaluate them based on two criteria:

- *Integration Level*: This describes how tightly coupled the ACME client is with the HTTP server. Clients that have a high integration level are embedded within the HTTP server runtime and can directly influence the certificate lifecycle and modify web requests. Standalone tools that operate outside the web server have a low integration level. For our purpose, a high integration is ideal.
- *Extensibility*: This refers to how easily the ACME client can be extended. A low extensibility means that the client provides no or a limited plugin API, while a highly extensible client has a modular architecture. Our approach benefits from high extensibility.

Certbot [10] is a widely used command-line ACME client maintained by the Electronic Frontier Foundation. It

 doi: 10.2313/NET-2025-11-3_10

is written in Python and can be used both as a standalone application as well as in combination with an existing web server like *Apache* or *nginx*. Other, not officially supported web servers can integrate Certbot through plugins. [11]

Certbot stores obtained certificates at predefined paths (`/etc/letsencrypt/live/$domain`) and relies on external configuration in the web server to deploy them [11]. Thus, it is loosely coupled with the actual web server. Working with predefined locations should not pose an issue when creating the binding: we know the location of the TLS certificates and can define a path where website operators should place QWACs. However, we would need to additionally extend the web server to serve the binding and modify the HTTP response headers.

cert-manager [12] is a Kubernetes certificate controller written in Go. It automates the TLS certificate issuance and management in a Kubernetes cluster. As part of a modular issuer framework, it also incorporates an ACME client. cert-manager consists of three Kubernetes pods – a controller, a webhook, and a CA injector –, making extension for binding logic more complex. Its integration is specific to Kubernetes and less applicable to general-purpose HTTP server deployments.

Caddy [13] is a modular and extensible HTTP server written in Go. It has built-in modules with different implementations to obtain TLS certificates, one of them being an ACME v2 client. Since the ACME module is implemented directly in the web server, it has a very high integration. This means we should be able to effectively modify the HTTP header and add endpoints. Caddy's modularity should aid in extending the ACME client to control the certificate obtaining process.

4. Automated QWAC Binding Renewal

From the options we discussed, Caddy provides the most promising foundation for automating the QWAC binding renewal process due to its high extensibility and high integration of the ACME module with the web server. Thus, in the following, we analyze how QWAC binding renewal can be achieved in Caddy.

4.1. Architectural Overview of Caddy

Caddy is a statically linked Go binary, but follows a modular and extensible architecture by offering a plugin system. Architecturally, Caddy can be divided into three parts: [14]

- The *command* is Caddy's command line interface.
- Caddy's *core* reads the configuration file and is responsible for module orchestration.
- A set of *modules*, both built-in and by external developers. They contain most of Caddy's functionality and implement server features.

Modules define their own configuration options. They follow a well-defined lifecycle: The load phase loads the module into memory. The provision phase contains the module setup code. The use phase executes the module logic and the cleanup phase unloads the module. [14]

4.2. Implementation Approaches

In the following, we describe three strategies to hook into certificate issuance in order to implement automated binding renewal in Caddy.

Storage module. By implementing the interface `certmagic.Storage`, it is possible to detect file changes, including certificate file changes. Compared to a naive file polling approach, this method should be able to detect certificate changes with more precision. However, while precise in detecting changes, this method lacks context: we cannot immediately distinguish TLS certificate from other certificates, nor determine whether the event corresponds to issuance or renewal.

Issuer wrapping. `certmagic.Issuer` is an interface implemented by certificate issuers. By wrapping the existing `ACMEIssuer` and registering it as new `Issuer`, we can intercept calls to the `Issue()` function. This provides full access to the CSR and the issued certificate, enabling precise and controlled execution of binding logic. However, our second approach is not as general as our first approach since it is specific to the issuer we wrap. Additionally, this modification of the internal issuer pipeline risks compatibility issues with future versions of Caddy.

Event-based hook. Our third approach uses Caddy's built-in event module. Caddy emits a `cert_obtained` event when a TLS certificate is successfully obtained. We can register a listener for this event and trigger the binding logic. Since the event only contains the certificate path, we need to parse the raw certificate bytes, which adds additional complexity. However, we can reuse existing libraries for this task. This approach is non-intrusive and generic across issuers, while offering us the control over the issuance process we require.

4.3. Proof-of-Concept Implementation

To show the feasibility of automated binding renewal for 2-QWAC, we implement a proof-of-concept binding renewal for Caddy. We choose the event-based approach for its precision and robust compatibility. It comprises two Caddy modules.

Middleware module. We implement a middleware that modifies the `ServeHTTP()` function. It adds the `Link` header to HTTP responses pointing to a configurable endpoint. Additionally, it serves the corresponding configurable endpoint which returns a static dummy JWT.
Event listener module. The event handler subscribes to the `cert_obtained` event. Upon a newly created or issued TLS certificate, it resolves the certificate paths and parses the raw certificate bytes using Go's `crypto/x509` package. Then, it computes a SHA-256 thumbprint and prints it as proof that the certificate renewal has been detected. The event handler then generates an HMAC-SHA-256 signature using a static secret as a placeholder for the QWAC private key to simulate a digital binding operation.

4.4. Evaluation

We evaluate the implementation using Pebble, a lightweight and local ACME test server [15]. We configure our test setup with 15 seconds certificate lifetime in Pebble, ten seconds renewal interval in Caddy and disable

the challenge validation in Pebble for rapid testing. Then, using those configurations, we start a local Pebble server and a local Caddy server including our modules.

Using curl [16], we send a GET request to the Caddy server. We inspect the HTTP response and validate the presence and correctness of the `Link` header. We then verify that requests to the `Link` header URL return the static dummy JWT. Thus, we conclude that the middleware module is working correctly. Next, we observe the certificate renewals. Each certificate renewal triggers an event within the Caddy server once Caddy's polling has detected it. The event listener module creates a log entry including the certificate thumbprint and the HMAC-SHA-256 signature. In conclusion, the results confirm that the automation functions correctly under continuous rotation.

5. Conclusion and Future Work

The European Union's eIDAS 2.0 regulation has initially received criticism from both browser and academia, partly due to the introduction of QWACs [4], [5]. A recent change to the QWAC specification introduces 2-QWAC, an approach which does not require the addition of EU-mandated certificate authorities to global root trust stores [3]. 2-QWAC offers a viable compromise between verified organizational identity and browser trust autonomy.

However, the 2-QWAC approach introduces more complexity; by separating identity verification and TLS handshake, a binding between QWAC and TLS certificate is necessary. Manual management of bindings would not be feasible for websites with frequent certificate rotation.

It is our belief that the adoption of QWACs depends on the ability to integrate with modern, automated TLS workflows. Our work demonstrates that automation of 2-QWAC binding generation and renewal is both feasible and practical. By leveraging Caddy's modular architecture and integrated ACME client, we implemented a proof-of-concept that extends the ACME flow and is able to dynamically serve bindings. We aim to make our implementation fully ETSI-compliant by implementing JAdES binding generation.

For the adoption of QWACs to succeed, however, implementations for other popular ACME clients like Certbot are necessary. We identified multiple approaches to automate the 2-QWAC binding, which future work could generalize for other ACME clients. Furthermore, we discussed characteristics of other popular ACME clients hindering their extension with QWAC binding logic; future work should find ways to address these challenges.

A comprehensive ACME client survey is left for future work. While there exist lists of ACME implementations [17], [18], a comprehensive overview and feature comparison of ACME clients, evaluating support for hooks, plugin APIs, and web server integration depth, has – to our knowledge – not yet been published.

QWACs themselves offer opportunities for further research, as well. Since the current specification is still very recent, it has not yet been implemented in major web browsers. How the parallel specification of two QWAC variants, 1-QWAC and 2-QWAC, will affect adoption and whether website operators will prefer one of the approaches is an interesting question that requires further observation.

As QWACs continue to evolve, automation will be one key factor to their adoption. We believe our results provide a foundational step toward fully integrated, secure, and scalable QWAC usage in real-world environments.

References

[1] European Union, "Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC," Oct. 2024.

[2] Chromium Developers, "EV UI Moving to Page Info," Available: https://chromium.googlesource.com/chromium/src/+/HEAD/docs/security/ev-to-page-info.md, [Accessed: Jun. 22, 2025].

[3] ETSI, "Etsi ts 119 411-5, v2.1.1," Available: https://www.etsi.org/deliver/etsi_ts/119400_119499/11941105/02.01.01_60/ts_11941105v020101p.pdf, Feb. 2025, [Accessed: May 1, 2025].

[4] Mozilla, "November 2021 position paper on the European Commission's legislative proposal to revise the eIDAS Regulation," Available: https://blog.mozilla.org/netpolicy/files/2021/11/eIDAS-Position-paper-Mozilla-.pdf, Nov. 2021, [Accessed: Jun. 10, 2025].

[5] "Global website security ecosystem at risk from EU Digital Identity framework's new website authentication provisions," Available: https://www.eff.org/files/2022/03/02/eidas_cybersecurity_community_open_letter_1_1.pdf, Mar. 2022, [Accessed: Jun. 10, 2025].

[6] Chromium Developers, "Implement support for QWACs," Available: https://issuetracker.google.com/issues/392931065, 2025, [Accessed: Jun. 19, 2025].

[7] M. Akram, W. Barker, R. Clatterbuck, D. Dodson, B. Everhart, J. Gilbert, W. Haag, B. Johnson, A. Kapasouris, D. Lam, B. Pleasant, M. Raguso, M. Souppaya, S. Symington, P. Turner, and C. Wilson, "Securing Web Transactions: TLS Server Certificate Management," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication (SP) 1800-16, Jun. 2020.

[8] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten, "Automatic Certificate Management Environment (ACME)," Internet Engineering Task Force, Request for Comments RFC 8555, Mar. 2019.

[9] European Union, "Directive (EU) 2022/2555 of the European Parliament and of the Council of 14 December 2022 on measures for a high common level of cybersecurity across the Union, amending Regulation (EU) No 910/2014 and Directive (EU) 2018/1972, and repealing Directive (EU) 2016/1148 (NIS 2 Directive)," Dec. 2022.

[10] Certbot Developers, "Certbot," Available: https://certbot.eff.org/, [Accessed: May 26, 2025].

[11] ——, "User Guide," Available: https://eff-certbot.readthedocs.io/en/stable/using.html, 2018, [Accessed: Jun. 15, 2025].

[12] cert-manager Developers, "Cert-manager," Available: https://cert-manager.io/, 2025, [Accessed: May 26, 2025].

[13] ZeroSSL, "Caddy," Available: https://caddyserver.com/, 2025, [Accessed: May 26, 2025].

[14] ——, "Architecture," Available: https://caddyserver.com/docs/architecture, 2025, [Accessed: May 21, 2025].

[15] Pebble Developers, "Letsencrypt/pebble," Available: https://github.com/letsencrypt/pebble, Jun. 2025, [Accessed: Jun. 9, 2025].

[16] "Curl," Available: https://curl.se/, [Accessed: Jun. 9, 2025].

[17] Developers of Certify The Web, "ACME Clients," Available: https://acmeclients.com/, [Accessed: May 26, 2025].

[18] Let's Encrypt (ISRG), "ACME Client Implementations," Available: https://letsencrypt.org/docs/client-options/, Feb. 2025, [Accessed: May 21, 2025].

Market Models in the European Digital Identity Wallet Ecosystem

Timm Bauer, Stefan Genchev*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: timm.v.bauer@tum.de, genchev@net.in.tum.de

***Abstract*—The European Digital Identity (EUDI) Wallet creates new opportunities for issuing verifiable digital diplomas as Qualified Electronic Attestations of Attributes (QEAA). This paper explores and compares market models for QEAA-based diploma issuance and verification. The models under consideration include university-operated Qualified Trust Service Providers (QTSPs), outsourcing to external providers, government-supported schemes, pay-per-verification, and industry-sponsored approaches. By examining stakeholder incentives and dependencies, this paper highlights benefits and limitations per model. The analysis reveals that there is no dominant market model, as the optimal strategy depends on the use case and stakeholders. Hybrid approaches can help adapt and apply models to specific use cases, and an adoption phase may help gaining trust and experience at a lower initial investment.**

***Index Terms*—EUDI wallet, QEAA, eIDAS 2.0, verifiable credentials, digital diplomas, market models**

1. Introduction

The European Digital Identity (EUDI) Wallet, introduced in the eIDAS 2.0 framework [1], is changing how digital credentials are issued, stored, and verified. One key application is the issuance and verification of Qualified Electronic Attestations of Attributes (QEAA) [2], such as higher education diplomas. This process involves multiple stakeholders, each with distinct roles and incentives.

Although the wallet infrastructure is evolving, there is a lack of analysis regarding sustainable market models, particularly for the use case of digital university diplomas. Actors here are universities, graduates, Qualified Trust Service Providers (QTSPs), and employers [3], and their interaction affects the sustainability and accessibility of the EUDI wallet. This paper examines market models that could govern the issuance and verification of QEAA-based diplomas within the EUDI wallet ecosystem.

Section 2 provides an use-case-independent overview of the legal and technical foundations. Section 3 delves into the use case, by describing the problem and examining stakeholders, their roles, and incentives. By analyzing incentives and funding mechanisms, Section 4 explores applicable models, considering university-, government-, and verifier-centric approaches. Using the QEAA-based diploma use case as a representative scenario, this paper aims to identify which market models can sustainably support the issuance and verification of QEAA, while balancing incentives, accessibility, and financial obligations.

2. Background and Regulatory Framework

Regulation (EU) No. 910/2014 [4], known as eIDAS, established a legal framework for electronic identification, authentication, and trust services across the EU. Aimed at enabling cross-border interoperability of services such as eID, signatures, and seals, it faced limited adoption due to restricted user control and lack of support for attribute issuance [5], [6]. Additionally, its inflexibility with respect to supporting diverse use cases caused it to be "unable to respond to new market demands for Identity Management" [7, p. 439]. These issues led to its 2021 revision and introduction of eIDAS 2.0 [1], which mandates that Member States provide citizens an EUDI Wallet [8].

2.1. European Digital Identity Wallet Ecosystem

The EUDI Wallet is a standardized and user-centric digital wallet proposed by the European Union. Its main objective is enabling citizens and organizations to store, manage, and verify digital certified attributes [9]. This wallet is related to the concept of *Self-Sovereign Identity* (SSI), a model where users fully control their identity data and can selectively share verified credentials issued by trusted entities [10]. In the EUDI ecosystem, Trust Service Providers (TSPs) are trusted entities that issue Personal Identification Data (PID), a person's core identity attributes like name and date of birth. Comparable to the SSI approach, the ecosystem is based on a trust triangle betweet its participants, with the roles depicted in table 1:

TABLE 1: Digital Identity Ecosystem Participants [9]

Participant EUDI	SSI Equivalent	Description
PID Provider	Issuer	Issues identity attributes
End User	Holder	Wallet user who stores and presents credentials
Relying Party	Verifier	Validates credentials

In this relationship, TSPs issue verified credentials to users, who share them with a relying party to prove their identity or attributes [10]. Wallet users control what information they present to the relying party for verification. This concept is referred to as *Selective Disclosure* [9].

2.2. Electronic Attestations of Attributes

In the EUDI Wallet Ecosystem, electronic attributes refer to verified pieces of information, belonging to a person or organization, that can be stored, shared, and

 doi: 10.2313/NET-2025-11-3_11

verified digitally. The attestation of attributes, such as certificates, licenses, and professional or educational qualifications can be issued by TSPs or public authorities [2]. Based on the required level of assurance, a distinction is made between different types of attribute attestations, affecting their issuance and legal recognition [11]:

- **Qualified Electronic Attestations of Attributes** (QEAA) are issued by QTSPs which must be accredited according to the eIDAS 2.0. Examples of QEAA include education diplomas, and their legal effect is equivalent to paper documents [12].
- **Public Electronic Attestations of Attributes** (Pub-EAA) are attributes originating from official government records, for example driver's licenses [11]. Issuers are public authorities, and the legal effect is also equivalent to paper documents.
- **Electronic Attestations of Attributes** (EAA), also called non-qualified EAAs, do not have to be issued by a QTSP as they do not require the same level of assurance. Issuers can be non-qualified TSPs, but the legal effect is not equivalent to paper documents. Examples of non-qualified EAAs include gym membership attestations [12].

2.3. Monetization Strategies

One of the core requirements outlined in the eIDAS 2.0 regulation is that the EUDI Wallet should be provided to citizens free of charge [9]. Additional costs may arise for the other ecosystem participants based on the services they provide, particularly for QTSPs due to strict qualification requirements and regular conformity assessment [8], [13]. Developing sustainable business models and monetization strategies for attribute attestations is essential for the successful implementation of a self-sustaining wallet ecosystem [14]. Castaldo et al. [14] identify three different pricing models, given that the system architecture incorporates fitting verification and attribute management methods to support the corresponding model:

- **Issuance-based**: In a *pay-per-issuance* fashion, the user covers the cost of receiving a verified attribute [15].
- **Verification-based**: Contrary to the above model, the relying party would be required to *pay-per-use* upon verifying a presented credential [15].
- **Free of charge**: Neither the wallet user nor the verifier have to pay. To ensure sustainability, this would most probably require external funding such as government subsidies [13].

With the foundational principles established, this paper now focusses on the issuance and use of digital university diplomas as a representative use case of QEAA.

3. Use Case: QEAA for University Diplomas

This section showcases how the abstract technical components and ecosystem participants translate into real-world implementations, namely digital university diplomas. Graduates typically need to prove their qualifications for employment or further study. Paper-based verification is slow, error-prone, and susceptible to fraud [13], [16], with additional challenges in cross-border recognition [17]. The EUDI Wallet provides an opportunity for an interoperable, and user-oriented verification of educational qualifications [18]. Diplomas as QEAA are legally recognized, cross-border operational [12], and can be seamlessly integrated into digital workflows, e.g. supported by the EU's Single Digital Gateway, which aims to streamline access to public services across Member States [19]. In a sustainable ecosystem, participants require financial compensation for provided services. This paper therefore explores market models for issuing digital diplomas in the EUDI ecosystem.

3.1. Stakeholders and Roles

The stakeholders are the degree-awarding institution, a QTSP, the graduate, and the employer or higher education institution the graduate is applying to [3], [18]. The university acts as the issuer, but might partner with an external TSP instead of fulfilling that role exclusively itself. [3]. The graduate acts as the wallet user and the employer as the relying party.

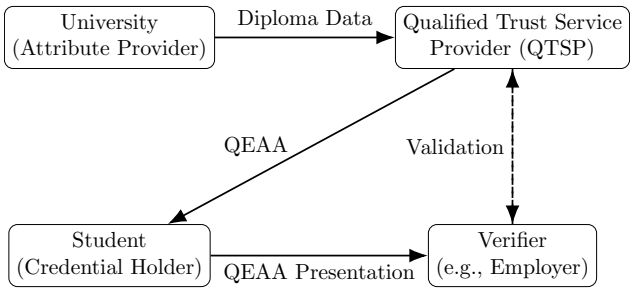


Figure 1: Attribute Issuance Workflow

Figure 1 shows the workflow for issuing and verifiying QEAA diplomas. Solid arrows indicate attribute data flow, and the dotted arrows the verification process. The university transfers the diploma information to the QTSP via a secure channel. Upon receiving the information and identifying the student, the QTSP issues a QEAA for the diploma to the student's wallet. Within an application process, the student presents these credentials to the employer, who verifies them via the QTSP's signature [3].

3.2. Incentive Structure and Value Gain

To identify applicable market models for the presented use case, it is necessary to understand the stakeholders' incentives, especially their potential financial benefits. This is important when considering the central question of who is willing to pay for the attribute attestation.

- Issuing universities may benefit from improved process efficiency, enhanced reputation, and compliance with EU digital standards [3].
- Graduates gain portable, verifiable credentials that improve cross-border access to jobs and education [18].
- Employers can reduce hiring costs and impede fraud through digital verification [16].
- QTSPs directly monetize their services through their attestation business models [14].

 doi: 10.2313/NET-2025-11-3_11

3.3. Related Work

Vaziri et al. [3] analyze the use case of universities issuing digital diplomas, considering the regulatory and operational challenges under the eIDAS 2.0 regulation. A comparison of the approaches of diplomas as QEAA as opposed to qualified sealed documents, reveals that "QEAA offers superior interoperability and holder binding, [but] it faces regulatory uncertainty and implementation complexities" [3, p. 183]. Vaziri et al. explore scenarios where the university becomes a QTSP, outsourcing to an external QTSP as shown in Figure 1, university alliances, and a "bring-your-own-QTSP" [3, p. 189] model. Vaziri et al. [3] and Seegebarth et al. [2] also explore the possibility of credentials electronically signed as qualified electronic seals (QSeals) according to eIDAS 1.0, to ease adoption until QEAA are fully applicable in the market.

4. Market Models for QEAA-Based Diploma

Having established the operational dynamics of digital diplomas in the EUDI ecosystem, we now focus on the market models that could emerge to support, govern, and monetize such use cases. Understanding the value exchange mechanisms and the economic and social implications for each market model is important for assessing the benefits and limitations for stakeholders. Implementing QEAA-based diplomas requires upfront investment and ongoing compensation. Strong incentives for stakeholders, in the form of substantial value gains, are necessary to promote stakeholders' willingness to pay. Network effects also play a role, as a full transition from paper-based to QEAA diplomas depends on widespread issuance and employer adoption of digital application processes. While graduates do benefit, as outlined in Section 3.2, their benefits are not financial and their willingness to pay is low, as a consequence [16]. Additionally, as the EUDI Wallet is intended to be free for users [13], market models must focus on other stakeholders.

This section presents five models, based on either the university, the government, or the verifier bearing the primary cost. As the academic literature regarding formalized market models for QEAA is scarce, these models are mainly adopted from SSI-related monetization strategies, or transferred from related trust services.

4.1. University-Centric

Similarly as proposed by Vaziri et al. [3], the university-centric approach considers both the possibility of the university becoming a QTSP and enlisting an external QTSP.

4.1.1. Internal QTSP. In this market model, the university itself becomes a QTSP and directly issues QEAA for diplomas. This model provides a high level of trust, since the university acts as both the authentic source of the academic credential and is responsible for its authenticity [3]. For verifiers, this direct issuance simplifies trust relationships as there is no dependency towards external issuers. The university's motivations might include the ability to exercise full control over credential issuance and remaining the technical authority for authenticity, which is also visible in the attribute attestation [3]. This approach could also help avoid vendor lock-in by reducing reliance on third-party infrastructure.

To gain and maintain the qualification status, universities must meet strict requirements and regularly face conformity assessments. Vaziri et al. [3] conclude that the regulatory and financial requirements are too high, especially for smaller universities. University alliances, such as EuroTeQ [20], could reduce the operational cost per university. This approach could be considered a hybrid version of the internal and external QTSP model.

4.1.2. External QTSP. In the external QTSP model, universities enlist an already accredited QTSP. The university maintains its role as the authentic source, while delegating the technical processes of credential creation, signing, and lifecycle management to the external QTSP [3]. The primary benefit in comparison to the internal approach is that universities can implement eIDAS-compliant credentials without substantial upfront investments. For verifiers, trust in the credential depends on both the external QTSP, and the university that is referenced in the QEAA as the attribute source [3]. However, it introduces a long-term dependence on the external service provider.

The university's financial gain depends on the price for issuing a QEAA-diploma compared to a paper-diploma, and whether the paper-based process can be replaced entirely. As of now, there is no standard price for QEAA in the European market. Since both services are regulated trust services under eIDAS, the pricing model might be similar to that of remote qualified electronic seals, allowing for an estimate. Sign8 is a German QTSP, that charges approximately 2.50 EUR per signature, within a volume-based pricing model [21]. In the year 2023, the Technical University of Munich (TUM) had 9 541 graduates [22]. Assuming no more than 12 000 graduates per year, TUM would be charged up to 30 000 EUR per year by Sign8.

4.2. Government-Centric

In the use case of QEAA diplomas, the government was not considered a stakeholder so far, despite its significant role in the EUDI ecosystem. It plays a crucial part either by directly providing the wallet infrastructure or by enabling private providers to do so in a competitive market [13]. This role was neglected in the previous consideration because the infrastructure provider does not significantly influence the presented market models. However, given the government's involvement, government-supported market models will also be considered.

External funding via direct subsidies could reduce the financial barrier for the issuance of digital diplomas by making the outsourcing model more financially sustainable, especially for smaller universities. Alternatively, state-operated TSPs could be established, issuing the diplomas on behalf of universities free of charge, comparable to the issuance of PuB-EAA [11]. At the same time, this could introduce bureaucratic burdens, slow down innovation cycles, and may raise concerns about institutional autonomy. Nonetheless, government-supported issuance could enhance accessibility to QEAA diplomas for all universities and support financial sustainability, if the ecosystem is not self-sustaining.

4.3. Verifier-Centric

Long-term self-sustainability in the EUDI Wallet ecosystem can also be achieved if the verifier contributes to the financial compensation. As verifiers, such as employers, benefit from improved credential verification processes, making manual verification obsolete and preventing fraud, they have an incentive to invest [13], [16]. Similarly as for the university-centric models, the willingness to pay depends on the opportunity cost compared to current paper-based verification processes.

4.3.1. Pay-Per-Verification. The Pay-Per-Verification model is based on the verification-based monetization strategy on the side of the QTSP. This model is transferred from market models for digital identification services, such as know-your-customer processes, in which the relying party pays an identity provider per user identification [23]. In this analogy, the QTSP corresponds to the identity provider and the employer to the relying party.

Universities collaborate with external QTSPs, which employ a verification service that charges the verifier to establish a usage-based monetization [14]. Cost for universities and students would be eliminated and transferred to the employer. However, if different universities use different QTSPs without a unified verification interface, this could introduce process inefficiencies as employers have to adapt to the corresponding interfaces.

Additionally, privacy concers arise from tracking verification requests for billing purposes. This tracking could potentially lead to exposure of information regarding the verification of credentials by specific parties. Depending on the implementation, service providers could potentially link users and credentials to verification requests, and ultimately to the verifier [14], [16]. However, Castaldo et al. [14] suggest a verification approach that would maintain user anonymity and eIDAS compliance, even considering a verification-based monetization strategy.

4.3.2. Sponsored Attestation. Academic credentials are typically static, therefore they would be issued once and potentially verified multiple times [13]. The pay-per-verification model has the potential to establish long-term sustainability and support ongoing maintenance of the trust infrastructure. However, frequent verification fees might discourage adoption from the employer's side.

In the Sponsored Attestation model, companies that directly benefit from the academic training of students cover the costs of issuing eIDAS-compliant digital diplomas. This model is particularly interesting when empirical evidence indicates that a significant proportion of university graduates are likely to apply to a particular employer. Examples of this use case could be dual study programs or university-industry partnerships, such as those between TUM and SAP [24].

Similarly to government funding, universities would benefit due to the reduced financial demand while strengthening institutional ties with partners. However, this approach could potentially result in unequal access, where students of universities without industry partners might lack digital academic credentials. This could lead to a heterogeneous credential ecosystem and negative network effects.

Nonetheless, this model could support the adoption of QEAA and a self-sustainable ecosystem by aligning financial responsibility with financial benefits for the employer. These employers could also benefit from this model if it is implemented in the early stages when digital credentials are not yet established as market standard. This would give them early access to the benefits of digitally verifiable academic credentials.

4.4. Adoption Phase

The implementation of verifiable credentials in business processes is subject to network effects [13]. Integrating QEAA into processes requires substantial investments in infrastructure and system integration, while the return on investment depends on the usage rate. This effect could hinder the use of QEAA-diplomas in the early stages of EUDI ecosystem development, as the potential return on investment may seem too low.

Seegebarth et al. [2] suggest splitting the transition towards QEAA into two phases, whereas the first phase represents an adoption phase. During this phase, verifiable credentials would be implemented as eIDAS compliant qualified sealed documents [2], [3]. The infrastructure for this approach is already implemented and stakeholders could gain trust and experience with the technology at a lower cost, before QEAA will be fully utilized in the second phase. Ultimately, QEAA are superior to the QSealed document approach due to higher interoperability, seamless process integration, and selective disclosure features [3].

5. Conclusion

As eIDAS 2.0 facilitated the widespread adoption of the EUDI wallet and QEAA-based credentials, the question of how market models will shape the issuance and verification of digital diplomas becomes increasingly critical. The models explored in this paper differ in funding, monetization strategy, and stakeholder dependencies and obligations, based on their respective incentive structures.

The internal QTSP approach provides the university with maximum institutional control, but it requires the university to meet strict conformity requirements, which introduces high costs, making it not viable especially considering smaller universities. Outsourcing to external QTSPs offers better operational efficiency but introduces dependencies on third parties.

Government-supported models would improve accessibility independent of the university's size and budget. Public funding may also support adoption, as network effects limit stakeholders' willingness to invest during early stages with low returns. Both direct subsidies and central public services could support equity and adoption but might reduce flexibility and institutional autonomy.

The pay-per-verification and sponsored attestation approaches would directly link costs to verifiers' benefits but could introduce privacy risks or unequal access. The willingness to pay for both the issuing universities and employers depends on the transition cost and operational cost compared to paper-based diplomas.

 doi: 10.2313/NET-2025-11-3_11

Further work could include comparative analyses regarding the cost and effort between paper-based and digital credential issuance and verification. Hybrid approaches could also be applied since it is unlikely that one single model will universally fit all educational contexts. Ultimately, the success and sustainability of QEAA-diplomas will depend on balancing financial sustainability, privacy, and institutional autonomy.

References

[1] European Parliament and Council of the European Union, "Regulation (eu) no 1183/2024 amending regulation (eu) no 910/2014 to establish the european digital identity framework," https://eur-lex.europa.eu/eli/reg/2024/1183/oj, 2024, [Online; accessed 07-June-2025].

[2] C. Seegebarth, P. Bastian, and M. Kraus, "Enabling attribute attestations: Road from verifiable credential to qeaa," *Datenschutz und Datensicherheit-DuD*, vol. 48, no. 4, pp. 237–240, 2024.

[3] A. Vaziry, L. Vetter, and A. Küpper, "eidas 2.0: Evaluating the issuance of digital university diplomas," in *Open Identity Summit 2025*. Gesellschaft für Informatik eV, 2025, pp. 183–190.

[4] European Parliament and Council of the European Union, "Regulation (eu) no 910/2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec," https://eur-lex.europa.eu/eli/reg/2014/910/oj, 2014, [Online; accessed 07-June-2025].

[5] European Commission, "Study to support the impact assessment for the revision of the eidas regulation - final report," https://digital-strategy.ec.europa.eu/en/library/study-support-impact-assessment-revision-eidas-regulation, 2021, [Online; accessed 07-June-2025].

[6] C. Busch, *eIDAS 2.0: Digital Identity Services in the Platform Economy*. Centre on Regulation in Europe, 2022.

[7] J. Inza, "The european digital identity wallet as defined in the eidas 2 regulation," in *Governance and Control of Data and Digital Economy in the European Single Market: Legal Framework for New Digital Assets, Identities and Data Spaces*. Springer Nature Switzerland Cham, 2025, pp. 433–452.

[8] S. Schwalm, "The possible impact s of the eidas 2.0 digital identity approach in germany and europe," in *Open Identity Summit 2023*. Gesellschaft für Informatik eV, 2023, pp. 109–120.

[9] N. Urbach, T. Guggenberger, H. Pfaff, J.-C. Stoetzer, S. Feulner, M. Babel, M. Principato, and J. Lautenschlager, "Eu digital identity wallet - anwendungsfälle, nutzungspotenziale und herausforderungen für unternehmen," Projektgruppe Wirtschaftsinformatik des Fraunhofer-Institut für Angewandte Informationstechnik FIT, Bayreuth, 2024.

[10] J. Strüker, N. Urbach, T. Guggenberger, J. Lautenschlager, N. Ruhland, V. Schlatt, J. Sedlmeir, J.-C. Stoetz, and F. Völter, "Self-sovereign identity - grundlagen, anwendungen und potentiale portabler digitaler identitäten," Projektgruppe Wirtschaftsinformatik des Fraunhofer-Institut für Angewandte Informationstechnik FIT, Bayreuth, 2021.

[11] Potential, "What are the 3 types of electronic attestations of attributes (eaa)?" Apr 2025, [Online; accessed 07-June-2025]. [Online]. Available: https://www.digital-identity-wallet.eu/news/what-are-the-3-types-of-electronic-attestations-of-attributes-eaa/

[12] European Commission, "Eu digital identity wallets for issuers," [Online; accessed 07-June-2025]. [Online]. Available: https://ec.europa.eu/digital-building-blocks/sites/display/EUDIGITALIDENTITYWALLET/Wallet+for+Issuers

[13] K. Degen and T. Teubner, "Wallet wars or digital public infrastructure? orchestrating a digital identity data ecosystem from a government perspective," *Electronic Markets*, vol. 34, no. 1, p. 50, 2024.

[14] L. Castaldo, G. Cortese, S. Izzo, and F. Balsamo, "Electronic attestation of attributes extended validation services," *TDI 2025: 3rd International Workshop on Trends in Digital Identity*, 2025.

[15] M. Panfilio, "Possible architectures of digital european wallets: national certifications and the roles of key stakeholders," Namirial, Feb 2025, [Online; accessed 07-June-2025]. [Online]. Available: https://www.namirial.com/en/inspiration/possible-architectures-of-digital-european-wallets/

[16] M. Kubach and H. Roßnagel, "Economically viable identity ecosystems: Value capture and market strategies," in *Open Identity Summit 2024*. Gesellschaft für Informatik eV, 2024, pp. 27–38.

[17] Your Europe, "Recognition of academic diplomas," [Online; accessed 09-June-2025]. [Online]. Available: https://europa.eu/youreurope/citizens/education/university/recognition/index_en.htm

[18] P. Herbke and H. Yildiz, "Elmo2eds: transforming educational credentials into self-sovereign identity paradigm," in *2022 20th International Conference on Information Technology Based Higher Education and Training (ITHET)*. IEEE, 2022, pp. 1–7.

[19] Bundesministerium für Digitales und Staatsmodernisierung, "Die single digital gateway-verordnung (sdg)," [Online; accessed 23-August-2025]. [Online]. Available: https://www.digitale-verwaltung.de/Webs/DV/DE/onlinezugangsgesetz/info-sdg/info-sdg-node.html

[20] EuroTeQ, "About euroteq - how six universities engineer the future," [Online; accessed 20-June-2025]. [Online]. Available: https://euroteq.eurotech-universities.eu/about-us/

[21] SIGN8, "Volumenbasierte modelle - digitale unterschriften sign8," [Online; accessed 10-June-2025]. [Online]. Available: https://sign8.eu/volumebased/

[22] TU München, "Tum in zahlen 2023," [Online; accessed 10-June-2025]. [Online]. Available: https://mediatum.ub.tum.de/doc/1774468/1774468.pdf

[23] V. Schlatt, J. Sedlmeir, S. Feulner, and N. Urbach, "Designing a framework for digital kyc processes built on blockchain-based self-sovereign identity," *Information & Management*, vol. 59, no. 7, p. 103553, 2022.

[24] TU München, "Sap@tum collaboration lab," [Online; accessed 20-June-2025]. [Online]. Available: https://www.ioc.tum.de/sap-colab/startseite/

MASQUE-based Performance Enhancing Proxies

Patrick Bokelmann, Daniel Petri*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: patrick.bokelmann@tum.de, petriroc@net.in.tum.de

***Abstract*—Transport-layer protocols like TCP were originally designed to operate end-to-end, providing reliable and congestion-controlled transport. However, with the growth of the global Internet and the integration of wireless links, this characteristic has been suppressed by Performance Enhancing Proxies (PEPs) in an effort to boost transmission performance. Yet, the widespread use of these middleboxes has hindered the advancement of the TCP protocol by effectively requiring network traffic to conform to their standards. Novel transport protocols like QUIC combat this ossification with extensive header encryption, forcing the discontinuation of these transparent middleboxes. As the demand for these performance enhancements persists, this paper proposes the implementation of PEPs for QUIC connections using successive Multiplexed Application Substrate over QUIC Encryption (MASQUE) tunnels, allowing the tunneled connection to disable its congestion control mechanisms.**

***Index Terms*—transport protocols, performance-enhancing proxies, middleboxes, MASQUE, QUIC**

1. Introduction

In previous decades, the global network has departed from the transport layer's end-to-end design in favor of in-network performance enhancement functions. Operating on multiple network layers, Performance Enhancing Proxies (PEPs) have transparently altered and optimized network traffic. For instance, at the transport layer, middleboxes can filter or space out TCP acknowledgements, retransmit packets that were lost on a path segment, send localized acknowledgements, or terminate TCP connections and insert themselves as a man-in-the-middle [1]. Other use-cases include the implementation of tunneling, compression, or prioritized connection multiplexing.

These practices are particularly beneficial in wireless networks. For example, (geostationary) satellite links suffer from long round-trip times (RTTs) due to their distance from the Earth, which in turn cause long feedback loops. Congestion events on the network path after the satellite link may negatively affect the sending behaviour on the satellite link, effectively underutilizing the link [1]. In contrast, W-LAN links are prone to latency variations or packet losses caused by layer-two retransmits and handovers. By caching TCP segments, using local retransmit timers, suppressing duplicate acknowledgements, or outright TCP connection splitting, PEPs may isolate the wireless and wired path segment.

However, the rise of widespread encryption has increasingly hindered the operation of traditional PEPs. Novel transport layer protocols like QUIC authenticate the endpoint of a connection, preventing or impeding the impersonation of endpoints. With transparent middleboxes' reliance on specific protocol definitions, which significantly hinders the evolution of new protocol iterations, most of QUIC's header fields are encrypted. While this is effective in combating protocol ossification, it further restricts the deployment of transparent PEPs. Nevertheless, with a congestion control scheme similar to TCP, the protocol is still susceptible to the same issues mitigated by PEPs.

Necessitated by the departure from these transparent enhancements, there seems to be a general consensus that connection endpoints should select network functions and provide them with the necessary information that is otherwise protected through encryption.

This paper explores a novel approach to realizing similar features for QUIC connections by encapsulating packets using MASQUE, an HTTP/3 proxy. Our design proposes a sequence of MASQUE tunnels to provide reliable transport with split congestion control loops, which allows the congestion control algorithm of the inner connection to be disabled. The remainder of this work is organized as follows: Section 2 outlines the mechanisms of the QUIC protocol and provides an overview of the MASQUE proxying. Section 3 discusses related work. We then detail our proposed approach in Section 4. Finally, we conclude the paper and identify points for future work in Section 5.

2. Background

The *QUIC* protocol [2] is a novel transport protocol built as a successor to TCP and used as the underlying transport protocol for HTTP/3. Built on top of UDP, it departs from port numbers and IP addresses to connection IDs to identify a connection. These allow the connection to migrate across different network paths. Furthermore, it integrates TLS to provide a confidential and authenticated connection. This not only results in a reduced connection establishment duration in comparison to TLS on TCP, but also directly authenticates the server at the transport layer, ensuring an end-to-end connection. Additionally, it introduced two different header types: one for packets used to establish a connection, needing more header fields during that phase. The second header type is used in 1-RTT packets after the connection has been established, which reduces the overhead of unused header fields. As

 doi: 10.2313/NET-2025-11-3_12

explained in Section 1, most header fields in a QUIC packet are encrypted in an effort to combat protocol ossification. During the connection establishment process, QUIC enables the use of 0-RTT packets to transmit application data from a previous connection using an already existing cryptographic context. While the encryption of 0-RTT packets does not guarantee perfect forward secrecy, this mechanism may be exploited to accelerate the establishment of tunnelled end-to-end connections.

The contents of a QUIC packet are further organised into different frames, which are used to carry control information and application data. These frames allow QUIC to support multiplexing multiple data streams, eliminating head-of-line blocking issues commonly found in TCP. Streams are used to transmit data reliably and are subject to congestion and flow control. In addition to reliable transport, the datagram extension [3] introduces support for unreliable payloads contained in datagram frames. While subject to congestion control, they do not have to be retransmitted in case of a packet loss and are not subject to flow control limitations.

MASQUE is a group of protocols that define the tunneling of UDP, IP, or Ethernet packets over an HTTP/3 connection. The *CONNECT-UDP* method [4] allows the proxying of UDP packets inside an HTTP connection. The client connects to a MASQUE proxy and issues a CONNECT-UDP upgrade request, providing the server's IP address and port number. Packets are then encapsulated in HTTP datagrams, which directly translate to QUIC datagrams [5], and sent through the QUIC connection to the proxy. A packet the proxy receives is then decapsulated and forwarded to the server. UDP packets with a QUIC packet payload can be sent in a MASQUE connection, tunneling the end-to-end connection, as shown in Figure 1.

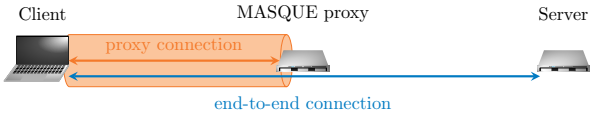


Figure 1: MASQUE Proxy [6]

Should the underlying transport protocol not support unreliable data transmission, HTTP datagrams can instead be enclosed in HTTP Capsules [5], which are then transmitted reliably in a QUIC stream. This mechanism also enables support for legacy protocols like TCP, thus maintaining backward compatibility with previous HTTP versions such as HTTP/1.1 or HTTP/2. HTTP Capsules can be re-encoded into HTTP Datagrams in transit. However, this is discouraged in [5] to avoid disturbances in the packet flow. While MASQUE was originally conceived as a measure to improve user privacy, Krämer et al. [7] voiced a proposal for usage in a performance-enhancing context, using reliable transmissions to counteract the effects of an unreliable local link.

3. Related Work

Kosek et al. [8] utilized QUIC's connection migration capabilities in combination with reworked cryptographic mechanisms to split QUIC connections at a proxy while maintaining an encrypted payload. The authors derive additional keying material during the handshake to introduce an additional layer of encryption for the application data beyond the built-in QUIC transport layer encryption. This enables the entire connection state, except for the additionally derived keys, to be shared with a middlebox while maintaining the application data's end-to-end cryptographic protections. To split an existing connection at a so-called SMAQ middlebox, the client opens an additional QUIC connection to the proxy, instructs it to act as a SMAQ proxy, and shares the connection state. The proxy then duplicates the connection state to gain a separate state for each endpoint. Next, it sends a PING to both the client and server using the new connection states, thereby initiating a connection migration. Posing as the other endpoint for both the server and client, the proxy is thus able to split the connection and insert itself in the middle while forwarding all traffic between the endpoints. Before sending application data, the endpoints encrypt it using the additional keying material. This largely maintains end-to-end privacy, integrity, and authenticity, despite sharing the QUIC connection keys with a third party. This design notably resembles TCP connection splitting with an underlying TLS connection.

Yuan et al. [9] employ acknowledgements sent by a proxy to simulate the behaviour of split congestion control loops at the client. The proxy computes packet identifiers of encrypted packets, which it then bundles into acknowledgements sent to a connection endpoint. Leveraging these early acknowledgements provided by the middlebox, the packet's sender can determine the path segments the packet has successfully traversed and adjust its congestion control algorithm accordingly. The authors observe that the sum of the congestion windows of split TCP connections using the QUBIC congestion control algorithm should equal the congestion window of a single, end-to-end congestion control loop. If a loss occurs on a given path segment in the split scenario, only the congestion window of that path would be decreased. Thus, the sum only decreases proportionally to the path segment's share of the congestion window. To emulate this behaviour in an end-to-end congestion control loop with access to out-of-band acknowledgements over a path segment, the congestion window is decreased proportionally to the path segment's share of the overall path.

4. Design

To achieve split congestion control loops, we establish a sequence of MASQUE tunnels for the QUIC connection over the entirety of the network path, as depicted in Figure 2. Using HTTP capsules enables endpoints to delegate the responsibility for end-to-end congestion and flow control to the proxy connections on each path segment. As each packet is tunneled using streams, it is transmitted reliably to the next proxy. Furthermore, the path segment to the (next) proxy is congestion-controlled by the connection to the proxy, effectively realizing split congestion

 doi: 10.2313/NET-2025-11-3_12

control loops. Note that this design relies on the server either being able to accept a MASQUE connection or to be able to receive incoming connections through a MASQUE tunnel as proposed in [10]. Furthermore, middleboxes must not convert HTTP capsules into QUIC datagrams, as this would break end-to-end connection reliability. While endpoints can disable their inner congestion control mechanisms, they should still adhere to the flow control limitations. Furthermore, acknowledgements should still be processed in case a packet loss occurs at the server in transit from the MASQUE tunnel connection to the end-to-end connection. The frequency of the acknowledgements, and the retransmit timers can be set higher than usual, and acknowledgements can be bundled together.

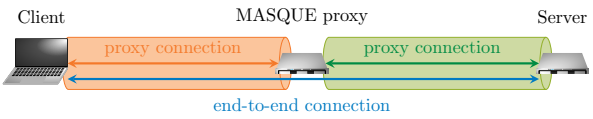


Figure 2: End-to-end tunnel with chained MASQUE connections

4.1. Connection Establishment

The connection establishment procedure is depicted in Figure 3. We assume that no previous cryptographic context is available from previous connections, i.e., no 0-RTT packets can be used. First, the client establishes the connection to the proxy. Then, it upgrades the HTTP connection and includes the initial end-to-end handshake packet encapsulated in an HTTP capsule. Upon receiving the Connect request, the proxy extracts the server address and port and initiates a connection to the server. After issuing a Connect request to the server and establishing another MASQUE tunnel, the proxy forwards the initial end-to-end handshake packet to the server. Finally, the server responds with the end-to-end handshake packet, which is tunneled through both MASQUE connections using HTTP capsules. The connection is established as soon as the client receives the server handshake, and data can be exchanged end-to-end.

4.2. Performance Overhead

A conventional QUIC connection requires one RTT to establish the cryptographic keys to send protected data. Assuming the MASQUE proxy is on the direct network path, establishing the connection through a series of MASQUE tunnels requires an additional handshake for each path segment. Because the HTTP upgrade token containing the server destination address is application data, it can only be transmitted once the handshake is complete. Thus, establishing the tunnels can not be parallelized. The tunnel for the next path segment can only start to be established after the handshake for the previous tunnel has been completed, and the first 1-RTT packet has been sent. Assuming that the first initial packet of the end-to-end connection is already carried in the first 1-RTT packet of each MASQUE connection, it reaches the server after 1.5 consecutive RTTs on each path segment. With the MASQUE tunnels fully established after the first end-to-end packet, the server's handshake response does not experience similar delays. In total, this setup requires at least two end-to-end RTTs to establish the end-to-end connection. Note that these calculations do not account for potential path asymmetries and processing delays in the proxies, and only serve to provide an understanding of the minimal latency overhead. Furthermore, 0-RTT packets, or using already-established MASQUE tunnels, have the potential to reduce this overhead significantly.

The latency of an established connection may be further affected by the internal stream buffers of a proxy. In particular, buffering stream data may lead to repeatedly dis- and reassembling packets in the proxy, resulting in a processing overhead. Additionally, Kühlewind et al. [6] demonstrated that internal stream scheduling algorithms may lead to an increase in packet losses for a proxied connection if the MASQUE tunnel is shared with multiple end-to-end connections.

Importantly, encapsulating QUIC packets in an HTTP connection incurs an overhead in the number of bits needed to transmit a packet. This decreases the available bandwidth, as more bytes of the MTU are needed to send the additional QUIC and HTTP headers, leaving less space for the payload in each packet. When using HTTP capsules to tunnel QUIC packets, the bit overhead, including the end-to-end header, is approximately 10 % for a packet size of 1440 B, whereas the overhead for a packet without tunneling and a packet size of 1380 B is approximately 3 % [6]. It should be noted that the overhead when using a MASQUE tunnel may change in the future, with efforts to introduce the capability to forward packets instead of tunneling [11], reducing the overhead induced by the additional headers of the outer QUIC connection.

4.3. Comparison to Existing Approaches

By migrating the existing connection to the proxy, Kosek et al. [8] avoid the tunneling overhead incurred by additional headers of the outer connection, as discussed in Section 4.2. However, with packets' contents consisting largely of application data, their approach should exhibit similar processing overheads at the endpoints and middleboxes. Both approaches doubly encrypt the application data as well as rebuild the outer encryption layer at the proxy. However, our proposal does not necessitate the connection to be established end-to-end first and allows established tunnels to be used by parallel connections, reducing the connection establishment latency. Furthermore, it avoids having to rework QUIC's cryptographic handshake and exposing privacy-sensitive connection details to a third party.

In contrast, Yuan et al. [9] limit the role of the proxy to providing additional information to an endpoint in the connection. As the proxied packet flow is only observed but not modified by the middlebox, the additional information necessitates a communication side-channel. Additionally, this requires the proxy to operate on-path.Furthermore, the endpoint receiving the acknowledgements needs to implement a congestion control algorithm that can capitalize on them. Even then, such algorithms will only be able to

 doi: 10.2313/NET-2025-11-3_12

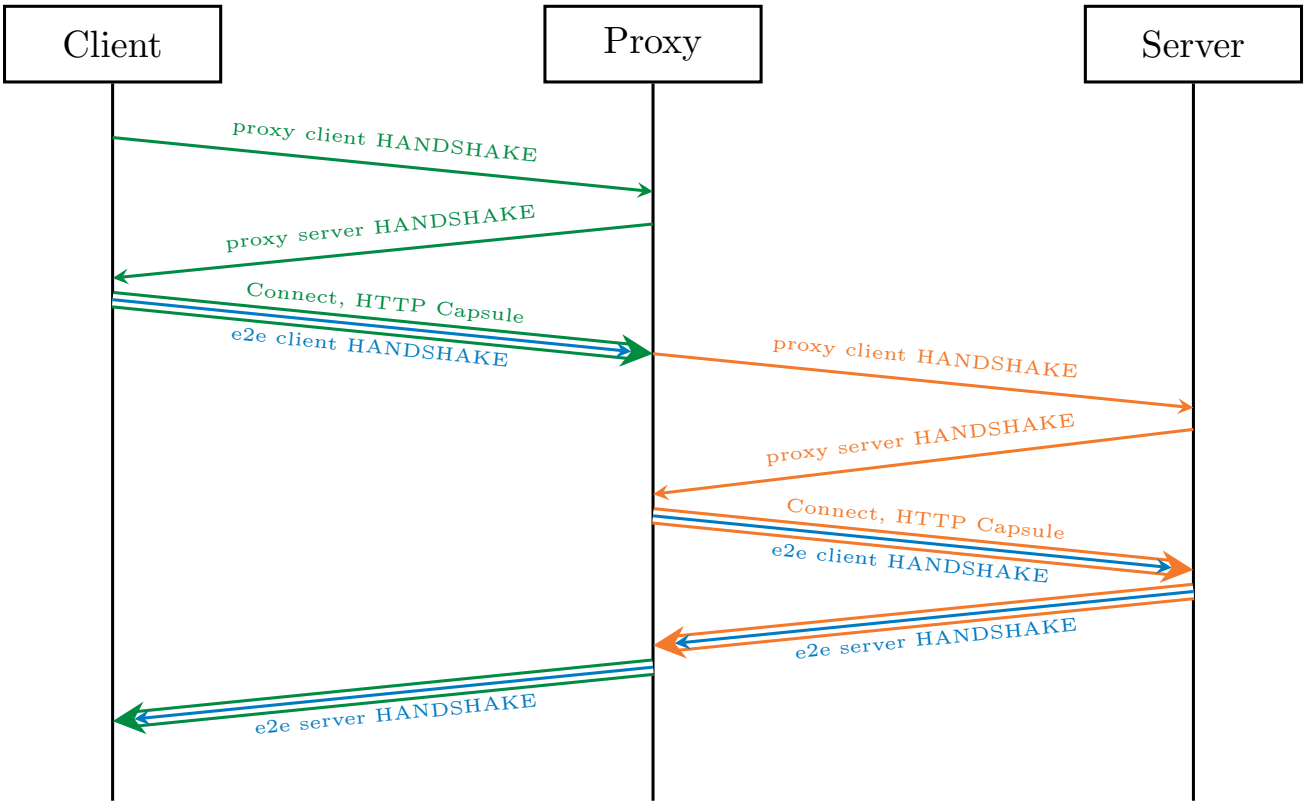


Figure 3: Connection Establishment

approximate the behavior of split congestion control loops. Unless endpoints also exchange packet identifiers with a middlebox that buffers sent packets, packets will always have to be retransmitted end-to-end in case of a loss.Such designs have not been investigated by academia as of the writing of this paper. However, unlike the solutions proposed in this paper and by Kosek et al., operating such a proxy does not require support from both endpoints. Finally, determining whether calculating and distributing packet identifiers or de- and reencrypting packets requires more computational resources remains an open question.

5. Conclusion and Future Work

In this paper, we motivated the continued relevance of Performance Enhancing Proxies for post-TCP transport protocols like QUIC and revisited the general mechanisms of the QUIC protocol and MASQUE proxies. Our main contribution is putting forward a proposal to realise PEPs with split connection control loops for QUIC connections. We further discussed the performance overheads of the solution and compared alternative approaches.

Future work could empirically examine the presented approach and evaluate its performance in comparison with the alternative approaches discussed in Section 3. Moreover, finding alternative solutions to establish a reliable and congestion-controlled tunnel between the last middlebox and the server could eliminate a possible obstacle for adopting the proposed scheme.

References

[1] J. Griner, J. Border, M. Kojo, Z. D. Shelby, and G. Montenegro, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135, Jun. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3135

[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] T. Pauly, E. Kinnear, and D. Schinazi, "An Unreliable Datagram Extension to QUIC," RFC 9221, Mar. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9221

[4] D. Schinazi, "Proxying UDP in HTTP," RFC 9298, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9298

[5] D. Schinazi and L. Pardue, "HTTP Datagrams and the Capsule Protocol," RFC 9297, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9297

[6] M. Kühlewind, M. Carlander-Reuterfelt, M. Ihlar, and M. Westerlund, "Evaluation of QUIC-based MASQUE proxying," in *Proceedings of the 2021 Workshop on Evolution, Performance and Interoperability of QUIC*, ser. EPIQ '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 29–34. [Online]. Available: https://doi.org/10.1145/3488660.3493806

[7] Z. Krämer, M. Kühlewind, M. Ihlar, and A. Mihály, "Cooperative performance enhancement using quic tunneling in 5g cellular networks," in *Proceedings of the 2021 Applied Networking Research Workshop*, ser. ANRW '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 49–51. [Online]. Available: https://doi.org/10.1145/3472305.3472320

[8] M. Kosek, B. Spies, and J. Ott, "Secure middlebox-assisted quic," in *2023 IFIP Networking Conference (IFIP Networking)*, 2023, pp. 1–9.

[9] G. Yuan, M. Sotoudeh, D. K. Zhang, M. Welzl, D. Mazières, and K. Winstein, "Sidekick: In-Network assistance for secure End-to-End transport protocols," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. Santa Clara, CA: USENIX Association, Apr. 2024, pp. 1813–1830. [Online]. Available: https://www.usenix.org/conference/nsdi24/presentation/yuan

[10] Y. Rosomakho, “Reverse HTTP CONNECT for TCP and UDP,” Internet Engineering Task Force, Internet-Draft draft-rosomakho-masque-reverse-connect-00, Apr. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-rosomakho-masque-reverse-connect/00/

[11] T. Pauly, E. Rosenberg, and D. Schinazi, “QUIC-Aware Proxying Using HTTP,” Internet Engineering Task Force, Internet-Draft draft-ietf-masque-quic-proxy-05, Mar. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-masque-quic-proxy/05/

 doi: 10.2313/NET-2025-11-3_12

Time-Sensitive Networking on virtualized network components

Simon Burger, Florian Wiedner*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: simon.burger@tum.de, wiedner@net.in.tum.de

***Abstract*—Time-Sensitive Networking (TSN) and network virtualization are integral to our modern networking landscape. While TSN enhances Ethernet with mechanisms for deterministic data transmission, network virtualization provides the flexibility and scalability required for rapidly deploying applications. However, their convergence remains a challenging topic.**

This paper surveys the individual challenges of TSN and network virtualization before deriving issues that complicate the integration of TSN in virtualized networks (e.g., timing, scheduling, and hardware constraints). It also gives an overview of possible synergetic effects of TSN in virtualized environments before concluding with potential research directions that would benetif effective integration.

***Index Terms*—time-sensitive networking, tsn, network virtualization, tsn in virtualized networks**

1. Introduction

Time-Sensitive Networking (TSN) can be seen as the critical enabler of real-time networking. Industrial and automotive systems, as well as communication networks, have increasingly become real-time applications [1]. With this, the real-world need for deterministic communication with strict latency and jitter guarantees has increased drastically. For example, a missed deadline in an airbag control system is a serious safety hazard.

TSN extends the Ethernet standard to support such guarantees and aims to enable the convergence of time-critical and best-effort network traffic [2]. Simultaneously, network virtualization has become irreplaceable. It enables flexible and programmable network infrastructures that allow virtual networks to share the same physical components [3]. This allows for separating physical infrastructure and network services, which is essential for technologies like 5G, edge computing, and for modern data center environments, as agile deployment and multitenancy are indispensable [4], [5]. Although these technologies developed independently, their combination promises appealing opportunities. However, successfully integrating them is a challenging task. While virtualized environments provide flexibility, scalability, and resource efficiency, they can undermine the timing determinism that TSN was designed to guarantee [3], [6]. We will investigate the challenges in deploying TSN within virtualized network environments. Furthermore, we will discuss what is currently missing to make this process seamless. The paper is structured as follows: Section 2 provides the necessary background on TSN and network virtualization concepts. Section 3 focuses on challenges intrinsic to TSN, whereas Section 4 then outlines the challenges of virtualized networks. Lastly, in Section 5, we explore the integration challenges, highlighting conflicts and mutual benefits that arise when attempting to guarantee deterministic communication over a virtualized infrastructure, before concluding by summarizing the findings and suggesting directions for future research in Section 6.

2. Background

To grasp how TSN may converge with virtual networks in the future, it is necessary to understand each technology separately first. Therefore, this section briefly outlines essential mechanisms of TSN and network virtualization.

2.1. Time-Sensitive Networking

TSN is a set of standards concerning deterministic communication that builds upon Ethernet, as Ethernet can not guarantee latency or low jitter (i.e., variation in packet delay) [1]. Essential concepts include:

- *Scheduling* means to decide which frames to transmit at what time. While numerous optimization objectives exist, latency and determinism are the most relevant ones for TSN [1].
- *Traffic shaping* is done by prioritizing frames and delaying the queuing of others. Combined with a fitting schedule, this establishes a traffic profile that complies with Quality of Service (QoS) requirements (e.g., latency) [1].
- The *Precision Time Protocol* (PTP) is used to synchronize clocks of each node in a system by choosing a Grandmaster Clock, which propagates timing information to all other clocks [7].

2.2. Virtualization

Network virtualization is used to create multiple virtual networks (VNs) on shared physical infrastructure, which is referred to as the substrate network [3]. The mapping of virtual nodes and links onto the physical substrate is known as Virtual Network Embedding (VNE) [8].

Different virtualization techniques are used to create virtualized environments. These include virtual machines (VMs), bare-metal hypervisors, and containers, each with different performance and flexibility properties [4]. In a virtual network, these can constitute virtual nodes. Regular

 doi: 10.2313/NET-2025-11-3_13

Switches, Network Interface Cards (NICs), and Routers can be virtualized and part of a VN. As these nodes are independent of their physical location, they can be placed anywhere, which impacts resource usage, load balancing, and other aspects [9]. By leveraging kernel bypassing techniques (e.g., Data Plane Development Kit (DPDK)), the virtualization overhead caused by the kernel networking stack can be reduced. Here, network device drivers implemented in the userspace are used to send or receive data directly, avoiding the kernel networking stack. [10]. Furthermore, with direct device assignment, VMs can avoid any hypervisor involvement and access I/O network devices directly, thus reducing virtualization overhead [11].

3. Challenges in Time-Sensitive Networking

Modeling and simulating time-sensitive networks. Network modeling is essential for developing and testing novel frameworks, architectures, or protocols. For real-time networks in particular, a robust validation is inevitable. A tight analysis is fundamental to achieving given requirements with minimal resources (i.e., better utilization). For instance, several different modeling languages already exist for automotive embedded systems capable of modeling high bandwidth Ethernet connections. However, Ashjaei et al. note that most of the existing open-source solutions (e.g., AMALTHEA4public, COMDES) cannot support the requirements of TSN [12].

Performance simulation, which can be done in hardware (HW) or software, is another crucial but difficult topic. Due to only a few studies in this field, existing TSN performance simulation methods, especially physical simulation, while still beneficial, are not yet fully mature and must be improved further (e.g., timing precision) [13]. Their ideal usecase can be seen in validating network characteristics under various conditions and enabling rapid testing of different TSN parameters (e.g., within TSN schedulers).

Clock synchronization. To enable real-time communication in a convergent network, both end devices and network devices must have the same notion of time [14].

Firstly, the PTP relies on precise time stamps of each message. Less precision in clock synchronization induces jitter and decreases determinism [15].

Secondly, once a common understanding of time has been established, time is crucial for assigning transmission time slices to packets of various priorities [2]. As discussed by Chahed et al., even minor misalignments can have a devastating effect on latency and jitter, possibly violating the given QoS requirements [1].

Reliability. Any deterministic system has to be reliable. In TSN, reliability is usually guaranteed by a combination of mechanisms. These include Path Control and Reservation, Frame Replication and Elimination for Reliability, and Per-Stream Filtering and Policing. These manage and reserve network resources along the chosen path, send replicated frames along disjoint paths, and filter traffic to prevent overload, improving network reliability. While necessary, these practices lead to increased complexity and resource overhead (e.g., bandwidth). Furthermore, it introduces a high risk of misconfiguration [13].

Scheduling and traffic shaping. In state-of-the-art network architectures, the delays for propagation, processing, and transmission of messages can be considered deterministic and constant. Therefore, the non-determinism is primarily a result of scheduling and queuing delays [14]. To minimize the jitter that a convergent real-time network experiences, appropriate scheduling strategies are essential. The schedule sets the ideal traffic profile (e.g., type and volume of network traffic), while the selected shaping mechanism ensures that the traffic adheres to said profile. Different shaping mechanisms have been developed, each with their respective use cases. Some mechanisms, like Asynchronous Traffic Shaping, do not require a centralized clock but may not be able to meet certain QoS requirements regarding latency. More common approaches like Cyclic Queuing and Forwarding and Time-Aware Shaping (TAS) require time synchronization across all network parts, which is, as previously described, challenging to achieve. Depending on the application area, you must weigh different goals (e.g., performance v. dynamism) to choose the best-fitting traffic-shaping and scheduling algorithms [1]. The transmission schedule in TSN networks is usually computed offline (i.e., all requests known a priori). While there are a range of proposed mechanisms (e.g., TAS), the problem can still not be solved in polynomial time, even if the arrival time of every time-triggered traffic is known in advance [6]. For this reason, heuristic solutions are often applied (e.g., HLS [16]). They do not deliver optimal schedules, but are faster and more efficient in complex networks and under restricted resources [17].

Hardware support. Specialized hardware can be expensive. Together with an increasing need for time-critical enabled networks, this has led to increased efforts towards achieving real-time capabilities using commercial off-the-shelf (COTS) hardware [7], [18] or software-based implementations [15], [19], [20]. Without specialized hardware, some precision will be lost. For example, if a NIC does not support the IEEE 802.1AS standard, the oftentimes required timing precision in the nanosecond range can not be achieved [7].

Moreover, clock synchronization is a significant difficulty for software-only implementations of the PTP. This is due to time stamps of messages being introduced in higher network layers instead of specialized hardware interfaces, leading to more jitter [15]. Especially in embedded systems, it can be necessary to implement specific time-sensitive functions directly in hardware to reduce load on the Central Processing Unit (CPU) [12].

On the other hand, some feasible and precise enough software-only implementations of the PTP do already exist [15]. Furthermore, newer Linux kernel versions are equipped with a TAS implemented in the TAPRIO (Time-aware priority shaper) queuing discipline (qdisc) [7]. Software-only frameworks for emulation and performance testing of TSN networks are also starting to be used in areas like mobile networks [13].

4. Challenges in Network Virtualization

Mapping of physical resources. The mapping of physical resources (i.e., the VNE problem) constitutes one of the most critical and complex challenges in network virtualization. This assignment of VN components

 doi: 10.2313/NET-2025-11-3_13

to the physical substrate must satisfy various constraints, such as CPU capacity, link bandwidth, and other QoS guarantees [21]. The VNE problem is shown to be NP-hard, both in offline settings and in more realistic online scenarios (i.e., requests arrive dynamically over time). For this reason, heuristic or approximation-based algorithms are employed to achieve feasible mappings. However, these often operate under simplifying assumptions such as infinite substrate capacities or static topologies, which limit their applicability in real-world settings [8].

Another dimension to the resource mapping challenge is the dynamic nature of virtualized environments. Virtual machines and containers may be migrated or scaled in response to changing requirements or workloads [22]. The virtual machine placement can be optimized for different objectives, like energy consumption or response time [9]. These changes require a new mapping of resources and impact the previously computed transmission schedules.

Moreover, virtualization technologies themselves introduce overhead that must be factored into the resource mapping in case of strict QoS requirements. In particular, hypervisor-based virtualization introduces additional latency and jitter. To mitigate these issues, recent approaches advocate kernel-bypassing techniques (e.g., DPDK) and direct device assignment [11]. Tail latency and throughput improvements can be an order of magnitude when utilizing these. [23].

Interoperability between heterogeneous networks. As virtual networks scale across administrative domains and physical infrastructures, interoperability issues between heterogeneous networks emerge.

The deficiency in standardized interfaces between infrastructure providers (InPs), service providers (SPs, i.e., organizations that create VNs on the infrastructure of different InPs), and end users is one issue [8]. Proprietary network orchestration implementations hinder cross-domain compatibility and complicate VN provisioning for the SPs [24]. A standardized model (e.g., XML-based) is required to express requirements, making processes like signaling and bootstrapping across diverse infrastructure domains seamless and simplifying the establishment of end-to-end virtual links [8].

Another issue is the naming and addressing incompatibility across network domains. Each virtualized environment may implement its own naming and addressing schemes. This heterogeneity further complicates end-to-end communication. Chowdhury et al., therefore, consider it an important research challenge to find a suitable framework for enabling global connectivity. They further mention one such framework called iMark, which would theoretically be viable but is not feasible in practice [8].

Resource discovery also becomes non-trivial, as inter-domain virtual link provisioning requires infrastructure providers to maintain and expose accurate representations of their physical topology and link capacities. This can be done event-based or periodically [8]. Cross-provider resource allocation mechanisms must be established to allow end-to-end virtual links spanning multiple administrative domains. Without it, VNs would be restricted to a single domain, which may not always be sufficient. This again necessitates standardized interfaces and interoperable signaling protocols between service and infrastructure providers. The diversity of physical networking technologies, each with different characteristics, makes this task particularly challenging [3], [8].

Security in VNs. Isolation is one of the design criteria network virtualization should fulfill. While VNs leverage mechanisms like secure tunnels and advanced encryption to achieve some degree of security, some threats on the physical layer still have to be addressed [3]. Examples would be different forms of physical tampering.

An important aspect in that regard is the monitoring and efficient isolation of failures within the substrate network [3]. Network Virtualization is equipped with several monitoring tools (e.g., for bandwidth or memory usage) that can be used to detect malfunctioning or malicious nodes, which have to be isolated and then removed from the network [25]. Moreover, the flexibility and programmability of each virtualized network element [3], [25] imply an increase in the importance of secure programming.

Lastly, to help prevent failures that could lead to compromised network security, the InPs are required to employ admission control. Primarily due to dynamic resizing of allocated resources, it is challenging to constantly account for the physical network's available resources. However, this is necessary for upholding QoS guarantees and ensuring a secure network [8].

5. Combining Time-Sensitive Networking and Network Virtualization

As these technologies mature, integrating one into the other is a logical step. This section will explore difficulties arising from said integration, like timing precision and scheduling, while highlighting synergetic effects, making this a promising area of research.

5.1. Challenges and limitations

Timing Precision and Hardware Constraints. In a TSN-enabled network, precise time synchronization across all network components is typically achieved by the PTP. However, in software-only implementations of the PTP, time stamps are obtained at higher network layers due to the lack of specialized hardware interfaces, introducing jitter and undermining timing precision [15]. This issue is particularly critical in virtualized environments because the often-used general-purpose hardware limits support for specialized functionality.

Virtualized environments add complexity due to the lack of hardware timestamping support and delays introduced by hypervisors and virtual switches [10], [15]. In such settings, achieving nanosecond precision is difficult. For example, using the software-based TAPRIO qdisc in Linux systems may result in synchronization failures when combined with the PTP if the NIC lacks IEEE 802.1AS support [7]. Frame drops or delays and clock synchronization become unpredictable when hardware queues are full, or timing mechanisms are unavailable or not properly emulated [6]. This additional unpredictability caused by virtualization can not be tolerated in real-time environments.

Moreover, VMs and containers introduce additional overhead through multiple network stack traversals, which

 doi: 10.2313/NET-2025-11-3_13

can considerably impact timing guarantees [10]. For example, it takes roughly 3 to 10 µs to enter and exit a VM and perform an I/O operation [26]. While the use of kernel-bypassing technologies like the DPDK and direct device assignment improve performance, they reduce flexibility by tightly coupling virtual machines to specific physical resources (i.e., complicating VM migration). They also come at the cost of increased CPU load, which can further jeopardize the strict QoS requirements of TSN [11].

Additionally, software-only TSN stacks are inefficient with regard to checksum calculations and memory operations, as specialized HW like NICs and Direct Memory Access controllers typically handle these. In such cases, these tasks consume considerable CPU resources and can introduce latency, which is detrimental to real-time performance and reliability. The software-based network stack with TSN support proposed by Denzler et al. tolerates the mentioned loss in performance in favor of reducing dependence on hardware-specific capabilities [27].

Lastly, heterogeneous networks and inconsistent hardware capabilities across network segments further worsen the synchronization issues. The need for priority translation and the degradation of synchronization across domains can result in inconsistent scheduling behavior and increased jitter, which are difficult to compensate for [1]. Depending on the given TSN requirements, these conditions may not be sufficient.

Scheduling Complexity and Resource Contention. TSN relies on centralized scheduling and shaping mechanisms (e.g., TAS). The schedules are usually computed offline, which requires a static view of the network topology and traffic flows [6]. These mechanisms become more complex when integrated into virtualized environments, where topologies are dynamic due to VM migrations or scaling. In such cases, offline scheduling approaches can become ineffective, as real-time dynamic reconfiguration is not scalable [6], [28].

Virtualization techniques amplify scheduling complexity as they allow virtual components such as talkers and listeners to be placed flexibly across the network nodes. This increases the dimensionality of the scheduling problem since both placement and timing must be optimized to preserve end-to-end latency guarantees [1].

Moreover, many incremental scheduling algorithms are designed to accommodate new flows without modifying previously allocated time slots to prevent jitter and inconsistencies. While these algorithms support some dynamic updates, it becomes problematic in virtualized networks where frequent and flexible reconfiguration is often required [14].

Additionally, optimal resource scheduling is itself, as discussed in section 3, NP-hard and typically requires heuristic solutions [8]. These heuristics may not be accurate enough to support the hard real-time requirements of TSN, even more so in scenarios where multiple infrastructure providers are required to coordinate themselves.

Highly parallel processing platforms, like in automotive systems, introduce even more uncertainty because multiple virtualized components compete for shared resources (e.g., memory, I/O bandwidth). This may lead to execution time variability (i.e., jitter), presenting another challenge in virtualized TSN environments [12]. The poor timing predictability is especially problematic for traffic with hard deadlines, meaning such virtualized parallel systems are not well-suited for real-time guarantees.

Finally, complexity further increases because of the physical resource constraints (e.g., limited processing or buffer capacity at gateways) to which deployed virtual network functions must adhere. It is not possible to minimize the resulting jitter in polynomial time [29].

5.2. Possible synergetic effects

Although they do not solve any of the challenges of implementing one another, TSN and network virtualization each address some weaknesses of the other. As previously mentioned, real-time-enabled virtualized networks have applications across a wide range of industries. The following will be a concise overview of how both technologies can benefit from each other once the previously discussed challenges are overcome.

- Virtualized networks may introduce unpredictable latency, jitter, and even packet loss due to shared physical resources and virtualization overhead [6], [11]. The lack of real-time guarantees can make VNs impracticable in ultra-low latency systems. If TSN is introduced to such systems, a bounded latency and deterministic transmission can be guaranteed through the various mechanisms described in the TSN standard.
- TSN is typically rather strict, assuming static topologies and centralized offline scheduling [6]. This can become infeasible in more dynamic environments, like cloud-based or industrial systems. Network virtualization can enable dynamic deployment of TSN-enabled functions across VMs, increasing flexibility and scalability.
- As TSN traditionally requires hardware support (e.g., by NICs [7]), experimenting with it can be expensive. Many TSN features can be implemented in software on general-purpose COTS HW, enabling better academic research and simpler testing using virtualized networks.

6. Conclusion and future work

The paper examined TSN and network virtualization separately before investigating the challenges of deploying TSN within virtualized network environments. This highlighted how their realization poses timing, scheduling, and resource management issues. Identified challenges include a lack of hardware support in virtualized environments (e.g., for timestamping), limitations of centralized offline scheduling in dynamic topologies, and the difficulty of achieving deterministic behavior in systems that inherently tend towards flexibility.

Several research directions must be pursued to fully leverage the benefits of TSN in virtualized networks. Firstly, ongoing research on developing accurate clock synchronization protocols should be directed towards fully software-based implementations. Moreover, flexible scheduling algorithms must be specifically designed and evaluated for dynamic environments while still enabling real-time guarantees. Furthermore, practical solutions for

solving the VNE problem in real time are needed. Further improving kernel-bypassing techniques could be a viable research direction to increase performance while preserving the configurability and scalability of virtualized networks.

References

[1] H. Chahed and A. Kassler, "Tsn network scheduling—challenges and approaches," *Network*, vol. 3, no. 4, pp. 585–624, 2023. [Online]. Available: https://www.mdpi.com/2673-8732/3/4/26

[2] A. Weder, "TIME SENSITIVE NETWORKING: Eine Einführung in TSN," Fraunhofer IPMS, Dresden, Germany, White Paper, 2019. [Online]. Available: http://s.fhg.de/time-sensitive-networking

[3] N. M. Mosharaf Kabir Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20–26, 2009.

[4] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, programmable, and virtualized 5g networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, p. 107516, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620311786

[5] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 909–928, 2013.

[6] M. Pahlevan, "Time sensitive networking for virtualized integrated real-time systems," Doctor of Engineering Dissertation, University of Siegen, Siegen, Germany, 2019.

[7] F. Rezabek, M. Bosk, G. Carle, and J. Ott, "Tsn experiments using cots hardware and open-source solutions: Lessons learned," in *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2023, pp. 466–471.

[8] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128609003387

[9] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1409–1434, 2019.

[10] A. Garbugli, L. Rosa, A. Bujari, and L. Foschini, "Kubernetsn: a deterministic overlay network for time-sensitive containerized environments," in *ICC 2023 - IEEE International Conference on Communications*. IEEE, 2023, p. 1494–1499. [Online]. Available: http://dx.doi.org/10.1109/ICC45041.2023.10279214

[11] A. Garbugli, L. Rosa, L. Foschini, A. Corradi, and P. Bellavista, "A framework for tsn-enabled virtual environments for ultra-low latency 5g scenarios," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 5023–5028.

[12] M. Ashjaei, L. Lo Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, "Time-sensitive networking in automotive embedded systems: State of the art and research opportunities," *Journal of Systems Architecture*, vol. 117, p. 102137, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121001028

[13] J. Pei, Y. Hu, and L. Tian, "A review on key mechanisms of time-sensitive networking," in *2021 International Conference on Advanced Computing and Endogenous Security*, 2022, pp. 01–07.

[14] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2066–2075, 2018.

[15] K. Correll, N. Barendt, and M. Branicky, "Design considerations for software only implementations of the ieee 1588 precision time protocol."

[16] M. Pahlevan, N. Tabassam, and R. Obermaisser, "Heuristic list scheduler for time triggered traffic in time sensitive networks," vol. 16, no. 1, 2019. [Online]. Available: https://doi.org/10.1145/3314206.3314208

[17] C. Xue, T. Zhang, Y. Zhou, M. Nixon, A. Loveless, and S. Han, "Real-time scheduling for 802.1qbv time-sensitive networking (tsn): A systematic review and experimental study," in *2024 IEEE 30th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2024, pp. 108–121.

[18] J. Coleman, S. Almalih, A. Slota, and Y.-H. Lee, "Emerging cots architecture support for real-time tsn ethernet," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2019, p. 258–265. [Online]. Available: https://doi.org/10.1145/3297280.3297542

[19] M. K. Hany, K. Montgomery, and R. Candell, "An analytical evaluation for software-based tsn in industrial wi-fi networks," in *2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2024, pp. 1–6.

[20] R. Candell, K. Montgomery, M. Kashef Hany, S. Sudhakaran, and D. Cavalcanti, "Scheduling for time-critical applications utilizing tcp in software-based 802.1qbv wireless tsn," in *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*, 2023, pp. 1–8.

[21] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[22] O. Smimite and A. Karim, "Containers placement and migration on cloud system," *International Journal of Computer Applications*, vol. 176, 07 2020.

[23] J. Fried, G. I. Chaudhry, E. Saurez, E. Choukse, I. Goiri, S. Elnikety, R. Fonseca, and A. Belay, "Making kernel bypass practical for the cloud with junction," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Apr. 2024, pp. 55–73. [Online]. Available: https://www.usenix.org/conference/nsdi24/presentation/fried

[24] A. Francescon, G. Baggio, R. Fedrizzi, R. Ferrusy, I. G. Ben Yahiaz, and R. Riggio, "X–mano: Cross–domain management and orchestration of network services," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–5.

[25] N. C. Fernandes, M. D. D. Moreira, I. M. Moraes, L. H. G. Ferraz, R. S. Couto, H. E. T. Carvalho, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte, "Virtual networks: isolation, performance, and trends," *annals of telecommunications - annales des télécommunications*, vol. 66, no. 5, pp. 339–355, 2011.

[26] D. B. Oljira, "Low latency communication in virtualized and multipath networks," Ph.D. dissertation, 2020.

[27] P. Denzler, T. Frühwirth, C. Lehr, and J. Auffray, "Time-predictable software-based tsn-enabled network stack for mixed criticality traffic," in *2024 IEEE 20th International Conference on Factory Communication Systems (WFCS)*, 2024, pp. 1–8.

[28] A. Stage and T. Setzer, "Network-aware migration control and scheduling of differentiated virtual machine workloads," in *2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, 2009, pp. 9–14.

[29] Y. Zhang, Q. Xu, M. Li, C. Chen, and X. Guan, "Qos-aware mapping and scheduling for virtual network functions in industrial 5g-tsn network," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

Applications of MASQUE-proxies in TEE Environments

Martin Halfen, Lion Steger*, Daniel Petri*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: martin.halfen@tum.de, stegerl@net.in.tum.de, petriroc@net.in.tum.de

***Abstract*—This paper will talk about two privacy and anonymity enhancing technologies: MASQUE-Proxies and Trusted Execution Environments (TEEs). TEEs offer hardware-based confidentiality and integrity for sensitive computations, whereas MASQUE enables encrypted and obfuscated traffic tunneling over QUIC and HTTP/3, enhancing resistance to traffic analysis. This paper investigates the possibilities of merging these two technologies to create secure, privacy-preserving proxy infrastructures. We investigate the technical foundation of TEEs and MASQUE, assess practical implementations like Apple's iCloud Private Relay, and discuss the role of TEEs in enhancing trust through verifiable execution. We assess performance trade-offs using related TEE-based proxy implementations.**

***Index Terms*—confidential computing, cloud computing, MASQUE-Proxy, TEE**

1. Introduction

Today the demand for privacy and anonymity in the online realm is higher than ever, New technologies such as Trusted Execution Environments (TEEs) and MASQUE proxies are promising ways to address these concerns. TEEs leverage hardware-based security to ensure that sensitive data is processed securely, even if the surrounding software stack is compromised by an adversary.

MASQUE (Multiplexed Application Substrate over QUIC Encryption) is a new proxying protocol built upon the modern foundations of HTTP/3 and QUIC. The protocol allows the user to tunnel arbitrary data through connections that appear to external observers as standard encrypted HTTP traffic [1]. This obfuscation method helps protect metadata and improves resistance against traffic analysis. MASQUE is already deployed in services like Apple's iCloud Private Relay, which aims to hide users' network traffic from adversaries [2].

The combination of TEEs and MASQUE proxies creates a powerful framework for both secure computation and private communication. This synergy opens up promising opportunities for privacy-preserving applications in this area.

This paper explores the technical foundations, potential applications, and challenges of combining these technologies. In Section 2.1, we provide an overview of TEEs and their specification. In Section 2.2 we cover the MASQUE protocol and its technical foundations. With this background we discuss in Section 4 the potential applications of combining these technologies, as well as the technical challenges that can arise with their integration.

2. Background

This section provides the essential background needed to understand how MASQUE proxies are integrated with Trusted Execution Environments. Our starting point will be a description of the fundamental principles and security guarantees of TEEs, with an emphasis on the mechanisms that render them especially appropriate for safeguarding sensitive computations on untrusted infrastructure. Subsequently, we investigate the MASQUE protocol and its distinctive proxying features which uses QUIC and HTTP/3, emphasizing how it facilitates efficient and private network tunneling. These technologies can be combined to establish the foundation for proxy architectures that are secure, high-performance, and privacy-preserving.

2.1. Trusted Execution Environments (TEEs)

TEEs leverage hardware-based security mechanisms to protect the integrity and confidentiality of software running on potentially untrusted platforms, such as public cloud infrastructure, third-party data centers, or infrastructure with uncontrolled access. In this paper, we assume an adversarial model where the adversary has full control over the entire software stack and can execute any arbitrary privileged program, and has also full control over all OS duties like CPU scheduling and IO operations [3]. TEEs offer two critical functionalities: **remote attestation** and **runtime protection**.

Remote attestation enables an external verifier to determine the integrity of a TEE instance before interacting with it as Li et. al show in their work [3]. This process involves the TEE instance requesting an attestation from a trusted attestation authority, typically a secure manufacturer TEE instance or a secure hardware component on the chip [3]. The authority generates a signed attestation report that includes a cryptographic measurement of the TEE's initial state, covering both the deployed software and critical system components, including CPU features, memory layout, and operating system parameters [3]. The verifier can then use this report to determine whether the TEE instance is trustworthy before providing sensitive data or workloads.

After verifying the initial state of the TEE through remote attestation, the system must also ensure its integrity during runtime. This phase, referred to as *Runtime Protection* (RP), is responsible for managing system resources such as CPU, memory, and I/O—while protecting the TEE from interference by the untrusted components of the OS [3].

 doi: 10.2313/NET-2025-11-3_14

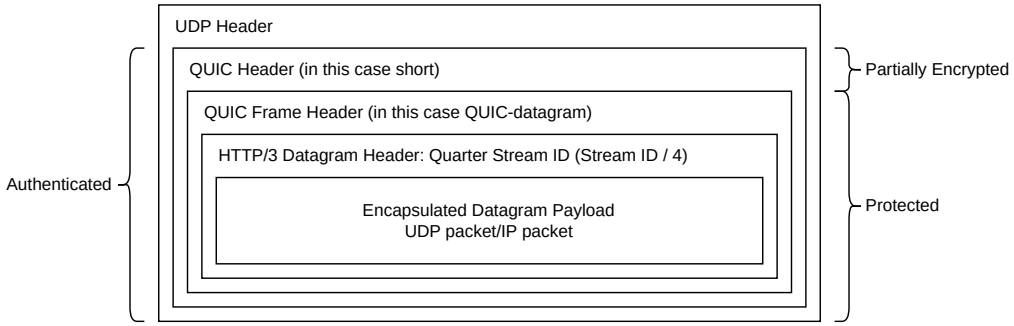


Figure 1: UDP/IP over MASQUE-Proxy taken from [1]

Several runtime protection strategies have been proposed by Li et. all [3] in thier paper:

1) **Unprotected Mode**: The TEE relies fully on the untrusted OS for resource management, especially for I/O operations like networking or disk access [3]. This simplicity comes at the cost of exposure to side-channel attacks and manipulation.
2) **Isolated Runtime Management**: TEE resources are managed separately by the runtime layer, while the OS handles non-TEE workloads. This is typical for context switching scenarios [3].
3) **Guarded Runtime**: A mediation layer observes and verifies OS actions, such as memory allocation, to maintain integrity without full detachment from the OS [3].
4) **TEE-Managed Mode**: The TEE manages its hardware resources independently (e.g., virtual memory). While highly isolated, this mode can conflict with system schedulers and degrade performance [3].

Each of these modes represents a trade-off between security, performance and trust assumptions. Selecting the appropriate runtime protection strategy depends on the threat model and the level of trust placed in the underlying OS.

2.2. MASQUE Proxy

The core idea behind the *Multiplexed Application Substrate over QUIC Encryption* (MASQUE) protocol is to enable the tunneling of arbitrary data streams over HTTP/3 using the QUIC transport protocol [1]. MASQUE supports two main tunneling modes: `CONNECT-UDP` and `CONNECT-IP`, both of which allow traffic to be obfuscated and multiplexed in a way that resembles standard web traffic, thereby enhancing privacy [1].

In `CONNECT-UDP` mode, the client must first specify the target IP address and port to which UDP packets should be forwarded. After the QUIC connection with the MASQUE-proxy is established, raw UDP packets are wrapped within HTTP/3 datagrams and sent to the MASQUE-proxy. The proxy then unwraps the UDP-packages from the QUIC-header and forwards the UDP payloads to the specified endpoint and relays the responses back to the client. This mode is particularly suitable for use cases such as Voice-over-IP, where UDP is already the underlying transport protocol [1].

In `CONNECT-IP` mode, the client encapsulates raw IPv4 or IPv6 packets into HTTP/3 datagrams, which are then transmitted over a QUIC connection to the MASQUE proxy. The proxy unwraps the packets and forwards them to their intended destination, enabling the tunneling of arbitrary IP traffic. This mechanism allows MASQUE to emulate a full-tunnel VPN, where the client's entire IP-layer traffic is routed through the proxy in a way that is indistinguishable from standard encrypted web traffic [1].

In both tunneling modes, MASQUE leverages the encryption of QUIC and commonness of HTTP/3 traffic to hide the nature of tunneled traffic, making it more difficult for an adversary to perform traffic classification, surveillance or correlation. This makes MASQUE a compelling candidate for privacy-preserving proxy deployments in trusted or semi-trusted network environments.

3. Related Work

TEEs have been proposed as a foundation for secure computing in an untrusted environment such as data centers. Li et al. [3] provide a comprehensive systematization of the design choices and pitfalls in modern TEE architectures, including remote attestation and runtime protection models [3]. Practical deployments methods such as SCONE [4] have demonstrated how TEEs can be used to secure containerized applications.

In the context of encryption proxies, Bouhairi et al. [5] evaluate a SCONE-based implementation of the Eperi Gateway, an encryption proxy, and demonstrate how TEEs can ensure data confidentiality. However, their findings also reveal a measurable performance overhead, with latency increasing from 423 ms to 912 ms, effectively more than doubling [5]. Their findings provide insights into the practical trade-offs when deploying proxy services within secure enclaves.

The MASQUE protocol is a relatively recent development, offering encrypted tunneling over HTTP/3 using QUIC. It has been implemented in Apple's iCloud Private Relay, which uses a two-hop architecture to decouple user identity from destination servers [2]. Probst [1] presents a MASQUE-based proxy prototype for lower OSI-layer traffic, illustrating the flexibility of MASQUE for privacy-preserving proxying.

Kühlewind et al. [6] published a paper about the performance of MASQUE without a TEE environment.

To date, no published work has examined the integration of MASQUE proxies with TEEs. This paper proposes such a combination, outlines the benefits for

 doi: 10.2313/NET-2025-11-3_14

privacy-preserving analytics, and identifies future research directions to realize this architecture in practice.

4. Applications of MASQUE-Proxies in TEE Environments

The integration of MASQUE proxies with TEEs presents a promising pathway for achieving secure and more private communication over the internet. TEEs, such as Intel SGX, can help eliminate the need to trust proxy providers blindly by providing a mechanism to verify the running TEE instance. The combination of MASQUE's effective multiplexed proxying via the QUIC protocol with the confidentiality assurances of TEEs enables a new way of secure proxy-based architectures.

This section investigates three particular domains of application in which TEE integration can be advantageous for MASQUE proxies.

First, we discuss in Section 4.1 how TEE can be used in way that traffic obfuscation techniques are used to improve user anonymity. Next, we examine in Section 4.2 how TEEs can enhance trust in systems like *iCloud Private Relay*. Finally, we analyze in Section 4.3 the performance trade-offs observed in TEE-based proxy implementations to understand their practical implications.

4.1. Confidential Proxying with Traffic Obfuscation

When browsing the internet, an adversary observing the network traffic can often determine which website a user is accessing. The primary purpose of a proxy is to conceal the user's intended destination, thereby enhancing privacy by preventing direct association between the user and the target website [1].

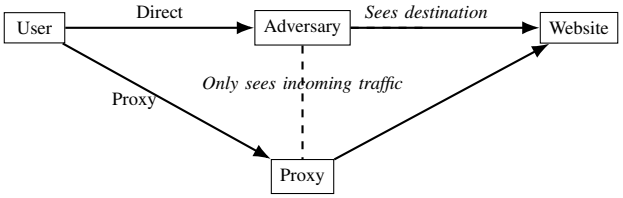


Figure 2: Using a proxy to hide the destination website from network observers

Ensuring user anonymity when using proxies is a critical requirement when using a proxy. One effective method to achieve this is through a technique known as *mixing* [7]. In this approach, data packets originating from multiple users are collected at an intermediate node. These packets are then shuffled and subjected to an intentional delay before being forwarded to the destination [7]. This randomized reordering and timing obfuscation significantly complicate the task of correlating input and output flows, both for the proxy provider and for a global passive adversary, thereby enhancing traffic-level privacy, as depicted in Figure 4.1.

A fundamental challenge in proxy based anonymization is the need to trust the proxy provider not to log incoming and outcoming traffic. As such logs would enable the correlation of network activity with a specific

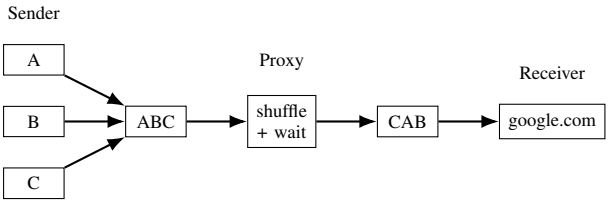


Figure 3: The mixing Workflow

users. This trust requirement can be mitigated through the use of TEEs combined with remote attestation. If the cloud provider supplies both the proxy source code and the corresponding TEE measurement, remote attestation can be employed to verify that the proxy instance is indeed running within a genuine, untampered TEE. An additional advantage of this approach is the strong isolation provided by TEEs. It becomes significantly more difficult for an adversary, even one with access to the same physical hardware, to escape their execution context and observe traffic processed by the TEE instance [3]. This strengthens the anonymity of the traffic obfuscation mechanism.

4.2. Enhancing Trust in iCloud Private Relay through TEEs

Another approach to enhancing privacy is known as *re-routing*, where traffic is relayed through one or more intermediary nodes [7]. Apple implements this concept in its proprietary service *iCloud Private Relay* [2], which uses the MASQUE protocol as the underlying proxying mechanism.

The core idea behind this method is to separate the knowledge of the data source and its destination. Apple receives the user's incoming traffic but does not see the final destination. Instead, it forwards the encrypted request to a third-party relay, which then delivers the data to the intended receiver [2]. This two-hop architecture ensures that no single entity has full visibility of both the sender and the receiver [1], [2].

However, this model relies on a degree of trust in both Apple and the third-party relay provider. Users must assume that neither party colludes or logs identifying information [2]. This is where *TEEs* can strengthen the trust model. If both relay providers executed their services inside a TEE environment, published their code and attestation measurements, users could independently verify that their data is handled as promised.

Unfortunately, this level of transparency is currently unrealistic. Apple does not typically open source its codebase, and Apple did not state anything about public attestation in their white paper [2]. Nonetheless, this idea highlights the potential role of TEEs in increasing accountability and verifiability in privacy-preserving relay systems.

4.3. Performance Implications of TEE-Based Proxy Implementations

Bouhairi et al. [5] investigate how the Eperi Proxy can be implemented using SCONE, a secure Docker container

 doi: 10.2313/NET-2025-11-3_14

framework based on Intel SGX technology. SCONE enables the creation of secure Dockerfiles from any Docker image and offers additional features such as filesystem shielding, network shielding and secure system calls [4]. Through this implementation, encryption and decryption operations are executed entirely within the secure enclave, ensuring both data confidentiality and integrity.

The Eperi Gateway shares similarities with MASQUE, which also emphasizes secure and confidential data transmission, but uses TCP/TLS over the QUIC protocol [5]. Therefore, several observations from the SCONE-based implementation may provide valuable insights for a potential MASQUE-based TEE deployment.

One of the main findings from their evaluation is a significant increase in latency, rising from 423 ms to 912 ms, more than double [5]. This latency loss is critical, especially for latency-sensitive applications such as *iCloud Private Relay*. Several factors contribute to this overhead, including cache flushes, the cost of integrity checks performed at each context switch, and the memory encryption overhead introduced by Intels SGX architecture [3].

Another key metric analyzed is throughput. While the TEE-based implementation achieves similar throughput levels to the non TEE implementation, it demands substantially more CPU resources, 910 % instead of 790 % at maximum throughput [5]. As a result, the SCONE system reaches a throughput ceiling at approximately 100req/s, due to near saturation of the available virtual CPU cores which will result in a latency jump [5].

5. Conclusion and Future Work

In conclusion, TEEs offer promising new opportunities to enhance the privacy guarantees of MASQUE-based proxy architectures. By isolating sensitive operations, like encryption and decryption, and enabling verifiable execution, TEEs could mitigate trust assumptions inherent in current systems. However, challenges remain, including susceptibility to denial-of-service (DoS) attacks and enclave resource exhaustion, which must be systematically addressed to ensure robust and practical deployments [3].

However, given that MASQUE is a relatively new protocol, there is currently little research on the integration of TEEs in practical MASQUE proxy deployments. Most available comparisons rely on alternative systems, such as the Eperi Gateway.

For future work, a prototype implementation of a MASQUE proxy within a TEE instance, using frameworks such as SCONE, could provide valuable insights into performance overheads and practical feasibility. Additionally, further investigation is needed into mechanisms that allow end users to verify that a MASQUE proxy is genuinely running inside a trusted TEE instance on a cloud provider, possibly through remote attestation.

Such developments could significantly increase the trustworthiness and transparency of next-generation privacy infrastructures.

References

[1] C. Probst, "Rust-based MASQUE-Proxying for Lower OSI Layer Traffic," Jul 2023, accessed: 2025-06-01. [Online]. Available: https://oc.net.in.tum.de/s/MsdcpDFrJ2ikHsa/download/thesis_probst.pdf

[2] Apple Inc., "iCloud Private Relay Overview," Whitepaper, Dec 2021, accessed: 2025-06-01. [Online]. Available: https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf

[3] M. Li, Y. Yang, G. Chen, M. Yan, and Y. Zhang, "Sok: Understanding design choices and pitfalls of trusted execution environments," in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1600–1616. [Online]. Available: https://doi.org/10.1145/3634737.3644993

[4] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. L. Stillwell, D. Goltzsche, D. Eyers, R. Kapitza, P. Pietzuch, and C. Fetzer, "Scone: secure linux containers with intel sgx," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16. USA: USENIX Association, 2016, p. 689–703.

[5] M. J. A. Bouhairi, M. Mullick, M. Wolf, I. Gudymenko, and S. Clauß, "Encryption Proxies in a Confidential ComputingEnvironment," feb 2023, accessed: 2025-06-08. [Online]. Available: https://wwwpub.zih.tu-dresden.de/~s0278016/publications/Encryption_Proxies_in_Conf_Comp_Environments.pdf

[6] M. Kühlewind, M. Carlander-Reuterfelt, M. Ihlar, and M. Westerlund, "Evaluation of quic-based masque proxying," in *Proceedings of the 2021 Workshop on Evolution, Performance and Interoperability of QUIC*, ser. EPIQ '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 29–34. [Online]. Available: https://doi.org/10.1145/3488660.3493806

[7] F. Shirazi, M. Simeonovski, M. R. Asghar, M. Backes, and C. Diaz, "A survey on routing in anonymous communication protocols," *ACM Comput. Surv.*, vol. 51, no. 3, Jun. 2018. [Online]. Available: https://doi.org/10.1145/3182658

Evaluation of sources for IPv6 Hitlists

Finn Johannes Hartmann, Lion Steger*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: finn.hartmann@tum.de, stegerl@net.in.tum.de

***Abstract*—IPv6 hitlists are essential tools for conducting large-scale internet measurements. This paper evaluates various data sources used to construct IPv6 hitlists, focusing on IPv6 address stability and Border Gateway Protocol (BGP) distribution. Our analysis reveals significant differences in the quality and characteristics of each source's data and highlights the importance of source selection in measurement studies.**

***Index Terms*—IPv6, measurement, ASN, historical analysis**

1. Introduction

The ongoing adoption of IPv6 has introduced many challenges due to the large scale of the address space (2^{128} addresses). Due to this size, it is infeasible to scan the whole space. To address this need, the 'IPv6 Hitlist Service' maintained by the Chair of Network Architectures and Services at TUM [1] provides a continuously updated list of responsive IPv6 addresses. In the following, we will refer to it as 'hitlist'. The hitlist contains IPv6 addresses from different public sources and verifies their responsiveness regularly. Hence, it has become an essential tool for internet scans.

Previous work by Steger et al. [2] established this hitlist methodology, while Gudehege [3] analyzed the source quality deeply.

In this paper, we will analyze such an IPv6 hitlist, and focus on two other aspects:

- The stability of Autonomous System Numbers (ASNs) [4] for IPv6 addresses over time (Section 4.1)
- The source distribution across ASNs (Section 4.2)

With these analyses, we try to find a choice of sources that will improve future measurement efforts in their overall representativeness. We aim to inform future measurement efforts by better understanding the relationship between sources and ASNs and address stability.

2. Data Overview

This section provides foundational information on the hitlist dataset and the methods used to collect and process the data. The subsequent analysis presented in Section 4 is based on this background.

We are using a dataset collected by the IPv6 Hitlist Service [1]. The hitlist contains a filtered list of reachable IPv6 addresses gathered from their last scan [5]. To perform these scans, they use various sources, e.g., the hitlist, Bitnodes, and IPInfo (Section 2.2). Hence, the hitlist shows IPv6 addresses from the active Internet address space and is updated regularly, nowadays on a monthly basis, but in the earlier days weakly or daily, during each scan.

These addresses are subsequently categorized based on their responsiveness to different network protocols, enabling a deeper understanding of IPv6 address space utilization [2].

For our evaluation, we worked on an SQL import of the hitlist by Gudehege [3], which is structured into two main components:

- **Inputdata**: All collected IPv6 addresses, irrespective of their responsiveness, are active unless our scan reverts this assumption.
- **Outputdata**: A filtered subset of the inputdata that includes only addresses that responded during active scans.

2.1. Data Structure

The distinction between the inputdata and outputdata tables is essential for our analysis. Not all relevant metadata is presented in both tables, necessitating the use of both to perform a comprehensive evaluation.

- **Inputdata Table Fields**: `ip`, `source`, `date`, `bgpIp`, `bgpMask`, `asn`
- **Outputdata Table Fields**: `ip`, `protocol`, `date`, `bgpIp`, `bgpMask`, `asn`

The key difference lies in the `protocol` field in the outputdata and the fact that only responsive IPv6 addresses are retained. This makes the outputdata particularly useful for filtering reachable IPs.

2.2. Data Collection

The IPv6 addresses were aggregated from a diverse range of publicly available sources, including e.g.:

- Bitnodes — provides active Bitcoin nodes, which operate over IPv6 [6]
- IPInfo — offers a dataset which contains enriched metadata for their observed IPs, including e.g. geolocations and ASN [7]
- Rapid7DNS — contains IPv6 responses from large-scale zone file scans [8]
- RIPE — collects data by using globally distributed probes, which are based on a community supported measurement [9]

 doi: 10.2313/NET-2025-11-3_15

- Hitlist — is maintained by the Chair of Network Architectures and Services which combine all these different sources and perform their scans on this data to update their list of responsive IPv6 addresses and performance DNS resolution [1], [5]

This comprehensive aggregation strategy ensures broad coverage of the active IPv6 address space. The database on which our analyses are based contains approximately 11B entries. Therefore, we limited the data volume for our evaluation by using a sampling condition in which we only included every 100th entry.

3. Related work

This paper is based on data collection from the IPv6 hitlist maintained by the Chair of Network Architectures and Services since 2018 by Gasser et al. [1]. They have introduced a methodology for generating IPv6 hitlists by aggregating data from various sources. The hitlist structure on which we are working is based on the SQL structure from Gudehege [3]. This structure is explained in the background (Section 2.1).

Our work extends these studies by providing a comparative analysis of data sources concerning IPv6 address stability and BGP [10] distribution, offering insights into the suitability of each source for different measurement objectives.

4. Analysis

During the analysis in this chapter, we focus on two main aspects: IPv6 address stability (Section 4.1) and source ratio per ASN (Section 4.2). The analysis is based on measurements from the `hitlistdb.inputdata` dataset. Each source is analyzed over time to detect volatility patterns and systematic biases in network distribution.

4.1. Analysis of ASN stability

This subsection examines, we examine the ASN changes from November 2023 to October 2024, focusing on the temporal patterns and source-specific characteristics of IPv6 address stability.

4.1.1. Data preparation. In the following analysis, we work with a subset of our inputdata, specifically querying the dataset for all IPv6 addresses that have appeared with more than one Autonomous System Number (ASN). From this query, we gather a list of 'unstable' IPs exhibiting ASN changes, which were used for a detailed timeline visualizations of ASN stability patterns.

4.1.2. Observations. Figure 1 shows the monthly count of IPv6 addresses that changed their ASN, broken down by data source. Additionally, Figure 2 presents the relative fraction of IPs with ASN changes per source, providing normalized insights into source-specific volatility patterns.

Monthly ASN Change Patterns: The monthly ASN change analysis reveals significant temporal and source-specific variations in IPv6 address stability. From November 2023 to February 2024, we observe a noticeable peak in the total number of ASN changes in IPs, reaching approximately 2,500–3,000 per month. Afterward, the total number of IPs with ASN changes declines and stabilizes around 1,000–1,500 changes per month. Such a noticeable difference could be caused by increased network infrastructure changes or BGP hijacking [11]. The latter refers to a falsely announced BGP prefix by an Autonomous System (AS).

Source-Specific Volatility Patterns: The clustering of sources shows a clear behaviour pattern across different sources. High-volatility sources (including *ipinfo*, *yarrp*, and *rapid7dns*) contribute disproportionately to ASN changes throughout the observation period. However, such a high proportion could also be based on the size of these sources (TABLE 1), which is, in our case, very likely. In peak months, *ipinfo* and *rapid7dns* show a similar pattern to the total cumulation. Such a pattern indicates that these sources capture IP addresses that are associated with dynamic or ephemeral infrastructure. This leads to the effect that they are assigned only temporarily and change more frequently. Each newly assigned IPv6 address may also map to a different Autonomous System due to load balancing or routing optimization. Against that, there are stable sources such as *bitnodes*, *openipmap*, and *ripe*, where assigned IP addresses infrequently change over time. These sources rarely exceed 200–300 changed ASNs monthly, making them more suitable for long-term network topology studies.

Source	Count
ipinfo	8.168.647.739
hitlist	2.349.036.709
yarrp	541.657.853
rapid7dns	228.492.805
ripe	202.662.048
bitnodes	50.812.377
openipmap	45.478.545

TABLE 1: IPv6 address counts per data source

However, unlike the total number of IP changes, the relative numbers show that *ipinfo* and *yarrp* have consistent IP changes. The relative numbers are based on the total number of IP changes for the corresponding source. More than 50% of *ipinfo* and more than 40% of *yarrp* IPs change their ASNs monthly. Generally, there is nearly the same pattern as shown in Figure 1. Initially, all sources have a high change rate, spreading from 10% to 100%. From March to October, we can observe clusters. One has a high change rate of roughly 50%–90%, and the other from around 0%–45%.

Temporal Stability Trends: The data shows an overall decline in ASN changes with a constant decrease from the peak in early 2024 to approximately 1,000–1,500 per month.

Figures 3 and 4 illustrate the ASN change patterns per IP over time by plotting temporal ASN assignment intervals for individual IP addresses.

The frequency of ASN changes observed for specific IP addresses, particularly those shown in Figure 3, represent an uncommon behavior in typical internet infrastructure. The analysis reveals that most IPv6 addresse with frequent ASN share a common BPG prefix. Two

 doi: 10.2313/NET-2025-11-3_15

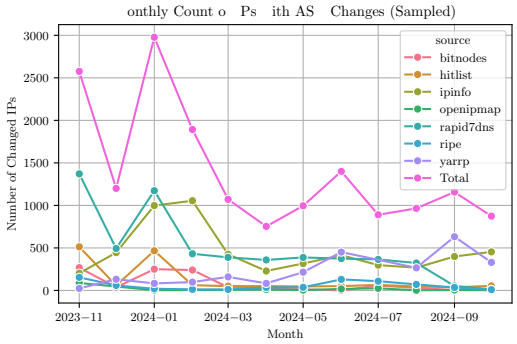


Figure 1: Monthly ASN changes per source

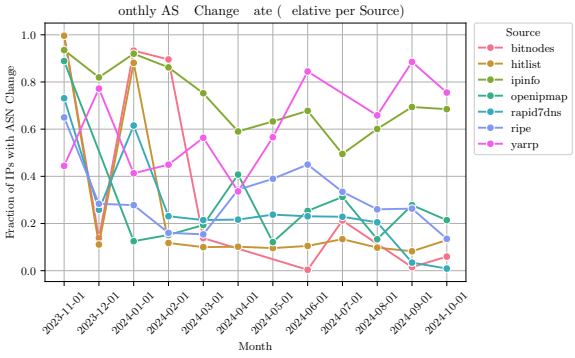


Figure 2: Monthly ASN change rate (relative per source)

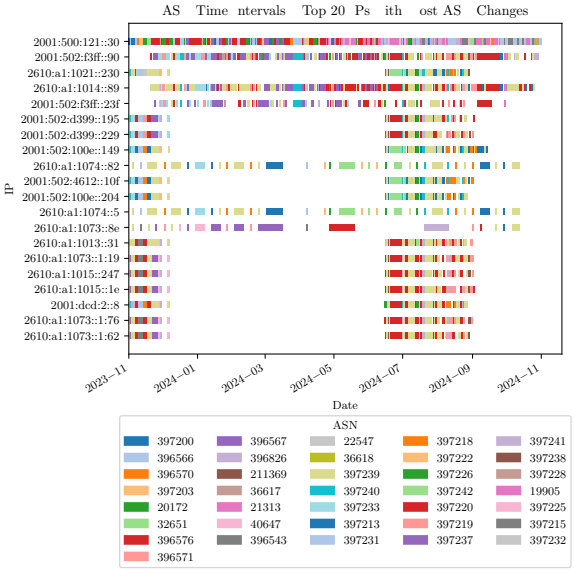


Figure 3: Top 20 IPv6 addresses with most ASN changes

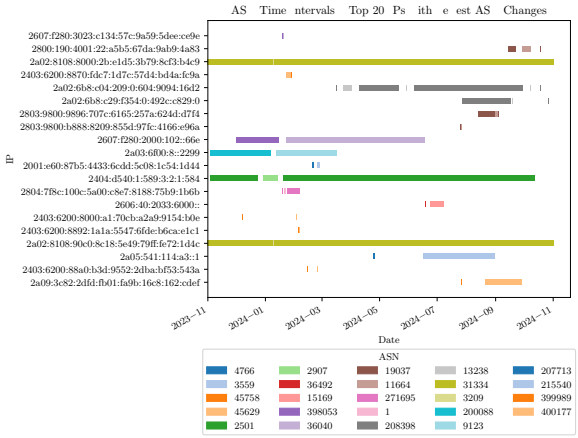


Figure 4: Top 20 IPv6 addresses with the fewest non-zero ASN changes

representative examples include:

`2001:500:121::30` (1)

`2001:502:f3ff::90` (2)

This behavior is most likely attributable to Multi-Origin AS (MOAS) prefixes, where the same IPv6 BGP prefix is announced by multiple ASNs simultaneously or over time [12]. This phenomenon serves as a strong indicator for dynamic, ephemeral infrastructure deployment patterns or as a way of load balancing on the AS-level (`2001:500:121::/48` or `2001:502:f3ff::/48` [13], [14]).

These addresses switch ASNs multiple times between 2023 and 2024, including changes to AS397239 (Vercara, LLC), AS397220 (Vercara, LLC), and AS396566 (VeriSign Global Registry Services) [15]. By often using the same ASNs, we can strongly assume that these addresses are used temporarily.

Conversely, Figure 4 presents IPv6 addresses with stable ASN assignments throughout the observation window. Examples include:

`2a02:8108:8000:2b:e1d5:3b79:8cf3:b4c9` (3)

`2404:d540:1:589:3:2:1:584` (4)

These findings emphasize the need for stability-aware selection when analyzing IPv6 topologies. Ignoring ASN volatility may lead to biased assumptions about the persistence or reachability of observed prefixes. Using diversified and stable sources, as recommended in [2], [3], [12], strengthen measurement accuracy.

4.2. Source distribution per ASN

During this chapter, we analyze the source distribution per ASN for the 10 ASNs with the most entries in the hitlist.

 doi: 10.2313/NET-2025-11-3_15

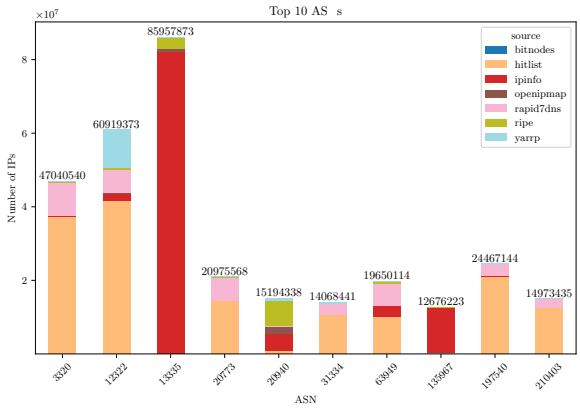


Figure 5: Top 10 ASNs with corresponding source contributions

4.2.1. Data preparation. To analyze the distribution of IPv6 addresses across ASNs by source, we extracted all records containing valid BGP prefix information and ASN. We grouped entries by BGP prefix, ASN, and source. Subsequently, we counted the number of IPs associated with each combination. If one IPv6 address had more than one source, it was counted for every source individually. Hence, we explicitly considered from which source an IPv6 address was imported. Finally, we identified the top 10 ASNs with the highest IP counts and visualized the source contributions, which are shown in a stacked bar plot.

4.2.2. Observations. Figure 5 reveals significant imbalances in source contributions across the top 10 ASNs by IP count, with entry counts ranging from approximately $1.2{\times}10^7$ to $8.6{\times}10^7$ across different ASNs.

ASN Distribution Imbalances: An essential observation from the distribution is the pronounced source dominance pattern. AS13335 (Cloudflare) shows the highest total count (approximately $8.6{\times}10^7$ entries) with dominant contributions from *ipinfo* and *hitlist* sources.

Representativeness Implications: The significant imbalances cause several concerns for IPv6 measurement studies — Geographic bias from single-source dominance may skew results toward specific regions, source-specific collection methodologies may introduce systematic temporal biases, and certain sources may preferentially capture specific infrastructure types (e.g., CDN nodes, residential addresses, or enterprise networks) [3], [16].

Certain ASNs, such as AS13335 (Cloudflare Inc.), AS3320 (Deutsche Telekom AG), and AS12322 (Free SAS) [15], are heavily dominated by data from one or two sources, e.g., *ipinfo* or *hitlist*. This skew can result in an overrepresentation of specific network regions, which may bias scanning results if only single-source hitlists are used.

Hence, incorporating multiple diverse sources is crucial to improve ASN diversity and global representativeness in IPv6 measurements [3].

5. Conclusion and Future Work

After analyzing ASN stability and source distribution per ASN, the evaluation of IPv6 hitlist entries indicates significant stability of IPv6 addresses and variability of BGP distribution. Sources like Bitnodes and Yarrp provide more stability, making them preferable for longitudinal studies. In contrast, while offering broader coverage, sources such as IPInfo and Rapid7DNS may introduce volatility and skewed ASN representation.

Recommendations for Future Measurements: Based on our analyses and the distribution patterns observed, we recommend multi-source integration to mitigate single-source biases, source-aware sampling with weighted contributions to achieve balanced ASN representation, stability-informed selection, and prioritizing stable sources for longitudinal studies. At the same time, volatile sources should be included for comprehensive coverage and geographical validation by cross-validating source contributions with known infrastructure distributions.

Future research should focus on developing methodologies to balance the trade-off between data diversity and stability. Additional metadata, such as geolocation and latency measurements, could enhance the utility of IPv6 hitlists. Moreover, exploring machine learning techniques to predict IPv6 address stability may improve the hitlist quality. One such approach is called Target Generation Algorithms (TGAs) [2].

References

[1] O. Gasser, J. Zirngibl, and L. Steger, “IPv6 Hitlist Service,” https://ipv6hitlist.github.io/, 2025, [Online; accessed 02-June-2025].

[2] L. Steger, L. Kuang, J. Zirngibl, G. Carle, and O. Gasser, “Target Acquired? Evaluating Target Generation Algorithms for IPv6,” in *Proceedings of the Network Traffic Measurement and Analysis Conference (TMA)*, Jun. 2023, best Paper Award.

[3] J. Gudehege, “Analysis of IPv6 Hitlist sources,” Academic Research Paper, 2024, [Provided by advisor; accessed 02-June-2025].

[4] G. Huston, “Exploring autonomous system numbers,” *The Internet Protocol Journal*, vol. 9, no. 1, pp. 2–23, 2006.

[5] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle, “Scanning the IPv6 Internet: Towards a Comprehensive Hitlist,” in *Proc. 8th Int. Workshop on Traffic Monitoring and Analysis*, Louvain-la-Neuve, Belgium, Apr. 2016. [Online]. Available: https://net.in.tum.de/pub/ipv6-hitlist/

[6] “Bitnodes ipv6 snapshot dataset,” https://bitnodes.io, 2021, [Online; accessed 13-June-2025].

[7] “Ipinfo.io ipv6 dataset,” https://ipinfo.io/data, 2024, [Online; accessed 13-June-2025].

[8] “Rapid7 forward dns dataset,” https://opendata.rapid7.com/sonar.fdns_v6/, 2022, [Online; accessed 13-June-2025].

[9] “Ripe atlas ipv6 measurement data,” https://atlas.ripe.net, 2024, [Online; accessed 13-June-2025].

[10] “Border Gateway Protocol (BGP),” RFC 1163, Jun. 1990. [Online]. Available: https://www.rfc-editor.org/info/rfc1163

[11] “What is BGP hijacking?” https://www.cloudflare.com/learning/security/glossary/bgp-hijacking/, [Online; accessed 14-June-2025].

[12] K. Z. Sediqi, A. Feldmann, and O. Gasser, “Live long and prosper: Analyzing long-lived moas prefixes in bgp,” 2023, [Online; accessed 05-June-2025].

[13] “Internet health report,” https://www.ihr.live/en/prefix/2001:500:121::/48, [Online; accessed 18-June-2025].

[14] “Internet health report,” https://www.ihr.live/en/prefix/2001:502:f3ff::/48, [Online; accessed 18-June-2025].

[15] “AS2org API Doc,” https://api.data.caida.org/as2org/v1/doc, 2025, [Online; accessed 15-June-2025].

[16] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, “Towards ip geolocation using delay and topology measurements,” in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006, pp. 71–84.

Modeling the Architecture of QUIC Implementations with rustviz

Leopold Jofer, Daniel Petri*, Marcel Kempf*,
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: leo.jofer@tum.de, petriroc@net.in.tum.de, kempfm@net.in.tum.de

***Abstract*—QUIC implementations differ vastly in their support for the different QUIC standards and extension-RFCs. A lack of documentation and an architectural overview make it difficult to judge which library supports what, how extensible they are and how they are structured. This paper examines two QUIC implementations: tquic by Tencent and neqo, developed by Mozilla for the Firefox web browser. Utilizing our custom tool, `rustviz`, we break down their architecture and subsequently model their design and intended usage via the C4 software model. From that, we infer their RFC support and extensibility and try to find potential anti-features or deterrents to including them in a project. Our findings were added to the QUIC Explorer, which makes it easy for developers to filter and assess the suitability of QUIC libraries.**

***Index Terms*—QUIC, QUIC Explorer, architecture, software, tquic, neqo, Rust, C4 Model**

1. Introduction

With the advent of the Web as a dominant platform in our lives, the speed, latency and reliability of web transport protocols have gained an increasingly important role. Traditional network communication, such as HTTP or SSH, use TCP for transmitting data. However, TCP was designed for the Internet of a different time and therefore has significant downsides and problems. Head-of-line blocking, connection establishment overhead and poor performance are just some of the baggage that TCP brings when using it [1]. As such, a need for a more efficient, faster and vertically integrated network stack has arisen. The next stage of this evolution is marked by QUIC, a new TCP alternative that aims to surpass these restrictions thanks to a more advanced design [2].

QUIC, a comparatively new protocol, solves and mitigates many of these problems, by integrating encryption, multiplexing streams and establishing connections in a more efficient manner. It is a user-space protocol, building on the properties of UDP, with various congestion control algorithms, that can detect packet loss across streams without blocking. On top of QUIC, HTTP/3 was designed as an efficient application layer protocol that aims to replace and improve HTTP/1.1 and HTTP/2 with better header compression and without head-of-line blocking [3].

However, when using QUIC in their application, developers may face challenges in finding a suitable implementation that fits their needs. Information about which libraries exist, what features they support and how standard-compliant they are is scattered, scarce and often out of date.

1.1. Adoption Landscape

Google, Cloudflare, Apple, Tencent and many more each have their own QUIC implementations with varying support for the different iterations and RFCs related to QUIC [4], [5]. As a developer, finding the right library or tool for your use case can therefore be quite hard, as public information is scattered across different websites, tables, repositories and documentations. Sometimes, the information given about what an implementation supports is quite scarce. Finding out what really is in the codebase is challenging.

1.2. Scope of the paper

In this paper, we cover two implementations, tquic by Tencent and neqo by Mozilla, which is used in firefox. We use our newly developed tool, called `rustviz`, to dive deeper into the codebase and find anomalies and features regarding the implementations. Our findings will be added to the QUIC explorer.

2. Background

QUIC is not a singular standard or RFC. Rather, it is composed out of multiple RFCs, extensions, and drafts that describe different parts and additions to the protocol. For example, HTTP/3's header compression with QPACK is specified in a separate RFC from the underlying QUIC protocol [6]. Some parts of the QUIC protocol itself (like congestion control) are described by the RFC, yet their concrete algorithm or implementation is not. Libraries can evoke security and compatibility concerns, like the choice of the TLS library used - which is a frequent target for attacks [7] - or implementation correctness.

2.1. QUIC Explorer

There exist many different QUIC implementations with different characteristics, maturity levels, and use cases. Judging which one is right for a certain use case can be a fairly complex problem. The QUIC Explorer [4] provides a central repository that enables users to filter QUIC implementations by feature and language and provides them with a simple overview.

 doi: 10.2313/NET-2025-11-3_16

2.2. C4 model

In this paper, we use the C4 modeling approach to examine the architecture of the libraries that are considered. The C4 model defines four abstraction layers:

- the *context* in which a software system exists,
- the *containers* of which the aforementioned software system is composed,
- the *components* of such a container, e.g. code modules
- the actual *code* of the application [8] [9]

C4 helps to frame the context in which a library is being used, as well as the inner workings of the library itself. Since we are examining the architecture of libraries — not entire software systems — in this paper, the relevant layers for this paper are the container layer and the component layer. Modeling the actual code would mostly yield low-level implementation details, which are not relevant for us. Instead, small peeks into the codebase when necessary are deemed adequate.

2.3. Encrypted Client Hello (ECH)

ECH is a mechanism to hide the site the user is visiting. When a client connects to a server with TLS, the Server Name Identification (SNI) as well as other TLS extensions are unencrypted, allowing potential adversaries to uncover which server the client is connecting to. ECH uses a public key (most commonly from a DNS record) to encrypt the SNI, therefore making it substantially more difficult to reveal what web address a client is visiting [10].Hence, it yields a considerable privacy benefit. The main threat actors that this protocol protects against are network observers and ISPs. It is however only effective when multiple websites use the same ECH origin. If every web origin/domain had its own ECH address, that address would still uniquely identify a visitor.

QUIC does not create connections layered over a TLS connection, but instead integrates it directly into the protocol. The initial handshake also establishes the encryption as well as the data connection. Encrypted Client Hello does not require a separate TLS layer to function. Instead, the QUIC-TLS handshake (`CRYPTO` packets, see [11]) is encrypted and wrapped in a so called ClientHelloOuter-Message. This outer message contains a SNI that is generic enough to be useless to an adversary. The receiving QUIC server decrypts this message, thus uncovering the actual destination of the packet and redirecting the stream.

2.4. Sans-I/O

The Rust package ecosystem is split into synchronous (using the standard library and threads) and asynchronous packages (using a third-party executor for the futures). The choice of executor furthers this fragmentation, as every executor has a slightly different I/O interface. For example, a library written for `mio`, Tokios I/O implementation is not compatible with `async-std`, another async executor [12].

To circumvent this issue, Rust libraries for protocols typically resort to feature flags to allow for switching executor. In other cases, they just pick one. Sans-I/O libraries instead expose a set of handles that hook into I/O for the caller to implement, like e.g. quiche by Cloudflare [13]. Another advantage of this approach is that it future-proofs the library to a certain extent and makes it easier to use in all kinds of environments, like embedded or IoT (Internet of Things) devices.

3. Design and Methodology

In order to not exceed the scope of the paper, go into meaningful depth and be able to make a sensible comparison, we chose the following two QUIC implementations: *neqo* by Mozilla and *tquic* by Tencent.

3.0.1. neqo by Mozilla. neqo is Mozilla's QUIC implementation, written in the Rust programming language [14]. Its primarily used in Firefox, tightly integrated as a part of the greater — not to be confused by name — `Necko` network stack [15]. In fact, neqo's primary purpose does not entail being used outside of the Firefox/`Necko` ecosystem. It is not published on crates.io, the Rust package registry. As a result, neqo on its own has not seen wider adoption beyond that scope. But, due to its use in Firefox, which has around 155 million monthly active users [16], we nonetheless consider it relevant to analyze.

3.0.2. tquic by Tencent. *tquic* is a performant QUIC library developed by Tencent, also written in Rust. Its primary use is in the Tencent cloud [17]. In contrast to neqo, tquic is being developed as a standalone library that can be integrated into various codebases and has also been released as a crate on crates.io [18]. Since existing public information on the library is sparse even though it is used quite widely, it is the second library that we analyze.

3.1. Diagram Creation

Creating component-level diagrams by hand is tedious and can be error-prone. It is easy to miss or misunderstand a part of the codebase and create a wrong schematic. We use a custom-made tool called `rustviz`[1] to generate these diagrams automatically. To reduce noise, our tool can filter it's output so that test suites and binaries can be omitted.

In addition, to showcase how the projects fit into the greater scope of the technical landscape and how they are used, we hand-create C4 diagrams on a container level. In select cases the source code of the implementation is examined as well.

3.2. Analyzed Features

Since our goal is to add our findings to the QUIC Explorer, the key features we examine partially stem from its list of features. Below is a list of the features that we look for in this paper and why.

- Encrypted Client Hello Support, since it is not yet part of the QUIC Explorer and relevant for privacy requirements.
- TLS Library choice, since old, untested or wrong crypto poses a huge threat to security.

1. github.com/leopoldlabs/rustviz

 doi: 10.2313/NET-2025-11-3_16

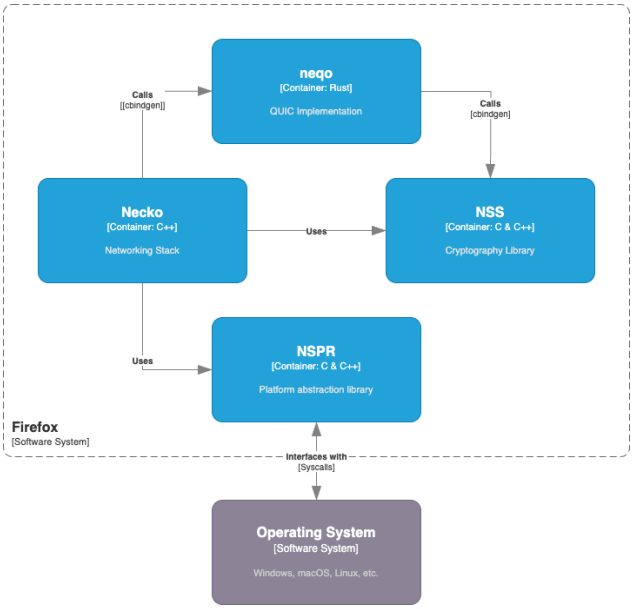


Figure 1: container view of neqos usage in Firefox

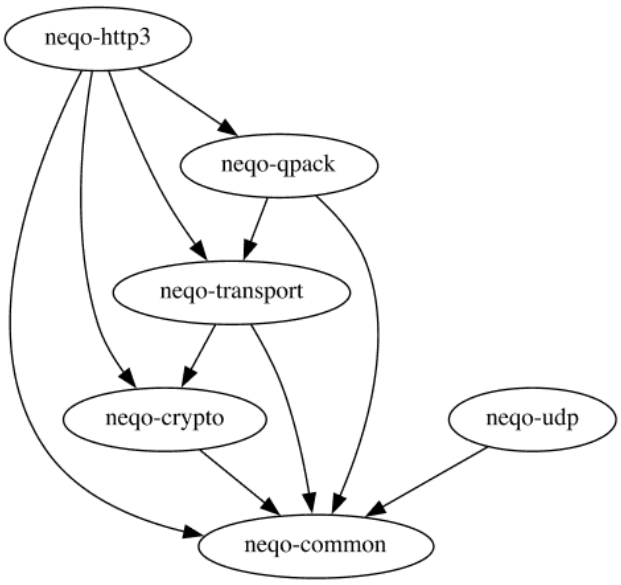


Figure 2: the individual crates that make up the neqo library. generated with `rustviz --filter fuzz,test-fixture,neqo-bin`, visualized with graphviz

4. Implementation

`rustviz` can operate on two abstraction layers of a Rust project. It is able to infer the dependencies between different crates in a cargo workspace by parsing and discovering the corresponding `Cargo.toml` files. From this workspace, an internal graph is built. Also, it can visualize the relationships between different modules, so it tracks which module imports which. This is done by generating documentation for a given crate with `rustdoc`. This documentation is then parsed and traversed with a breadth-first search in order to build a simplified module graph. In both cases, this Graph can then be output in textual form as C4-Diagrams or GraphViz-Diagrams. These can then be viewed with the MermaidJS Diagram Viewer or a GraphViz visualizer. Unnecessary modules can be omitted with the `--filter` option. This process uncovers architectural patterns and gives an easy overview of how the codebase is structured, making it easier to uncover support for different RFCs and standards. This provides a good entry point for exploring the codebase. The interactions between the various components of the codebase also unveil insights into its modularity and extensibility.

5. Evaluation

5.1. neqo

neqo's unique place in the Firefox codebase means that it depends on packages that are fairly unusual for the Rust ecosystem. Figure 1 shows that if the browser wants to establish a QUIC connection, Necko calls neqo via `cbindgen`. The sockets are handled by `NSPR`, a library that provides a cross-platform abstraction for sockets, files and other primitives [15], [14]. It does not implement I/O itself.

As visible in Figure 1, neqo uses the *NSS* (Network Security Services) library for cryptographic operations, the well-maintained library that is also used elsewhere in Firefox. It does so with C bindings, generated by `cbindgen`. Since there is no significant abstraction layer over NSS, swapping out the cryptography library is not easy in neqo and would require drastic code changes. neqo's NSS dependency and lack of publicly available crate makes it harder to integrate into regular Rust projects, as the build process becomes more complicated. NSS and NSPR are external C/C++ dynamic libraries [19] that are often out of date on older distributions. Consumers of neqo can build their own version of the NSS and NSPR. This process is outlined in the neqo documentation [14].

Digging deeper, Figure 2 shows the internal structure of the library and how different features are compartmentalized into various crates. Common abstractions, utilities and other shared pieces of code, that are used throughout the library, such as QLOG support, are inside `neqo_common`. On the other hand, higher-level abstractions of QUIC such as streams, network methods and UDP datagram parsing are all concentrated inside `neqo_transport`.

In order to provide the necessary cryptographic primitives for QUIC and to interface with the cryptographic library `NSS`, neqo provides C bindings. The definitions for these bindings are defined in `neqo_crypto`. HTTP/3 support can be layered on top using `neqo_http3`, which uses the QPACK decoder and encoder from the separate `neqo_qpack` crate. neqo has full support for client-side HTTP/3 and experimental support for server-side HTTP/3, which is not meant for production use but rather client-code testing [14, neqo-http3/src/lib.rs:21-23]. Taking a closer look at the `neqo_transport` crate reveals that neqo has support for ECH. The `Connection` struct contains methods to enable ECH on the server and client side respectively, as well as methods to retrieve the current ECH configuration of the connection.

5.2. tquic

tquic is being developed for general usage in Rust projects on the server side and is also a sans-io QUIC

 doi: 10.2313/NET-2025-11-3_16

library [17]. In contrast to neqo, it is not divided into different crates with different functionalities, instead it is only organized with Rust modules. It hooks itself into a server (or client) as a user space library. When a connection reaches a server, the kernel of the server forwards those UDP packets to the respective web server. The web server then calls the delegated handlers provided by tquic, so that the task of parsing them and handling the connection is delegated from the main program to tquic, the library. Just like neqo, tquic also contains an HTTP/3 and a QPACK implementation, but they are hidden behind the `h3` feature flag and contained inside of the `h3` module. We are going to update this feature on the QUIC Explorer for both.

tquic uses Google's BoringSSL, a mature crypto library forked from openSSL that is also used in Google Chrome. It does so in conjunction with the `ring` crate, which has been explicitly marked as an experiment in it's README [20]. Presumably this was done to protect the author of any liability. Yet, the seeming risk of this dependency is unfortunately not mentioned anywhere in the tquic docs. tquic is not presented as experiment either. An explanation would be sensible and helpful in this context. It does not contain an ECH configuration. There is no mention of it in the codebase and official documentation.

6. Conclusion and future work

Both tquic and neqo are solid, architecturally sound choices when picking a QUIC library and they will stay so for the foreseeable future due to their large supporters.

Due to the QUIC specification and extensions being so expansive, the number of elements to consider when picking a QUIC library is substantial. Finding out the extent of what RFC and feature is actually implemented remains a tedious task. While the QUIC Explorer mostly mitigates this and provides a nice overview, it still needs to be kept up-to-date and expanded as well. Library developers should put more emphasis on compatibility documentation and provide more usage examples.

6.1. Future outlook

Code comments and docs can state that a feature is supported, but whether it is implemented lacklusterly or properly can hardly be inferred from that.

6.1.1. Spec compliance. An example for this kind of testing is the QUIC tests developed by Microsoft for their protocol testing language, Ivy. The tests can verify that a implementation is spec compliant.

The tquic developers claim to have used these tests for their library, however their test environment and test harness version were not published. Ivy itself is unmaintained [21], just like the tests that were published for QUIC within the ivy repository. Not only would the language require updates, the tests would also have to be rewritten in order to be compliant with the RFC, not just the old drafts.

6.1.2. Automated Testing. New versions of libraries can potentially create unintended regressions. While both neqo and tquic come with Unit tests, full integration tests, potentially using ivy have the potential to catch more errors. A public dashboard could show the percentage of passing tests, and how the resulting spec compliance evolves over time.

6.1.3. Better generation of C4 Diagrams. This is regarding the generation of code-level diagrams that showcase the relationships between functions, structs, traits, etc. This is presumably also useful in a broader scope than QUIC library analysis, for example as an educational help for Rust learners and teachers.

7. Related Work

There are a few similar tools to rustviz.

- `cargo tree` in the Rust Toolchain outputs a tree visualization of a crate's dependency graph [22].
- `cargo-modules`, a separate tool, shows a tree overview of a crate's modules [23].

References

[1] M. contributors, "Head-of-line blocking - Glossary," 2025, [Online; accessed 11-Aug-2025]. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Head_of_line_blocking

[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] G. Perna, M. Trevisan, D. Giordano, and I. Drago, "A first look at http/3 adoption and performance," *Computer Communications*, vol. 187, pp. 115–124, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422000421

[4] M. Kempf, "QUIC Explorer," https://quic-explorer.net, accessed: 2025-08-12 (Commit 105a3a4). [Online]. Available: https://quic-explorer.net

[5] I. Q. W. Group, "implementations.md - quic working group." [Online]. Available: https://github.com/quicwg/quicwg.github.io/blob/main/implementations.md

[6] C. B. Krasic, M. Bishop, and A. Frindell, "QPACK: Field Compression for HTTP/3," RFC 9204, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9204

[7] "Vulnerabilites - OpenSSL," 2025, [Onlin,e accessed 12-Aug-2025]. [Online]. Available: https://openssl-library.org/news/vulnerabilities/index.html

[8] S. Brown, "The C4 Model for Software Architecture," 2018, [Online; accessed 21-June-2025]. [Online]. Available: https://www.infoq.com/articles/C4-architecture-model/

[9] ——, "The C4 model for visualising software architecture," 2019, [Online; accessed 21-June-2025]. [Online]. Available: https://c4model.com/

[10] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, "TLS Encrypted Client Hello," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-esni-25, Jun. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-esni/25/

[11] M. Thomson and S. Turner, "Using TLS to Secure QUIC," RFC 9001, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9001

[12] M. Endler, "The State of Async Rust," 2025, [Online; accessed 20-Aug-2025]. [Online]. Available: https://corrode.dev/blog/async/

[13] A. G. (Cloudflare), "Enjoy a slice of QUIC, and Rust!" 2019, [Online; accessed 21-June-2025]. [Online]. Available: https://blog.cloudflare.com/enjoy-a-slice-of-quic-and-rust/

[14] Mozilla, "Neqo, the Mozilla Firefox implementation of QUIC in Rust," 2025, [Online; accessed 21-June-2025]. [Online]. Available: https://github.com/mozilla/neqo/tree/main

[15] ——, “Firefox Source Docs,” 2025, [Online; accessed 15-Aug-2025]. [Online]. Available: https://firefox-source-docs.mozilla.org/networking/http/http3.html

[16] ——, “Firefox User Activity Dashboard,” 2025, [Online; accessed 15-Aug-2025]. [Online]. Available: https://data.firefox.com/dashboard/user-activity

[17] Tencent, “tquic Documentation,” 2025, [Online; accessed 15-Aug-2025]. [Online]. Available: https://tquic.net/docs/intro/

[18] “tquic - Crates.io,” 2025, [Online; accessed 15-Aug-2025]. [Online]. Available: https://crates.io/crates/tquic

[19] “nspr: Summary - Mozilla Mercurial,” 2025, [Online; accessed 21-June-2025]. [Online]. Available: https://hg-edge.mozilla.org/projects/nspr

[20] B. Smith, “ring,” 2025, [Online; accessed 21-June-2025]. [Online]. Available: https://github.com/briansmith/ring

[21] Microsoft, “IVy is a research tool intended to allow interactive development of protocols and their proofs of correctness and to provide a platform for developing and experimenting with automated proof techniques.” 2020, [Online; accessed 21-June-2025]. [Online]. Available: https://github.com/microsoft/ivy

[22] “cargo-tree(1),” 2025, [Online; accessed 24-Aug-2025]. [Online]. Available: https://doc.rust-lang.org/cargo/commands/cargo-tree.html

[23] regexident, “cargo-modules,” 2025, [Online; accessed 24-Aug-2025]. [Online]. Available: https://github.com/regexident/cargo-modules

Autoencoder-Based Anomaly Detection in Networks

Yavuzalp Kaplan, Johannes Späth*, Max Helm*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: yavuzalp.kaplan@tum.de, spaethj@net.in.tum.de, helm@net.in.tum.de

***Abstract*—Modern computer networks produce large amounts of traffic, and unusual patterns in this traffic can signal failures, misconfigurations, or security threats. Autoencoders are used for anomaly detection, as they capture normal patterns and identify anomalies through reconstruction error. In this paper, we review the use of autoencoder-based models for network anomaly detection. We discuss preprocessing techniques, loss functions, and hyperparameters, and we summarize experimental results from previous studies on datasets such as NSL-KDD, UNSW-NB15, KDDCUP'99, and InSDN. The results show that while autoencoders achieve promising detection performance, outcomes vary depending on the dataset and parameter choices.**

***Index Terms*—Autoencoder, Anomaly Detection, Preprocessing, Activation Functions, Loss Functions**

1. Introduction

In October 2016, a large Distributed Denial of Service (DDoS) attack caused major problems across the internet [1]. Websites like Twitter, Netflix, Reddit and many others became unreachable for several hours. The attack was executed using a botnet called Mirai, which had infected many Internet of Things (IoT) devices such as digital cameras and DVR players. These devices, which normally send very little traffic, began to generate large amounts of repeated Domain Name System (DNS) requests to Dyn, a major DNS provider. Traditional intrusion detection systems (IDS), which usually detect known attack patterns, failed to notice the early signs of this attack because the traffic did not match any known signatures. But this kind of abnormal network behavior, especially many quiet devices sending a lot of similar traffic at the same time, could have been detected earlier using machine learning methods like autoencoders.

Attacks like the Mirai botnet show that network security is becoming more important. Today, cyberattacks happen more often and are harder to detect [2]. Attacks such as DoS, port scans, data theft can cause serious damage, e. g., by disrupting critical services, mapping network vulnerabilities, or exfiltrating sensitive information. Traditional IDS methods use signatures to compare traffic to known attack types. This helps with known threats, but it does not work well for new or unknown attacks, which are called zero-day attacks [3].

Anomaly detection offers a different approach. Instead of relying on predefined attack signatures, it tries to find unusual behavior in the network. This makes it useful for detecting new and unknown threats.

In this paper, we explain how autoencoders can be used for network anomaly detection. We describe the preprocessing steps, how the autoencoder model works and how reconstruction loss helps to detect anomalies. We also discuss the types of attacks that autoencoder-based models can find and look at results from previous studies.

2. Background

Autoencoders are a type of neural network that are used for anomaly detection. They work by learning normal patterns in the data. An autoencoder has three parts: an encoder, which compresses the input into a smaller representation, a latent space that stores this compressed form and a decoder, which reconstructs the original data from it. The model is trained to minimize the difference between the input and the output. If the model sees something very different from what it has learned, the difference, or reconstruction loss, becomes larger, which signals a possible anomaly.

However, network data often contains different types of features. Some are strings, like `protocol_type` and `flag`, while others are numbers, like `src_bytes` and `dst_bytes`. Some are binary, like `is_guest_login`. These different types of data cannot be used directly in an autoencoder. Preprocessing steps are needed to convert them into numbers. One common method is *One-Hot-Encoding*, which turns categorical features into binary vectors. Datasets like KDDCUP'99 and NSL-KDD provide examples of such features and are often used in network anomaly detection research [4].

Several previous studies [3], [5]–[14] have applied autoencoder-based models to network anomaly detection tasks, using public datasets such as NSL-KDD, KDDCUP'99, UNSW-NB15, InSDN, etc. These works show that autoencoders can successfully detect a variety of attacks, including DoS, Probe, R2L, U2R, Backdoors, Exploits, Fuzzers and more. However, the performance of these models depends on many factors, such as the choice of hyperparameters, the preprocessing techniques and the characteristics of the dataset [15].

3. Autoencoders

An autoencoder is a type of machine learning model based on a neural network architecture. Its main goal is to reconstruct given input. It is a structure that learns to

 doi: 10.2313/NET-2025-11-3_17

compress the input data into a lower-dimensional space, and then learns to make as faithful an approximation of the input data as possible from this compressed representation. In the process, the model learns the key features and patterns of the data, filtering out unnecessary information and noise.

The structure of an autoencoder usually consists of three main parts: encoder, latent space and decoder. The encoder part takes the input data such as an image, an audio file or a network connection log and transforms it into a smaller and denser representation, the latent space, also known as bottleneck. This latent space is a compressed and informative representation of the input data. The decoder takes the compressed representation and tries again to produce a prediction of the original input data. Figure 1 illustrates the simple architecture of an autoencoder, including the encoder, latent space and decoder components [16].

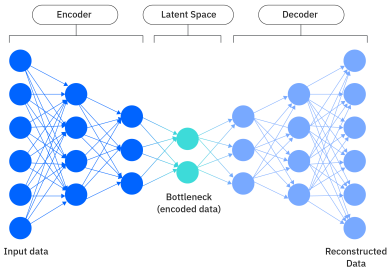


Figure 1: Simple Autoencoder Architecture

3.1. Learning Techniques

Within the concept of machine learning, there are different learning techniques. These are divided into supervised, semi-supervised and unsupervised learning. Supervised learning is a type of learning where each input data has a corresponding label. The model learns the relationship between the input data and the label, and aims to predict the label for new data. Semi-supervised learning is an approach that combines labeled and unlabeled data to improve learning accuracy when labeled data is scarce.

However, these methods fail to detect zero-day attacks that can occur daily in networks [3]. For this reason, an unsupervised learning technique is applied to the autoencoder for anomaly detection in networks. In this technique, the input data is not labeled, but is learned from the data itself. The model tries to discover patterns, relationships, groupings or hidden structures in the data. Since there are no labels, the model learns entirely from the distribution of the data and tries to discover normal and abnormal states or clusters within the data.

3.2. Activation Functions, Encoder and Decoder

Activation functions enable the network to build a nonlinear structure and help it learn complex data patterns. Which activation function is used for encoding and decoding the data directly affects the success and the performance of the model. Therefore, this function, which determines what and how the model learns, should be carefully selected before training [15]. If the activation function is not chosen correctly, the model may struggle to learn complex relationships and may not produce the expected results. Commonly used activation functions are sigmoid, tanh, rectified linear unit (ReLU), and scaled exponential linear unit (SELU), which are defined in Eq. (1) – (4).

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1}$$

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2}$$

$$\mathrm{ReLU}(x) = \max(0, x) \tag{3}$$

$$\mathrm{SELU}(x) \begin{cases} \lambda x & \text{if } x > 0 \\ \lambda\alpha(e^{x}-1) & \text{if } x \leq 0 \end{cases} \tag{4}$$

We will now focus on how the encoder and decoder work with activation functions. An encoder is a neural network that compresses the input data into a lower-dimensional latent space, i. e., it maps a high-dimensional input vector X into a smaller latent vector Z.

$$Z = \sigma(WX + b) \tag{5}$$

Here, Z is the compressed representation in latent space, σ is the activation function, W is the weight matrix and b is the bias vector. From this latent representation produced by the encoder, the decoder tries to produce a prediction of the input data. The goal is to output $\hat{X}$ as close as possible to the input X [5].

$$\hat{X} = \sigma(W'Z + b') \tag{6}$$

4. Anomaly Detection

As discussed in Section 3, the main purpose of autoencoders is to compress the input data into a smaller latent space and then reconstruct the original data from this representation. In this process, the success of the model is measured by the similarity between the input data and the reconstructed data. One of the most common methods used to measure this similarity is reconstruction loss [15].

4.1. Loss Functions

Reconstruction loss is a numerical measure of how accurately the model can reproduce a data given as input. Functions such as Mean Squared Error (MSE) or Binary Cross-Entropy (BCE) are usually preferred [15]. MSE is a common choice, especially when working with numerical data. MSE takes the average of the squares of the differences between the input and model output and is defined as follows:

$$L_{\mathrm{MSE}}(X, \hat{X}) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2 \tag{7}$$

Where n is the number of features, X_i is the i-th feature in the original input and $\hat{X}_i$ is the i-th feature in the

 doi: 10.2313/NET-2025-11-3_17

reconstruction output. Alternatively the BCE loss function can also be applied. This function, which is used when the input data is binary (0 or 1), calculates the binary cross-entropy loss difference between the expected and actual output.

$$L_{\text{BCE}}(X, \hat{X}) = -\frac{1}{n}\sum_{i=1}^{n}\Big(X_i \cdot \log(\hat{X}_i) + (1 - X_i) \cdot \log(1 - \hat{X}_i)\Big) \quad (8)$$

The loss function should be chosen carefully so that the model can learn correctly. A loss function that is not appropriate for the data type or the problem can negatively affect the model's learning process and cause inaccurate results. For example, using BCE loss in numerical data can lead to the model's outputs being stuck in a wrong range. Hence, the loss function must match the data type and the model's purpose.

4.2. Preprocessing

Network logs often contain various types of features, such as `protocol_type`, `src_bytes`, `dst_bytes`, `flag`, `is_guest_login`, `num_access_files` etc. Public datasets like KDDCUP'99, NSL-KDD, UNSW-NB15 include such examples [4]. However, these raw data features cannot be directly fed into an autoencoder model, because some of them are strings like `protocol_type`, some are numeric like `src_bytes`, some are binary like `is_guest_login`. Autoencoders only use numerical data for training and testing [17]. To make them suitable for training, a preprocessing step is necessary. One common approach is to use *One-Hot-Encoding*, which converts categorical and string-based features into numerical vectors. This process transforms all data into a fully numerical format, allowing the autoencoder to process the input effectively.

For example, the feature `protocol_type` has three distinct attributes: `tcp`, `udp` and `icmp`, each of which is encoded into a three-dimensional binary vector, such as [1,0,0], [0,1,0] and [0,0,1], respectively. In other words, the single feature `protocol_type` is encoded into three features by one-hot-encoding [6].

After preprocessing, the original data is transformed into a fully numerical vector. A simplified example of such a vector is shown in (9).

$$x = [0.0,\ 1.0,\ 0.78,\ 0.54,\ 0.12,\ 0.0,\ 1.0,\ 0.33,\ \ldots] \quad (9)$$

4.3. Security

Anomaly detection models are important tools for protecting computer networks. They find unusual activities that are different from normal network behaviour. Autoencoder-based models are especially useful because they learn what normal traffic looks like and detect anything that seems unusual, even if the attack is new or unknown. Unlike signature-based systems, they do not need labeled attack data and can work without knowing the exact type of attack in advance.

These models detect many different kinds of network attacks by checking how much a new traffic pattern differs from what was learned during training. The most common attack types that anomaly detection systems aim to protect against include Denial of Service (DoS), Probe, Remote to Local (R2L), User to Root (U2R), Backdoors, Exploits, Fuzzers, Generic, Portscans, Reconnaissance, Shellcode, and Worms [18].

Autoencoder-based models detect these attacks by spotting patterns that do not match normal traffic, helping to protect networks from a variety of threats. The NSL-KDD and KDDCUP'99 datasets are commonly used in tests for detecting DoS, Probe, R2L and U2R attacks, while the UNSW-NB15 dataset is used to evaluate models against attacks such as Backdoors, DoS, Exploits, Fuzzers, Generic, Portscans, Reconnaissance, Shellcode and Worms [18].

5. Evaluation

Evaluating the performance of an anomaly detection system is essential to understand how efficiently it can detect unusual patterns in network traffic. This section introduces the commonly used performance metrics in anomaly detection and discusses the factors that affect model performance, such as the choice of loss function, network architecture, and training settings.

5.1. Evaluation Metrics

To evaluate the performance of autoencoder-based anomaly detection models, four common metrics are used: *precison*, *recall*, *f1-score*, and *accuracy*. These metrics are computed from four basic measures, true positive (TP), false positive (FP), true negative (TN), false negative (FN):

- TP, defined as the number of anomalies correctly identified as anomalies.
- FP, referring to the number of normal samples incorrectly identified as anomalies.
- TN, defined as the number of normal samples correctly identified as normal.
- FN, referring to the number of anomalies incorrectly identified as normal.

The metrics for performance measurement in anomaly detection systems are as follows:

Precision: Indicates the proportion of samples predicted as anomalies that are actually anomalies.

$$\textit{Precision} = \frac{TP}{TP + FP} \quad (10)$$

Recall (Sensitivity): Measures the proportion of actual anomalies correctly identified by the model.

$$\textit{Recall} = \frac{TP}{TP + FN} \quad (11)$$

F1-Score: The harmonic mean of *Precision* and *Recall*.

$$F_1 = 2 \cdot \frac{\textit{Precision} \cdot \textit{Recall}}{\textit{Precision} + \textit{Recall}} \quad (12)$$

Accuracy: The overall correct prediction rate of the model.

$$\textit{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (13)$$

 doi: 10.2313/NET-2025-11-3_17

5.2. Hyperparameters

The performance of autoencoder-based anomaly detection systems depends on various factors and hyperparameters [15], including:

- **Number of hidden layers:** More layers can capture complex patterns but may also lead to overfitting, where the model memorizes the training data and fails to generalize.
- **Number of neurons per layer:** Controls the model's ability to learn patterns; more neurons improve learning but increase computational cost.
- **Size of the latent space:** A smaller space forces stronger compression and may help anomaly detection, but too small a space can cause information loss.
- **Activation function:** The choice of activation functions affects the model's ability to capture nonlinear relationships.
- **Loss function:** Defines the optimization target; MSE and BCE are the most common choices.
- **Learning rate:** Sets the update step size; high rates can cause instability, while low rates may slow convergence.
- **Number of epochs:** The number of training cycles; too few may lead underfitting, where the model fails to capture patterns in the data, while too many may result in overfitting.
- **Batch size:** Number of samples used per training step; large batches provide more stable updates, while small batches add variability but may improve generalization.

These hyperparameters must be carefully tuned to balance model complexity, training efficiency and detection performance.

5.3. Experimental Results

Anomaly detection tasks have been extensively studied in the literature, and several works have demonstrated the potential of autoencoder-based models in identifying anomalous network traffic. This section provides a summary of experimental results reported in previous studies, highlighting the performance of autoencoder architectures across different datasets and metrics.

Table 1 summarizes the results on the NSL-KDD dataset. Study [3] tested a memory-augmented deep autoencoder (MemAE) to improve anomaly detection by using a memory mechanism. Study [7] applied a sparse autoencoder to learn sparse features for better detection. Study [6] chose the Mean Absolute Error (MAE) as loss function for better performance, while Study [8] tested a denoising autoencoder (DAE) to make the model more robust to noise. Additionally, Study [9] applied a Fully Connected Network (FCN) and a Shallow Long Short-Term Memory network (SLSTM), achieving strong results in terms of precision and recall.

Table 2 covers the UNSW-NB15 dataset. Study [3] tested MemAE again on this dataset. Additionally, Study [10] used an autoencoder for different attacking scenarios such as DoS, Backdoors, Shellcode, etc. Study [11] evaluated other machine learning models on this dataset, including Decision Tree (DT), K-Nearest Neighbours (KNN), Naïve Bayes (NB), and Random Forest (RF). Furthermore, Study [12] explored the use of more complex models such as Stacked Sparse Autoencoder (SSAE), Variational LSTM (VLSTM), and CNN-LSTM, highlighting the diversity of approaches in handling anomaly detection tasks.

Table 3 presents results for the KDDCUP dataset. For example, Study [13] used an autoencoder model on this dataset. Study [14] used the Deep Autoencoding Gaussian Mixture Model (DAGMM) on the same dataset, combining clustering and autoencoder reconstruction errors to detect anomalies.

Table 4 shows results on the InSDN dataset. Study [5] applied both a One-Class Support Vector Machine (OC-SVM) and a Long Short Term Memory autoencoder (LSTM) on this dataset.

TABLE 1: Evaluation Metrics on NSL-KDD Dataset

Model	Accuracy	Precision	Recall	F1-Score
MemAE [3]	0.8951	0.9062	0.8951	0.8993
Sparse AE [7]	0.8839	0.8544	0.9595	0.904
AE [6]	**0.9061**	0.8683	0.9843	0.9226
AE [8]	0.8828	0.9123	0.8786	0.8951
DAE [8]	0.8865	0.9648	0.8308	0.8928
FCN [9]	-	**0.997**	0.874	0.931
SLSTM [9]	-	0.983	**0.996**	**0.99**

TABLE 2: Evaluation Metrics on UNSW-NB15 Dataset

Model	Accuracy	Precision	Recall	F1-Score
MemAE [3]	0.853	**0.8774**	0.853	0.8526
AE [10]	**0.9516**	-	-	-
DT [11]	0.8455	0.864	0.846	0.8549
KNN [11]	0.8449	0.855	0.845	0.85
NB [11]	0.7639	0.782	0.764	0.7729
RF [11]	0.8363	0.869	0.836	0.8522
SSAE [12]	-	0.731	0.963	0.832
VLSTM [12]	-	0.86	**0.978**	**0.907**
CNN-LSTM [12]	-	0.801	0.956	0.872

TABLE 3: Evaluation Metrics on KDDCUP Dataset

Model	Accuracy	Precision	Recall	F1-Score
AE [13]	0.9282	0.9236	**0.9923**	**0.96**
DAGMM [14]	-	**0.9297**	0.9442	0.9369

TABLE 4: Evaluation Metrics on InSDN Dataset

Model	Accuracy	Precision	Recall	F1-Score
OC-SVM [5]	0.875	0.89	**0.93**	0.91
LSTM [5]	**0.905**	**0.93**	**0.93**	**0.93**

6. Conclusion and Future Work

Autoencoder-based models are a useful tool for detecting anomalies in network traffic by learning normal patterns and identifying unusual data points. These models can help identify various types of attacks, including both

 doi: 10.2313/NET-2025-11-3_17

known and unknown threats. However, the performance of an autoencoder model depends on careful preprocessing, the choice of loss function, and tuning of hyperparameters.

The experimental results summarized in Tables 1 – 4 show that no single autoencoder-based model consistently outperforms others across all datasets and metrics. For example, on the NSL-KDD dataset, FCN achieved the best precision, while SLSTM reached the highest recall. On UNSW-NB15, MemAE achieved the highest precision, but VLSTM provided the strongest recall. For KDDCUP, AE and DAGMM both delivered strong results, with AE performing better in recall and DAGMM achieving the highest precision. On the InSDN dataset, LSTM clearly outperformed OC-SVM across all metrics. These findings suggest that the dataset and selected hyperparameters play a major role in determining which model performs best.

While existing studies show promising results, further research is needed to optimize models for real world network environments and large scale data. Future work should focus on making these models more scalable, more robust, and easier to deploy in practice.

References

[1] N. Woolf, "DDoS Attack That Disrupted Internet Was Largest of Its Kind in History, Experts Say," https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet, 2016, [Online; accessed 18-Jun-2025].

[2] J. Martin, "Cybersecurity Statistics: IoT, DDoS, and Other Attacks," https://explodingtopics.com/blog/cybersecurity-stats#iot-ddos-and-other-attacks, 2025, [Online; accessed 19-Aug-2025].

[3] B. Min, Y. Jihoon, S. Kim, D. Shin, and D. Shin, "Network Anomaly Detection Using Memory-Augmented Deep Autoencoder," *IEEE Access*, vol. PP, pp. 1–1, 07 2021.

[4] M. S. Yadav and R. Kalpana, "Data Preprocessing for Intrusion Detection System Using Encoding and Normalization Approaches," in *2019 11th International Conference on Advanced Computing (ICoAC)*, 2019, pp. 265–269.

[5] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network Anomaly Detection Using LSTM Based Autoencoder," in *Proc. 16th ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet)*, Alicante, Spain, 2020, pp. 37–45.

[6] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, and F. Sabrina, "Improving Performance of Autoencoder-Based Network Anomaly Detection on NSL-KDD Dataset," *IEEE Access*, vol. PP, pp. 1–1, 2021.

[7] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," *EAI Endorsed Transactions on Security and Safety*, vol. 3, no. 9, 2016.

[8] R. C. Aygun and A. G. Yavuz, "Network Anomaly Detection with Stochastically Improved Autoencoder Based Models," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2017, pp. 193–198.

[9] R. K. Malaiya, D. Kwon, J. Kim, S. C. Suh, H. Kim, and I. Kim, "An Empirical Evaluation of Deep Learning for Network Anomaly Detection," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, 2018, pp. 893–898.

[10] L. Ashiku and C. Dagli, "Network Intrusion Detection System using Deep Learning," *Procedia Computer Science*, vol. 185, pp. 239–247, 2021, big Data, IoT, and AI for a Smarter Future.

[11] F. A. Khan and A. Gumaei, "A Comparative Study of Machine Learning Classifiers for Network Intrusion Detection," in *Artificial Intelligence and Security*, X. Sun, Z. Pan, and E. Bertino, Eds. Cham: Springer International Publishing, 2019, pp. 75–86.

[12] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational LSTM Enhanced Anomaly Detection for Industrial Big Data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2021.

[13] M. Ahmed, A. N. Mahmood, and J. Hu, "A Survey of Network Anomaly Detection Techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[14] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection," in *International Conference on Learning Representations*, 2018.

[15] K. Berahmand, F. Daneshfar, E. S. Salehi, and et al., "Autoencoders and their applications in machine learning: a survey," *Artificial Intelligence Review*, vol. 57, no. 28, 2024.

[16] IBM, "Variational Autoencoder," https://www.ibm.com/de-de/think/topics/variational-autoencoder, 2024, [Online; accessed 26-May-2025].

[17] H. Torabi, S. Mirtaheri, and S. Greco, "Practical Autoencoder-Based Anomaly Detection by Using Vector Reconstruction Error," *Cybersecurity*, vol. 6, 2023.

[18] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.

Blockchain Governance and Tokenomics

Vadym Khyzhniak, Holger Kinkelin*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: vadym.khyzhniak@tum.de, kinkelin@net.in.tum.de

***Abstract*—Resource sharing among distributed research testbeds demands contribution rewards, transparent membership, and fair use of hardware. We propose a dual-token Decentralized Autonomous Organization (DAO) on Algorand — "cash" Payment Tokens for real-time resource payments and "prepaid" Governance Tokens for contribution-based voting. We define initial issuance, token lifecycles, and two core workflows (new-member onboarding and monthly distributions tied to usage) and add safeguards to prevent pool depletion and abuse. Our framework focuses on tokenomics and offers a simple, automated solution for equitable testbed coordination.**

***Index Terms*—DAO, Algorand, tokenomics, payment tokens, governance tokens, resource sharing, testbed coordination**

1. Introduction

Resource sharing across multiple research testbeds poses specific challenges: how to reward contributions, ensure fair access to hardware, and make membership decisions in an optimal, transparent way. A *Decentralized Autonomous Organization* (DAO) built on Algorand can address these challenges by encoding payment and governance rules directly in on-chain logic.

1.1. Goal

This paper examines the tokenomics of a DAO that coordinates resource sharing among distributed research testbeds. By leveraging *Algorand Standard Assets* (ASA), the DAO issues two token types (proposed by us):

- **Payment Tokens**, used to "book" specialized hardware resources (GPU clusters, 6G testbeds, etc.),
- **Governance Tokens**, used to vote on membership and parameter changes.

Our objectives are to:

- Define the roles and lifecycle of each token type.
- Describe the initial minting and allocation to founding sites and the seeding of on-chain pools.
- Explain how those pools function under "cash" versus "prepaid" models.
- Analyze critical corner cases — such as empty pools and inactive token holders.
- Identify open questions and recommendations for sustainable operation.

1.2. Outline

The paper is structured as follows:

- **Section 2** presents a concise overview of the DAO testbed scenario and defines the key concepts and terminology (DAO, governance processes, Algorand smart contracts, ASA).
- **Section 3** analyzes our dual-token model (Payment & Governance), details initial issuance and ongoing token flows, including member admission workflows, and addresses critical corner-case protections.
- **Section 4** discusses remaining open questions and directions for future work, such as parameter tuning, cycle lengths, and replenishing mechanisms.
- **Section 5** concludes with the main takeaways and an outlook on prototyping.

Following this structure, the paper systematically develops the tokenomics framework and demonstrates its practical applicability to the research testbed ecosystem.

2. Background and Definitions

In order to design a robust tokenomics framework, we first explain the core concepts and terminology that form our DAO model. We focus on DAOs, their governance processes, the Algorand platform's smart contract capabilities, and the role of ASA.

2.1. What is a DAO?

A Decentralized Autonomous Organization (DAO) encodes its rules in on-chain smart contracts, automating proposals, fund management, and membership without centralized boards [1].

2.2. Governance

Governance in a DAO typically follows three stages [2]:

1) **Proposal Creation (Off-Chain & On-Chain):** Members come up with and discuss proposals off-chain (e. g. forums or meetings), A finalized proposal is then submitted on-chain, registering its details in the governance contract.
2) **Voting:** Token holders lock (or stake) their GTs to cast votes for or against the proposal within

 doi: 10.2313/NET-2025-11-3_18

a predefined period. Vote weight can be proportional to holdings or adjusted by other rules.

3) **Execution:** After voting closes, if the quorum and approval thresholds are met, a smart contract automatically carries out the decision implementation, such as minting new tokens, transferring assets, or updating the system's parameters.

Such a hybrid off-/on-chain model balances the organization's low-cost coordination with transparency.

2.3. Algorand and Smart Contracts

Algorand is a public blockchain that uses a pure proof-of-stake (PPoS) consensus mechanism to achieve fast finality and low transaction fees. Its design prioritizes scalability and energy efficiency, making it suitable for high-frequency micro-transactions such as resource payments in our scenario.

Algorand supports two types of smart contracts [3]:

- **Stateless Contracts:** Lightweight scripts that approve single transactions based on static conditions. They cannot store data between calls.
- **Stateful Contracts:** Programs with an on-chain key-value store, which allows to retain state across multiple transactions, what is essential for tracking token-pool balances, proposal counts, and vote tallies over time.

Our DAO uses *stateful contracts* because they "remember" pool reserves and governance state, enabling multi-step workflows like minting, reclaiming, and member admission without using any external databases.

2.4. Algorand Standard Assets (ASA)

ASAs provide native support for issuing and managing fungible tokens without a contract code [4]. ASAs are defined by parameters including total supply, decimal precision, and designated addresses with minting, freezing, or management privileges. ASA tokens benefit from Algorand's high throughput and low transaction fees, making them suitable for frequent transactions (resource payments) and governance actions (proposals). Our system will create two ASA instances: one for PT and one for GT — leveraging Algorand's built-in support for transparent token management.

3. Tokenomics

The proposed DAO's design depends on two tokens: *Payment Tokens* (PT), operating as "cash" for resources, and *Governance Tokens* (GT), issued monthly in a "prepaid" way to reflect recent contributions. Such an approach enables payments while keeping voting power aligned with participation.

3.1. Payment Tokens (Cash Model)

PTs are the DAO's on-chain cash for resource bookings. At Genesis, the DAO mints a fixed PT supply of (illustratively) 1,000,000 tokens. Founding members (five testbeds) receive 50% of that supply (500,000 PT), split equally at 100,000 PT each to cover initial operations and establish baseline liquidity. The remaining 50% (500,000 PT) is held in the PT Pool, from which newly admitted testbeds receive a one-time allocation (determined by governance) when they join.

Under this model, whenever Testbed A books, say, 10 GPU hours or 10 6G time slots (defined by the rules) on Testbed B, A transfers 1,000 PT directly from its wallet to B's wallet. This immediate, wallet-to-wallet payment ensures providers are compensated in real-time. If a testbed exhausts its PT balance, it may either earn tokens by providing resources or, if necessary, subject to a governance vote and receive a capped one-time top-up from the PT Pool (e. g. up to 1,000 PT per defined cycle).

By granting founding members half the supply and reserving the rest for governed new-member allocations and emergency top-ups, the cash model provides both initial stability and the flexibility to onboard additional testbeds in a controlled, democratic fashion. The controlled top-up mechanism pushes testbeds to manage their PT budgets carefully and use resources wisely, preventing frivolous or simply wasteful bookings.

3.2. Governance Tokens (Prepaid Model)

GTs encode decision-making power within the DAO and are never used for resource payments. At launch, the DAO mints a total supply of 100,000 GT. Founding members (five testbeds) receive 50% of that supply (50,000 GT), split equally 10,000 GT each to establish initial governance rights. The remaining 50% (50,000 GT) is held in the GT Pool, reserved for periodic, contribution-based issuance to all active testbeds.

Rather than a one-off grant, GT are distributed (e. g. monthly) according to each testbed's contribution: at the start of each cycle, the DAO measures the PT each testbed earned during the previous month (e. g. A earned 20,000 PT and B earned 5,000 PT). To maintain a 1 GT : 10 PT ratio reflecting the GT Pool's ten times smaller size relative to PT, the contract issues 1 GT for every 10 PT earned. Thus, the smart contract transfers GT from the pool: A receives 2,000 GT, B receives 500 GT. Any GT that remain unspent by the end of the cycle are returned to GT pool, preventing accumulation and ensuring that voting power always reflects recent, active participation.

Since newcomers have no prior earnings, the DAO grants each newly admitted testbed a minimal allocation, 1,000 GT for example, ensuring they can immediately participate in governance. This welcome grant comes from the GT Pool and helps integrate new members without skewing voting power excessively.

This prepaid issuance model aligns governance influence with actual contributions and discourages passive or long-inactive members from hoarding voting power. By resetting GT balances every cycle and returning unused tokens, the DAO maintains a dynamic, merit-based governance structure in which decision-making rights are based on each testbed's real-time engagement.

3.3. Operational Workflows

This subsection describes the DAO's two routines that drive token distribution and resource coordination. First,

 doi: 10.2313/NET-2025-11-3_18

we detail a hypothetical New-Member Admission Flow, which governs how candidates join and receive their initial token allocations. Then, we describe the ongoing monthly token flow, showing how PT earnings translate into GT issuance, resource bookings are made, and how unused governance tokens return to the pool each cycle.

3.3.1. New-Member Admission Flow. Membership starts off-chain: candidates announce their hardware capabilities on the DAO forum. A sponsoring member then submits the candidate's details, such as address and capacity, to the on-chain governance contract, opening a forty-eight-hour vote. Token holders stake their cycle-issued GT to vote. If a simple majority supports admission, the contract atomically transfers the one-time PT allocation (e. g. 50,000 PT) and a welcome GT grant (e. g. 1,000 GT) to the newcomer's wallet, as shown in Fig. 1. Finally, the registry marks the testbed as active, enabling it to both consume and provide resources in the next cycle [5].

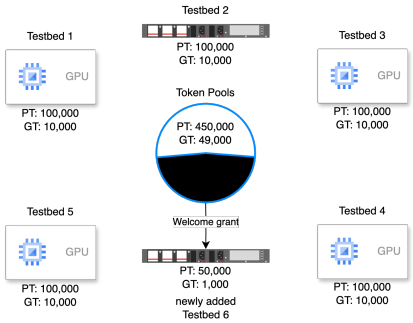


Figure 1: New-Member Admission, welcome grant.

3.3.2. Ongoing Monthly Token Flow. On the first day of each cycle, the contract tallies each testbed's PT earnings from the previous month and issues GTs at a 1 GT : 10 PT ratio from the GT Pool. Resource bookings occur via instant PT transfers between wallets as the month proceeds. Governance proposals may arise anytime: members stake GTs to vote, and once voting closes, the contract enacts the decision and returns staked GT. At the cycle's end, any unspent GTs return to the pool. This loop (PT earnings → GT issuance → PT resource bookings → GT return) ensures that token balances remain tightly coupled to actual contributions and usage while maintaining the integrity of both token pools.

3.4. Corner-Case Protections

Even the most robust tokenomics framework must prepare for exceptional situations. Below, we expand our original principles with complementary safeguard methods for PT/GT pools, treasury reclamation, emergency needs, and "rage quits" to ensure the DAO remains operational and equitable.

3.4.1. PT Exhaustion and Emergency Top-Ups. Because Payment Tokens are granted once upfront, heavy consumers may run their wallets to zero before earning new tokens. Our DAO offers a capped emergency top-up mechanism to prevent deadlock where a testbed cannot book critical resources. Any testbed in need may submit a governance proposal, requiring only a simple majority of staked GT, to unlock a one-time PT grant (e. g. limited up to 1,000 PT per cycle) from the PT Pool. This advance must be repaid through subsequent resource earnings. By tying such top-up to governance approval, we ensure that requests are reasonable and encourage careful PT usage. Beyond liquidity, contract-level security risks must be considered. Attacks such as double-reporting of resource usage, Sybil identities, or malicious proposals could undermine fairness. These risks suggest the need for trusted measurement oracles and rate-limiting mechanisms in future prototypes.

3.4.2. PT Pool Depletion. If repeated top-ups exhaust the PT Pool, the DAO faces a shortage of reserve tokens for new member allocations and emergencies. Two recovery paths are available:

- **Controlled Minting:** The DAO can pass a vote to mint additional PTs at a predetermined percentage of the original supply (e. g. 5%) to replenish the pool, balancing liquidity needs with token scarcity.
- **Inactive Balance Reclamation:** If a testbed remains inactive (no PT earnings or bookings) for two consecutive cycles, its leftover PT balance can be returned to the PT Pool via a routine governance action, reclaiming tokens without affecting total supply.

These methods ensure that reserve tokens remain available for both new members and emergencies.

3.4.3. GT Pool Depletion. A similar risk arises if continuous GT issuance to existing and new members drains the Governance Token Pool. Once depleted, no further GTs can be allocated, meaning governance for newcomers is frozen, and active contributors are disabled. To guard against this, two ways are possible:

- **Mint Voting:** Voting approves the creation of new GTs, similarly to PT minting, to refill the pool for upcoming cycles.
- **Inactivity Confiscation:** Members who neither earn PT nor participate in votes for three consecutive cycles lose their remaining GTs, which return to the pool. Thus, active participants are supported without increasing supply.

3.4.4. Imbalanced Resource Usage. A handful of very large testbeds might monopolize resource consumption and earn disproportionate GT allotments, threatening to tilt governance in their favor [6]. To maintain equitable influence, the DAO can cap monthly GT allocations per member (e. g. no more than 20% of total GTs issuable per cycle). In such a way, the system is safeguarded against "plutocracy".

3.4.5. Departing Members. As mentioned in [7], the right to exit should be part of the core protocol so that any member can exit at any time and for any reason.

 doi: 10.2313/NET-2025-11-3_18

When a testbed voluntarily exits the DAO, it should return its PT and GT holdings to the pools rather than retain them. Upon an on-chain exit call, the governance contract returns any unused PTs/GTs to the respective pools rather than burning them, thereby preserving supply for future cycles. The system will have no orphaned tokens and maintains pool integrity.

4. Open Questions and Future Work

While our dual-token framework lays the groundwork for coordinating resource sharing, some parameters merit further study. For example, the emergency top-up cap must balance availability and token scarcity: too low, and testbeds lack critical access; too high, and PT loses value. Simulations will be crucial to pinpoint an effective threshold.

The cadence of GT issuance also deserves scrutiny. We currently use monthly cycles tied to PT earnings, but shorter intervals (e. g. weekly) might align voting power more closely with activity, whereas longer intervals could cut overhead. Likewise, the newcomer GT grant must strike a balance — large enough to include new members yet small enough to avoid temporary imbalances. Allowing the community to vote on these parameters offers flexibility.

Our framework would benefit from empirical validation. Even simple simulations of token flows could show whether pools deplete too quickly or whether voting power stabilizes over time. Future work should include cost comparisons against existing resource-sharing federations (e. g. GENI, PlanetLab) to quantify advantages in fairness and overhead.

Finally, our pool-replenishment strategies, like minting via majority vote or reclaiming inactive balances, have to be tested in real environment. We must determine when to mint additional tokens rather than reclaiming them based on pool levels and member activity. Adding adaptive logic that monitors these metrics might result in a more efficient system.

5. Conclusion

We have presented an architecture for a research testbed DAO: PTs enable instant, peer-to-peer transfers for resource usage, and GTs are distributed cyclewise in proportion to contributions, ensuring decision rights align with activity. Algorand-implemented solution via stateful smart contracts and ASA enables streamlined membership onboarding, token issuance, resource exchanges, and token "recycling" in a standalone system.

By incorporating emergency votes, controlled minting, and user exits, we are establishing the DAO principles and providing an efficient structure for a distributed testbed ecosystem. At the same time, several trade-offs remain to be explored, such as balancing emergency liquidity against token scarcity, calibrating cycle lengths for governance, and ensuring that heterogeneous resources are priced in a fair and flexible way.

Our contribution should therefore be seen less as a final protocol and more as a blueprint for experimentation. A logical next step will be to implement a minimal prototype and run simulations of token flows under different workloads. Pilot deployments across a small federation of research sites would further validate scalability and robustness in real-world settings. Such evaluations will ultimately determine how well the proposed dual-token system generalizes and how it can be integrated with broader DAO frameworks.

References

[1] S. Wang, W. Ding, J. Li, Y. Yuan, L. Ouyang, and F.-Y. Wang, "Decentralized autonomous organizations: Concept, model, and applications," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 870–878, Oct. 2019.

[2] K. Pakseresht and H. Kinkelin, "Governance of a distributed autonomous organization," Chair of Network Architectures and Services, Technical University of Munich, Tech. Rep., 2024, seminar paper.

[3] J. Weathersby, "Linking algorand stateful and stateless smart contracts," https://developer.algorand.org/articles/linking-algorand-stateful-and-stateless-smart-contracts, Nov. 2020, Algorand Developer Portal.

[4] "Algorand Standard Assets (ASAs)," https://developer.algorand.org/docs/get-details/asa/, [Online; accessed 22-June-2025].

[5] Upstream, "Joining a dao," https://guide.upstreamapp.com/dao-members/joining-a-dao, 2024, [Online; accessed: 19-June-2025].

[6] M. Esposito, T. Tse, and D. Goh, "Decentralizing governance: exploring the dynamics and challenges of digital commons and daos," *Frontiers in Blockchain*, vol. 8, p. 1538227, 2025, published: 16 May 2025.

[7] Moloch, "Ragequit," https://moloch.daohaus.fun/features/ragequit, 2025, [Online; accessed: 19-June-2025].

Securing BGP - Mechanisms to Prevent Routing Leaks

Leon Spörl, Michael Oberrauch*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: leon.spoerl@tum.de, oberrauc@net.in.tum.de

***Abstract*—The Border Gateway Protocol (BGP) is today's prevailing interdomain routing protocol, and due to its widespread adoption, it is likely to remain so for the foreseeable future. BGP was designed with scalability and efficiency in mind but lacks fundamental security features. Routing leaks caused by misconfiguration or malicious actions are among the main vulnerabilities. We evaluate the leading security mechanisms IRR, RPKI, and BGPsec with respect to performance, scalability, adoption, and their ability to fulfill specific security objectives. Often, inconsistent adoption is the limiting factor of the described mechanisms as security features only take full effect if rules are uniformly applied and strictly enforced. We conclude by reviewing promising new advancements regarding the implementation of ASPA and outline potential future developments.**

***Index Terms*—Autonomous System Provider Authorization (ASPA), Border Gateway Protocol (BGP), Border Gateway Protocol Security (BGPsec), Internet Routing Registry (IRR), Resource Public Key Infrastructure (RPKI)**

1. Introduction

Since its introduction over 30 years ago, the Border Gateway Protocol has become the global de facto standard of inter-AS routing. Despite its excellent routing capabilities, it is insecure by design.

One of the key problems of BGP is that an AS has no control over its resources, i.e., how the AS number and IP prefixes that have been uniquely assigned by the Regional Internet Registry are announced and propagated. This lack of control allows malicious or misconfigured networks to distribute incorrect routing statements. Routing leaks are a common type of security violation caused by erroneous route propagation. Minor routing errors can affect large regions, with even simple attacks posing a major threat to the Internet's backbone. There have been numerous incidents, one of them being an accidental route leak induced by Google in August 2017, which caused massive Internet disruptions in Japan [1].

Incidents like this stress the need for globally deployed BGP security mechanisms. In this paper, we analyze the most relevant innovations that allow us to prevent routing leaks and other BGP-related attacks. We begin by reviewing related work in this research area, followed by some background information and definitions. We then examine popular security approaches and end with an outlook on future developments.

2. Related work

In the early days of BGP, most research papers only proposed new solutions or evaluated a single security proposal [2], [3]. A few years later, the first systematic reviews were published. As BGP security is a quickly evolving field of research, the findings of literature surveys from around 2010 are outdated and do not address most of today's security mechanisms [4]–[6]. A reasonable recent survey by Mitseva et al. [1] provides an extensive overview of BGP security properties and solution approaches.

Regarding the individual security mechanisms, recent research includes a quantitative analysis of RPKI deployment by Chung et al. [7] and a survey by Rodday et al. [8] summarizing RPKI-related studies, challenges and solution proposals. Research by Du et al. [9] highlights vulnerabilities of the Internet Routing Registry, by quantifying and analyzing irregular IRR records. The challenges of BGPsec deployment and optimization approaches are summarized by Abdelhafez and Fadlalla [10].

3. Background

Before discussing the attributes of BGP and the need for specific security measures, we have to take a look at the general structure of the Internet and its historical background.

3.1. Autonomous Systems

The Internet consists of many interconnected, independently administered networks, so-called autonomous systems (ASes). From a technical perspective, an autonomous system is a set of routers that follow a uniform routing plan to allow for intra- and inter-AS network communication. [11] An AS is managed by a single organization and is identified by a unique 16 or 32-bit number. The AS numbers (ASNs) and IP address spaces are assigned by the Regional Internet Registries (RIRs) to the respective autonomous systems. [1], [12] Depending on the geographical location, a different RIR applies [13]:

- **AfriNIC**: Africa
- **APNIC**: parts of Asia, Pacific Region
- **ARIN**: North America, parts of the Caribbean
- **LACNIC**: Latin America, parts of the Caribbean
- **RIPE NCC**: Europe, Middle East, parts of Asia

A variety of interior gateway protocols are used for routing within an AS, i.e., intradomain routing [14]. Some

 doi: 10.2313/NET-2025-11-3_19

organizations may even run multiple routing protocols in parallel or use proprietary standards. While intradomain routing is transparent to other ASes and does not have to follow a common standard, interdomain routing requires an operational standard so all autonomous systems can span a global network.

3.2. The Border Gateway Protocol

The Border Gateway Protocol (BGP) was first introduced in 1989 [15], specified in RFC 1105, and replaced the Exterior Gateway Protocol (EGP) [16]. The most recent version, BGP-4 [11], is today's de facto standard for global inter-AS routing [1]. In the early days of the Internet, there were also other interdomain routing protocols proposed, e.g. the OSI Inter-Domain Routing Protocol (IDRP) [17], which was expected to replace BGP but has no relevance today [18], [19].

The BGP specification describes how autonomous systems can exchange reachability information and announce IP prefixes to each other [11]. BGP is a path-vector protocol, meaning that not only the distance but the whole path to a destination is announced. Therefore, each AS adds itself to the path before propagating the route to others. [5] Using that information, every AS can maintain its own reachability graph, make informed routing decisions, and derive a routing table. Topology changes are propagated within minutes, making BGP a dynamic and resilient protocol. An AS can take different metrics into account when making routing decisions, the most relevant ones being path and prefix length. The shortest path method is used to minimize the number of hops between source and destination. If announced IP address ranges overlap, the announcement with the more specific IP prefix is usually prioritized. [12]

Business agreements and financial considerations also play a role when selecting paths and propagating routes to other ASes. Two connected autonomous systems can have a customer-provider relationship, i.e., the customer pays the provider for the traffic routed over the path or a peer-to-peer relationship where traffic is routed over the link free of charge. [20] If each AS tries to maximize its revenue and only propagates routes that incur financial gain, we speak of valley-free routing. After a packet has traversed a provider-to-customer or peering link, it is only allowed to take additional provider-to-customer links and must not again move upwards in the hierarchical BGP model. Valley routes may lead to increased path lengths and ASes unintentionally transiting traffic [21].

3.3. BGP Route Leaks

Although the term route leak already appears in literature from the early 2000s, it lacked a uniform definition for a long time [22]. RFC 7908, published in 2016, defines a route leak as "the propagation of routing announcement(s) beyond their intended scope" [23]. A route leak occurs when an AS propagates a learned route to another AS and thereby violates the policies of an AS alongside the resulting path. More specifically, policies typically refer to the business relations described in Section 3.2. A policy violation can be the infringement of the valley-free property [1].

RFC 7908 provides a taxonomy of route leaks based on observed incidents and differentiates between six types. The first five depict distinct scenarios where learned BGP routes from non-customer ASes are announced to provider or peering ASes. Figure 1 illustrates a type 1 route leak where an AS receives a route announcement from a provider AS and propagates it to another provider AS. The leaked route is probably preferred by AS 3 because customer ASes are more highly prioritized than peering ASes, resulting in a hairpin turn at AS 2.

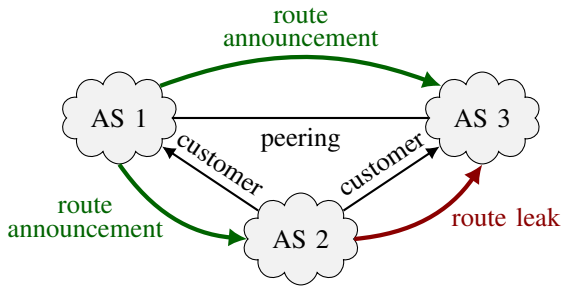


Figure 1: BGP route leak type 1

The remaining route leak category, type 6, refers to an issue that can occur if BGP is used as a routing protocol inside an AS. In this case, internal and external BGP must be strictly separated. Misconfiguration can lead to internal routes being exported to the outside and announced to other autonomous systems. These internal prefixes may be more specific than externally routed ones and are therefore preferred by other ASes.

This behavior is similar to a route hijack, also known as prefix hijack, which occurs when an AS originates a prefix it is not the holder of. If the address space belongs to another AS, this can have severe consequences as the illegitimately announced route may be selected by other ASes across the Internet due to a shorter path or a more specific prefix. [1], [5]

The distinction between BGP route leaks and hijacks is not very sharp in the literature. While Wijchers and Overeinder [24] draw a clear line between the terms, the RFC on route leak classification cites resources about prefix hijacks and lists hijacking incidents as examples for route leaks [23]. Both incidents can happen by accidental misconfiguration or with malicious intent [5], [23].

4. Security Mechanisms

BGP uses the Transmission Control Protocol (TCP) [25], which offers some basic error correction and retransmission mechanisms [11]. However, BGP provides no confidentiality or integrity protection for its messages [5]. By default, route advertisements are not authenticated, so autonomous systems can announce arbitrary prefixes, no matter if they are the legitimate holders [12]. There are various approaches to make BGP more secure and reliable. We will analyze the most popular ones regarding their security properties, performance, practicability, and adoption.

4.1. Internet Routing Registry

The Internet Routing Registry (IRR), proposed in 1995, was one of the first attempts to make BGP routing

 doi: 10.2313/NET-2025-11-3_19

more transparent and secure [26]. The IRR is a set of public databases where ASes can upload routing information on prefixes they hold using the Routing Policy Specification Language (RPSL) [27]. Other ASes can then fetch the data and create BGP route filters [1].

As of May 2025, there exist 18 IRR databases[1] hosted by companies as well as Regional and National Internet Registries. Many of the databases mirror other IRRs to provide exhaustive data sets. The different IRR providers do not use standardized authentication and validation mechanisms, resulting in varying quality of database records [9]. RIRs can authenticate their members and approve ownership of resources before publishing routing information for a certain address space. Other routing registries lack the ability to perform this kind of authorization but may publish routing information on an address space anyway. [12]

Malicious actors exploited these circumstances in the past. Du et al. [9] describe a case where attackers were able to hijack a prefix belonging to Amazon and added a manipulated route object to the ALTDB routing registry. They were thereby able to reroute customers of a company that used Amazon's cloud resources to a phishing page and stole cryptocurrency worth $235,000. This is not an isolated incident. Over the period of 1.5 years, Du et al. analyzed RADB, which is the largest IRR database holding over 1.5 million route objects. By investigating consistency across IRR databases and checking if database entries match BGP announcements, they detected 34,199 irregular entries and classified 6,373 of them as suspicious.

Proposals to improve IRR's security did not resolve all vulnerabilities or were simply not implemented on a broad scale [1], [28], [29]. IRR's main issue is the lack of global uniformity, making route validation unreliable and error-prone. While it was a great advance in terms of routing transparency in the 90s, it does not meet the expectations of BGP security today. [12]

4.2. Resource Public Key Infrastructure

The Resource Public Key Infrastructure (RPKI) [30] is an out-of-band system that enables resource holders to cryptographically prove their identity and digitally sign statements on the intended use of their resources. The infrastructure uses X.509 certificates [31] with two extensions that bind AS numbers and IP address prefixes to the holder of the certificate's private key [32]. The hierarchical structure of RPKI is based on five trust anchors, with each Regional Internet Registry operating its own root Certificate Authority (CA). AS administrators can request resource certificates over the member portal of their respective RIR. This procedure is called hosted RPKI, as certificate creation, publication, and key rollover are all done by the RIR. Although this process is relatively simple and convenient for most members, it is not sufficient for all of them. Some organizations must host their own child CA in order to delegate RPKI administration to their customers. This operating mode is called delegated RPKI and is supported by all RIRs except for AFRINIC (as of May 2025). [7]

Using their resource certificate, autonomous systems can sign Route Origin Authorizations (ROAs) [33]. A ROA authorizes an AS to originate a certain IP address space. Apart from the prefix and ASN, a ROA contains the maximum prefix length the specified AS is allowed to advertise. The issued ROAs are typically published by the RIRs. Other ASes can then fetch the data sets, verify the ROA signatures and derive validated ROA payloads (VRPs). The verification of incoming BGP announcements, known as route origin validation (ROV) [34], can return three different results:

- **Valid**: announcement covered by a VRP
- **Invalid**: announcement from unauthorized AS or announcement violates the VRP's maximum prefix length attribute
- **NotFound**: announced prefix not (fully) covered by any VRP

Given that RPKI is still far from universal adoption, announcements returning the *NotFound* status should be accepted for now. [7], [12]

The deployment of RPKI started in 2011 and was impaired by many configuration errors in the early adoption stages. Chung et al. [7] evaluated RPKI statistics that were collected from 2011 to 2019. Throughout 2011, they observed 48.92% of VRP-covered announcements to be invalid. This value drastically improved over the years, settling between 2% and 5% in 2019. During the whole measurement period, around half of the invalid announcements were caused by too specific prefixes. Many of the published ROAs were missing the MaxLength attribute, indicating misconfiguration as a cause. In recent years, the general adoption has improved.

The ratio of unique IPv4 prefix-origin pairs (combinations of IP prefix and origin AS number) covered by VRPs has increased from 12% in early 2019 to over 56% in May 2025 according to the NIST RPKI Monitor[2]. The IPv6 coverage shows similar numbers, recording almost 58% validatable IPv4 prefix-origin pairs in May 2025. While there is less historic data available for IPv6, the observable trend is similar to IPv4 [7].

RPKI only takes full effect if all ASes systematically drop invalid announcements. Some major service providers, such as AT&T, are already enforcing this policy. [7] RPKI's route origin validation prevents prefix hijacks but does not target route leaks as defined in Section 3.3. An AS can restrict route origination for its resources, but the path an announcement takes is unknown [8].

4.3. Border Gateway Protocol Security

Border Gateway Protocol Security, better known as BGPsec [35], is a BGP extension that relies on RPKI's resource certificates to provide cryptographic path validation. BGPsec replaces the AS_PATH attribute with a BGPsec_PATH attribute inside the BGP update messages. The new attribute contains digitally signed path information. Each AS that an announcement traverses signs the previous path, its own AS number, and the number of the AS the announcement will be propagated to. Every router

1. https://irr.net/registry

2. https://rpki-monitor.antd.nist.gov

 doi: 10.2313/NET-2025-11-3_19

in the announcement chain verifies the signatures and can detect path forgery. [10]

The main downside of BGPsec is its poor adoption due to the high entry barrier. Most ASes would need to replace their hardware to support BGPsec, without deriving a direct benefit from the investment [1]. If one router along the announcement path has no BGPsec capabilities, the whole path immediately becomes invalid. There is no *NotFound* state like in RPKI, so partial BGPsec deployment has little utility. [36] Aside from that, the extension generates significant computational overhead slowing down the BGP operations by a factor of 70, as shown by Kim and Kim [37]. There have been optimization efforts to reduce CPU overhead and memory consumption, but no optimization algorithm could be implemented at scale yet. While there is some room for improvement, well-performing BGPsec requires specialized hardware accelerators. [10]

4.4. Autonomous System Provider Authorization

Autonomous System Provider Authorization (ASPA) [38] is a draft that has been discussed by the Internet Engineering Task Force (IETF) over the last years. ASPA objects are part of the RPKI, similar to ROAs [7]. By generating and signing an ASPA object, an AS can specify a list of provider ASes. Using these statements, each hop along a path can be identified as

- **Provider**,
- **Not Provider**, or
- **No Attestation** if no ASPA can be retrieved from the customer.

Given this information, we can make plausibility checks and detect implausible paths potentially caused by route leaks. ASPA explicitly supports incremental deployment. If some ASes make no attestation, the plausibility check succeeds if no other policy violations are detected, no matter how sparsely the feature is deployed. ASPA is capable of detecting (accidental) route leaks and even protects against some forms of prefix hijacks. The wider the adoption of ASPA, the harder it becomes for attackers to perform unrecognized route hijacks. However, Route Origin Authorization is still the most reliable method of hijacking prevention. ASPA's deployment model is very similar to ROAs. Cryptographic operations are handled by certificate authorities, while routers can handle verification without much computational overhead.

RIPE officials stressed ASPA's significance at the "RIPE 90" meeting in May 2025. It is yet to be standardized and is planned to be implemented in RIPE's hosted RPKI in summer 2025. ARIN and APNIC also intend to test their implementations in 2025. [39]

4.5. Other Approaches

Apart from the already presented security mechanisms, there were many other proposals in the past, most of them having no relevance nowadays. One of the first approaches was Secure-BGP (S-BGP) [2], which is conceptually very similar to RPKI. It uses an out-of-band public key infrastructure to authenticate prefix announcements and provide route origin validation. Since S-BGP's adoption was obstructed by performance issues, Secure Origin BGP (soBGP) [3] was developed at Cisco Systems in 2003. It also relies on a PKI and is characterized by a lower-overhead implementation, but uses proprietary certificates and, hence, lacks interoperability.

Further proposals were made in the following years, but none of them was realized at a large scale until RPKI was standardized in 2012 [1]. More recent research by Hari and Lakshman [40] suggests blockchain-based BGP security, omitting the PKI's central root of trust. However, the scalability and performance of a blockchain-based solution have not yet been demonstrated with a practicable implementation.

5. Conclusion and Outlook

When assessing the features, performance, and adoption of BGP security mechanisms, we can conclude that efficient solutions have already been found, but advancements are impeded by a heterogenous system topography and rigid legacy systems.

The resource public key infrastructure provides a solid base that allows developments like Route Origin Authorization and Autonomous System Provider Authorization to build upon. BGPsec is a proven solution for path validation, which was well integrated into BGP and RPKI but is constrained by computational overhead. This substantial entry barrier slows down the adoption process. The most important advancement in the upcoming years will be the further increase of RPKI adoption. The objective should be a full coverage allowing for strict ROA policy enforcement without locking out autonomous systems from global routing. Even though ASPA is still an IETF draft, it can be seen as a promising technique. As soon as the standardization is finished, it will profit from a low entry barrier, given the persisting RPKI and ASPA's excellent support for incremental deployment.

References

[1] A. Mitseva, A. Panchenko, and T. Engel, "The state of affairs in BGP security: A survey of attacks and defenses," *Computer Communications*, vol. 124, pp. 45–60, 2018. [Online]. Available: https://doi.org/10.1016/j.comcom.2018.04.013

[2] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (S-BGP)," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 582–592, 2000.

[3] R. White, "Securing BGP Through Secure Origin BGP," 2003.

[4] M. O. Nicholes and B. Mukherjee, "A survey of security techniques for the border gateway protocol (BGP)," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 1, pp. 52–65, 2009.

[5] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A Survey of BGP Security Issues and Solutions," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 100–122, 2010. [Online]. Available: https://dx.doi.org/10.1109/JPROC.2009.2034031

[6] G. Huston, M. Rossi, and G. Armitage, "Securing BGP — A Literature Survey," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 2, pp. 199–222, 2011.

[7] T. Chung, E. Aben, T. Bruijnzeels, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, R. v. Rijswijk-Deij, J. Rula, and N. Sullivan, "RPKI is Coming of Age: A Longitudinal Study of RPKI Deployment and Invalid Route Origins," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 406–419. [Online]. Available: https://doi.org/10.1145/3355369.3355596

[8] N. Rodday, I. Cunha, R. Bush, E. Katz-Bassett, G. D. Rodosek, T. C. Schmidt, and M. Wählisch, "The Resource Public Key Infrastructure (RPKI): A Survey on Measurements and Future Prospects," *IEEE Transactions on Network and Service Management*, vol. 21, no. 2, pp. 2353–2373, 2024. [Online]. Available: https://doi.org/10.1109/TNSM.2023.3327455

[9] B. Du, K. Izhikevich, S. Rao, G. Akiwate, C. Testart, A. C. Snoeren, and k. claffy, "IRRegularities in the Internet Routing Registry," in *Proceedings of the 2023 ACM on Internet Measurement Conference*, ser. IMC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 104–110. [Online]. Available: https://doi.org/10.1145/3618257.3624843

[10] M. Abdelhafez and Y. Fadlalla, "BGPsec Deployment Challenges and Optimization Efforts," in *2024 IEEE 22nd Student Conference on Research and Development (SCOReD)*, 2024, pp. 277–281. [Online]. Available: https://doi.org/10.1109/SCOReD64708.2024.10872667

[11] Y. Rekhter, S. Hares, and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4271

[12] A. Band, "RPKI Documentation," https://rpki.readthedocs.io, 2018, [Online; accessed 13-May-2025].

[13] R. Housley, J. Curran, G. Huston, and D. R. Conrad, "The Internet Numbers Registry System," RFC 7020, Aug. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc7020

[14] M. Athira, L. Abrahami, and R. G. Sangeetha, "Study on network performance of interior gateway protocols — RIP, EIGRP and OSPF," in *2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2)*, 2017, pp. 344–348.

[15] K. Lougheed and Y. Rekhter, "Border Gateway Protocol (BGP)," RFC 1105, Jun. 1989. [Online]. Available: https://www.rfc-editor.org/info/rfc1105

[16] D. L. Mills, "Exterior Gateway Protocol formal specification," RFC 904, Apr. 1984. [Online]. Available: https://www.rfc-editor.org/info/rfc904

[17] C. Kunzinger, "OSI INTER-DOMAIN ROUTING PROTOCOL (IDRP)," Internet Engineering Task Force, Internet-Draft draft-kunzinger-idrp-ISO10747-01, Nov. 1994. [Online]. Available: https://datatracker.ietf.org/doc/draft-kunzinger-idrp-ISO10747/01/

[18] J. A. Hawkinson and T. J. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)," RFC 1930, Mar. 1996. [Online]. Available: https://www.rfc-editor.org/info/rfc1930

[19] K. Butler, T. Farley, P. McDaniel, and J. Rexford, "A survey of BGP security," *ACM*, vol. 5, pp. 1–35, 2004.

[20] L. Gao and J. Rexford, "Stable Internet routing without global coordination," in *Proceedings of the 2000 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 307–317. [Online]. Available: https://doi.org/10.1145/339331.339426

[21] S. Y. Qiu, P. D. McDaniel, and F. Monrose, "Toward Valley-Free Inter-domain Routing," in *2007 IEEE International Conference on Communications*, 2007, pp. 2009–2016.

[22] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 3–16. [Online]. Available: https://doi.org/10.1145/633025.633027

[23] K. Sriram, D. Montgomery, D. R. McPherson, E. Osterweil, and B. Dickson, "Problem Definition and Classification of BGP Route Leaks," RFC 7908, Jun. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7908

[24] B. Wijchers and B. Overeinder, "Quantitative Analysis of BGP Route Leaks," NLnet Labs, Nov. 2014. [Online]. Available: https://ripe69.ripe.net/presentations/157-RIPE-69-Routing-WG.pdf

[25] W. Eddy, "Transmission Control Protocol (TCP)," RFC 9293, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9293

[26] T. Bates, E. Gerich, L. Joncheray, J.-M. Jouanigot, D. Karrenberg, M. Terpstra, and J. Yu, "Representation of IP Routing Policies in a Routing Registry (ripe-81++)," RFC 1786, Mar. 1995. [Online]. Available: https://www.rfc-editor.org/info/rfc1786

[27] C. Villamizar, T. J. Bates, C. Alaettinoglu, D. Meyer, M. Terpstra, D. Karrenberg, and E. P. Gerich, "Routing Policy Specification Language (RPSL)," RFC 2280, Jan. 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2280

[28] E.-y. Kim, L. Xiao, K. Nahrstedt, and K. Park, "Secure Interdomain Routing Registry," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 2, pp. 304–316, 2008.

[29] G. Siganos and M. Faloutsos, "Analyzing BGP policies: methodology and tool," in *IEEE INFOCOM 2004*, vol. 3, 2004, pp. 1640–1651 vol.3.

[30] M. Lepinski and S. Kent, "An Infrastructure to Support Secure Internet Routing," RFC 6480, Feb. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6480

[31] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, May 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5280

[32] D. C. W. L. Jr., K. Seo, and S. Kent, "X.509 Extensions for IP Addresses and AS Identifiers," RFC 3779, Jun. 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3779

[33] M. Lepinski, D. Kong, and S. Kent, "A Profile for Route Origin Authorizations (ROAs)," RFC 6482, Feb. 2012. [Online]. Available: https://wwfw.rfc-editor.org/info/rfc6482

[34] R. Bush, "Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI)," RFC 7115, Jan. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7115

[35] M. Lepinski and K. Sriram, "BGPsec Protocol Specification," RFC 8205, Sep. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8205

[36] R. Lychev, S. Goldberg, and M. Schapira, "BGP security in partial deployment: is the juice worth the squeeze?" *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, p. 171–182, Aug. 2013. [Online]. Available: https://doi.org/10.1145/2534169.2486010

[37] K. Kim and Y. Kim, "Comparative analysis on the signature algorithms to validate as paths in bgpsec," in *2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, 2015, pp. 53–58.

[38] A. Azimov, E. Bogomazov, R. Bush, K. Patel, J. Snijders, and K. Sriram, "BGP AS_PATH Verification Based on Autonomous System Provider Authorization (ASPA) Objects," Internet Engineering Task Force, Internet-Draft draft-ietf-sidrops-aspa-verification-22, Mar. 2025. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-verification/22/

[39] T. Bruijnzeels, "RPKI Functionality Roadmap," May 2025. [Online]. Available: https://ripe90.ripe.net/wp-content/uploads/presentations/110-RIPE-NCC-RPKI-Features-2025.pdf

[40] A. Hari and T. V. Lakshman, "The Internet Blockchain: A Distributed, Tamper-Resistant Transaction Framework for the Internet," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, ser. HotNets '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 204–210. [Online]. Available: https://doi.org/10.1145/3005745.3005771

Demonstrating Encrypted Client Hello (ECH) Privacy Benefits

Jasper Christian Stritzke, Tim Betzer*, Christian Dietze*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: jasper.stritzke@tum.de, betzer@net.in.tum.de, diec@net.in.tum.de

***Abstract*—Encrypted Client Hello (ECH) extends TLS 1.3 by encrypting the entire ClientHello, eliminating plaintext Server Name Indication (SNI) leakage. A reproducible container-based testbench, built from CoreDNS, an ECH-enabled Nginx instance, and an experimental curl client, allows direct, side-by-side observation of ECH-protected versus conventional TLS handshakes with the same endpoint. Packet captures produced by the framework show that ECH conceals the actual destination hostname, exposing only an innocuous outer SNI to passive observers. In contrast, plaintext TLS exposes the real domain in every trace. The work demonstrates that ECH eliminates the plaintext metadata leak in TLS 1.3 without altering the overall handshake semantics, compatibility, or application-layer behavior.**

***Index Terms*—TLS, Encrypted Client Hello, ECH, SNI, Privacy, Network Security, HPKE**

1. Introduction

1.1. Evolution of Internet Privacy and TLS Challenges

The increasing demand for internet privacy has driven significant advancements in cryptographic protocols, particularly within the Transport Layer Security (TLS) suite.

While TLS 1.3 encrypts most handshake data, a critical privacy vulnerability persists: the Server Name Indication (SNI) extension and other sensitive fields remain plaintext. The SNI, transmitted during the initial `ClientHello` message, reveals the target domain to network observers, enabling pervasive monitoring, censorship, and traffic analysis [1], [2], compromising user privacy despite encrypted communication.

Following post-Snowden surveillance concerns, the Internet Engineering Task Force (IETF) formally identified pervasive monitoring as an attack in RFC 7258 [3], transforming privacy from a technical feature into a foundational protocol requirement.

ECH represents a critical infrastructure upgrade for internet privacy, directly countering surveillance capabilities that undermine encrypted communication confidentiality.

1.2. ECH as a Comprehensive Privacy Solution

Encrypted Client Hello represents a significant advancement in addressing TLS privacy vulnerabilities through a comprehensive, protocol-level approach. Rather than addressing individual metadata leaks, ECH fundamentally restructures the TLS handshake to encrypt all sensitive client information from initial connection.

This enhancement eliminates piecemeal privacy solutions by providing complete ClientHello metadata protection, creating a unified defense against passive surveillance and traffic analysis across TLS 1.3 implementations. ECH prioritizes immediate privacy benefits and long-term protocol sustainability, ensuring infrastructure compatibility while establishing foundations for future privacy enhancements.

1.3. Purpose and Significance of the ECH Testbench Project

The ECH Testbench project provides a local, reproducible framework demonstrating ECH functionality and privacy benefits. By comparing ECH-enabled with non-ECH TLS traffic to the same server, the testbench shows how ECH obscures real target domains, challenging passive network observation. This setup provides an interactive playground for exploring ECH's operational flow and privacy implications.

2. Background on TLS and Encrypted Client Hello

This section provides essential background on TLS handshake mechanics, the privacy vulnerabilities in current implementations, and the evolution toward ECH as a comprehensive solution.

2.1. Fundamentals of TLS Handshake and SNI

TLS handshakes establish secure channels where client and server negotiate cryptographic parameters and authenticate identities. The Server Name Indication (SNI) extension allows clients to specify target hostnames in ClientHello messages. When multiple domains share IP addresses, servers use this field to present correct certificates [2].

Despite subsequent application data encryption, the ClientHello with SNI extension remains unencrypted in TLS 1.3, exposing target domains to on-path observers including ISPs, government agencies, and malicious actors [1]. This plaintext exposure undermines user privacy by revealing browsing habits and enabling censorship or traffic analysis [4]. Intermediaries can inspect the first TLS packet to determine specific targets, despite full communication encryption.

 doi: 10.2313/NET-2025-11-3_20

2.2. Evolution from ESNI to ECH

The privacy risks of plaintext SNI motivated the proposal of `Encrypted SNI (ESNI)`, an early attempt to hide the SNI field [2]. However, ESNI proved insufficient by protecting only SNI; remaining ClientHello extensions (e.g., ALPN list) stayed visible for client fingerprinting or service inference. This led to the comprehensive `Encrypted Client Hello (ECH)` protocol, encrypting the entire ClientHello. ECH distributes public keys and parameters via DNS `SVCB/HTTPS` resource records, more appropriate than ESNI's TXT records, bootstrappable over encrypted DNS transports, reducing sensitive metadata leakage. The unencrypted "outer" ClientHello carries only "innocuous" values [1].

2.3. ECH Technical Mechanisms

ECH operates by dividing the TLS ClientHello message into two parts: an "outer" ClientHello and an "inner" ClientHello [4]:

Outer ClientHello: This part is unencrypted and contains innocuous values for sensitive extensions, and a generic "outer SNI" (also known as a public name or fronting domain), extracted from the HTTPS SVCB record, visible to network observers, and used as a fallback if ECH fails.

Inner ClientHello: This part is encrypted using a public key retrieved by the client and contains the actual sensitive extensions, including the real target domain (the "inner SNI").

ECH uses the Hybrid Public Key Encryption (HPKE) standard for inner ClientHello encryption/decryption [5]. The server's ECH configuration, including HPKE public key, cipher suite, and public domain name, is conveyed via DNS. This DNS mechanism bootstraps ECH, allowing clients to discover encryption parameters before TLS handshake initiation.

This split ClientHello structure lets network observers only see the generic public name (e.g., `web.local` in the testbench, or `cloudflare-ech.com` in Cloudflare's deployment) in the outer SNI, while the actual target domain (e.g., `ech.test`) is hidden within the encrypted inner ClientHello [6]. This means multiple unrelated sites can appear indistinguishable to inspection tools if hosted by the same ECH-enabled provider, significantly enhancing privacy by obscuring the destination. This effect is especially relevant for companies like Cloudflare, where all ECH-enabled users could share the same `cloudflare-ech.com` domain. Sites that self-host their entire infrastructure still benefit from this technology, but the fronting domain would still maintain a 1:1 mapping. The privacy benefits in this scenario would not be as great as with shared ECH-compatible providers.

2.4. Challenges and Advanced ECH Mechanisms

Even with ECH, a potential challenge arises if only sensitive services adopt it, creating a "Do not stick out" problem where ECH usage itself becomes identifiable [1]. To address this, ECH-supporting clients always include the ECH extension: either a real ECH extension when server configuration is available, or a GREASE (Generate Random Extensions And Sustain Extensibility) ECH extension to mask which servers support ECH. Additionally, "Implicit ECH" allows clients to choose any outer SNI and config_id, further obfuscating ECH usage and preventing fingerprinting [7].

Implementing Implicit ECH requires servers to perform trial decryption across multiple potential keys, which adds computational complexity but significantly enhances the robustness of ECH against sophisticated traffic analysis and fingerprinting attempts. This demonstrates a nuanced understanding of privacy that extends beyond simple encryption, addressing the challenge of hiding the fact that privacy mechanisms are in use to avoid drawing unwanted attention and prevent network observers trying to filter or flag ECH traffic.

3. ECH Testbench Architecture and Experimental Setup

This section details the containerized testbench architecture designed to demonstrate ECH functionality through direct comparison of encrypted and plaintext TLS handshakes.

3.1. Overview of Testbench Design Principles

The ECH testbench is a self-contained environment demonstrating ECH privacy benefits through direct comparison of plaintext and ECH-TLS traffic, showcasing ECH's SNI and sensitive field protection. The architecture emphasizes simplicity and reproducibility, using containerization for deployment ease and consistent cross-environment setup.

The project supports both `podman compose` and `docker compose`.

3.2. Key Components and Their Functions

The testbench comprises three primary components, each with a distinct role in demonstrating ECH functionality:

3.2.1. DNS Server (CoreDNS). CoreDNS serves DNS records, including HTTPS SVCB records for `ech.test`. This HTTPS record contains ECH configuration (Base64-encoded ECHConfigList) for automatic client discovery of ECH-enabled connections [8]. HTTPS records standardize ECH configuration for service binding. The `coredns/zones/ech.test.zone` file defines the DNS zone with HTTPS record containing `ech` SvcParamKey for ECH configuration.

3.2.2. ECH Server (Nginx). A single Nginx server, specifically compiled with ECH support, handles both normal and ECH-enabled TLS requests to the same endpoint. The server decrypts the inner ClientHello for ECH requests to identify the real target domain (`ech.test`) while processing standard TLS requests normally [5].

ECH support implementation in Nginx requires utilizing an ECH feature branch of OpenSSL and Nginx sources, as native, out-of-the-box ECH support is still maturing. This highlights practical challenges in deploying bleeding-edge privacy protocols, often necessitating development branches or patched software [9].

 doi: 10.2313/NET-2025-11-3_20

3.2.3. ECH Client (curl). The curl client, specifically compiled with ECH support, generates test traffic for both ECH and plain TLS connections. It also facilitates packet capture with `tcpdump`. These `pcap` files can be inspected to see ECH and non-ECH traffic in action. The `clients/curl/scripts/benchmark.sh` script automates traffic generation for both ECH and plain TLS connections, ensuring a controlled and comparable setup.

3.3. Network Layout and Configuration

The testbench operates within a defined isolated network environment to simulate real-world interactions and ensure controlled experimentation. Table 1 summarizes the network configuration.

TABLE 1: Network Layout and IP Addresses

Component	IP Address	Role in Network
DNS Server	13.37.0.53	ECH config, HTTPS RR
ECH-enabled Nginx	13.37.0.10	Web target
Curl Client	13.37.0.50	Sends traffic

3.4. Operational Flow and Experimental Procedure

The ECH-enabled connection follows a distinct sequence of operations designed to ensure the encryption of sensitive ClientHello information:

DNS Query for ECH Configuration: The ECH-enabled client (curl [1]) initiates the process by querying the DNS server (CoreDNS) for the HTTPS record associated with `ech.test` [5]. The server's ECH public key from the ECH configuration is used to encrypt its subsequent ClientHello message.

ECH Config Retrieval: The DNS server responds with the HTTPS record, which includes the ECH configuration. This configuration contains `web.local` as the designated public fronting domain, along with the necessary Hybrid Public Key Encryption (HPKE) public key and cipher suite. The client parses this Base64-encoded configuration to prepare for the encrypted handshake.

ClientHello Construction: The client constructs a TLS handshake message with two distinct parts [4]:

- **Outer ClientHello:** The actual ClientHello that contains non-sensitive information, including `web.local` as the visible SNI and the `encrypted_client_hello` extension (type 65037)
- **Inner ClientHello:** Encrypted section containing the sensitive information like the real target `ech.test` as its SNI, embedded within the ECH extension

Server Processing: The ECH-enabled Nginx server receives the ClientHello, processes the Outer ClientHello, then attempts to decrypt the "inner" ClientHello using its corresponding private key, identified by the ECH config_id. Upon successful decryption, the server identifies the real target domain, presents the appropriate certificate and completes the handshake.

1. As curl currently does not support DoH with untrusted certs, the HTTPS record is manually fetched and passed to curl

3.4.1. Plain TLS Flow (Unencrypted Connection). In contrast, a plain TLS connection follows the traditional, unencrypted handshake process with a standard ClientHello containing the plaintext SNI directly specifying `ech.test`, allowing any on-path observer to extract the target domain.

3.4.2. Traffic Generation and Analysis. The testbench employs specific scripts for traffic generation and analysis. The `benchmark.sh` script automates the generation of both ECH and plain TLS traffic in the client using the ECH-enabled curl build, while the `analyze_traffic.sh` script processes the captured network traffic to extract and compare key fields such as SNI values and the presence or absence of the ECH extension.

4. Results and Discussion

This section presents the experimental findings from packet-level analysis of ECH versus plaintext TLS traffic, demonstrating ECH's effectiveness in concealing sensitive metadata.

4.1. Traffic Analysis: SNI Visibility and ECH Extension Presence

The experimental results from the testbench quantify ECH's privacy protection effectiveness. Running the benchmark will result in pcap files that contain all the network traffic between the client, the DNS, and the Nginx server.

Below are excerpts from the packet captures, which were opened in Wireshark for Mac.

Figure 1 shows the client requesting an `A` (and `AAAA`) record for `ech.test` and the DNS server responding. After the client receives the DNS response, a TCP connection is initiated. After the initial SYN, ACK procedure, the client sends the ClientHello, with the SNI visible (`ech.test`) to network observers.

13.37.0.50	13.37.0.53	DNS	68	Standard query 0x1797 AAAA ech.test
13.37.0.50	13.37.0.53	DNS	68	Standard query 0x2c90 A ech.test
13.37.0.53	13.37.0.50	DNS	138	Standard query response 0x1797 AAAA ech
13.37.0.53	13.37.0.50	DNS	126	Standard query response 0x2c90 A ech.te
13.37.0.50	13.37.0.10	TCP	74	47160 → 443 [SYN] Seq=0 Win=64240 Len=0
13.37.0.10	13.37.0.50	TCP	74	443 → 47160 [SYN, ACK] Seq=0 Ack=1 Win=
13.37.0.50	13.37.0.10	TCP	66	47160 → 443 [ACK] Seq=1 Ack=1 Win=64256
13.37.0.50	13.37.0.10	TLSv1…	583	Client Hello (SNI=ech.test)
13.37.0.10	13.37.0.50	TCP	66	443 → 47160 [ACK] Seq=1 Ack=518 Win=647
13.37.0.10	13.37.0.50	TLSv1…	1494	Server Hello, Change Cipher Spec, Appli

Figure 1: Plaintext TLS showing SNI exposure

Figure 2 shows the ECH-enabled client requesting the `A` (and `AAAA`) records but also the `HTTPS Resource Record`. After receiving the IPv4 and the `ECHConfig`, the client starts the TLS handshake with a ClientHello. This time, as the client used ECH, a network observer can only see the fronting domain `web.local` defined in the `ECHConfig`, hiding the real SNI encrypted in the "inner" ClientHello.

The Nginx server processes both (ECH and non-ECH) connections as requests to `ech.test`, while network observers perceive only the generic `web.local` domain for the ECH-enabled request.

ECH's privacy effectiveness depends on deployment scale: when `ech.test` remains the sole service behind

 doi: 10.2313/NET-2025-11-3_20

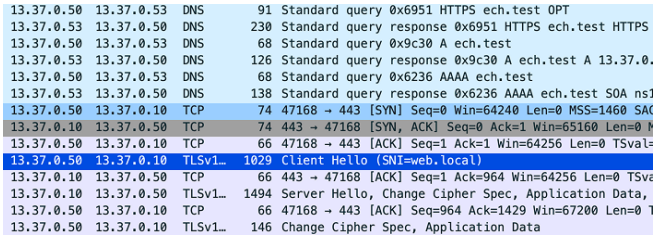


Figure 2: Demonstrating SNI concealment via ECH

a given outer SNI, the 1:1 mapping preserves some fingerprintability. However, when multiple unrelated services share the same fronting domain, such as `web.local` in this testbench or `cloudflare-ech.com` in Cloudflare's production deployment [4], all ECH-protected connections become indistinguishable to external observers, maximizing privacy through traffic aggregation.

4.2. Implications for Network Security

ECH effectively prevents several classes of passive attacks:

SNI-based censorship becomes infeasible as censors cannot identify target domains from ClientHello inspection. **Traffic correlation attacks** are significantly hindered when multiple services share the same outer SNI. **Pervasive monitoring** capabilities are reduced as ISPs and intermediaries lose visibility into user browsing patterns.

However, ECH does not protect against active attacks or endpoint-based monitoring. Network administrators must adapt security strategies toward behavioral analysis rather than content-based inspection of TLS metadata.

5. Related Work

This section positions our testbench contribution within the broader landscape of TLS privacy research and practical ECH evaluation approaches.

Privacy Analysis and Formal Verification. Hoang et al. [10] conducted the first comprehensive empirical study of domain name encryption privacy benefits, analyzing k-anonymity properties of 7.5M domains across nine global vantage points. Only 30% of domains achieve meaningful privacy (k>100), while 20% gain no benefit from one-to-one domain-IP mappings, establishing ESNI effectiveness limitations. Bhargavan et al. [11] provided the first mechanized formal analysis of TLS 1.3 privacy properties with ECH, using ProVerif to define client identity privacy, unlinkability, and extension privacy. They identified early ECH draft vulnerabilities and established theoretical foundations for encrypted handshake security.

Deployment Measurement Studies. Tsiatsikas et al. [12] measured ECH and ESNI adoption across top 1M domains, finding minimal ECH deployment despite theoretical benefits. Less than 19% of domains supported ESNI, with practically no ECH support as of 2023, highlighting deployment gaps.

Practical Implementation and Testing. While Cloudflare and other CDNs have enabled ECH in production environments [4], controlled evaluation frameworks remain limited. Hoang et al. [10] performed large-scale active DNS measurements to assess real-world co-hosting patterns, but focused on privacy quantification rather than demonstrating ECH functionality. Most evaluation approaches rely on theoretical analysis or internet-scale measurements rather than controlled, reproducible testbeds for direct ECH protocol comparison.

Censorship Circumvention Context. Recent work has examined ECH's role in circumventing SNI-based censorship [13], though ECH deployment remains insufficient to provide widespread censorship resistance. Research shows that authoritarian regimes have actively blocked ESNI traffic, emphasizing the need for robust ECH adoption and evaluation frameworks.

6. Conclusion and Future Work

This section synthesizes the key findings from our ECH testbench evaluation, assesses its broader implications for internet privacy, and outlines future research directions.

6.1. Summary of Findings

This paper analyzes Encrypted Client Hello (ECH) privacy benefits through a practical testbench framework demonstrating critical privacy enhancements in modern TLS communications. By comparing ECH-enabled and plain TLS connections, we prove ECH's effectiveness in hiding real target domains from network observers through encrypted inner ClientHello and generic "outer" SNI, cloaking real domain names.

The key findings include:

- ECH effectively conceals target domains (`ech.test`) behind generic public names (`web.local`), especially when multiple sites share the same outer SNI
- The same server infrastructure can seamlessly handle both ECH and plain TLS connections
- ECH presents significant challenges for traditional network security monitoring approaches

6.2. Significance for Internet Privacy

ECH eliminates the last major plaintext metadata leak in TLS 1.3, completing privacy protection by encrypting previously exposed ClientHello fields. The testbench demonstrates protection without functional compromises, maintaining TLS 1.3 compatibility while preventing passive domain monitoring. However, widespread adoption faces implementation complexity, evidenced by required experimental software builds.

Protocol effectiveness scales with adoption density as ECH maximizes privacy when multiple services share common outer SNIs, creating indistinguishable traffic for network observers.

ECH development reflects broader internet protocol design shifts toward privacy-by-design principles, where user privacy becomes fundamental rather than optional. Current deployment by major CDN providers like Cloudflare indicates ECH's transition from experimental to production-ready technology, with advancing browser support. This directly responds to the IETF's recognition of pervasive monitoring as an infrastructure attack.

 doi: 10.2313/NET-2025-11-3_20

6.3. Challenges and Limitations

While ECH provides substantial privacy benefits, challenges remain for successful deployment.

Implementation Complexity: ECH requires systematic DNS infrastructure, server software, and client application updates. Our testbench demonstrates experimental software needs, highlighting maturity challenges.

Adoption Incentives: The "Do not stick out" problem suggests ECH effectiveness depends on widespread adoption to avoid making ECH-enabled traffic a signal to adversaries.

Infrastructural Concentration: ECH offers greatest privacy gains when endpoints are shared, favoring large CDN deployment like Cloudflare, potentially incentivizing further internet infrastructure centralization.

6.4. Future Work and Extensions

Future research directions based on this work include:

Extended Testbench Scenarios: Expanding the testbench to include more complex scenarios such as testing ECH performance under varying network conditions, evaluating compatibility with different TLS proxies or middleboxes, and exploring the implications of Implicit ECH for further obfuscation [7].

Performance Analysis: Comprehensive latency and throughput measurements under varying network conditions, quantifying ECH's operational overhead in production-like environments.

Enterprise Security Solutions: Research into new methods for enterprise network security monitoring in an ECH-pervasive environment would be valuable, potentially exploring endpoint-based security solutions or privacy-preserving analytics techniques that can operate effectively without requiring plaintext SNI access.

Deployment Studies: Large-scale studies of ECH deployment challenges and adoption patterns in real-world environments.

Mitigating Infrastructural Concentration: Examine ECH deployment approaches that lessen reliance on dominant CDNs, thereby expanding the range of participating operators and enhancing end-user privacy.

References

[1] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, "TLS Encrypted Client Hello," Internet Engineering Task Force, Internet-Draft ietf-tls-esni-25, Jun. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-esni/25/

[2] NordVPN, "What is encrypted SNI, and how does it relate to censorship?" https://nordvpn.com/blog/encrypted-sni/, 2024, [Online; accessed June 21, 2025].

[3] S. Farrell and H. Tschofenig, "Pervasive Monitoring Is an Attack," RFC 7258, 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7258.html

[4] Cloudflare, "Encrypted Client Hello - the last puzzle piece to privacy," https://blog.cloudflare.com/announcing-encrypted-client-hello/, 2023, [Online; accessed June 21, 2025].

[5] T. Probst, "Decoding TLS Encrypted Client Hello extension," https://thibautprobst.fr/en/posts/ech/, 2025, [Online; accessed June 21, 2025].

[6] Security.com, "Navigating Encrypted Client Hello (ECH): Insights from RSAC 2025 Conference," https://www.security.com/expert-perspectives/navigating-encrypted-client-hello-ech-insights-rsac-2025, 2025, [Online; accessed June 21, 2025].

[7] N. Sullivan, "Implicit ECH Configuration for TLS 1.3," Internet Engineering Task Force, Internet-Draft draft-sullivan-tls-implicit-ech-00, Feb 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-sullivan-tls-implicit-ech/

[8] B. Schwartz, M. Bishop, and E. Nygren, "Bootstrapping TLS Encrypted ClientHello with DNS Service Bindings," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-svcb-ech-08, Feb 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-svcb-ech/

[9] Nginx, "Request for Encrypted Client Hello (ECH) Support in Nginx," https://github.com/nginx/nginx/issues/266, 2024, gitHub Issue #266, [Online; accessed June 21, 2025].

[10] N. P. Hoang, A. A. Niaki, N. Borisov, P. Gill, and M. Polychronakis, "Assessing the privacy benefits of domain name encryption," *CoRR*, vol. abs/1911.00563, 2019. [Online]. Available: http://arxiv.org/abs/1911.00563

[11] K. Bhargavan, V. Cheval, and C. A. Wood, "A symbolic analysis of privacy for TLS 1.3 with encrypted client hello," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: ACM, 2022, pp. 279–292. [Online]. Available: https://doi.org/10.1145/3548606.3559360

[12] Z. Tsiatsikas, G. Karopoulos, and G. Kambourakis, "Measuring the adoption of TLS encrypted client hello extension and its forebear in the wild," in *Computer Security. ESORICS 2022 International Workshops*, ser. Lecture Notes in Computer Science, vol. 13785. Cham: Springer, 2023, pp. 177–190. [Online]. Available: https://doi.org/10.1007/978-3-031-25460-4_10

[13] S. Wendzel, W. Mazurczyk, L. Caviglione, and A. Mileva, "A survey of internet censorship and its measurement: Methodology, trends, and challenges," 2025. [Online]. Available: https://arxiv.org/abs/2502.14945

ISBN 978-3-937201-85-6
DOI 10.2313/NET-2025-11-3
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)