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

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

Winter Semester 2025/2026

August 5, 2025 – March 8, 2026

Munich, Germany

Editors

Georg Carle, Marcel Kempf, Daniel Petri, Stefan Genchev

Publisher

Chair of Network Architectures and Services

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

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

Winter Semester 2025/2026

Munich, August 5, 2025 – March 8, 2026

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

Network Architectures
and Services
NET 2026-05-1

Proceedings of the Seminar
Innovative Internet Technologies and Mobile Communications (IITM)
Winter Semester 2025/2026

Editors:

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

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

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

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

Cataloging-in-Publication Data

Seminar IITM WS 25/26
Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)
Munich, Germany, August 5, 2025 – March 8, 2026
ISBN: 978-3-937201-86-3

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2026-05-1
Innovative Internet Technologies and Mobile Communications (IITM) NET 2026-05-1
Series Editor: Georg Carle, Technical University of Munich, Germany

# Preface

We are pleased to present to you the proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM) during the Winter Semester 2025/2026. 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 Regina Bayer with the paper *Efficient and Accurate Network Modeling with ML* and Sebastian Almeida Henriques da Silva with the paper *TCP Cookies in the Linux Kernel* .

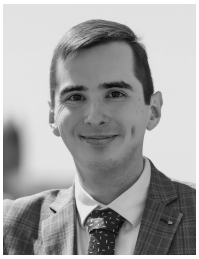
We hope that you appreciate the contributions of these seminars. If you are interested in further information about our work, please visit our homepage `https://net.in.tum.de`.

Munich, May 2026

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
*Technical University of Munich*

Christian Dietze
*Technical University of Munich*

Sebastian Gallenmüller
*Technical University of Munich*

Christopher Harth-Kitzerow
*Technical University of Munich*

Kilian Holzinger
*Technical University of Munich*

Marcel Kempf
*Technical University of Munich*

Holger Kinkelin
*Technical University of Munich*

Lorenz Lehle
*Technical University of Munich*

Michael Oberrauch
*Technical University of Munich*

Daniel Petri
*Technical University of Munich*

Nina Schwanke
*Technical University of Munich*

Johannes Späth
*Technical University of Munich*

Lion Steger
*Technical University of Munich*

Florian Wiedner
*Technical University of Munich*

# Seminar Homepage

`https://net.in.tum.de/teaching/ws2526/seminars/`

# Contents

## Blockseminar



## Seminar

# Happy Optometrist: A Summary of the Development and Implementation of Happy Eyeballs

Tim Albrecht, Tim Betzer*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: tim.albrecht@mytum.de, betzer@net.in.tum.de

***Abstract*—**

**IPv4 address space has become increasingly exhausted and workarounds such as Carrier Grade NAT have had to be introduced. This has led to IPv6 proportionally gaining importance. While IPv6 offers a larger address space, built-in security options, more efficient routing, as well as many other advantages, its adoption has been slow and riddled with problems. As a way to ease this transition, many clients and services have opted for a dual-stack approach, which aims to enable connectivity via both protocols.**

**Happy Eyeballs (HE) was introduced as a way to prioritize IPv6 in these setups while not worsening user experience. This is achieved by keeping IPv4 as a viable backup option in case of IPv6 failure. The first version of HE was proposed as a standard by the IETF in 2012. Since then HEv2 has replaced HEv1 and a draft of a third version is currently under development.**

**This paper aims to summarize the development of the HE protocol and related works, as well as evaluate the current state of HE and its function in the modern internet landscape. The main focus will be on the parameters defined in HE, its relation to newly introduced protocols like QUIC and how the basic concept of HE could be adopted in more contexts.**

***Index Terms*—Happy Eyeballs, IPv6, Dual-Stack**

## 1. Introduction

Dual-stack setups have established themselves as one of the most important tools in supporting the transition from IPv4 to an IPv6-only internet. By allowing connections with both address families, clients can use IPv4-only, IPv6-only and other dual-stack services while providers can offer their services to clients who might only support one of the two address families. One might assume that this practice would result in an overall improved user experience, since, if both client and service use a dual-stack setup, two options for establishing a connection are offered.

### 1.1. Dual-stack before Happy Eyeballs

The state of dual-stack devices before the introduction of HE is explained in detail by Bajpai and Schönwälder in [2], [3]. They find that in 2009, only around ~1% of ALEXA 1M websites announced `AAAA` entries. Of these, about 40% were not accessible via IPv6. Furthermore, for 12% of all websites with `AAAA` entries, more than half of their content failed over IPv6. References [2], [3] attribute this brokenness mostly to the majority of IPv6 connections using Teredo [4] or 6to4 [5], as shown by Google [1]. Native traffic overtook these in about March 2010 (see Fig. 1a). Additional studies [6] by Colitti et al. and [7] by Zander et al. find that "even in situations where relays work, Teredo / 6to4 add noticeable latency when compared to native IPv4 and IPv6" [3]. Of course, some innate brokenness can also be attributed to native IPv6 [8].

Still, the majority of devices tried establishing a connection over IPv6 first. This behavior is explained by the usage of `getaddrinfo()` [9], which returns a list containing both `A` and `AAAA` records. These are then ordered according to [10], resulting in all IPv6 addresses being pushed to top of the list. Consequently, any service aiming to establish a connection would try all IPv6 addresses first before falling back to IPv4.

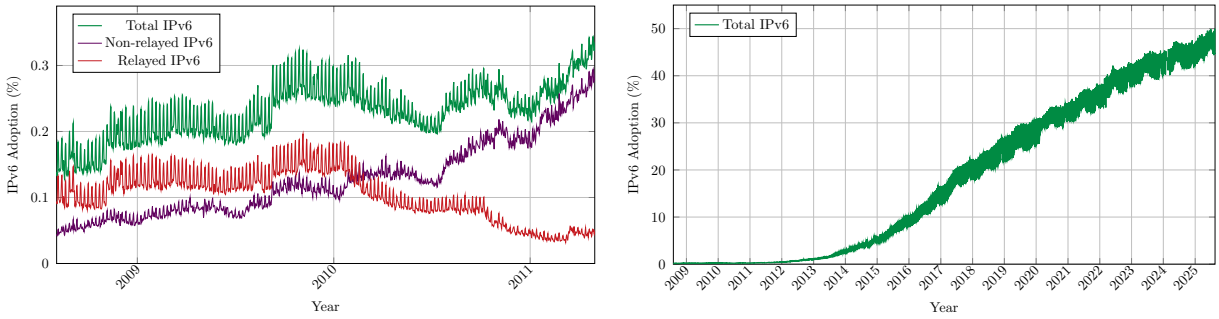
This convention affected user experience greatly. Repeated connection attempts over broken IPv6 were causing delays of around 20 s to 40 s, as was analyzed by Chown and Morse in [11], while a connection over IPv4 would have succeeded as normal. This behavior is extremely harmful, since it would discourage clients and hosts alike from using a dual-stack setup, thereby sabotaging the transition from IPv4 to IPv6.

Corroborating these findings, Google's IPv6 adoption statistics [1] show that less than 1% of all connections reaching Google were made over IPv6 until about the first quarter of 2013 (see Fig. 1).

### 1.2. Happy Eyeballs

The first version of the Happy Eyeballs protocol [12] was published by the IETF in 2012. As an RFC it proposes an internet standard that aims to improve user experience when working with dual-stack setups. The name stems from the goal to "make users' eyeballs happy" by reducing waiting time for websites to load. HEv1 defines that if a connection attempt over one address family does not succeed, another using the remaining one should be started quickly after. A recommended value for this delay will be discussed in 2.1. Since IPv6 is preferred, it will be the address family that is tried first. In case of failure, a fallback to IPv4 would only cause a few milliseconds of delay, massively improving user experience compared to the 20 s to 40 s of delay mentioned in [11].

 doi: 10.2313/NET-2026-05-1_01




(a) From August 2008 to May 2011

(b) From August 2008 to August 2025

Figure 1: Percentage of connections reaching Google over IPv6. Figure reproduced from [1]

The next iteration, HEv2 [13], was proposed in 2017. Concepts, such as the delay between connections, are further specified and additional parameters such as a Resolution Delay and a First Address Family Count were added. These help in further prioritizing IPv6 while still ensuring a safe and fast fallback to IPv4. Sattler et al. [14] show that HEv2 is not yet widely implemented.

HEv3 [15] is the latest version, being worked on by the IETF since 2023. It focuses less on the specific parameters of HE implementations and more on the selection of additional DNS records, which are `SVCB` and `HTTPS`. This influences the `L4` protocol being used and the ordering of the addresses that will be used to make connection attempts. Therefore, HEv3 does not iterate on the handling of IPv6 much but rather proposes new rules to optimize the selection of additional protocols with the goal of improving user experience.

As of 2025, IPv6 speed and reliability have significantly improved [3]. Accordingly, IPv6 adoption has continually trended upwards and is approaching almost 50% (see Fig. 1b), as reported by Google [1]. A number of studies, such as [3], [14] and [16] by Papastergiou et al., as well as [17] by G. der Kinderer, show that the focus of the HE concept is gradually shifting more towards optimization in contexts, where a decision between multiple protocols has to be made. This is corroborated by the HEv3 draft. While the underlying purpose of improved user experience is still present, HE is no longer used solely to decide between IPv6 and IPv4.

## 2. Background

This section provides technical details on each version of the HE protocol. It will summarize their functionality and discuss their differences.

### 2.1. Happy Eyeballs v1 (RFC 6555)

As the first HE protocol, HEv1 [12] aims to resolve the most harmful problems of dual-stack setups that were present at the time of its release in 2012. It identifies the multiple connection attempts over possibly broken IPv6 as the cause of "significant connection delay compared to an IPv4-only client" [12] and proposes a *Connection Attempt Delay* (CAD) of 150 ms to 250 ms (see Table 1) to combat this behavior.

A dual-stack client trying to establish a connection would first execute a DNS request. If it receives both an `AAAA` and an `A` record it should initiate a connection by sending a `SYN` segment over IPv6 first. If the client would not receive a `SYN/ACK` response from the server it would usually attempt further connections using the same IPv6 address [8] causing significant delays. This behavior may be explained by the usage of `TCP`, which is innately persistent [8]. In contrast, HEv1 proposes a fast fallback, should an IPv6 connection not succeed. This is achieved by immediately sending out a `SYN` segment over IPv4 after a specific amount of time, which is defined by the CAD. This does not mean that connections, which are already being tried, should be interrupted. Rather, all connections should be raced, with some addresses being prioritized. Only if one connection succeeds, should the other attempts be terminated. According to HEv1, clients must cache this information for about 10 min to avoid having to fall back during subsequent connection attempts. A behavior in cases, where both IPv6 and IPv4 fail and the client received multiple `AAAA` and/or `A` records is not yet defined.

In conclusion, HEv1 addresses the biggest problem of dual-stack setups when encountering broken IPv6, that being a significant connection delay, while still staying rather simple and mainly focusing on the fallback to IPv4.

### 2.2. Happy Eyeballs v2 (RFC 8305)

HEv2 [13] significantly refines the HE algorithm, expanding its functionality to cover the DNS request, among other . It introduces a *Resolution Delay* (RD), that defines how long a client should wait for an `AAAA` record response after already receiving an `A` record. If no `AAAA` record arrives, a connection attempt over IPv4 is initiated (see Fig. 2). HEv2 now also recommends behavior in cases where the DNS request yields multiple records of the same type. To this end, a new parameter, *First Address Family Count* (FAFC), is introduced. It defines how many addresses of the same address family should be tried before falling back to the other one. A value of 1 is recommended but 2 may also be used to favor a particular address family more strongly [13]. This practice is employed only by the Safari browser [14] to further prioritize IPv6. Should the fall back attempt fail, it is recommended to try all the remaining addresses from the favored address family. In cases where these fail as well, connections using the leftover addresses from the other family should be attempted.

Besides the RD, HEv2 introduces additional parameters, which specify the CAD. First established in HEv1

TABLE 1: Comparison of parameters defined for HEv1 [12], v2 [13] and the draft for v3 [15] as of September 2025. Table reproduced from [14]

| Parameter | HEv1 (2012) | HEv2 (2017) | HEv3 (2025-ongoing) |
|---|---|---|---|
| Considered protocols | IPv4, IPv6 | IPv4, IPv6, DNS | IPv4, IPv6, DNS, QUIC |
| DNS Records | – | `AAAA`, `A` | `SVCB`, `HTTPS`, `AAAA`, `A` |
| Resolution Delay | – | 50 ms | 50 ms |
| Address selection | IPv6 once, then IPv4 | alternating | IP family and L4 protocol |
| Fixed Conn. Attempt Delay | 150 - 250 ms | 250 ms | 250 ms |
| Min/Rec./Max when dynamic | – | 10 ms / 100 ms / 2 s | 10 ms / 100 ms / 2 s |

with a recommended value between 150 ms and 250 ms, now a static value of 250 ms is recommended. Other new parameters are a minimum and maximum CAD, with suggested values being 100 ms and 2 s respectively. The minimum CAD is defined to never be less than 10 ms. This helps to avoid cases, where high packet-loss rates cause congestion collapse [13]. Caching the round trip time (RTT) of prior connections allows this data to be used to influence both the ordering of addresses and the CAD, giving priority to those with a low delay.

### 2.3. Happy Eyeballs v3 (IETF Draft)

While HEv3 [15] is not yet an RFC, it is an IETF draft, in development since 2023. Currently, it does not modify any of the existing parameter values described in HEv2 but extends the functionality to include additional resource records. These are `HTTPS` and `SVCB`, which can provide further information, useful in prioritizing the better connection. This includes the protocol, distinguishing between traditional TCP + TLS and HTTP/3 [18] over QUIC [19], the latter being faster and more efficient. Furthermore, services may advertise support for Encrypted Client Hello (ECH) or include 'address hints', which supply IP addresses alongside additional routing information, potentially reducing the need for separate `A` and `AAAA` lookups. Overall, these newly introduced features allow clients to apply the HE algorithm more dynamically without relying solely on previously cached optimizations, since they are directly provided during DNS resolution.

## 3. Related Work

Several prior studies have investigated the HE protocol with respect to its effect on the IPv6 landscape, different implementations and their functionality, as well as how HE may be used as a tool for optimization in various situations. The following section will summarize some of them and outline the transformation the HE protocol has undergone in the 13 years since its introduction in 2012.

### 3.1. Happy Eyeballs as a fix for broken IPv6

Even before the standardization of the first HE version, article [20] by E. Aben was released in 2011. They compare the already existing implementations of different browsers, including Chrome, Firefox and Safari, especially in unison with two versions of Mac OS X. The article reveals that browsers like Chrome had already implemented an HE algorithm, while, e.g. Firefox did not. Additionally, a harmful behavior called 'hampering Eyeballs' is described, which was mostly exhibited by Mac OS X Lion and caused connections to prefer IPv4 even though IPv6 worked perfectly fine. In conclusion, they showed that HE can be a useful tool in improving user experience but might also introduce some noticeable drawbacks if not implemented correctly.

Similarly, article [8] by G. Huston also examines HE implementations as of 2012. Additionally, they first investigate IPv6 brokenness using Windows 7 and an Internet Explorer browser to find concrete delays of about 22 s in dual-stack setups. This is corroborated by the findings of Chown and Morse in [11], who examined similar delays using Internet Explorer, as well as delays of up to 45 s using browsers like Firefox or Safari. As for the implementations, [8], like [20], also investigates Safari with Mac OS X, Chrome and Firefox. It is shown that both Chrome and Firefox connect to the first address family received by a DNS query, not prioritizing IPv6. Furthermore, Firefox is praised for racing both protocols as soon as their DNS responses arrive, despite this behavior being prone to lead to 'hampering Eyeballs' [20]. In contrast, Chrome waits the recommended amount of 300 ms before falling back to the other address family. Chrome also displayed longer delays in cases where both protocols failed. Here, the need for new specifications like RD, FAFC and a minimum CAD, which were later introduced in HEv2 [13], is showcased. Lastly, it is discussed that binding too many ports when establishing a TCP connection might be harmful when using Carrier Grade NAT (CGN). In light of this, prioritizing IPv6 with a delay like Chrome's 300 ms is emphasized to be better than a true parallel implementation.

The most recent studies examining HE and IPv6 performance are [2] and [3], both written by Bajpai and Schönwälder, released in 2016 and 2019 respectively. These are long term studies measuring IPv6 adoption, brokenness and latency, as well as the effect of HE since 2013. Since [3] expands on [2], this paper will focus on the former. Similar to [8], they name Teredo [4] and 6to4 [5] as the main cause of IPv6 brokenness and find that it has significantly decreased from 40% in 2009 to 1.9% in 2019. Furthermore, they examine that from 2012 to January 2019, the number of ALEXA 1 M websites that announce `AAAA` record entries has increased from ~1% to ~19.2% [3]. Additionally, they name Google's adoption statistic, which has increased from less than 1% in 2012 to ~26% in Jan 2019 [3] and now to almost 50% (see Fig. 1b). Still, 27% of websites with `AAAA` entries suffer from partial IPv6 failure. Most important

 doi: 10.2313/NET-2026-05-1_01

to the functionality of the HE protocol, they find that for 99% of ALEXA 10 K websites (with `AAAA` entries), an IPv6 connection is preferred 96% of the time [3]. In 81% of these cases, this behavior prefers a slower IPv6 connection, which is almost always only ~1 ms behind. They demonstrate that a CAD of 150 ms, compared to the recommended 250 ms, would not significantly impact IPv6 preference. Therefore, they suggest specifying the CAD to a more useful value of 150 ms.

### 3.2. The state of Happy Eyeballs implementations

While studies like [8] and [20] investigate early HE implementations, the only study measuring these in recent years has been [14] by Sattler et al., using a local testbed, as well as an online tool which can be found under **https://www.happy-eyeballs.net/**. They expanded their investigation to include not only browsers but recursive resolvers and the terminal commands curl [21] and wget [22]. Their findings show that all implementations prefer IPv6 and that all except wget offer a fallback to IPv4. As for the CAD, curl, Firefox and Chromium based browsers use static values of 200 ms, 250 ms, and 300 ms respectively, while Safari employs a dynamic approach, exhibiting CADs ranging from 50 ms up to 2 s. Safari is also shown to be the only browser implementing an RD. It uses the recommended value of 50 ms. This results in both other browser types failing completely should no `A` record arrive, in spite of a DNS response providing a valid `AAAA` record. Additionally, only Safari implements its HE algorithm with a FAFC of 2 choosing to more heavily prioritize IPv6. In conclusion, most implementations are still using HEv1 with Safari being the only exception.

### 3.3. The role of Happy Eyeballs in a modern Internet landscape

As demonstrated by [3], discussed at the end of 3.1, HE is suggested to be further specified with a CAD of 150 ms. Additionally, the HEv3 draft [15] incorporates new protocols. Both of these suggestions demonstrate that Happy Eyeballs can be used for optimization. Similarly, reference [16] also discusses how HE can be used to decide between two `L4` protocols, these being TCP and SCTP. They find that this approach works, while introducing acceptable additional CPU load for servers, which can be improved with caching. This validates HEv3's idea of using the HE algorithm to not only decide between IPv6 and IPv4, but also different HTTP versions, since HTTP/3 over QUIC uses UDP instead of TCP. Lastly, reference [17] shows how the HE approach can be extended to be used by other protocols, XMPP in this case, with the general goal of improving user experience.

## 4. Conclusion

The Happy Eyeballs protocol was introduced during a time of prevalent IPv6 issues. Its use as a fix for these problems has been documented to work very well. HE succeeds in significantly improving user experience while not slowing down the transition from IPv4 to IPv6 and is therefore seeing widespread implementation. Still,

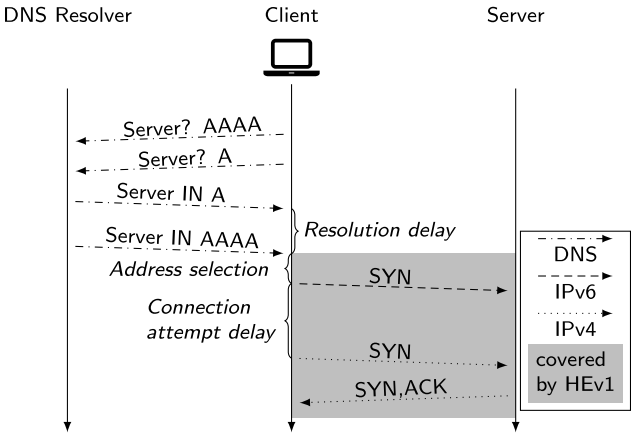



Figure 2: Happy Eyeballs (v2) connection process. Figure reproduced from [14]

HEv1 had to be further specified in HEv2 to more safely guarantee a fallback to IPv4 and optimize the algorithm with concrete values that cover additional cases. For that reason, it is alarming to see that mostly only HEv1 has been implemented, especially since HEv2 secures safe connections to IPv6-only services [14], which are the end goal in the ongoing transition from IPv4.

As of 2025, the state of IPv6 has greatly improved, being used more often with connection times on par with IPv4 most of the time. Furthermore, the internet infrastructure has become increasingly complicated with the introduction of various different protocols. In such an environment HE offers a very powerful approach in selecting between these, being used more for optimization, than only as a tool for consolidation brokenness a specific protocol. Since HE uses a fairly simple idea of racing two options, giving priority to one of them, it could find even more widespread adoption. Future work might further expand on this functionality or integrate this concept into other applications.

In conclusion, Happy Eyeballs was introduced as a fix for broken IPv6 and has developed into an algorithm to select between different protocols, with the option to prioritize some of them.

## References

[1] Geoff Huston. (2025) Google ipv6 statistics. Accessed: 2025-09-21. [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html

[2] V. Bajpai and J. Schönwälder, "Measuring the effects of happy eyeballs," in *Proceedings of the 2016 Applied Networking Research Workshop*, ser. ANRW '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 38–44. [Online]. Available: https://doi.org/10.1145/2959424.2959429

[3] V. Bajpai and J. Schönwälder, "A longitudinal view of dual-stacked websites—failures, latency and happy eyeballs," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 577–590, 2019.

[4] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," RFC 4380, Feb. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4380

[5] B. E. Carpenter and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds," RFC 3056, Feb. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3056

[6] L. Colitti, S. H. Gunderson, E. Kline, and T. Refice, "Evaluating ipv6 adoption in the internet," in *Passive and Active Measurement*, A. Krishnamurthy and B. Plattner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 141–150.

[7] S. Zander, L. L. Andrew, G. Armitage, G. Huston, and G. Michaelson, "Investigating the ipv6 teredo tunnelling capability and performance of internet clients," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, p. 13–20, Sep. 2012. [Online]. Available: https://doi.org/10.1145/2378956.2378959

[8] Google. (2012) Bemused eyeballs: Tailoring dual stack applications for a cgn environment. Accessed: 2025-09-20. [Online]. Available: https://labs.apnic.net/index.php/2012/05/13/bemused-eyeballs-tailoring-dual-stack-applications-for-a-cgn-environment/

[9] (2025) getaddrinfo(3) – linux manual page. man7.org Linux man-pages project. Accessed: 2025-09-21. [Online]. Available: https://man7.org/linux/man-pages/man3/getaddrinfo.3.html

[10] R. P. Draves, "Default Address Selection for Internet Protocol version 6 (IPv6)," RFC 3484, Mar. 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3484

[11] T. Chown and J. Morse, "Experiences of host behavior in broken ipv6 networks," Presentation at IETF 80, V6OPS Working Group, March 2011, slides–PDF. [Online]. Available: https://www.ietf.org/proceedings/80/slides/v6ops-12.pdf

[12] D. Wing and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts," RFC 6555, Apr. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6555

[13] D. Schinazi and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency," RFC 8305, Dec. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8305

[14] P. Sattler, M. Kirstein, L. Wüstrich, J. Zirngibl, and G. Carle, "Lazy eye inspection: Capturing the state of happy eyeballs implementations," *arXiv preprint*, vol. arXiv:2412.00263v3, November 2024. [Online]. Available: https://arxiv.org/abs/2412.00263

[15] T. Pauly, D. Schinazi, N. Jaju, and K. Ishibashi, "Happy Eyeballs Version 3: Better Connectivity Using Concurrency," Internet Engineering Task Force, Internet-Draft draft-ietf-happy-happyeyeballs-v3-01, Jul. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-happy-happyeyeballs-v3/01/

[16] G. Papastergiou, K.-J. Grinnemo, A. Brunstrom, D. Ros, M. Tüxen, N. Khademi, and P. Hurtig, "On the cost of using happy eyeballs for transport protocol selection," in *Proceedings of the 2016 Applied Networking Research Workshop*, ser. ANRW '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 45–51. [Online]. Available: https://doi.org/10.1145/2959424.2959437

[17] G. der Kinderen, "Happy eyeballs." [Online]. Available: https://xmpp.org/extensions/xep-0495.html

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

[19] 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

[20] Emile Aben. (2011) Hampering eyeballs - observations on two "happy eyeballs" implementations. Accessed: 2025-09-17. [Online]. Available: https://labs.ripe.net/author/emileaben/hampering-eyeballs-observations-on-two-happy-eyeballs-implementations/

[21] D. Stenberg, "curl," https://curl.se/, 2024, retrieved 2025-09-22.

[22] T. Rühsen, D. Shah, and G. Scrivano, "Gnu wget," https://www.gnu.org/software/wget/, 2025, retrieved 2025-05-01.

# TCP Cookies in the Linux Kernel

Sebastian Almeida Henriques da Silva, Lion Steger*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: sebastian.silva@tum.de,stegerl@net.in.tum.de

***Abstract*—This paper examines TCP SYN cookies as a defence against backlog exhaustion during connection setup. The central question is whether any server-side state is created while cookie mode is active before the final ACK is verified. We also outline two secondary issues, namely the feasibility of computing a valid cookie in advance for a given connection key and the CPU cost of cookie processing at high connection attempt rates. We analyze the Linux 6.16 TCP stack by tracing the passive open path and the cookie routines, and we validate the readings with kernel counters under controlled floods. We find that Linux keeps no persistent per-connection state between the initial SYN and the validated final ACK while cookie mode is active. In an off-path model, blind one-round-trip completions are implausible, since validation reduces to an effective 24-bit search per minute, and kernel exploits are likewise implausible as they require full system control. The per-attempt hashing work is small compared to packet processing in the network stack, which remains the practical bottleneck.**

***Index Terms*—TCP syn cookies, syn flooding, listen backlog exhaustion, Linux kernel**

## 1. Introduction

Transmission Control Protocol is the dominant reliable transport on today's Internet. It provides ordered delivery, congestion control and connection management by means of a three-way handshake. During this exchange, the server retains a provisional state after receiving a SYN and before the final ACK arrives [1]. The design is robust to ensure reliability, but it exposes a simple vector for denial of service. The technical document RFC 4987 describes the effect succinctly: "The SYN flooding attack is a denial of service method affecting hosts that run TCP server processes" [2]. By forcing the listener to keep many half-open entries, an attacker can exhaust the finite backlog and block legitimate clients [2].

SYN cookies were introduced in the 1990s to keep services available under flood conditions. Instead of reserving a backlog entry on a SYN, the listener encodes verification data into its initial sequence number and postpones allocation until the final acknowledgement. The original note states the intended outcome plainly: a server using cookies does not "have to drop connections when its SYN queue fills up", at the cost of rejecting some TCP options [3]. Modern kernels enable cookies automatically under stress, for example in Linux [4].

This paper revisits SYN cookies in the Linux kernel and addresses three research questions: (i) Does a listener create any server-side state between the initial SYN and the verified final ACK while cookie mode is active (see methodology §3.1 and evaluation §4.1). (ii) Is it feasible to compute a valid cookie in advance for a chosen connection key and a time counter so that the handshake can be completed with only the final acknowledgement (see methodology §3.2 and evaluation §4.2). (iii) What CPU cost do cookie generation and validation introduce at high connection attempt rates? (see methodology §3.3 and evaluation §4.3).

The next section summarizes the attack surface and the defence, after which we discuss these questions in turn.

## 2. Background and Related Work

This section recalls the TCP parts relevant for connection setup and the SYN-flood vulnerability, then summarizes SYN cookies as implemented in modern systems.

### 2.1. How TCP works

TCP segments carry ports, sequence numbers, acknowledgements, flags, and options in a fixed header with optional extensions [1]. Figure 1 shows only the fields relevant for the handshake and option negotiation.

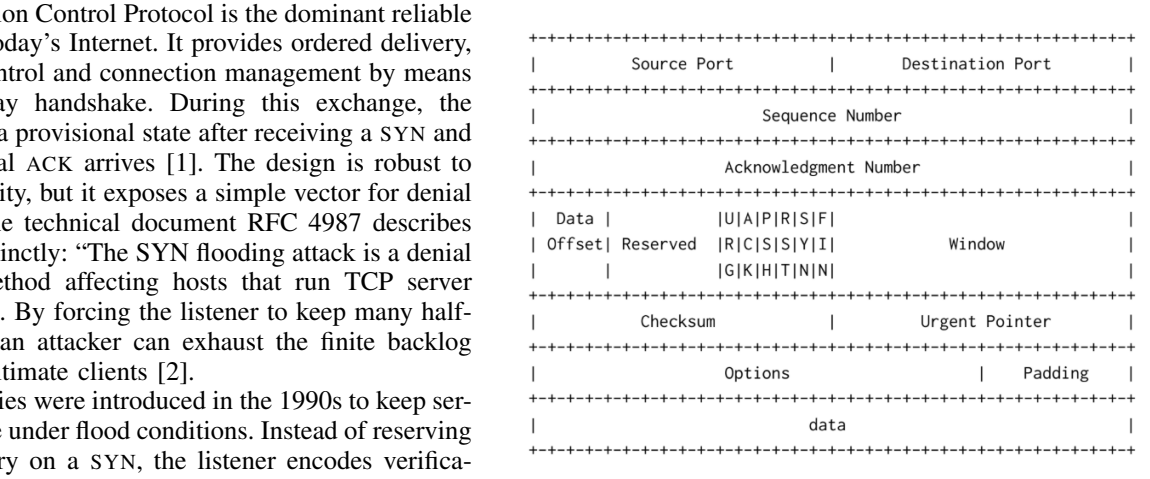



Figure 1: TCP header fields relevant to connection setup and options [5].

A new connection uses the three-way handshake: the client sends a SYN with initial sequence number $x$; the server replies with SYN–ACK and its initial sequence number $y$; the client completes by acknowledging $y + 1$. We refer back to this standard handshake in what follows; a sketch is in Figure 2.

 doi: 10.2313/NET-2026-05-1_02

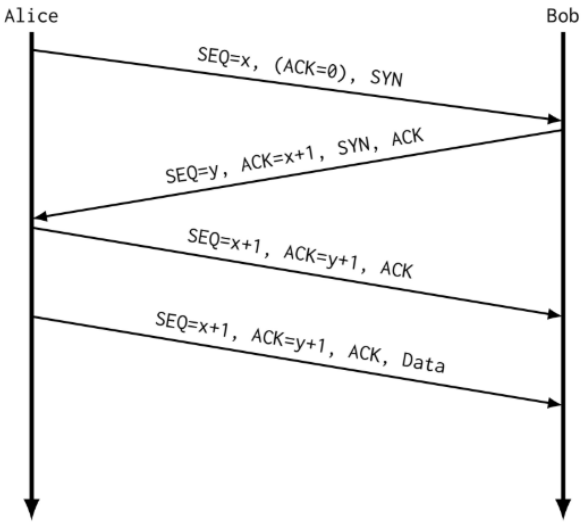



Figure 2: TCP three-way handshake. The server allocates temporary state after the SYN, which leaves the backlog vulnerable to flooding [5].

Two properties matter for both attack and defence: a listener must remember enough to validate the final ACK, and the listen backlog is finite. Too many half-open entries delay or drop new connections.

## 2.2. SYN floods: the attack

A SYN flood sends many SYNs, often with spoofed sources, and never completes the handshake. Each SYN forces the server to allocate a small temporary connection object (called a *request socket*), which represents state kept on the server side until the handshake either completes or times out. Enough fake entries fill the backlog and block real clients [2]. Figure 3 shows the effect.

**Threat model and scope:** We consider an off-path attacker, meaning an attacker that can send large volumes of spoofed SYN segments and so-called *blind* final ACKs. By blind, we mean that the attacker cannot see the server's replies and therefore must guess values such as sequence numbers without feedback. We study the IPv4 passive open path, the sequence of kernel functions that handle incoming SYN segments on a listening socket. In Linux, this path includes logic to create or validate a temporary request socket depending on backlog state, followed by decision routines that determine whether to enable cookie mode. Our analysis is based on Linux 6.16 with timestamps enabled and cookie mode triggered either by backlog overflow or forced via `net.ipv4.tcp_syncookies = 2`. Our criterion for "no state" is the absence of any persistent per-connection object (such as a `request_sock`) between the SYN and the validated final ACK in either the request queue or the accept queue. On-path adversaries, middlebox interference, and transport features that require full option negotiation are out of scope for this paper.

## 2.3. SYN cookies: the defence

SYN cookies remove server state during the vulnerable phase. When cookie mode is active, the server encodes

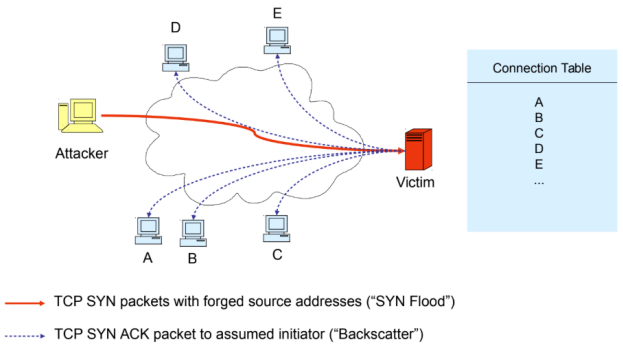



Figure 3: Backlog exhaustion via half-open connections [5].

the minimum needed information into its initial sequence number, replies with SYN–ACK, and defers allocation until the final ACK [3].

Operationally, a cookie is a keyed function of the four-tuple, a coarse time counter, the peer's initial sequence number (ISN), and a compact encoding of options (e.g., MSS). On the final ACK, the server recomputes and validates the cookie and recovers the encoded option. This protects the backlog at the cost of extra hashing and limits on option negotiation [3], [4]. The Linux kernel reflects these choices in `net/ipv4/syncookies.c` [4], and RFC 4987 discusses pros and cons in the wider mitigation set [2].

## 2.4. Cookie construction in Linux

Linux TCP SYN cookies are a keyed function of the four-tuple, a coarse time counter, and the peer's initial sequence number, with compact encodings for a minimal option set. The implementation uses a small constant number of `SipHash-2-4` invocations and integer combination in `net/ipv4/syncookies.c`.

**Inputs:** Let $(s_{\text{addr}}, d_{\text{addr}}, s_{\text{port}}, d_{\text{port}})$ be the four-tuple, let $x$ be the peer ISN from the SYN, and let $t$ be a coarse time counter with about one-minute steps. The code computes a keyed hash over these values and combines it with the peer ISN and the time counter to form the server cookie. The source comment summarizes the construction succinctly: "Compute the secure sequence number: `HASH(...) + sseq + (count * 2^24)`" [4]. Here `count` is the minute counter, and the multiplication by $2^{24}$ shifts the counter into the top bits of the server ISN.

**Option packing:** The MSS is encoded by index into a fixed table declared in the source as "`static __u16 const msstab[] = {536, 1300, 1440, 1460}`" [4]. When TCP timestamps are enabled, Linux packs a compact option summary into the low bits of the SYN–ACK timestamp. The code comment states: "`TCP Timestamp: 6 lowest bits of timestamp sent in the cookie SYN-ACK stores TCP options.`" [4]. These six bits comprise a four bit window scale field where the value `0xF` signals no scaling, plus one bit each for SACK permission and ECN capability [4]. If timestamps are not in use, only the limited MSS choice is retained.

**Creation and validation:** While cookie mode is active, the listener does not allocate a request queue entry on

receipt of the SYN. It replies with a SYN–ACK whose ISN is the cookie defined above. On the final ACK, the kernel recomputes and age checks the cookie, decodes the MSS index and the timestamp option bits, and only then creates the child socket on the accept path. The reconstruction occurs in `cookie_v4_check()` and the child is created by `tcp_get_cookie_sock()` [4].

# 3. Methodology

To answer our research questions, we now outline the steps we took to trace the kernel code paths, design our experiments, and build a simple cost model.

## 3.1. Server-side state under SYN cookies

To verify whether Linux TCP SYN cookies avoid server-side state creation during the half-open phase, we performed a code-level analysis of the upstream Linux kernel TCP stack. We focused on the IPv4 passive open path and the SYN-cookie implementation. Specifically, we traced the control flow on receipt of a SYN segment through `tcp_conn_request()` in `net/ipv4/tcp_input.c`, the flood-handling decision in `tcp_syn_flood_action()`, and the cookie construction and validation routines in `net/ipv4/syncookies.c`. We recorded the conditions under which SYN cookies are activated, the exact constants used in cookie encoding, and whether a `request_sock` and associated queue entries are retained prior to the final ACK. Our criterion for "no state" is the absence of any persistent per-connection object in the SYN (request) queue or accept queue between the SYN and the final ACK.

## 3.2. Feasibility of a one-RTT handshake by cookie prediction

**Method overview:** We distinguish two possible avenues to attempt a blind one-RTT handshake.

**(1) Brute force guessing:** The attacker repeatedly issues blind final ACKs, each with a guessed cookie value. Since validation checks a $2^{24}$ space within the active one-minute counter bucket, this reduces to a probabilistic search problem.

**(2) Kernel compromise:** The cookie value is derived from a per-boot secret, declared in `syncookie_secret` in `net/ipv4/syncookies.c`. This key is randomized at boot and never exported to user space. Only by modifying kernel code (e.g., replacing the random key with a fixed constant and rebuilding the kernel) could an attacker fully control cookie generation. Such a step presupposes privileged access and is therefore equivalent to total host compromise.

**Simulation setup:** We forced cookie mode with `net.ipv4.tcp_syncookies=2`. A local listener bound to a fixed port while kernel counters were tracked with `nstat` and socket states monitored with `ss`. For brute force, experiments aligned to the start of a fresh minute and then issued $N$ blind final ACKs per bucket. For kernel modification, we rebuilt a Linux kernel after replacing the secret with a constant, which allows deterministic reproduction of cookies for given four-tuples.

## 3.3. Analytical cycle cost model

When SYN cookies are active, Linux replaces request queue allocation with keyed hash work. On receipt of a SYN, the listener performs two calls to `siphash_4u32`; on the final ACK, it performs two more and reconstructs the minimal context [4]. Let $H$ be the cycle cost of a single `siphash_4u32` on short inputs. Published timings report about 140 cycles for 16 bytes on an AMD FX 8150 class CPU, with similar magnitudes on contemporary cores [6]. We model

$$C_{\mathrm{SYN}} \approx 2H + c_{\mathrm{syn}},$$
$$C_{\mathrm{ACK}} \approx 2H + c_{\mathrm{ack}},$$
$$C_{\mathrm{total}} \approx 4H + c,$$

where $c_{\mathrm{syn}}$ and $c_{\mathrm{ack}}$ are constant bookkeeping costs such as option handling and MSS lookup, independent of backlog size.

# 4. Evaluation

This section reports the outcome of our analysis. We examine when SYN cookies are activated, whether blind one-RTT completions are feasible, and the computational cost introduced by cookie processing.

## 4.1. Server-side state under SYN cookies

**Activation conditions:** On receipt of a SYN, `tcp_conn_request()` consults `net.ipv4.tcp_syncookies`. If set to 2, the listener unconditionally operates in cookie mode; if set to 1, cookie mode is entered only when the SYN request queue is full. Queue fullness is tested via `inet_csk_reqsk_queue_is_full(sk)`, which "returns `inet_csk_reqsk_queue_len(sk) > READ_ONCE(sk->sk_max_ack_backlog)`" [7]. The function `tcp_syn_flood_action()` is documented to "Return true if a syncookie should be sent" [8] and finalizes the decision to send cookies. This matches the documented semantics of `tcp_syncookies` (0 disabled, 1 on overflow, 2 always).

**State retention prior to ACK:** When cookie mode is active, the kernel initializes a temporary `request_sock` (256 bytes on a standard `x86_64` build) to compute the SYN–ACK, but does not attach it to the listener or insert it into the SYN queue. Immediately after transmitting the SYN–ACK, the temporary request object is freed, and control returns without queuing any per-connection state [4]. Only upon receipt of a valid final ACK does `cookie_v4_check()` reconstruct sufficient context and `tcp_get_cookie_sock()` either create the child socket or free the request and return `NULL`, at which point state is materialized and the usual accept path proceeds [4].

**Implications:** Under SYN cookies, the request queue occupancy is held constant during a flood, since no half-open entries are retained. The design avoids memory exhaustion from unacknowledged SYNs at the cost of reduced option fidelity (limited MSS set and option encoding) and the known protocol trade-offs for SYN cookies. Based on our code inspection the amount of server-side persistent state created per incoming SYN before a valid ACK is zero.

 doi: 10.2313/NET-2026-05-1_02

### 4.2. Feasibility of cookie prediction within one RTT

**Observed result:** In all blind trials we observed no sockets in `ESTAB` on the listener port. `ESTAB` is the TCP "connected" state reached only after a full three-way handshake, so its absence means no successful prediction occurred. We did see resets increase, which is expected when the final ACK carries a wrong cookie. Normal (non-blind) connections behaved as usual.

**Kernel mod check:** We rebuilt the kernel with a fixed `syncookie_secret` and disabled the one time randomization so that cookies are deterministic for a given four tuple and minute. We then used a small user space script that starts a local server, captures one handshake on loopback, and computes a predicted cookie from the observed addresses, ports, and an MSS index. With `tcp_syncookies=2`, the predicted value differed from the kernel cookie; therefore, a one RTT handshake did not follow. We verified the tuple, the minute counter, the address word order, and the setting of `tcp_syncookies`, and we repeated the run many times with the same result. It may work in theory, but after numerous attempts we did not find a way to make this approach succeed.

**Probability model:** Each blind guess has probability $p = 2^{-24}$. After $N$ guesses in one minute, $P(\text{at least one hit}) \approx 1 - e^{-N/2^{24}}$. Thousands of guesses yield essentially zero hits; about $1.16 \times 10^7$ per minute are needed for $50\,\%$, and $\approx 5.0 \times 10^7$ for $95\,\%$. This matches the negative result above.

```
static siphash_aligned_key_t syncookie_secret[2] = {
    {.key = {4242ULL, 2121ULL}},
    {.key = {4242ULL, 2121ULL}}
};
```

Figure 4: Kernel built with a fixed cookie secret for the determinism check.

### 4.3. Cost model results

Instantiating the model with $H \approx 140$ gives $C_{\text{total}} \approx 560$ cycles per completed attempt, which is about $0.16$ to $0.28\,\mu$s on $3.5$ to $2.0\,$GHz CPUs, plus the constant terms [6].

**Throughput bound from hashing alone:** For a core with clock $f$, hashing alone allows roughly $f/(4H)$ attempts per second. With $f = 3\,$GHz and $H = 140$, this is about $5.3 \times 10^6$ attempts per second.

**What actually dominates:** Moving and interpreting packets are the slow part in practice. Empirical analysis of the Linux host network stack shows that packet I/O and data movement dominate CPU time at high rates, rather than small per packet arithmetic [9].

**Scaling considerations:** Since stack processing and memory traffic dominate, throughput scales mainly with the number of cores that can process packets and with NIC queue capacity, while the small per packet hash cost has only marginal influence on end to end throughput [9].

## 5. Conclusion and Future Work

This paper examined TCP SYN cookies in Linux 6.16 as a defence against backlog exhaustion during connection setup. Tracing the passive open path and the cookie routines shows that, while cookie mode is active, the listener retains no persistent per-connection state between the initial SYN and the validated final ACK [4], [8]. A temporary request object used to craft the SYN-ACK is freed without being queued, and state is materialized only after a valid cookie is acknowledged [8]. We described the cookie construction as a keyed function of the connection four-tuple, the peer ISN, and a minute granularity counter, with compact option encodings via an MSS table and timestamp bits [4]. In an off-path model, blind one-RTT completions are implausible: validation reduces to an effective $2^{24}$ search per minute bucket [4], implying about $1.16 \times 10^7$ guesses for a $50\,\%$ success chance. Similarly, a kernel exploit requires full system control and is therefore equally implausible, albeit in a different sense. A simple micro-cost model indicates that the added hashing work is on the order of a few hundred cycles per attempt [6] and is unlikely to dominate end-to-end processing in the stack.

**Future Work:** Useful extensions include measuring costs and limits under real workloads and hardware, extending the analysis to IPv6 and different timestamp or option configurations, and studying interactions with NIC offloads, SYN proxying, and middleboxes. It would also be valuable to quantify the practical impact of reduced option fidelity under floods and to evaluate secret rotation and time-bucket sizes against active attackers.

## References


[1] J. Postel, "Transmission Control Protocol," Internet Engineering Task Force, Tech. Rep. RFC 793, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/rfc/rfc793

[2] W. M. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," Internet Engineering Task Force, Tech. Rep. RFC 4987, Aug. 2007. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4987

[3] D. J. Bernstein. (1996) SYN Cookies. Accessed 2025. [Online]. Available: https://cr.yp.to/syncookies.html

[4] (2024) Linux kernel source, syncookies.c. Accessed 2025. [Online]. Available: https://elixir.bootlin.com/linux/v6.16/source/net/ipv4/syncookies.c

[5] G. Carle, H. Kinkelin, L. Steger, K. Glas, and M. Oberrauch, "02_TCP_Attacks.pdf, Network Security (NetSec) Lecture Slides," 2024, iN2101 – Winter Semester 2024/25, Chair of Network Architectures and Services, Technical University of Munich. Lecture material.

[6] J. Aumasson and D. J. Bernstein, "SipHash: A Fast Short-Input PRF," in *Progress in Cryptology – INDOCRYPT 2012*, ser. Lecture Notes in Computer Science, vol. 7668. Springer, 2012, pp. 489–508. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-34931-7_28

[7] (2024) Linux kernel source, inet_connection_sock.h. Accessed 2025. [Online]. Available: https://elixir.bootlin.com/linux/v6.16/source/include/net/inet_connection_sock.h#L286

[8] (2024) Linux kernel source, tcp_input.c. Accessed 2025. [Online]. Available: https://elixir.bootlin.com/linux/v6.16/source/net/ipv4/tcp_input.c

[9] Q. Cai *et al.*, "Understanding Host Network Stack Overheads," in *USENIX ATC*, 2021. [Online]. Available: https://www.cs.cornell.edu/~qizhec/paper/tcp_2021.pdf

# Challenges and Strategies for Transitioning TLS to Post-Quantum Cryptography

Moritz Beckel, Holger Kinkelin*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: moritz.beckel@tum.de, kinkelin@net.in.tum.de

***Abstract*—With Harvest-Now-Decrypt-Later attacks, quantum computing already creates vulnerabilities in today's systems and with increasing research on quantum computing, other attacks might soon become feasible against classical cryptography. Fortunately, post-quantum cryptography already exists, and it works with new mathematical problems that are not efficiently solvable by quantum computers. Transport Layer Security (TLS) is one of the most critical security protocols, so integrating post-quantum cryptography is imperative. In this work, we examine what parts of the TLS protocol are vulnerable to quantum-based attacks, and we evaluate different replacements in terms of standardization, security, and performance. Finally, we examine a strategy to efficiently migrate to a TLS implementation with post-quantum cryptography.**

***Index Terms*—TLS, Post-quantum Cryptography**

## 1. Introduction

TLS, responsible for encryption and authentication of endpoints, is one of the most used protocols on the internet, making security in this protocol especially important [1]. Unfortunately, this has not always been the case, as there have been several well-documented attacks over the last few years on the TLS protocol (e.g., the BEAST attack, which abuses issues of the TLS 1.0 implementation of Cipher Block Chaining) [2]. Even though mitigations for TLS vulnerabilities are proposed, there is still the problem of older TLS versions on the internet. Because of end-of-life software, many endpoints that do not support up-to-date implementations of TLS still exist, which is why it is so important to not only look at current vulnerabilities but also proactively mitigate future vulnerabilities [1].

One significant case is the capabilities of future quantum computers, which can break classical Public-Key Cryptography (PKC). A typical example for PKC is the RSA algorithm, which uses the integer factorization problem to make decryption without the key a hard problem [3]. However, in 1994, Peter Shor proposed an algorithm that can solve several mathematical problems in polynomial time, including the integer factorization problem [4]. Since the underlying mathematical problem of RSA is efficiently solvable, the cipher is no longer considered secure once quantum computers are powerful enough. To break an RSA key with 2048-bit length, a future quantum computer would need roughly 4000 qubits, though, since current quantum computers exhibit noise with increasing qubits, the real number is likely higher [5]. Current estimates suggest the needed amount of qubits to be fewer than one million [6].

PKC is used in the TLS protocol to ensure the authenticity of a connection. Thus, a sufficiently powerful quantum computer can break authentication and even silently read alongside a TLS connection between two endpoints. What makes these kinds of attacks especially critical is that they already pose a threat to current TLS connections. With a Harvest-Now-Decrypt-Later attack, big institutions that have access to internet infrastructure can eavesdrop and store vast amounts of messages [7]. These encrypted messages can be recovered once a sufficiently powerful quantum computer is developed. Fortunately, several cryptographic ciphers are secure even with the use of a quantum computer. However, these so-called post-quantum cryptographic ciphers (PQC) have only recently been standardized and do not pose the historical strength that classical PKC has, as classical PKC has been well-studied and used for a long time [8].

We aim to explore the latest version of the TLS protocol (currently version 1.3) and analyze which operations can be broken using a quantum computer. We then evaluate the current state of PQC, the currently available algorithms, their performance, and the security impact on the TLS protocol. We examine the changes that need to be made to the protocol to efficiently support PQC and successfully mitigate attacks against adversaries with sufficient quantum computing power. Finally, we look at how to efficiently migrate to a TLS implementation with PQC and what is currently available.

Apart from TLS, there are many other areas where PQC must be integrated, which is why this topic has been heavily researched. Related research includes studies on the integration of PQC into several network protocols, such as DNSSEC [9] and IPSEC [10]. There are also approaches, such as POST [11], that aim to develop schemata to support PQC in the automotive industry.

## 2. Impact on the TLS protocol

TLS is a network protocol that works on top of the TCP protocol. Its main goal is to ensure confidentiality, authenticity, and integrity. We now examine the inner workings of TLS and single out those cryptographic operations/primitives that are not strong enough to withstand a post-quantum attack and need to be replaced by newer PQC. We will do so by observing a typical connection establishment of the TLS 1.3 protocol, which is defined in RFC 8446 [12]. As TLS is an extensive protocol with many extensions, we examine a connection where only

 doi: 10.2313/NET-2026-05-1_03

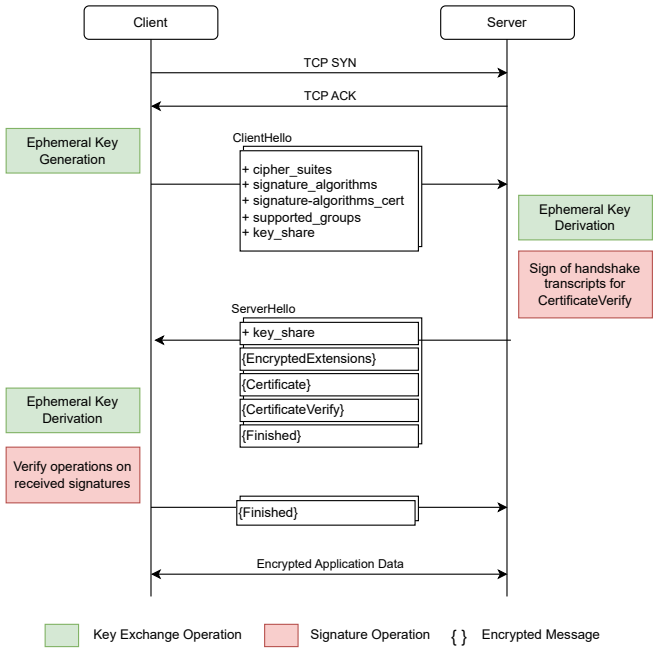



Figure 1: TLS 1.3 Handshake Overview.

the server authenticates itself, and the server and client do not have any pre-shared keys. A graphical overview is presented in Figure 1.

The TLS handshake starts once a TCP handshake is done and establishes a stateful connection. It begins by sending a *ClientHello* message, which contains several attribute fields. We omit most fields and only focus on the ones important for encryption. The first encryption-related field is the *cipher_suites* field, which contains a list of **symmetric ciphers** that the client supports. The next two fields are the *signature_algorithms* and *signature_algorithms_cert* fields that contain digital signature algorithms supported by the client. The final two fields are the *supported_groups* and *key_share* fields. These include all supported **key exchange algorithms** and the corresponding key material, which is used to share a secret key that both client and server can use for symmetric encryption. Upon receiving the message, the server selects a key exchange algorithm from the list of supported groups and calculates the shared secret.

The server responds with a *ServerHello* message that contains the *key_share* field with its own key material so that the client can deduce the shared secret as well. The message also contains the selected algorithm for symmetric encryption in the *cipher_suite* field. Afterward, the server sends the *EncryptedExtensions* message that contains additional extensions (e.g., the TCP segment size). Next, the server authenticates itself by sending the *Certificate* message that contains the server's certificate chain and the *CertificateVerify* message, which shows that the previous messages were exchanged with the server. The server does so by calculating a **hash** over the handshake context and the certificate chain and then calculating a **digital signature** over the hash with the server's private key. The client then verifies the digital signature with the server's public key, and the client verifies the public key (contained in the server's certificate) by going up the certificate chain and comparing the top with the client's own root certificate. The last two messages are the *Finished* messages, which both the client and the server send. They signal that neither server nor client has anything further to add to the handshake, and they contain a **Message Authentication Code (MAC)** of the entire handshake to ensure nothing was omitted. Once the *Finished* messages are exchanged, the TLS connection is established, and it is now possible to exchange user data. From now on, the shared secret is used for encryption, for both user data and status messages of the TLS protocol.

We can see that TLS makes use of several cryptographic operations: symmetric encryption, key exchange algorithms, digital signatures, hash functions, and MACs. As examined by Chen et al. [13], we can see that multiple operations are susceptible to quantum-capable attackers: Classical key exchange algorithms are based on the Diffie-Hellman algorithm, which uses the discrete logarithm problem for security [14]. However, with Shor's Algorithm, quantum computers can efficiently solve this problem, making it insecure [4]. The same goes for digital signatures that typically use ECDSA and RSA algorithms, which are again based on the discrete logarithm problem or the integer factorization problem, that are weak against quantum computers [3], [4]. For symmetric encryption, the impact is not as drastic, as no efficient solution to the underlying problems has been found yet. Though Grover [15] discovered that it is possible to speed up general search algorithms quadratically with quantum computers, which means that we can also search faster for a symmetric key. However, if we double the key size, we can maintain the same security as before. The same applies to hash functions [7]. They are also considered secure, given that they have a larger output. Message authentication codes are based either on symmetric ciphers or hash functions, which means they are also considered as secure as symmetric encryption or hashing [16]. An overview of the different cryptographic operations and their security is shown in TABLE 1. Fortunately, we can see that vulnerable operations are primarily used in the handshake, as later everything is symmetrically encrypted with the shared secret, and they are limited to key exchange and digital signatures, which means we have to find alternatives for key exchange and digital signature algorithms.

| Operation | Example Algorithm | Quantum-Resistance |
|---|---|---|
| Symmetric Cipher | AES128 | Requires larger keys |
| Hash Function | SHA256 | Requires larger output |
| MAC | HMAC | Requires larger output |
| Public Key Cipher | RSA-2048 | Insecure |
| Digital Signature | RSA-2048 | Insecure |
| Key Exchange | ECDHE | Insecure |

TABLE 1: Resistance of Classical Cryptography to Quantum-Based Attacks [13]

## 3. Exchanging Keys with Post Quantum Cryptography

In the previous section, we showed how the TLS protocol works and what cryptographic operations it uses. We now look at PQC operations that replace classical key exchange and signature algorithms. In this section, we examine different key exchange algorithms regarding security and efficiency. We focus mainly on algorithms

that have been standardized, as they are the most evaluated and the most likely to be considered for implementation in libraries such as OpenSSL.

The National Institute of Standards and Technology (NIST) is the most well-known entity that tries to agree on common ciphers that will be later implemented in network protocols such as TLS. Starting in 2016, NIST held a competition to find suitable PQC ciphers. In several rounds [8], [17]–[19], ciphers were submitted and analyzed for weaknesses. Forty ciphers have been included, and a select few were standardized over the course of 6 years. NIST publishes them as a Federal Information Processing Standard (FIPS), which contains an exact definition of the used algorithms. Currently, three have been published [20]–[22] and two more have already been announced [23], [24]. NIST also specifies five security levels for the proposed ciphers [25]. The levels resemble the difficulty it takes to break the cipher in ascending order, and the difficulty is specified by comparing the cipher to symmetric cryptography, i.e., AES, and SHA. Starting with the first level, an adversary should require the same computational resources needed to break AES128 (SHA256, AES192, SHA384, AES256 with higher levels). Apart from NIST, other institutions are also trying to standardize PQC, mainly the International Standardization Organization (ISO), which includes other algorithms that NIST has not chosen [26].

At the moment, all institutions propose PQC suitable methods for exchanging keys. Instead of a Diffie-Hellman-based key exchange, all institutions agree on the use of the Key Encapsulation Mechanism (KEM): During a key exchange between a server and a client, the client first sends its public key. The Server then uses the public key to encapsulate (encrypt) a randomly-chosen secret and sends it back to the client. The client then decapsulates (decrypts) the key with its private key and gets hold of the shared secret. As KEM works very similarly to Diffie-Hellman, no changes in the TLS protocol flow need to be made in order to use it; however, KEM is not forward secure by default because it reuses the key pair. To achieve forward secrecy, the key pair needs to be regenerated for each connection [27].

We now examine ciphers that are being recommended by the institutions mentioned above. We compare security, performance, and power consumption. Security is defined by the underlying mathematical problem (e.g., integer factorization) and by the ciphertext indistinguishability property that formally verifies the security of a cryptosystem given that the previous mathematical problem holds. For ciphertext indistinguishability, NIST suggests Indistinguishability under adaptive chosen-ciphertext (IND-CCA2) [8]. Classifying the mathematical problems is not as simple because they inherently differ and require deep knowledge to do a proper comparison. Furthermore, there is no proof of hardness for these problems. They are only considered secure so long as no efficient solution has been found, which generally means the older a problem is, the more secure it is considered. For this reason, we only consider the recency of a used mathematical problem. For performance, we compare the execution time to Elliptic-Curve based Diffie-Hellman (ECDHE) in a TLS 1.3 handshake. We chose ECDHE as it is currently a supported algorithm in the TLS 1.3 standard and is widely used for comparison in other work. We show the relative runtime overhead (e.g., 1.5x times higher execution time). Finally, we show the relative overhead for power consumption (e.g., 1.5x times higher power consumption):

**Kyber** (also called ML-KEM) was specified in FIPS 203 [22] and is currently the first choice for KEM ciphers. It is based on the Module-Learning-With-Error, a more recent specialization of a lattice problem, and currently has IND-CCA2 hardness [28]. Performance-wise, it is the best among lattice-based ciphers, and it is comparable to a classical key exchange with ECDHE for NIST level 1 when used in TLS 1.3, and for higher levels, Kyber can even outperform ECDHE [29]. Kyber also has similar power consumption in TLS 1.3 when compared to ECDHE [30].

**HQC** was chosen to be standardized as an alternative to Kyber by NIST because of the different underlying mathematical problem [24]. A standardization draft is expected to be released in 2026. It is based on the Quasi-Cyclic Syndrome Decoding problem, which is a specialization of a code-based problem and also has IND-CCA2 hardness [31]. Performance-wise, it is the best among code-based ciphers. On NIST level 1, HQC has a slight overhead of 1.45x - 1.8x when compared against ECDHE [7], [32]. HQC's power consumption is double that of ECDHE [30].

Additionally, NIST also lists other ciphers which still have not been disqualified but have not been chosen for standardization so far, including: Saber, NTRU and NTRU Prime, Classic McEliece, BIKE, and FrodoKEM [8], [19]. Most are not preferred either because of worse performance or stronger hardness assumptions of the ciphertext indistinguishability property. So far, they have been listed as alternative algorithms by NIST and are still being improved, so they remain an option if a cipher chosen for standardization is broken. A subset of alternative ciphers is also included in other standards:

**FrodoKEM** was specified in ISO 18033-2 [26] and is also recommended by BSI [33]. ISO considers it a conservative choice over Kyber as it uses the unmodified and well-studied Learning-With-Errors problem instead of the modified version used by Kyber [33], [34]. It has IND-CCA2 hardness and its performance overhead on NIST level 1 is 2.5x when compared with ECDHE [7]. FrodoKEM's Power consumption is roughly 14x times that of ECDHE [30].

**Classic McEliece** was also specified in ISO 18033-2 [26] and is recommended by BSI [33]. It is also considered a conservative choice as it is based on the Goppa-Code, which has the same maturity as RSA [33]. It has IND-CCA2 hardness [35]. The key size of Classic McEliece is significantly larger than that of other ciphers; in fact, it is too large to fit into the maximum size of $2^{16} - 1$ bytes of the TLS protocol's key share field, which means that it is not possible to use Classic McEliece in an unmodified TLS version [7]. There is work [36] that shows integrating Classic McEliece into the TLS protocol is possible, and a key share extension for the TLS protocol [37] has already been proposed. While there is currently no proper performance evaluation for the use in TLS, it was shown that the key generation of Classic MCEliece is much slower than any other cipher mentioned here, and encapsulation and decapsulation are faster than HQC [38].

Now we have several choices as to what key exchange cipher we want to use. If we go with NIST's choices, we have two ciphers that perform similarly to old key exchange ciphers. However, their underlying mathematical problems are still relatively new and not as field-tested. On the other hand, ISO's choices rely on older, well-studied mathematical problems, which are less likely to be broken, but at the same time, they have a higher overhead. Fortunately, this overhead is manageable even on weaker hardware, e.g., it was shown by Wagner et al. [36] that it is possible to implement even Classic McEliece with its large key sizes on an embedded system. There is, however, a third option which offers a tradeoff between performance and security: Hybrid PQC.

**Hybrid PQC** makes use of both classical Diffie-Hellmann and PQC KEM ciphers to provide a resilient cipher, so long as quantum computing attacks are infeasible. Once post-quantum attacks are feasible, the PQC cipher will hopefully have matured enough for it to be considered as secure as the old cipher. The first proposal was made by Bos et al. [39], which concatenated the material of algorithms and put them in the same field. When a key is exchanged, both algorithms are run in parallel, and the result is concatenated again. This has the advantage that no changes to the TLS protocol need to be made, and it was shown that with Kyber and ECDHE, the performance is similar to a regular ECDHE key exchange [40]. There also exist other proposals, such as an Additional Shared Secret extension [41] in the TLS protocol to add a field for a second key or explicitly adding hybrid encryption fields to allow multiple cipher combinations [42]. This makes the separation explicit and would allow a TLS implementation to actively decide which cipher to use. The performance of these modifications has not been tested yet; however, the performance is likely similar to the concatenation approach because we execute the same algorithms. However, this can have an advantage if there are non-ideal network conditions. For connections with high packet loss, it was observed that some ciphers perform worse, and switching to other ciphers will result in better performance [40].

We now have several methods to implement a key exchange, which will securely generate a shared secret that we can use for symmetric encryption. However, adversaries still have the ability to spoof an active connection by issuing false certificates, which is why we also need to make use of post-quantum digital signatures.

## 4. Providing Certificates with Post Quantum Cryptography

Just like we did with key exchange algorithms, we examine different digital signature algorithms in terms of security and efficiency, and we will focus on ciphers that have been standardized. Functionally, post-quantum digital signatures behave exactly as classical signatures, which means they can act as a drop-in replacement in the TLS protocol. Again, we analyze security, efficiency, and power consumption, and we compare the signature algorithms to the Elliptic-Curve-based Digital Signature Algorithm (ECDSA) in a TLS 1.3 handshake. ECDSA is currently a supported algorithm in the TLS 1.3 standard and is widely used for comparison in other work. We show the relative overhead of the execution time and the power consumption (e.g., 1.2x times higher execution time). For security of the cryptosystem, NIST suggests Existential Unforgeability under Chosen Message Attack (EUF-CMA) [25].

**Dilithium** (also called ML-DSA) was specified in FIPS 204 [20] and is currently the first choice for post-quantum digital signatures. It makes use of the Module-LWE and Module-SIS problems, which are based on the lattice problem [43]. Its hardness is SUF-CMA and meets the requirements of NIST [25], [43]. The performance of Dilithium is comparable to classical ECDSA, and its power consumption is also comparable [29], [30].

**FALCON** (also called FN-DSA) was chosen to be standardized by NIST as an alternative algorithm to Dilithium with similar performance [23]. A standardization draft is expected to be released in 2026. It makes use of the NTRU problem, which is based on the lattice problem [7], [44]. Falcon's hardness is proof of unforgeability in the QROM, which is deemed to be comparable to Dilithium [8], [45]. Performance-wise, it is also similar to ECDSA [7], [29], though the power consumption is roughly 2.7x times higher than with ECDSA [30].

**SPHINCS+** (alo called SLH-DSA) was specified in FIPS 205 [21] also as an alternative algorithm. It makes use of a hash-based problem, and its hardness is solely based on the security of hash functions, which pose adequate security [8]. The performance of SPHINCS+ is an order of magnitude slower than ECDSA; however, when comparing different works, there is no clear agreement on the slowdown [7]. Many different results have been published, likely because of different setups and the complexity of SPHINCS+, which has 12 parameters [7]. The same goes for energy consumption, which is much higher than other algorithms [30]. Because of this high overhead, SPHINCS+ is deemed a fallback in case a weakness is discovered in the other mainly lattice-based algorithms [8].

NIST and ISO also recommend the **XMSS/XMMS-MT** and **LMS/HSS** schemes [46], [47]. These are also hash-based and have better performance characteristics than SPHINCS+, but they are stateful, meaning that they can only sign a limited number of objects and require extra steps for the setup [7]. This makes them impractical for usage in the TLS protocol.

Similarly to key exchange algorithms, we can use hybrid algorithms and use classical signatures with post-quantum signatures simultaneously. This, however, requires changes to the TLS protocol and was implemented in [48]. The performance of hybrid signatures was shown to have an overhead of 1.2x to 2x times [49].

## 5. Implementing PQC into TLS

We now look at the TLS protocol again and examine what kind of attacks the TLS protocol is vulnerable to if we do not make use of PQC. We also look at how best to navigate the migration from classical cryptography to PQC.

Key exchange algorithms need to be migrated immediately because of the possibility of a Harvest-Now-Decrypt-Later attack, wherein an adversary that is on-path can eavesdrop and store messages, and once quantum

computers are powerful enough to break key exchange algorithms, the adversary can use them to recover the shared secret and decrypt all messages of the entire session. This attack requires an adversary to store a session from the beginning and acquire the key material from the TLS handshake, as this is the point where the TLS protocol is most vulnerable.

Authentication via certificates is less of a problem for current TLS connections: Adversaries can impersonate a peer by breaking its public keypair and issuing false certificates before the lifetime of the keypair ends. This attack also depends on the TLS handshake at the beginning of a TLS session. However, an adversary needs to actively participate in the connection, impersonate a peer, and only perform the attack while the session is ongoing, meaning it is not possible to perform this attack as long as we do not have a powerful enough quantum computer. Nevertheless, early adoption is still important as authentication with certificates is based on chaining with global root certificates on top of the chain having a life span of 20 years [50]. This means that if quantum-based attacks become feasible in this time span, root certificates that are trusted by and stored on the majority of devices need to be removed.

Because of these vulnerabilities, the current system needs to be migrated as soon as possible, which is why there are also recommendations by the IETF [50]. For key exchange algorithms, it is suggested to immediately transition to either hybrid or pure post-quantum key exchange, with hybrid being the preferred solution. From now on, TLS implementations should either provide both key shares or at least ask for post-quantum capabilities during a TLS handshake. Applications that depend on a library that implements TLS should test if their current setup supports post-quantum key exchange. For authentication, no immediate change is recommended. Instead, one should evaluate their system in terms of how often updates are possible and how accessible it is. If the system is deemed suitable, again, a hybrid solution is preferred over pure post-quantum certificates. To mitigate some of the overhead that hybrid certificates generate, new TLS extensions, such as TLS Cached Information Extension or TLS Certificate Compression, should be used. Once Quantum computing becomes feasible, hybrid signatures should be deprecated immediately because the current protocol design makes hybrid certificates vulnerable to downgrade attacks [51].

As an implementation, the most common library for TLS support is OpenSSL, which currently supports ML-KEM, ML-DSA, and SLH-DSA. With the provider architecture of OpenSSL and liboqs, it is also possible to access other algorithms such as FrodoKEM and HQC. A good alternative for embedded systems is WolfSSL. It currently also supports the same algorithms as OpenSSL. Nevertheless, it is important to carefully consider the security of implementations as they are not mature yet [50].

## 6. Conclusion and Future Work

We examined the TLS protocol and saw that key exchange and digital signature algorithms are weak and jeopardize security. Next, we analyzed alternatives for key exchange and digital signature algorithms. For different algorithms and their underlying mathematical problems, it is mainly a tradeoff between security and performance, with mature problems having a higher overhead than newer problems. And finally, we showed guidelines on how to navigate the migration to PQC and what implementations to consider. Key exchange algorithms must be replaced immediately, while digital signature algorithms are less urgent. For now, hybrid algorithms are preferred over pure post-quantum algorithms, but classical public key cryptography must be deprecated once quantum-based attacks are feasible. Future work could encompass evaluating different implementations, such as OpenSSL and WolfSSL, in terms of formal verification and optimization for resource-constrained systems.

## References


[1] H. Lee, D. Kim, and Y. Kwon, "Tls 1.3 in practice: How tls 1.3 contributes to the internet," in *Proceedings of the Web Conference 2021*, 2021, pp. 70–79.

[2] Y. Sheffer, R. Holz, and P. Saint-Andre, "Summarizing known attacks on transport layer security (tls) and datagram tls (dtls)," RFC 7457, 2015.

[3] D. Mahto and D. K. Yadav, "Rsa and ecc: A comparative analysis," *International journal of applied engineering research*, vol. 12, no. 19, pp. 9053–9061, 2017.

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[5] T. Häner, M. Roetteler, and K. M. Svore, "Factoring using 2n+ 2 qubits with toffoli based modular multiplication," *arXiv preprint arXiv:1611.07995*, 2016.

[6] C. Gidney, "How to factor 2048 bit RSA integers with less than a million noisy qubits," *arXiv preprint arXiv:2505.15917*, 2025.

[7] N. Alnahawi, J. Müller, J. Oupickỳ, and A. Wiesmaier, "A comprehensive survey on post-quantum tls," *IACR Communications in Cryptology*, 2024.

[8] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller *et al.*, "Status report on the third round of the nist post-quantum cryptography standardization process," 2022.

[9] K. Shrishak and H. Shulman, "Negotiating pqc for dnssec," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 2021, pp. 9–10.

[10] S. Bae, Y. Chang, H. Park, M. Kim, and Y. Shin, "A performance evaluation of ipsec with post-quantum cryptography," in *International Conference on Information Security and Cryptology*, 2022.

[11] H. Kinkelin, F. Rezabek, M. Kempf, and G. Carle, "Post-quanten-sichere open-source schemata und technologien für automotive-anwendungen," 2024. [Online]. Available: https://netintum.de/projects/post/

[12] E. Rescorla, "The transport layer security (tls) protocol version 1.3," RFC 8446, 2018.

[13] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. A. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*. NIST, 2016, vol. 12.

[14] U. M. Maurer and S. Wolf, "The diffie–hellman protocol," *Designs, Codes and Cryptography*, 2000.

[15] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[16] R. Bhaumik, B. Chakraborty, W. Choi, A. Dutta, J. Govinden, and Y. Shen, "The committing security of macs with applications to generic composition," in *Annual International Cryptology Conference*, 2024.

[17] D. Moody, G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y.-K. Liu, C. Miller, R. Peralta, R. Perlner *et al.*, "Status report on the first round of the NIST post-quantum cryptography standardization process," NIST, Tech. Rep., 2019.

[18] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta *et al.*, "Status report on the second round of the NIST post-quantum cryptography standardization process," *US Department of Commerce, NIST*, 2020.

[19] G. Alagic, M. Bros, P. Ciadoux, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller *et al.*, *Status report on the fourth round of the NIST post-quantum cryptography standardization process*. US Department of Commerce, NIST, 2025.

[20] NIST, "Module-lattice-based digital signature standard," U.S. Department of Commerce, Washington, D.C., Tech. Rep. Federal Information Processing Standards Publications (FIPS) 204, 2024.

[21] ——, "Stateless hash-based digital signature standard," U.S. Department of Commerce, Washington, D.C., Tech. Rep. Federal Information Processing Standards Publications (FIPS) 205, 2024.

[22] ——, "Module-lattice-based key-encapsulation mechanism standard," U.S. Department of Commerce, Washington, D.C., Tech. Rep. Federal Information Processing Standards Publications (FIPS) 203, 2024.

[23] C. Boutin. (2024) NIST releases first 3 Finalized Post-Quantum Encryption Standards. [Online]. Available: https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards

[24] ——. (2025) NIST selects HQC as fifth algorithm for Post-Quantum Encryption. [Online]. Available: https://www.nist.gov/news-events/news/2025/03/nist-selects-hqc-fifth-algorithm-post-quantum-encryption

[25] NIST, "Submission requirements and evaluation criteria for the post-quantum cryptography standardization process," 2016.

[26] ISO/IEC 18033-2:2006, *ISO/IEC 18033-2:2006/DAmd 2 - Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers*. ISO, Geneva, Switzerland, 2025. [Online]. Available: https://www.iso.org/standard/86890.html

[27] S. Fluhrer, Q. Dang, J. P. Mattsson, K. Milner, and D. Shiu, "ML-KEM Security Considerations," Internet Engineering Task Force, Internet-Draft draft-sfluhrer-cfrg-ml-kem-security-considerations-03, May 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-sfluhrer-cfrg-ml-kem-security-considerations/03/

[28] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, D. Stehlé *et al.*, "Crystals-kyber algorithm specifications and supporting documentation," *NIST PQC Round*, vol. 2, no. 4, pp. 1–43, 2019.

[29] R. Döring and M. Geitz, "Post-quantum cryptography in use: Empirical analysis of the tls handshake performance," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–5.

[30] G. Tasopoulos, C. Dimopoulos, A. P. Fournaris, R. K. Zhao, A. Sakzad, and R. Steinfeld, "Energy consumption evaluation of post-quantum tls 1.3 for resource-constrained embedded devices," in *Proceedings of the 20th ACM International Conference on Computing Frontiers*, 2023, pp. 366–374.

[31] P. Gaborit, C. Aguilar-Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, E. Persichetti, G. Zémor, J. Bos *et al.*, "Hamming Quasi-Cyclic (HQC)," 2025.

[32] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Assessing the overhead of post-quantum cryptography in tls 1.3 and ssh," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 2020, pp. 149–156.

[33] BSI TR-02102-1, *BSI TR-02102-1 Kryptographische Verfahren: Empfehlungen und Schlüssellängen*. BSI, Bonn, Germany, 2025. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.html

[34] "Frodokem: Learning with errors key encapsulation preliminary standardization proposal." [Online]. Available: https://frodokem.org/files/FrodoKEM_standard_proposal_20241205.pdf

[35] "Classic mceliece: conservative code-based cryptography: design rationale." [Online]. Available: https://classic.mceliece.org/mceliece-rationale-20221023.pdf

[36] J. Wagner and Y. Wang. New key share extension for classic mceliece algorithms. [Online]. Available: https://www.ietf.org/archive/id/draft-wagner-tls-keysharepqc-03.html

[37] J. Roth, E. Karatsiolis, and J. Krämer, "Classic mceliece implementation with low memory footprint," in *International Conference on Smart Card Research and Advanced Applications*, 2020.

[38] O. Kuznetsov, S. Kandiy, E. Frontoni, O. Smirnov *et al.*, "Trade-offs in post-quantum cryptography: A comparative assessment of bike, hqc, and classic mceliece." in *CQPC*, 2023, pp. 1–11.

[39] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the tls protocol from the ring learning with errors problem," in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 553–570.

[40] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking post-quantum cryptography in TLS," in *International Conference on Post-Quantum Cryptography*, 2020.

[41] J. M. Schanck and D. Stebila, "A Transport Layer Security (TLS) Extension For Establishing An Additional Shared Secret," Internet Engineering Task Force, Internet-Draft draft-schanck-tls-additional-keyshare-00, Apr. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-schanck-tls-additional-keyshare/00/

[42] W. Whyte, Z. Zhang, S. Fluhrer, and O. Garcia-Morchon, "Quantum-Safe Hybrid (QSH) Key Exchange for Transport Layer Security (TLS) version 1.3," Internet Engineering Task Force, Internet-Draft draft-whyte-qsh-tls13-06, Oct. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-whyte-qsh-tls13/06/

[43] "Crystals-dilithium algorithm specifications and supporting documentation (version 3.1)." [Online]. Available: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf

[44] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, Z. Zhang *et al.*, "Falcon: Fast-fourier lattice-based compact signatures over ntru, specification v1. 2," *NIST Post-Quantum Cryptography Standardization Round*, vol. 3, 2020.

[45] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry, "Random oracles in a quantum world," in *International conference on the theory and application of cryptology and information security*, 2011.

[46] D. A. Cooper, D. C. Apon, Q. H. Dang, M. S. Davidson, M. J. Dworkin, C. A. Miller *et al.*, "Recommendation for stateful hash-based signature schemes," *NIST Special Publication*, 2020.

[47] ISO/IEC 14888-4:2024, *ISO/IEC 14888-4:2024 Information security — Digital signatures with appendix — Part 4: Stateful hash-based mechanisms*. ISO, Geneva, Switzerland, 2024. [Online]. Available: https://www.iso.org/standard/80492.html

[48] E. Crockett, C. Paquin, and D. Stebila, "Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH," Cryptology ePrint Archive, Paper 2019/858, 2019. [Online]. Available: https://eprint.iacr.org/2019/858

[49] D. Marchsreiter and J. Sepúlveda, "Hybrid post-quantum enhanced tls 1.3 on embedded devices," in *2022 25th Euromicro Conference on Digital System Design (DSD)*. IEEE, 2022, pp. 905–912.

[50] T. Reddy.K and H. Tschofenig, "Post-Quantum Cryptography Recommendations for TLS-based Applications," Internet Engineering Task Force, Internet-Draft draft-ietf-uta-pqc-app-00, Sep. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-uta-pqc-app/00/

[51] T. Reddy.K, T. Hollebeek, J. Gray, and S. Fluhrer, "Use of Composite ML-DSA in TLS 1.3," Internet Engineering Task Force, Internet-Draft draft-reddy-tls-composite-mldsa-05, Jul. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-reddy-tls-composite-mldsa/05/

# SCION Versus Hardened BGP

Philip Falk, Michael Oberrauch*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: ge38fec@mytum.de, oberrauc@net.in.tum.de

***Abstract*—BGP has been the backbone of the Internet since its beginning. However, it is outdated and contains many security flaws. Various security extensions like BGPsec, RPKI-ROA and ASPA exist to mitigate these flaws. SCION is a new routing protocol, currently in development, with the aim to replace BGP in the future. With *security by design*, it wants to overcome BGP's security problems. This paper explores the main security mechanisms of both protocols and compares them to each other. We conclude that BGP's extensions improve its security significantly, but SCION's built-in cryptography is superior.**

***Index Terms*—scion, border gateway protocol, bgpsec, resource public key infrastructure, aspa**

## 1. Introduction

Today's Internet is a network of many autonomous systems (ASes). Each AS is a network of routers which is governed by an independent organization such as Internet service providers (ISPs), universities, or tech companies [1]. ASes regularly exchange information, so that the underlying routers, typically belonging to end-users, can communicate with each other. The border gateway protocol (BGP), originally created in the 1980s, is the protocol responsible for these frequent information exchanges (routing) between neighboring ASes. At the time of its conception, the Internet was still relatively small and security aspects were not a big point of focus. BGP therefore fundamentally works on unverified trust between networks. [2] Nowadays, unchecked false information about ASes, malicious or accidental, can spread over the Internet and can lead to issues like the famous YouTube hijack in 2008, which led to an hours-long YouTube outage [3]. This imposes a considerable security risk, which is why various mechanisms have been developed to reduce the dangers and flaws by extending BGP. These systems however, are not directly part of BGP but optional additions.

Further efforts try to eliminate today's dependency on BGP entirely through a new routing protocol called SCION. Scalability, Control, and Isolation On Next-generation networks (SCION) was designed with built-in security since its conception and is supposed to replace BGP in the future. It is, however, still in active development by the ETH Zürich and no standardized version exists at the time of writing in October 2025. [2]

We aim to compare the security aspects of different variations of hardened BGP to the new approach by SCION and show the differences between both protocols. Section 2 first provides an overview on how both BGP and SCION work and presents the main security problems. The following Section 3 then describes the existing and proposed security mechanisms for both protocols to mitigate the addressed dangers. Section 4 discusses and compares the presented mechanisms, Section 5 indicates further research, and finally Section 6 highlights the key outcomes.

## 2. Background

This section provides necessary knowledge on the security challenges of plain BGP and presents an overview of the mechanisms of both BGP and SCION.

### 2.1. BGP

To understand BGP, it is essential to know the concepts of Autonomous System Numbers (ASNs) and IP prefixes. An ASN is a globally unique identifier assigned to each AS by a Regional Internet Registry (RIR) under the coordination of the Internet Assigned Numbers Authority (IANA) [4]. Each AS manages one or more IP prefixes, sets of IP addresses it can allocate internally, and advertises these prefixes to other ASes via BGP to enable inter-domain communication.

Consider AS1 advertising its prefix 192.0.20.0/24 to neighboring ASes, AS2 and AS3. After establishing a peering connection with both neighbors, AS1 sends an advertisement to each of them [4]. Advertisements are officially called *UPDATE* messages and contain certain attributes. The most relevant one for this context is AS_PATH. It is a sequence of ASNs that need to be traversed in order to arrive at the AS that owns the advertised prefix. This originating AS is the rightmost ASN in the sequence. [5]

The current example of the BGP functionality can be seen in Figure 1. AS1 is the originating AS which starts to advertise its prefix by adding its ASN to the initially empty AS_PATH. When an advertisement reaches an AS, the receiving AS will remember its routing information, prepend its own ASN to the AS_PATH, and propagate it further to other ASes. As a result, the originating ASN remains at the rightmost position of the sequence, allowing any AS that receives the advertisement to identify both the prefix owner and the sequence of ASes that must be traversed to reach the originating AS. [4]

As illustrated in Figure 1, it is possible that an AS receives multiple advertisements from different sources containing the same prefix. It is then up to the AS's

 doi: 10.2313/NET-2026-05-1_04

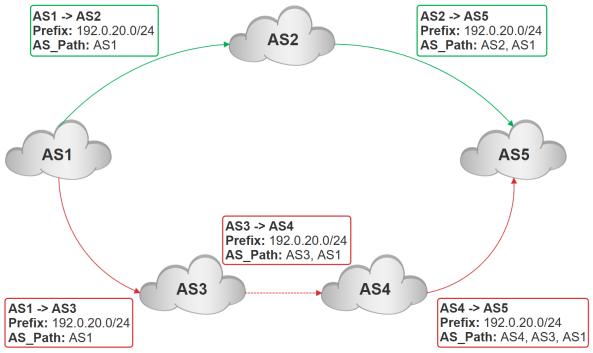



Figure 1: BGP routing process [4]

routing policy to prefer one path over the others. [4] It may also happen that some routes between neighboring ASes become unavailable. That is why these advertising messages get sent regularly, so a different path can be created instead.

## 2.2. BGP's security challenges

Plain BGP is solely based on trust relationships between ASes and has no security features. With nowadays over 80.000 active ASes, it is however, quite likely that not all are trustworthy [3]. An attacker controlling an AS has multiple options to hijack BGP traffic and subsequently force Internet traffic containing user-data through their malicious AS. The attacker can then either establish themselves as a Man in the Middle or simply drop the traffic, resulting in a Denial of Service (DOS) attack [6].

Suppose a malicious AS, controlled by an attacker, aims to replicate the mentioned YouTube hijack and force Internet traffic directed to YouTube through itself. The attacker could identify the prefixes belonging to YouTube and start advertising more specific subprefixes. These more specific prefixes subsequently spread through the Internet and users trying to access YouTube get redirected to the attacker instead. The reason for that is called longest prefix matching, which always redirects traffic to the most specific prefix available. [6] Alternatively, a prefix with a shorter path can also be announced [3]. The malicious AS takes the role of an originating AS wanting to advertise its prefixes when in reality it doesn't own the advertised prefixes. This type of attack works because plain BGP doesn't require origin validation, a check whether or not an AS owns its announced prefixes. Any AS can advertise any prefix, and any advertised prefix is implicitly trusted.

If an origin validator is however present, the same attacker has yet another way to hijack BGP traffic. Instead of pretending to own an advertised prefix, the malicious AS can also insert itself into a fake AS_PATH with the correct originating AS. An origin validator would check if the rightmost ASN in the AS_PATH owns the advertised prefix and would find it to be true, when in fact the malicious AS has placed its ASN at any place ahead of the origin ASN. [6] This makes it evident that a combination of origin validation and path validation is necessary to successfully validate an incoming prefix advertisement. Section 3 describes these two types of validation mechanisms.

It is important to note that route hijacks can also happen accidentally and are then referred to as route leaks. Usually, these are caused by misconfigurations. [7] An example for an accidental prefix hijack was the mentioned YouTube hijack where the government-owned Pakistan Telecom tried to block the access to YouTube country-wide. An accident caused this new BGP route, containing a more specific prefix, to spread over the Internet and effectively redirecting all YouTube traffic to Pakistan [3]. To fix the issue, an even more specific prefix had to be announced in order to hijack the traffic back [4].

## 2.3. SCION

SCION aims to become the next-generation Internet routing protocol by preventing the issues of BGP. It starts by logically grouping multiple ASes into Isolation Domains (ISDs), for example by country. Each ISD has a set of Core ASes which collectively create a common trust domain described in a Trust Root Configuration (TRC) which all Non-Core ASes agree upon. [2]

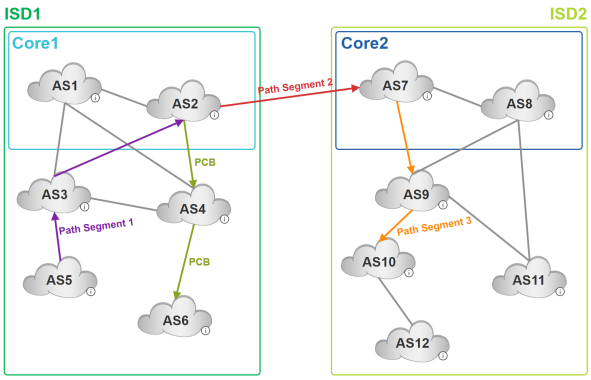



Figure 2: SCION beaconing and path segments [2]

As seen in Figure 2, communication inside of an ISD happens mainly between Core ASes and Non-Core ASes (Intra ISD) while communication between two different ISDs happens from Core1 to Core2 (Inter ISD). The goal of SCION is to create multiple path segments which are combined into a cryptographically secured forwarding path. An exemplary forwarding path, connecting AS5 from ISD1 to AS10 from ISD2, is illustrated in Figure 2, and consists of multiple path segments. A path segment is constructed through a process called beaconing, which works similarly to BGP's *UPDATE* messages. Intra ISD beaconing and Inter ISD beaconing function analogously. [8]

Intra ISD beaconing works by periodically sending Path Construction Beacons (PCBs) from Core ASes downwards to Non-Core ASes. PCBs consist of a sequence of AS Entries representing the path which the PCB has travelled so far. When a Non-Core AS receives a PCB, it first checks its validity as described in Section 3.4 and then adds its own AS Entry. An AS Entry cryptographically identifies the respective AS and contains a Hop Field (HF). An HF serves as a forwarding instruction for traffic within the respective AS, specifying how the traffic must travel to the next hop in order to correctly follow a path. The sequence of AS Entries, each containing an HF, is equivalent to the AS_PATH attribute in BGP with the

exception that each AS Entry is also cryptographically protected. [8]

When a PCB reaches a Leaf AS, the path discovery of that PCB is completed. Each Non-Core AS the PCB has passed through now has a valid path segment to its ISD's Core. Path segments are created from received PCBs and contain all the AS Entries from the respective PCB. PCBs are sent periodically in batches so that each Non-Core AS usually has multiple alternative path segments to choose from. [8]

SCION not only offers security in its path discovery beaconing process but also provides a SCION header for packet delivery. The SCION header encapsulates, next to the user-data, also the forwarding path to make sure that each packet stays on the selected path. The forwarding path is constructed from the Hop Fields from all AS Entries of all path segments. [9]

## 3. Security Mechanisms of BGP and SCION

This section explains the main security extensions for BGP and the built-in security features in SCION.

### 3.1. Security in BGP: RPKI (ROA)

As already stated, an AS can acquire its ASN and its prefixes from a RIR. Additionally, the AS also receives a certificate from its respective RIR, verifying that the AS owns said resources. In order to achieve origin validation, the AS can then create a Route Origin Authorization (ROA) record, which is a data object containing the resources (ASN and prefixes) owned by the AS [7]. ROAs essentially state which AS is authorized to originate which prefixes [10]. The AS can then cryptographically sign the ROA and have the certificate from the RIR validate its authenticity. The RIR thereby takes the role of a Certificate Authority (CA), a root of trust. ROAs can be saved in the RIR's repository publicly accessible for other ASes. When other ASes receive prefix advertisements, they can now validate their origin through the respective ROA by following the certificate chain to the trust anchor. [7]

This linking of resources to resource-owners by issueing certificates is called the Resource Public Key Infrastructure (RPKI) and it provides cryptographic origin validation [4]. Prefix hijacks, like the YouTube hijack, can be prevented this way.

### 3.2. Security in BGP: BGPsec

RPKI-ROA prevents ASes from originating prefixes which they do not own. What it does not prevent, however, are announcements of fake paths with correct origins. BGPsec is an extension to BGP which equips it with this necessary cryptographic path validation. It works on the basis of the already established RPKI by replacing the AS_PATH attribute from BGP advertisements with a cryptographically secured version called BGPsec PATH. [6] Figure 3 presents an example of BGPsec's functionality.

AS1 wants to announce its prefix to AS2. It therefore adds its ASN to the initially empty BGPsec PATH. To secure this initial advertisement, AS1 creates a cryptographic signature sig1, computed over the prefix, the

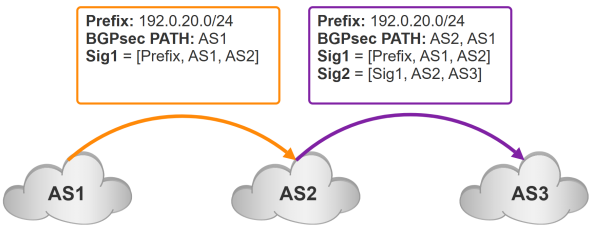



Figure 3: Example of BGP routing with BGPsec [4]

advertising AS (AS1) and the destination AS (AS2). This signature is added to the advertisement and sent to AS2. AS2 receives this advertisement and first checks the validity of all previous signatures. At this moment, that involves only sig1, which is validated by retrieving AS1's public key from its certificate through the RPKI. Advertisements with invalid signatures are dropped. If the signature is correct, then AS2 prepends its ASN to the BGPsec PATH and creates its own signature sig2. Sig2 and all further signatures are however not computed over the prefix anymore since that is already saved in sig1. Instead, sig2 is calculated over the current ASN (AS2), the destination AS (AS3) and the signature from the direct predecessor (sig1), thereby chaining the signatures together. All further ASes will have to follow the same steps as AS2, with the exception that the chain of signatures will grow and thereby involve more steps to validate. [4] This will result in increasing memory costs and processing power, which is one of the main drawbacks of BGPsec [6]. Another issue with BGPsec is the remaining vulnerability to an attack called wormhole attack [4].

### 3.3. Security in BGP: ASPA

Autonomous System Provider Authorization (ASPA) is a more lightweight approach to secure path validation. ASPA objects work very similarly to ROA records. They are data structures, created and signed by an AS and publicly available to other ASes at a RIR repository. ASPA also makes use of the RPKI by utilizing its CAs that hand out certificates to validate the signature of an ASPA object. The difference between ASPA and ROA is therefore only in their content. [11]

An ASPA object allows an AS (ASX) to specify a list of its authorized upstream providers. These are other ASes which are allowed to propagate any BGP advertisements that they have received directly from ASX. This customer-provider relationship (ASX is a customer) ensures that prefix advertisements can only propagate over trusted ASes. To fully secure a path, all ASes on the path must own ASPA objects in which they specify their respective providers. [12]

Consider the following exemplary AS_PATH: "AS3, AS2, AS1", with AS1 being the originating AS. ASPA path validation happens at every hop. This means that every AS, which receives an advertisement, must validate the entire received path. An AS receiving the stated path therefore has to acquire the ASPA object of AS1 and check if AS2 is part of AS1's list of upstream providers. If that is the case, then the next ASPA by AS2 is checked to validate the next customer-provider relationship between

AS2 and AS3. This continues until all relationships between two ASes of the path have been validated. Due to the nature of AS relationships, it is possible that at some point in the path, the relationship of "customer to provider" switches around to "provider to customer", resulting in an earlier AS in the path being the provider and the following AS being the customer. To successfully validate a path with ASPA, it is therefore allowed to have a single transition point. The validation of provider to customer relationships works the same by checking the customers' ASPA for its downstream providers. [11]

In the rare event that a compromised provider hijacks prefixes, ASPA is unable to detect such attacks [13]. Just like SCION, ASPA is still work in progress and no standardized version exists at the time of writing [11].

### 3.4. Security in SCION

SCION has two security mechanisms. First, there is a PKI that provides certificates for validation of PCBs in the beaconing process, and secondly, SCION cryptographically secures the forwarding path to ensure that packets travel along the selected path. In SCION, each ISD possesses a TRC, which serves as a trust root and enables a separate PKI per ISD. Each ISD has its own trusted CAs, which hand out certificates to the internal ASes. Unlike BGP, SCION therefore avoids global trust roots. By having multiple independent CAs, an ISD can still issue certificates even when one CA is compromised. [14]

In the intra-ISD beaconing process, PCBs are sent from Core ASes downwards to Non-Core ASes inside an ISD. Every AS on the path adds an AS Entry, which describes how traffic should flow through it. Each AS Entry is secured through a signature by the corresponding AS. The signature is computed over the current AS Entry, all previous AS Entries and all previous signatures, thereby effectively chaining the signatures. When an AS receives a PCB, it confirms its validity by checking the last signature through the respective AS's certificate. [8]

By having PCBs always originate from Core ASes, SCION does not have any origin validation like the RPKI. In SCION, prefixes are not advertised and origin validation is therefore not needed. [14]

SCION lets end-users choose the path their packets should travel along, by allowing them to combine their preferred path segments. To ensure that packets stay on the selected end-to-end forwarding path, SCION cryptographically secures the path information through Message Authentification Codes (MACs). These MACs are computed by all ASes of the path in the beaconing process, using each AS's local symmetric key. Every MAC secures a Hop Field which is added to the PCB and is computed the following way (1) [9]:

$$\mathrm{MAC}_i = K_i\big(\mathrm{SegID} \oplus \mathrm{MAC}_0 \oplus \cdots \oplus \mathrm{MAC}_{i-1}, \\ \mathrm{Timestamp},\ \mathrm{HF}_i\text{-content}\big) \tag{1}$$

As can be seen, each MAC is calculated over a chain of all previous MACs and a SegID, which is a random value initially selected by a Core AS. This computation makes the subsequent validation of each MAC more efficient in the packet forwarding process. When an AS receives a packet for forwarding, it recomputes its MAC and compares it with the corresponding MAC of the forwarding path to verify that the packet is following the intended path. An Accumulator (Acc) optimizes this recomputation by summarizing the SegID and all previous MACs into one value. The Acc is sent along with each packet and needs to be updated at each hop of the path. [9]

## 4. Security Comparison of BGP and SCION

One of SCION's biggest advantages compared to BGP is that it allows end-users to select a path they want their packets to travel along. This is especially important for national data protection when sending sensitive information related to healthcare, critical infrastructure or the military. BGP routing follows individual routing policies, which often send data across international borders over potentially unsecure hops. [2]

As soon as an end-user selects an end-to-end path, it is included and sent with each SCION packet. This mitigates computationally expensive longest prefix matching at each hop because the next hop is already included. MACs ensure that the chosen path is cryptographically secured. SCION therefore also provides security in the packet forwarding process and not only in path discovery. [9] Since the full path is contained in all SCION packets, SCION offers the option for returning packets to take the same route back, which also ensures a safe response [8]. Furthermore, SCION can be used interoperably with BGP through SCION-IP Gateways, which enable communication between SCION networks and BGP networks. This scenario however, loses SCION's property of cryptographically secured paths. [15]

SCION supports multipathing, which means that every hop always has multiple valid paths available. This gives SCION an advantage compared to BGP because it can switch to an alternative path more quickly, thereby making DOS attacks more difficult. [2]

BGP and SCION have different trust models. BGP's RPKI has global trust roots in all 5 RIR's while SCION's PKI splits its trust into individual ISDs, thereby having no reliance on third-party global CAs. [2]

BGP's security mechanisms are built on top of BGP's functionality and thereby optional and not universally employed. In 2024, only 51% of all prefixes were secured through RPKI-ROAs [6]. Mechanisms like ASPA require every AS on the AS_PATH to have created an ASPA object. Path validation cannot be fully guaranteed unless all ASes use ASPA [16]. SCION does not encounter these problems, as its security mechanisms are integrated from the start and enforced automatically [2].

BGP requires origin validation mechanisms like RPKI-ROA to verify if an AS is allowed to announce a prefix. SCION works fundamentally differently from BGP, as its PCBs always originate from Core ASes and carry path information rather than announcing prefixes. SCION therefore has no origin validation since it is unnecessary [14]. Path validation however is necessary in both SCION and BGP to protect the path discovery processes. In BGP, path validation is optional through extensions like BGPsec or ASPA, in SCION it is mandatory through the signing of AS Entries.

## 5. Related work

Recent research has further explored the security mechanisms and protocols discussed in this paper. As previously noted, BGPsec faces considerable performance challenges. Reference [6], therefore, presents several algorithms to optimize its path validation process. Another paper [17] analyzes the evolution and adoption of RPKI since its deployment in 2011, and a comprehensive survey [18] categorizes existing research on RPKI measurement studies. ASPA is less effective when it is not universally deployed, so a study [11] evaluates its security guarantees under partial deployment conditions. Furthermore, the performance of the SCION architecture, specifically latency and throughput, has been investigated in [19].

## 6. Conclusion

BGPsec, RPKI-ROA, and ASPA are optional but much-needed extensions designed to remedy BGP's lack of inherent security. BGPsec suffers from inefficient computing costs and still being vulnerable to attacks like the wormhole attack [4]. RPKI-ROAs are in active use, but their adoption remains limited and ASPA is still in development. These security mechanisms strengthen BGP, particularly against accidental route leaks, but malicious attackers still have various attack possibilities. It is clear that BGP's security extensions are far from ideal. Unlike BGP's optional extensions, SCION's security mechanisms are built-in and automatically enforced across all networks. From a security perspective, SCION clearly has the advantage. However, adopting a new routing standard also depends on factors such as performance and deployability, which lie beyond the scope of this paper.

## References


[1] "What is bgp? | bgp routing explained," https://www.cloudflare.com/en-gb/learning/security/glossary/what-is-bgp/, [Online; accessed 8-September-2025].

[2] "Next-generation internet routing," https://www.scion.org/about-scion/, [Online; accessed 8-September-2025].

[3] "What is bgp hijacking?" https://www.cloudflare.com/learning/security/glossary/bgp-hijacking/, [Online; accessed 10-September-2025].

[4] J. Oesterle, H. Kinkelin, and F. Rezabek, "Challenges with BGPSec," Technical University of Munich, Available: https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2022-01-1/NET-2022-01-1_02.pdf, 2021.

[5] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," Request for Comments: RFC 4271, IETF Datatracker, Available: https://datatracker.ietf.org/doc/html/rfc4271, 2006.

[6] M. Abdelhafez and Y. Fadlalla, "BGPsec Deployment Challenges and Optimization Efforts," *2024 IEEE 22nd Student Conference on Research and Development (SCOReD)*, 2024.

[7] J. Fleury and L. Poinsignon, "Rpki and bgp: our path to securing internet routing," https://blog.cloudflare.com/rpki-details/, [Online; accessed 14-September-2025].

[8] C. de Kater, N. Rustignoli, and S. Hitz, "SCION Control Plane," Active Internet-Draft, work in progress, IETF Datatracker, Available: https://datatracker.ietf.org/doc/draft-dekater-scion-controlplane/, 2025.

[9] C. de Kater, N. Rustignoli, J.-C. Hugly, and S. Hitz, "SCION Data Plane," Active Internet-Draft, work in progress, IETF Datatracker, Available: https://datatracker.ietf.org/doc/draft-dekater-scion-dataplane/, 2025.

[10] "Bgp origin validation," https://www.ripe.net/manage-ips-and-asns/resource-management/rpki/bgp-origin-validation/, [Online; accessed 14-September-2025].

[11] N. Umeda, T. Kimura, and N. Yanai, "The juice is worth the squeeze: Analysis of autonomous system provider authorization in partial deployment," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 269–306, 2023.

[12] A. Azimov, E. Uskov, R. Bush, J. Snijders, R. Housley, and B. Maddison, "A Profile for Autonomous System Provider Authorization," Active Internet-Draft, work in progress, IETF Datatracker, Available: https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-profile/, 2025.

[13] 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," Active Internet-Draft, work in progress, IETF Datatracker, Available: https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-verification/, 2025.

[14] C. de Kater, N. Rustignoli, and S. Hitz, "SCION Control Plane PKI," Active Internet-Draft, work in progress, IETF Datatracker, Available: https://datatracker.ietf.org/doc/draft-dekater-scion-pki/, 2025.

[15] D. Sieciński, "Traditional Internet vs. SCION Architecture," https://codilime.com/blog/scion-vs-traditional-internet/, 2023, [Online; accessed 23-September-2025].

[16] N. Geng, M. Huang, and Y. Wang, "An Analysis of ASPA-based AS_PATH Verification," Active Internet-Draft, work in progress, IETF Datatracker, Available: https://datatracker.ietf.org/doc/draft-geng-sidrops-aspa-analysis/, 2025.

[17] T. Chung, E. Aben, T. Bruijnzeels, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, R. van Rijswijk-Deij, J. Rula, and N. Sullivan, "Rpki is coming of age: A longitudinal study of rpki deployment and invalid route origins," *Association for Computing Machinery*, p. 406–419, 2019.

[18] N. Rodday, Ítalo 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.

[19] E. Ståhl, "Performance analysis of the scion protocol," KTH Royal Institute of Technology, Available: https://www.academia.edu/97173450/Performance_analysis_of_the_SCION_protocol, 2022.

# HTTPS-First as the New Standard

Doğa Ninsun Gezer, Lion Steger*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: ninsun.gezer@tum.de, steger@net.in.tum.de

***Abstract*—The modern web has largely shifted to encrypted connections, with the vast majority of traffic using the Hypertext Transfer Protocol Secure (HTTPS). This paper investigates how current web browsers initiate connections to servers and the use of encrypted channels for the initial contact. We analyze browser behavior empirically, revealing a shift towards HTTPS-First behaviour, but with varied implementations including direct HTTPS attempts, HTTP probes, and differing QUIC/HTTP3 adoption. Crucially, most retain an insecure fallback to HTTP. We categorize server-side upgrade mechanism like HTTP Strict Transport Security (HSTS) and HSTS preloading, and client-side upgrade mechanisms like HTTPS-Only Mode, analyzing their specific security guarantees and evaluating their effectiveness. By studying attacker methods like SSL stripping, we demonstrate how an insecure fallback behaviour to HTTP can be exploited. We explore the underlying reasons for HTTP persistence, rooted in backward compatibility and local network constraints.**

***Index Terms*—HTTPS, HTTP, HSTS, Network Security**

## 1. Introduction

The Hypertext Transfer Protocol Secure (HTTPS) is a well-established secure alternative to the Hypertext Transfer Protocol (HTTP). HTTPS provides confidentiality and integrity by using SSL/TLS protocols to establish an encrypted channel between the client and the server. [1] When a user enters a domain name without specifying a protocol (e.g., example.com), the browser's choice of default protocol becomes a critical security decision.

Historically, browsers defaulted to an insecure HTTP connection [**?**], exposing the initial interaction to Man-in-the-Middle (MitM) attacks before potential redirection. [2] Mitigation strategies like HSTS and HSTS preloading [2] were developed to address this vulnerability, signaling to browsers that all communication in the future should be through HTTPS. Consequently, all major browsers have shifted from the HTTP-First approach and now use HTTPS-First connection models.

This paper investigates the practical security implications within the web landscape. We first explain the context of HTTP's persistence and the available upgrade mechanisms. Then, we empirically analyze how major browsers currently implement their initial connection strategies. We subsequently analyze the specific risks arising from the HTTP fallback mechanism. Finally, we synthesize these findings to understand the current security posture regarding initial web connections.

## 2. Background

This section explains why HTTP persists despite HTTPS, clarifies historical context, and details the mechanisms aimed at upgrading connections, addressing the questions of why browsers cannot simply always use HTTPS and what problems each upgrade mechanism solves.

### 2.1. The Persistence of HTTP

Despite HTTPS becoming the new standard for the web [3], HTTP continues to persist as a fallback. The complete deprecation of HTTP is prevented by two primary challenges: backward compatibility and the unsolved problem of local network encryption. This section first examines these primary challenges. It then surveys the key mechanisms developed to enforce secure connections, distinguishing between server-side and client-side approaches [4].

**2.1.1. Backward Compatibility.** A primary reason for retaining HTTP support is backward compatibility and the persistent high prevalence of HTTPS configuration errors. We define these errors as issues within the SSL/TLS certificate, such as being from an untrusted Certificate Authority (CA), a mismatching domain name, or being expired, as well as mixed content warnings [5]. Early research, such as Jackson et al. (2008), mentions the "unknown intent" problem, where distinguishing between self-signed certificates on low-security sites and malicious attacks is difficult [6].

A 2022 study of 1,562 popular websites by Alabduljabbar et al. found that TLS certificate issues were significantly more common on free content sites (35.85%) than on paid content sites (6.99%). These problems included invalid certificates (17.42%), domain name mismatches (11.47%), and expired certificates (6.97%). This demonstrates that many websites still fail to implement HTTPS correctly, reinforcing the need for browsers to handle such cases without completely blocking access [5].

**2.1.2. The Local Network Problem.** The web's security model, based on the Public Key Infrastructure, does not extend to private networks. CAs are unable to issue trusted certificates for private IP addresses (e.g., 192.168.1.1) or reserved local hostnames (e.g., router.local) [7]. This makes HTTP the only practical protocol for essential tasks like configuring home routers, accessing corporate intranets, and running local development servers, forcing browsers to retain HTTP support.

 doi: 10.2313/NET-2026-05-1_05

## 2.2. Mechanisms for Upgrading to HTTPS

There are several mechanisms used to upgrade from HTTP to HTTPS. Implementations of server-side mechanisms are generally identical across the web as they follow standards from the World Wide Web Consortium or the Internet Engineering Task Force, whereas client-side mechanisms differ in their implementation in browsers [4].

| Mechanism | Share of Upgrades (%) |
|---|---|
| HSTS (incl. Preload) | 62.4 |
| HTTPS-First (Browser Default) | 30.4 |
| Web Extension | 3.3 |
| HTTPS-Only (Opt-in) | 2.3 |
| HTTPS RR (DNS) | 1.3 |
| CSP upgrade-insecure-requests | 0.3 |

TABLE 1: Effectiveness of Top-Level HTTP-to-HTTPS Upgrade Mechanisms (Source: [3])

HSTS, as specified in RFC 6797, allows a server to send a `Strict-Transport-Security` header over an HTTPS connection [2]. This instructs the browser to communicate with that domain exclusively over HTTPS in the future. However, HSTS relies on the principle of Trust On First Use, leaving the client's first connection unprotected from the MitM vulnerability [6].

To solve the first-visit problem, browser vendors maintain a list of domains that are hardcoded as HTTPS-Only [8]. For these domains, the browser enforces HTTPS from the very first connection, eliminating the window for SSL stripping attacks. While the list covers most high-traffic sites and together with HSTS is responsible for the majority (62.4%) of all successful HTTP-to-HTTPS upgrades as shown in the TABLE 1, it is not exhaustive [4]. The preload list could potentially include a wildcard for every top-level domain, but that would still create backward-compatibility issues [9].

The HTTPS-First Mode is the default behavior in most modern browsers. The browser first attempts to establish a secure HTTPS connection. If this fails (e.g., the server does not respond on port 443 or has an invalid certificate), it silently and automatically falls back to an insecure HTTP connection [10]. This opportunistic approach is the second-most effective upgrade mechanism, responsible for 30.4% of all upgrades as shown in TABLE 1, but its fallback behavior is an exploitable weakness [4].

The HTTPS-Only Mode is offered as a stricter, opt-in feature in major browsers, this mode also attempts to upgrade all connections but does not automatically fall back. If a secure connection fails, it blocks the navigation and presents the user with a security warning, prioritizing security over compatibility [10].

The `upgrade-insecure-requests` directive within a Content-Security-Policy (CSP) can upgrade same-origin navigations, while DNS-based HTTPS Resource Records (HTTPS RR) can signal HTTPS support during lookup [11], [12]. However, as shown in TABLE 1 their real-world impact on top-level upgrades is minimal, at 0.3% and 1.3% respectively [4].

# 3. Attacks Targeting the HTTPS-First Fallback

While HTTPS-First policies secure the data channel, they do not address all threats. Application-layer vulnerabilities like Cross-Site Scripting [13], client-side malware, and phishing attacks using valid TLS certificates remain potent threats. This analysis, however, focuses specifically on protocol downgrade attacks, which exploit the persistence of HTTP as a fallback mechanism. These attacks assume an adversary has a privileged MitM position, achievable by controlling a Wi-Fi access point, ARP spoofing, or DNS hijacking [14].

The primary threat in this context is SSL stripping, first demonstrated by Moxie Marlinspike [15]. A modern variant of this attack weaponizes the very compatibility feature designed to make HTTPS-First practical: its silent fallback to HTTP. An attacker can intentionally disrupt the browser's initial attempt to connect via HTTPS, for instance, by blocking traffic to port 443. This failure triggers the browser's automatic and silent fallback to insecure HTTP, handing the attacker the unencrypted request needed to initiate the attack.

Critically, this attack vector circumvents dynamic HSTS by exploiting its "Trust On First Use" vulnerability. If an attacker intercepts the very first connection, the browser never receives the `Strict-Transport-Security` header over a trusted channel. The attack itself prevents the deployment of the primary mechanism designed to stop it, as browsers ignore the header when sent over an insecure connection [2].

The attack leverages the browser's predictable fallback logic in three steps:

1) **Disrupt Initial HTTPS Attempt:** The MitM attacker actively prevents the browser's initial connection attempt to port 443 from succeeding. This can be done by blocking the traffic.
2) **Trigger Silent Fallback:** The browser, encountering a failure on port 443 and operating in the default HTTPS-First mode, automatically initiates a new connection attempt to port 80 (HTTP) without user notification.
3) **Intercept and Downgrade (SSL Stripping):** The attacker intercepts the unencrypted HTTP request. They establish their own secure HTTPS connection to the legitimate server, receive the content, strip all security elements (e.g., changing `https://` links to `http://`), and forward the modified, insecure HTTP version to the user [15].

From the user's perspective, the site simply loads over HTTP without any error, defeating the purpose of HTTPS. This demonstrates how a feature designed for compatibility (the silent fallback) becomes an exploitable flaw. Consequently, the HSTS Preload List remains the only mechanism capable of fully protecting users from this protocol downgrade attack on their first visit, as its policy is known to the browser before any connection is made [8].

# 4. Analysis of Modern Browser Behavior

To investigate the practical deployment of HTTPS-First connection strategies, this study analyzed the net-

 doi: 10.2313/NET-2026-05-1_05

work traffic of five widely used browsers as of September 2025 [16]: Google Chrome, Safari, Edge, Opera, and Firefox. Using the latest stable version of each browser after a clean installation, all network connections initiated when rendering page content were captured. .

Our experimental setup employed Wireshark, an open-source network protocol analyzer, to capture and inspect traffic. For each test, we entered the schemeless domain `example.com` into the browser's address bar. This domain was selected because it is accessible via both HTTP, HTTPS and QUIX but is not on Chromium's HSTS preload list [8]. We analyzed the sequence and destination ports (80 vs. 443) of initial TCP SYN packets, any subsequent UDP (QUIC) connection attempts, and the protocol used for the first application data transmission (TLS vs. HTTP vs. QUIC) The browser's behavior was categorized based on the first application-layer data packet sent:

- HTTPS-First Behavior: The browser sends a `TLSv1.x Client Hello` over a TCP connection to port 443. Any connections to port 80 are either closed or remain idle.
- HTTP-First Behavior: The browser sends an `HTTP GET` request over a TCP connection to port 80.

Following the initial connection, browsers showed differing tendencies to upgrade to QUIC/HTTP3 [**?**]. *Aggressive Upgraders (Chrome, Edge, Opera):* The Chromium-based browsers actively attempted QUIC connections to `example.com`, indicating a strategy favoring potential QUIC performance benefits. *Conservative Adopters (Firefox, Safari):* Firefox and Safari defaulted to using the established TCP/TLS connection for `example.com`. Firefox demonstrated QUIC capability via background traffic but applied it selectively, perhaps prioritizing TCP's maturity or specific performance heuristics not met by `example.com`. Safari generally adopts new protocols more cautiously.

This empirical analysis connects the need for HTTP support (Sec 2) to concrete browser implementation choices and their inherent trade-offs, leading into a discussion of the overall implications.

## 4.1. Google Chrome

Google Chrome began implementing an "HTTPS-First" mode in 2021 [17].The test of Google Chrome (v140.0.7339.213) showed that upon entering the schemeless URL, the browser initiated three parallel TCP connection attempts: two to port 80 (HTTP) and one to port 443 (HTTPS). While the TCP three-way handshakes completed for all three connections, the `TLSv1.3 Client Hello` was sent exclusively over the port 443 connection. The two connections to port 80 were established but remained idle, with no `HTTP GET` requests sent. Later Chrome also attempted to establish a QUIC connection for `example.com`.

This behavior is a strategy, where parallel connections to port 80 function as a possible browserside internal optimization. By establishing these connections preemptively, Chrome prepares for a potential fallback scenario where a server may not support HTTPS, aiming to improve performance without waiting for a server-side redirect. This is one of several security measures in the browser, which also include enforcing the HSTS Preload List [8] and adhering to the Mixed Content Specification by blocking or upgrading insecure sub-resources within a secure page [18].

## 4.2. Firefox

Firefox, developed by the Mozilla Foundation, is an open-source browser that operates on its own Gecko engine. Firefox has implemented security features such as an HTTPS-Only Mode and an HTTPS-First Mode [10], with the latter being the default in its Private Browsing mode since version 91 [19].

The analysis of Firefox (v143.0.1) demonstrated a strategy that initiates connections exclusively over HTTPS. Upon entering `example.com`, Firefox made no attempt to connect to port 80 (HTTP). Instead, it immediately opened two parallel TCP connections directly to port 443 (HTTPS). The `TLSv1.3 Client Hello` was sent over the first of these two secure connections to complete the handshake. Firefox used the established TCP/TLS connection and did not attempt a QUIC upgrade, although background traffic indicated QUIC capability for other services.

This method differs from the speculative fallback optimization observed in other browsers, as it completely bypasses the insecure protocol during the initial connection phase. This approach improves security by eliminating the brief window where an attacker might interact with an insecure connection. The HTTPS-First mode in Firefox is designed to be opportunistic, attempting an upgrade to HTTPS and only falling back to HTTP if a secure connection cannot be established [10]. The use of two parallel connections to port 443 is an internal performance optimization to ensure a fast and reliable TLS handshake.

## 4.3. Safari

In our test of Safari (v18.5), the browser implemented an HTTPS-First protocol by initiating two parallel TCP connection attempts: one to port 80 (HTTP) and one to port 443 (HTTPS). Following the successful completion of both TCP handshakes, Safari sent the `TLSv1.3 Client Hello` exclusively over the port 443 connection. The connection on port 80 was established but remained idle. Safari did not attempt to upgrade to QUIC/HTTP3 for `example.com`.

This behavior is a HTTPS-First approach that uses a speculative optimization, similar to Chromium-based browsers but with fewer parallel connections. By preparing a connection to port 80, Safari is ready to fall back to HTTP if the HTTPS attempt fails, but it prioritizes the secure connection. Safari also incorporates security features such as support for the HSTS Preload List [8] and mixed content blocking [18], aligning with modern web security practices.

## 4.4. Microsoft Edge

In 2021, Microsoft introduced an HTTPS upgrade feature termed "Automatic HTTPS" [20]. Our analysis of Microsoft Edge (v140.0.3485.94) confirmed that its

connection strategy is identical to that of Google Chrome. Upon entering the schemeless URL, Edge initiated three parallel TCP connection attempts: two to port 80 and one to port 443. All three handshakes completed, but the `TLSv1.3 Client Hello` was sent only over the port 443 connection. The other two connections to port 80 remained unused. Edge also attempted to establish a QUIC connection for `example.com` after the initial TLS handshake.

This behavior is a direct result of its Chromium foundation. The parallel connections serve as a speculative optimization, allowing the browser to proceed with a secure TLS handshake while being prepared for a fallback scenario. As part of the Chromium ecosystem, Edge also implements security features such as the HSTS Preload List and mixed content blocking [8], [18].

### 4.5. Opera

The analysis of Opera One (v122.0.5643.71) revealed a connection strategy identical to that of Firefox. Opera made no attempt to connect to port 80. Instead, it immediately opened two parallel TCP connections directly to port 443 (HTTPS) and sent the `TLSv1.3 Client Hello` over the first available secure connection. Opera also attempted a QUIC upgrade for `example.com` after establishing the TLS connection.

This finding shows a divergence from other Chromium-based browsers in the study. Opera has implemented a networking strategy for initial connections that aligns with Firefox's approach, prioritizing a secure-only initial attempt. This choice enhances security by removing the opportunity for downgrade attacks during the connection phase. While it customizes this initial step, Opera still utilizes the broader security architecture of the Chromium engine, including its handling of mixed content and support for HSTS.

TABLE 2: Observed Browser Behavior Summary for `example.com`

| Browser | Initial TCP SYNs | QUIC Attempt |
|---|---|---|
| Chrome | 1x HTTPS, 2x HTTP | Yes |
| Edge | 1x HTTPS, 2x HTTP | Yes |
| Safari | 1x HTTPS, 1x HTTP | No |
| Firefox | 2x HTTPS only | No |
| Opera | 2x HTTPS only | Yes |

## 5. Conclusion

The web has undergone a significant but incomplete transition towards a secure-by-default model. Our analysis shows that the default behavior for most major browsers has shifted from HTTP-First to a more secure HTTPS-First model. And we can cofnidently say that HTTPS has become the new standard for the web. However, this is fundamentally a web compatibility measure, not a security guarantee. The reliance on a silent, insecure fallback provides a clear and exploitable entry point for man-in-the-middle attacks like SSL stripping.

Truly reliable protection is only achieved through explicit, server-declared policies. HTTP Strict Transport Security, particularly when combined with the HSTS Preload List, remains the gold standard, offering the only comprehensive solution to the "first-visit" vulnerability that plagues all other mechanisms.

The complete deprecation of HTTP, while a desirable end-goal, is currently infeasible. The need to maintain access to legacy content and the fundamental architectural challenge of extending the web's trust model to private, local networks ensures that HTTP will persist as a necessary fallback for the foreseeable future. Future efforts must therefore focus not only on increasing HSTS adoption but also on developing new standards to solve the problem of local network encryption, which remains the most significant barrier to a fully encrypted web.

## References


[1] Internet Engineering Task Force (IETF), "HTTP Over TLS," https://tools.ietf.org/html/rfc2818, 2000, [Online; accessed September, 2025].

[2] ——, "HTTP Strict Transport Security (HSTS)," https://tools.ietf.org/html/rfc6797, 2012, [Online; accessed September, 2025].

[3] C. Kerschbaumer, F. Braun, S. Friedberger, and M. Jürgens, "The State of https Adoption on the Web," 2025.

[4] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz, "Measuring HTTPS Adoption on the Web," in *USENIX Security Symposium*, 2017.

[5] A. Alabduljabbar, R. Ma, S. Choi, R. Jang, S. Chen, and D. Mohaisen, "Understanding the Security of Free Content Websites by Analyzing their SSL Certificates: A Comparative Study," in *Proceedings of the 1st Workshop on Cybersecurity and Social Sciences (CySSS '22)*. ACM, 2022. [Online]. Available: https://doi.org/10.1145/3494108.3522769

[6] C. Jackson and A. Barth, "Forcehttps: protecting high-security web sites from network attacks," in *Proceedings of the International Conference on World Wide Web*, 2008.

[7] The Certification Authority Browser Forum (CA/Browser Forum), "Latest Baseline Requirements," https://cabforum.org/working-groups/server/baseline-requirements/requirements/, 2024, [Online; accessed September, 2025].

[8] C. Project, "Check HSTS preload status and eligibility," https://hstspreload.org/, 2012, [Online; accessed September, 2025].

[9] Mozilla, "Preloading HSTS," https://blog.mozilla.org/security/2012/11/01/preloading-hsts/, 2012, [Online; accessed September, 2025].

[10] C. Kerschbaumer, J. Gaibler, A. Edelstein, and T. van der Merwe, "HTTPS-Only: Upgrading all connections to https in Web Browsers," *Proceedings on Measurements, Attacks, and Defenses for the Web*, 2021.

[11] The World Wide Web Consortium (W3C), "Upgrade Insecure Requests," https://www.w3.org/TR/upgrade-insecure-requests/, 2015, [Online; accessed September, 2025].

[12] Internet Engineering Task Force (IETF), "HTTPS RR," https://datatracker.ietf.org/doc/draft-ietf-dnsop-svcb-https/00/, 2020, [Online; accessed September, 2025].

[13] KirstenS, "Cross site scripting (xss)," https://owasp.org/www-community/attacks/xss/, 2019, [Online; accessed October, 2025].

[14] K. Drakonakis, S. Ioannidis, and J. Polakis, "The Cookie Hunter: Automated Black-Box Auditing for Web Authentication and Authorization Flaws," in *Proceedings of the Conference on Computer and Communications Security*, 2020.

[15] M. Marlinspike, "New Tricks For Defeating SSL In Practice," https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf, 2009, [Online; accessed September, 2025].

[16] StatCounter, "Desktop Browser Market Share Worldwide," https://gs.statcounter.com/browser-market-share/desktop/worldwide, 2025, [Online; accessed September, 2025].

[17] C. Blog, “Increasing HTTPS adoption,” https://blog.chromium.org/2021/07/increasing-https-adoption.html, 2021, [Online; accessed September, 2025].

[18] The World Wide Web Consortium (W3C), “Mixed Content,” https://www.w3.org/TR/mixed-content/, 2023, [Online; accessed September, 2025].

[19] Mozilla, “Firefox 91 introduces HTTPS by Default in Private Browsing,” https://blog.mozilla.org/security/2021/08/10/firefox-91-introduces-https-by-default-in-private-browsing/, 2021, [Online; accessed September, 2025].

[20] Microsoft Edge Team, “Available for preview: Automatic HTTPS helps keep your browsing more secure,” https://blogs.windows.com/msedgedev/2021/06/01/available-for-preview-automatic-https-helps-keep-your-browsing-more-secure, 2021, [Online; accessed September, 2025].

 doi: 10.2313/NET-2026-05-1_05

# Systematic Review of Time-Series Machine Learning from a Networking Perspective

Liyan Qu, Johannes Späth*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: liyan.qu@tum.de, spaethj@net.in.tum.de

***Abstract*—Time-series machine learning (TSML) aims to handle data with temporal dependencies, which differs from traditional machine learning(ML) that assumes samples are independent and identically distributed (i.i.d.). TSML needs to consider dynamic characteristics such as trends, seasonality, and concept drift. Therefore, developing specific TSML tools is necessary. In recent years, plentiful TSML tools have emerged, but most studies focus on algorithm principles or single tasks, lacking systematic comparisons from the perspectives of tasks and tool characteristics. This paper takes a task-oriented approach to review and compare TSML tools covering major application scenarios, focusing on six tasks: imputation, feature extraction, forecasting, anomaly detection, change point detection, and classification. Based on this, the paper summarizes key characteristics to consider when selecting tools, such as time index awareness, backtest, multivariate support, scalability and output format. The review results can provide structured references for researchers without a machine learning background, helping them choose suitable TSML tools in practical engineering scenarios.**

***Index Terms*—time-series, machine learning, networking, forecasting, anomaly detection**

## 1. Introduction

Machine learning (ML) has achieved significant success in various fields such as image recognition, speech analysis, and financial forecasting [1]. In recent years, ML methods have also been applied to computer networking tasks, such as traffic forecasting, intrusion detection, and service classification [2]. Network-related data (e.g., traffic, latency, bandwidth utilization) naturally change continuously over time and are typical time-series data. However, time-series data often do not satisfy the independent and identically distributed (i.i.d.) assumption of traditional ML, as temporal dependencies, trends, seasonality, and concept drift exist between data samples [3]. Therefore, directly applying of traditional ML methods to time-series data is a significant challenge, and specific tools for TSML are required.

Recently, a large amount of research has been conducted on time-series machine learning (TSML) [4]. Existing reviews often focus on specific tasks [5], but systematic studies from the perspectives of tasks and tools remain limited. Especially for researchers without a machine learning background, there is still a lack of comprehensive review to guide practical tool selection. This paper aims to fill this gap by systematically reviewing and comparing actively maintained open-source tools (such as tsfresh, sktime, Darts, and Kats) from the perspective of TSML tasks. Rather than delving into the internal algorithms or model architectures, this paper focus on analyzing the tools' functional characteristics and practical applicability. While Python-based libraries(such as tsfresh, sktime, Darts, and Kats) are the main focus, several representative tools in other programming languages are also briefly discussed. The selected tools are mostly actively maintained open-source libraries that cover common time-series tasks such as imputation, feature extraction, forecasting, anomaly detection, change point detection, and classification.

This paper starts from the Darts library. We searched Google Scholar and GitHub using keywords such as "time series machine learning", "time series forecasting", and "time series review", and then filtered out the mainstream tools that are still actively maintained. Based on their official documentation and functional descriptions, we categorized and organized them according to typical TSML tasks, with an aim to cover a sufficient number of representative tools for each task type.

The structure of this paper is as follows: the second section introduces the essential prerequisite knowledge; the third section systematically analyzes various tools and their characteristics according to task types; the fourth section provides summary and limitations of this paper.

## 2. Prerequisite knowledge

In this section, we introduce some concept that will be used in the following sections.

### 2.1. Superviesd and Unsupervised Learning

In therms of what feedback can the learning model received, there are supervised learning, unsupervised learning [6]. Supervised learning models rely on data with class labels. Unsupervised learning does not rely on labels, it extracts patterns and structures from data spontaneously.

### 2.2. Online and Batch Learning

In terms of how data is fed into the learning model, there are online learning and batch learning [6]. In online learning, data is fed into the model one sample at a time, and the model updates itself incrementally with each new

 doi: 10.2313/NET-2026-05-1_06

sample. In batch learning, the model is trained on the entire dataset at once, and does not update itself until it is retrained with a new batch of data.

### 2.3. Features of TSML

The concepts below represent characteristics arising from the temporal nature of time-series data.

**Peek into the future:** using data information from future time points during model training or validation, which can lead to overly optimistic evaluation results and model overfitting [7].

**Trend:** a long-term sustained upward or downward direction in the time series, reflecting overall changes in the system over longer time scales [7].

**Seasonality:** periodic fluctuation patterns in the time series over time, such as daily, weekly, monthly, or annual cycles [7].

**Concept drift:** changes in data distribution or generation mechanisms over time, leading to model performance degradation on future data [8].

## 3. TSML Task Types and Tools

This section introduces common task types in TSML and their corresponding tools. The content is organized in a bottom-up sequence: basic mathematics and statistical tools, preprocessing tools, specific task tools, and comprehensive frameworks.

To provide clear guidance for selection, for each task type, this paper presents task definition, typical networking problems, key features that tools of this type should possess, and a comparison table of tools along with its explanation.

### 3.1. Foundational Libraries

Most TSML tools are based on similar underlying numerical and data processing libraries. NumPy provides matrix and vector operations [9], SciPy extends statistical and optimization methods [10], and pandas manages structured data and time indexing [11]. This section does not detailly introduce the specific dependencies of each tool, only indicates that, it is important to ensure the correct installation of relevant libraries.

### 3.2. Data Preprocessing

Raw time-series data often have problems such as redundancy, missing and noisy data, which adversely affect subsequent task processing. To improve performance and stability and reduce the overhead of downstream models, data preprocessing is usually required before implementing specific tasks, to reduce the burden on downstream algorithms.

**3.2.1. Imputaion.** Imputation [12] refers to estimating and filling in missing values in time-series data by leveraging existing observed data points. Its goal is to reconstruct data integrity and avoid model deviation or performance reduction due to missing data. For example, as discussed in [13], in network traffic monitoring, packet loss during sampling or transmission latency often leads to observation data missing, which needs to be reconstrcted through imputation to ensure the accuracy of subsequent processes such as, anomaly detection or capacity planning.

When selecting imputation tools, the following five aspects should be considered:

- **Time-Index-Aware**: Whether the tool can recognize and utilize timestamps, and adjust imputation logic according to time intervals.
- **No-Lookahead**: Whether the tool prevents using future observations during imputation, thus avoiding data leakage or causal violations.
- **Multivariate Support**: Whether the tool can jointly impute multiple correlated variables (columns) instead of processing each variable independently.
- **Pipeline-Compatible**: Whether the tool provides standardized interfaces or APIs that allow seamless integration into machine-learning pipelines.
- **Probabilistic Output**: Whether the tool can provide probabilistic estimates, such as confidence intervals, quantiles, or sample distributions, in addition to single point estimates.

This section introduces six representative imputation libraries commonly used in TSML: pandas-imterpolate() [14], SciPy-interpolate [15], sklearn-SimpleImputer [16], sklearn-KNNImputer [17], sklearn-IterativeImputer [18], and statsmodels-SARIMAX [19]. Table 1 compares these imputation tools in terms of the above features.

All comparison tables in this chapter use the following notation: "yes" indicates that the tool directly supports the feature, "partial" indicates that it is partially implemented only under specific conditions or in a limited way, "user implemented" indicates that it can be implemented through user-defined code, and "no" indicates that it is not supported at all.

**3.2.2. Feature Extraction.** Feature extraction [20], also known as representation learning, refers to the process of transforming raw or high-dimensional data into structured numerical features. Most ML models cannot directly handle raw unstructured data. Therefore, emphasizing important information and denoising through feature extraction, enables the data to be directly used by models. Feature extraction can reduce training costs, improve information relevance, enhance interpretability (although sometimes it may reduce it), and is often accompanied by dimensionality reduction.

Moreover, as we mentioned before, unlike traditional static ML, TSML also needs to capture temporal dependencies, trends, seasonality, and volatility. Therefore, when using non-deep learning models, specialized time-series feature extraction tools are required.

For instance, after missing traffic data are imputed, feature extraction is often applied to transform raw measurements into statistical indicators, such as packet inter-arrival variance, flow duration, or burstiness metrics, that help downstream anomaly-detection models identify network congestion or attacks.

When selecting feature-extraction tools, the following six aspects should be considered:

- **Programming Language**: The programming

TABLE 1: Comparison of Time-Series Imputation Tools

| Tool | Time-Index-Aware | No-Lookahead | Multivariate Support | Pipeline-Compatible | Probabilistic Output |
|---|---|---|---|---|---|
| pandas — interpolate | yes | partial | no | partial | no |
| SciPy — interpolate | yes | no | no | no | no |
| sklearn — SimpleImputer | no | partial | no | yes | no |
| sklearn — KNNImputer | no | partial | yes | yes | no |
| sklearn — IterativeImputer | no | partial | yes | yes | user implemented |
| statsmodels — SARIMAX | yes | partial | partial | user implemented | yes |

environments in which the tool is implemented.
• **Built-in Imputation**: Whether the tool provides built-in imputation functionality.
• **Multivariate Support**: Whether the tool can extract features from multiple variables (columns) at once.
• **Number of Features**: The number of features that the tool can extract; Catch22 can only extract a fixed set of 22 features.
• **Auto-Selection**: Whether the tool has mechanisms for automatically selecting features, rather than relying on user manual configuration.
• **Last Maintenance**: The date of the latest update to the tool, used to assess whether the project is still maintained and active.

This section introduces five representative feature extraction libraries commonly used in TSML: hctsa [21], Catch22 [22], tsfeatures [23], tsfresh [24], and TSFEL [25]. Table 2 compares these feature extraction tools in terms of the above features.

## 3.3. Specific Task Tools

According to different task goals, this section introduces four representative task types in TSML: forecasting, anomaly detection, segmentation/change-point detection, and classification.

**3.3.1. Forecasting.** Forecasting [26] is the most common and fundamental task type in TSML. Its goal is to utilize existing time-series data to build models, by capturing trends, seasonality, and random fluctuations, to predict explicit data values at future time points. Forecasting tasks require precise and stable results, to support decision-making and resource allocation.

Unlike traditional static ML forecasting, TSML forecasting emphasizes temporal continuity and causal constraints: training and testing must strictly follow the time order, and avoid the problem of "looking into the future". Moreover, modern TSML forecasting tools typically build a continuous loop: "forecast -> new data arrival -> monitoring -> model update", which is called online learning, rather than one-time training and output, known as batch learning.

As discussed in [27], link bandwidth or traffic variations forecasting is one of the typical applications in computer networks. Accurate traffic forecasting can help network systems to allocate resources and balance loads in advance, and to advoid congestion and performance degradation. Besides, network latency and packet-loss rate forecasting, data-center energy consumption forecasting, and server workload and task volume forecasting are also very practical applications [28].

When selecting forecasting tools, the following five aspects should be considered:
• **Backtest**: Whether the tool has built-in back test functionality, to prevent "looking into the future".
• **Online Update**: Whether the tool can continuously update models when new data arrive, to adapt to concept drift.
• **Output Format**: The types of forecast results the tool can generate—such as point estimates, prediction intervals, quantile values, sampled forecasts, or full predictive distributions. Richer outputs enable better uncertainty quantification.
• **Covariates Support**: Whether the tool allows incorporating external variables (such as temperature, traffic, or network load) to improve forecasting accuracy.
• **Dataset Scale**: The scale of datasets that the tool can efficiently handle, in terms of number of time series and length of each series. According to Hyndman [7], it can be classified as: Small: dozens to hundreds of series, each with less than 1000 time steps; Medium: hundreds to thousands of series, each with 1000-10000 time points; Large: tens of thousands of series, or each series exceeding 10000 time points.

This section introduces six representative forecasting libraries commonly used in TSML: Prophet [29], statsmodels-SARIMAX [30], StatsForecast [31], Darts [32], PyTorch-Forecasting [33], GluonTS [34], and MLForecast [35]. Table 3 compares these forecasting tools in terms of the above features.

**3.3.2. Anomaly Detection.** Anomaly detection [36] refers to the process of identifying data points or segments in a time series that significantly deviate from normal patterns. Its goal is to detect anomalous behaviors or potential system errors in time to prevent losses or performance degradation. Unlike static anomaly detection in traditional ML, TSML anomaly detection must consider temporal dependencies, periodic patterns, and inter-variable correlations.

As discussed in [37], distributed denial-of-service (DDoS) attack detection is a typical application scenario of anomaly detection in computer networks. This technique needs to identify sudden traffic surges or abnormal connection patterns in real time, to ensure network reliability. Intrusion detection, traffic matrix anomaly detection, and IoT network anomaly detection are also important applications.

When selecting anomaly detection tools, the following six aspects should be considered:
• **Backtest, Online Test, Multivariate Support**: Same as before.
• **Residual**: Whether the tool can remove trend and sea-

 doi: 10.2313/NET-2026-05-1_06

TABLE 2: Comparison of Time-Series Feature Extraction Tools

| Tool Name | Programming Language | Built-in Imputation | Multivariable Support | Number of Features | Auto-Selection | Last Maintenance |
|---|---|---|---|---|---|---|
| hctsa | Matlab | no | no | 7,700+ | yes | 2024-11-25 |
| Catch22 | C, Python, R Matlab, Julia | no | no | 22 | fixed 22 | 2022-06-21 |
| tsfeatures | R | no | no | 23+ | no | 2023-08-28 |
| tsfresh | Python | no | yes | 1200+ | yes | 2025-08-30 |
| TSFEL | Python | yes | yes | 65+ | no (select by config) | 2024-09-12 |

TABLE 3: Comparison of Time-Series Forecasting Tools

| Name | Backtest | Online Update | Output Format | Covariates Support | Dataset Scale |
|---|---|---|---|---|---|
| Prophet | built-in | yes | point /intervals | yes | small–medium |
| statsmodels (SARIMAX / ETS) | user-implemented | yes | point /intervals | yes | small–medium |
| StatsForecast | built-in | yes | point /intervals | yes (limited) | medium–large |
| Darts | built-in | yes | point /quantiles | yes | small–medium |
| PyTorch Forecasting | partial | yes | intervals /quantiles | yes | medium–large |
| GluonTS | built-in | yes | samples /distributions | yes | medium–large |
| MLForecast | partial | partial | point /quantiles | yes | large |

sonal components through forecasting models, and then detect anomalies in the residuals.

• **Anomaly Types**: The types of anomalies that the tool can identify, such as: Point Anomalies: Sudden spikes or drops at individual time points. Contextual Anomalies: Values that are normal in the overall trend,but anomalous under specific contexts or time conditions, e.g., high traffic at night. Subsequence Anomalies: Short sequences exhibiting anomalous patterns within a continuous time window, e.g., abnormal fluctuations within a cycle. Change-Point Anomalies: Persistent shifts in the baseline or statistical properties of the time series,e.g., significant changes in mean or variance. Multivariate Anomalies: Anomalous behaviors exhibited jointly across multiple time points or feature dimensions,which may appear normal when observed individually, but deviate from regular patterns overall.

This section introduces six representative time-series anomaly detection libraries: Merlion [38], ADTK [39], stumpy(Matrix Profile) [40], ruptures [41], Darts [32], and GluonTS [34]. Table 4 compares these anomaly detection tools in terms of the above features.

**3.3.3. Change Point Detection and Segmentation Tools.** Change point detection (CPD) [42] aims to identify timestamps in a time series where its statistical properties or structural patterns undergo significant and persistent changes, while segmentation divides the series into segments with similar behaviors and stable features based on these changes. Both tasks are usually implemented by the same tools, so they are introduced together in this paper. In TSML, this task is typically implemented in an unsupervised learning manner, without requiring predefined labels to detect changes. This process helps capture system status transitions, understand pattern evolution, and detect potential anomalies.

For example, as discussed in [43], online CPD on time series of latency and throughput in network measurements can be used to monitor real-time network service quality. Besides, CPD can also be used for network traffic structure change detection, network device or link status switching detection, network topology evolution/routing change detection, and network quality-of-service (QoS) baseline drift detection [44].

When selecting CPD or segmentation tools, the following five aspects should be considered:

• **Backtest, Online Test, Multivariate Support**: Same as before.

• **Unsuperviesd Types**: Whether the model support unsupervised training.

• **Confidence Quantification**: The degree of confidence quantification for each candidate change point result. Confidence levels include: high, medium, and low. High: Outputs significant quantities (such as p-values, confidence intervals, posterior probabilities), or provides calibrated probability metrics. Medium: Outputs probabilistic scores with built-in thresholds or adaptive calibration tools. Low: Only outputs raw scores or cost values without statistical calibration. This section introduces six representative CPD libraries commonly used in TSML: ruptures [41], changepoint(R) [45], stumpy(Matrix Profile) [40], ADTK [39], and Kats [46].

Table 5 compares these CPD tools in terms of the above features.

**3.3.4. Classification.** Time-series classification [47] aims to categorize time-series samples into predefined classes based on their overall patterns or features. Unlike traditional ML settings where "each sample corresponds to features at a single time point", in TSML, each sample consists of multiple time points, and models need to capture temporal dependencies, trends, and shapes simultaneously, to identify behavior types or system states.

As discussed in [48], in the field of network security, time-series classification is often used for malicious network traffic detection. Models analyze time-series features of network packets or connections (such as packet size distribution, inter-arrival times, flow duration) to learn

TABLE 4: Comparison of Time-Series Anomaly Detection Tools

| Name | Backtest | Online Update | Residual | Multivariable Support | Anomaly Types |
|---|---|---|---|---|---|
| Merlion | yes | yes | yes | yes | Point, Contextual, Collective |
| ADTK | partial | partial | partial | partial | Point, Contextual |
| stumpy (Matrix Profile) | user-implemented | partial | no | yes | Point, Subseq |
| ruptures | user-implemented | no | no | partial | Change |
| Darts/ GluonTS | yes | partial | yes | yes | Point, Contextual, Collective |

TABLE 5: Comparison of Time-Series CPD and Segmentation Tools

| Name | Backtest | Online Update | Multivariate Support | Unsuperviesd Types | Confidence Quantification |
|---|---|---|---|---|---|
| ruptures | partial | no | yes | yes | low |
| changepoint (R) | partial | no | user implemented | yes | high |
| stumpy (Matrix Profile) | partial | partial | user implemented | yes | low |
| ADTK | partial | partial | partial | yes | medium |
| Kats | yes | partial | yes | yes | high |

temporal patterns of different traffic types, thereby automatically distinguishing normal communication from malicious activities. Besides, there are applications such as encrypted traffic classification, protocol identification, and user behavior modeling [49].

When selecting TSC tools, the following five aspects should be considered:

• **Backtest**, **Online Test**, **Multivariate Support**: Same as before.

• **Variable-Length Handling**: Whether it can handle sequences with inconsistent time steps or varying lengths among samples.

• **Interpretability**: Whether the tool provides feature-importance analysis, visualization, or rule-based outputs.

This section introduces five representative TSC libraries commonly used in TSML: sktime [50], tsai [51], tslearn [52], pyts [53], and tsfresh [24].

Table 6 compares these TSC tools in terms of the above features.

### 3.4. Comprehensive Frameworks

At the same time, several comprehensive frameworks can handle multiple tasks: statsmodels supports imputation and forecasting with classical models (ARIMA, SARIMAX, ETS). tsfresh focuses on feature extraction and imputation for downstream ML tasks. Darts covers forecasting and anomaly detection using both statistical and deep models. GluonTS is designed for forecasting and anomaly detection with probabilistic outputs. ruptures is specialized in change-point and anomaly detection. stumpy detects anomalies and change points via the Matrix Profile. ADTK is dedicated to anomaly and change-point detection.

## 4. Conclusion and Limitations

This paper has systematically presented the essential prerequisite knowledge for time-series machine learning, the core task types, and task-oriented tool comparisons. Due to space limitations, all tables are presented in the appendix. The goal is to provide a clear path to quickly select appropriate tools for specific problems and to clarify the practical characteristics of each framework.

There are still some limitations: for each task type, only about five representative tools are included to cover approximately 90% of common scenarios; unconventional or niche applications are not discussed. Detailed algorithms, internal model architectures, and parameter settings of each library are beyond the scope of this paper.

Future work may expand this study by incorporating quantitative benchmarking, broader tool coverage, and unified evaluation criteria. Overall, this review serves as a structured reference for researchers and practitioners to navigate and select TSML tools effectively.

## References


[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, 2015.

[2] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, 2018.

[3] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data mining and knowledge discovery*, 2017.

[4] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Adversarial attacks on deep neural networks for time series classification," in *2019 International joint conference on neural networks (IJCNN)*. IEEE, 2019.

[5] M. Rußwurm and M. Körner, "Self-attention for raw optical satellite time series classification," *ISPRS journal of photogrammetry and remote sensing*, 2020.

[6] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, 2016.

[7] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[8] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, 2014.

[9] C. R. Harris, K. J. Millman, S. J. van der Walt *et al.*, "Array programming with NumPy," 2020.

[10] P. Virtanen, R. Gommers, T. E. Oliphant *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in python."

TABLE 6: Comparison of Time-Series Classification Tools

| Name | Backtest | Online | Multivariate | variable-length | Interpretability |
|---|---|---|---|---|---|
| sktime | partial | user implemented | yes | partial | yes |
| tsai | user implemented | user implemented | yes | user implemented | no |
| tslearn | user implemented | user implemented | partial | yes | partial |
| pyts | user implemented | user implemented | partial | partial | yes |
| tsfresh | user implemented | user implemented | yes | yes | yes |


[11] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, 2010.

[12] Y. Xu and R. Goodacre, "On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning," *Journal of analysis and testing*, 2018.

[13] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An information-theoretic approach to traffic matrix estimation," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 301–312.

[14] "pandas.DataFrame.interpolate," pandas official documentation, accessed: 2025-09-28. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html

[15] "scipy.interpolate," SciPy official documentation, accessed: 2025-09-28. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/interpolate.html

[16] "sklearn.impute.SimpleImputer," scikit-learn official documentation, accessed: 2025-09-28. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html

[17] "sklearn.impute.KNNImputer," scikit-learn official documentation, accessed: 2025-09-28. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html

[18] "sklearn.impute.IterativeImputer," scikit-learn official documentation, accessed: 2025-09-28. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html

[19] "statsmodels.tsa.statespace.SARIMAX," statsmodels official documentation, accessed: 2025-09-28. [Online]. Available: https://www.statsmodels.org/stable/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html

[20] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*, 2008.

[21] B. D. Fulcher and N. S. Jones, "hctsa: A computational framework for automated time-series phenotyping using massive feature extraction," *Cell systems*, vol. 5, no. 5, pp. 527–531, 2017.

[22] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, "catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis," *Data mining and knowledge discovery*, 2019.

[23] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications: with R examples*. Springer, 2006.

[24] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[25] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa, "Tsfel: Time series feature extraction library," *SoftwareX*, vol. 11, p. 100456, 2020.

[26] G. Bontempi, S. B. Taieb, and Y.-A. Le Borgne, "Machine learning strategies for time series forecasting," in *Business Intelligence*, 2013.

[27] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An information-theoretic approach to traffic matrix estimation," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003.

[28] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.

[29] "Prophet," accessed: 2025-09-28. [Online]. Available: https://facebook.github.io/prophet/

[30] "statsmodels," accessed: 2025-09-28. [Online]. Available: https://www.statsmodels.org/stable/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html

[31] "StatsForecast," accessed: 2025-09-28. [Online]. Available: https://nixtlaverse.nixtla.io/statsforecast

[32] "Darts," accessed: 2025-09-28. [Online]. Available: https://unit8co.github.io/darts/

[33] "PyTorch Forecasting," accessed: 2025-09-28. [Online]. Available: https://pytorch-forecasting.readthedocs.io/

[34] "GluonTS," accessed: 2025-09-28. [Online]. Available: https://ts.gluon.ai/

[35] "MLForecast," accessed: 2025-09-28. [Online]. Available: https://nixtlaverse.nixtla.io/mlforecast

[36] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, 2009.

[37] K. Garg and R. Chawla, "Detection of ddos attacks using data mining," *International Journal of Computing and Business Research (IJCBR)*, vol. 2, no. 1, pp. 2226–6166, 2011.

[38] "Merlion," accessed: 2025-09-28. [Online]. Available: https://salesforce.github.io/Merlion/

[39] "Anomaly Detection ToolKit (ADTK) documentation," accessed: 2025-09-28. [Online]. Available: https://adtk.readthedocs.io/

[40] "STUMPY (Matrix Profile)," accessed: 2025-09-28. [Online]. Available: https://stumpy.readthedocs.io/

[41] "ruptures," ruptures documentation (change point detection), accessed: 2025-09-28. [Online]. Available: https://centre-borelli.github.io/ruptures-docs/

[42] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and information systems*, 2017.

[43] C. Lévy-Leduc and F. Roueff, "Detection and localization of change-points in high-dimensional network traffic data," *The Annals of Applied Statistics*, pp. 637–662, 2009.

[44] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and information systems*, 2017.

[45] "R package 'changepoint' (CRAN)," accessed: 2025-09-28. [Online]. Available: https://cran.r-project.org/web/packages/changepoint/index.html

[46] "Kats," accessed: 2025-09-28. [Online]. Available: https://facebookresearch.github.io/Kats/

[47] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, 2019.

[48] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, 2012.

[49] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE transactions on network and service management*, 2019.

[50] "sktime," accessed: 2025-09-28. [Online]. Available: https://www.sktime.org/en/stable/

[51] "tsai (timeseriesAI)," accessed: 2025-09-28. [Online]. Available: https://timeseriesai.github.io/tsai/

[52] "tslearn," accessed: 2025-09-28. [Online]. Available: https://tslearn.readthedocs.io/

[53] "pyts," accessed: 2025-09-28. [Online]. Available: https://pyts.net/

# Enhancing QUIC with TAPS

İlhan Can Ateş, Marcel Kempf*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: ilhan.ates@tum.de, kempfm@net.in.tum.de

***Abstract*—QUIC has introduced a modern, encrypted, and multiplexed alternative to TCP. However, its adoption in general-purpose applications is limited by network ossification and API complexity. The IETF Transport Services (TAPS) architecture addresses these challenges by providing a protocol-independent interface that decouples applications from the underlying transport protocols. This paper explores and analyzes the mapping of QUIC's modern features, such as multi-streaming and connection migration, to the abstract TAPS concepts of "Messages" and "Connections," based on an IETF individual draft. We evaluate the design decisions behind this mapping and discuss how the two standards benefit each other. We conclude that TAPS facilitates safe QUIC deployment through automated fallback mechanisms, while QUIC serves as the ideal protocol to demonstrate the benefits of the TAPS architecture.**

***Index Terms*—quic, transport services, taps**

## 1. Introduction

The TCP/IP protocol suite has remained the dominant network architecture since its standardization, with Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) as the principal transport protocols deployed across the Internet [1]. However, as the Internet evolves towards mobile, encrypted, and latency-sensitive applications, the architectural limitations of traditional TCP—namely, head-of-line blocking and vulnerability to middlebox ossification—have become a bottleneck [2]. To address these challenges, the IETF standardized the QUIC protocol [3]. Although originally conceived by Google to accelerate HTTP web traffic [2], the IETF formalized QUIC as a general-purpose transport protocol. QUIC runs over UDP and provides modern features such as stream multiplexing, connection migration, and encrypted transport state.

Despite these technical advantages, network ossification remains a significant barrier to QUIC deployment. Numerous middleboxes and firewalls interfere with unrecognized protocols or options, thereby increasing the likelihood of connection failures for applications adopting new protocols [4]. While QUIC mitigates network ossification by operating over UDP, some restrictive networks block UDP traffic entirely, which necessitates a TCP fallback to maintain reliability [5]. This challenge is further compounded by legacy socket APIs that tightly couple applications to transport protocols, requiring developers to make static protocol choices during the design phase [6]. Consequently, there is limited incentive to migrate to newer alternatives such as QUIC, and developers frequently default to TCP.

The Internet Engineering Task Force (IETF) working group *Transport Services* (TAPS) was established to address deployment barriers associated with transport-layer protocols. The TAPS architecture specifies an abstract, protocol-independent interface that separates applications from underlying transport mechanisms [6]. Through this interface, applications can express requirements such as reliability and multiplexing, rather than selecting specific protocols. Additionally, the architecture supports dynamic racing mechanisms that attempt multiple protocols (e.g., QUIC and TCP) in parallel and automatically select the best available option without requiring application-level intervention.

This paper introduces the TAPS interface proposed by the TAPS working group, describes and analyzes the current draft mapping of QUIC for the TAPS interface, and discusses the remaining issues.

## 2. Background

To understand how a QUIC mapping can be built for TAPS, we first take a look at the QUIC protocol itself.

### 2.1. QUIC

QUIC is a transport protocol built on top of UDP, enabling stateful, encrypted, and reliable communication [3]. Rather than defining an entirely new protocol, it carries packets over UDP datagrams. This approach both circumvents severe forms of network ossification, which block non-standard transport protocols, and at the same time leverages the lightweight nature of UDP. As a result, QUIC reconstructs the reliability and state management features associated with TCP, while also adding confidentiality using integrated TLS [7].

A key feature of QUIC is its use of streams. These lightweight, independent byte streams are multiplexed within a single connection, ensuring reliable and ordered delivery [3]. Unlike TCP, where the loss of a single packet delays the delivery of all subsequent data until the missing packet is retransmitted—a vulnerability known as head-of-line (HOL) blocking— QUIC restricts this behavior to individual streams. Each stream manages its own flow control and retransmits lost packets independently, so packet loss in one stream does not affect others. The result is increased data transfer efficiency.

QUIC communication is encrypted by default, and QUIC handshakes include the TLS parameters. Setting

 doi: 10.2313/NET-2026-05-1_07

up a secure TLS layer on top of TCP requires two round trips: one TCP handshake and one TLS handshake. In comparison, QUIC allows for a significantly faster, one-round-trip (1-RTT) establishment of a secure channel. After a connection is established, 0-RTT connections are also possible, further reducing latency [7]. QUIC packets can contain multiple frames, with a frame being the smallest protocol unit. There are various frame types in QUIC. These objects are used to communicate not just application data (e.g., `STREAM` frames) but also to manage all protocol activity. Examples include `ACK` frames for acknowledgements, `CONNECTION_CLOSE` frames for immediate connection termination, and `MAX_DATA` and `MAX_STREAM_DATA` frames for flow control on a connection or stream basis, respectively [3]. By exchanging as much control information as possible in frames, QUIC ensures that this information cannot be read in transit, as the payload is encrypted. This design allows easy protocol updates by adding new frame types, unlike TCP, which is difficult to extend due to network ossification, as demonstrated by Honda et al. [4].

QUIC also considers the increasingly mobile nature of the Internet, where clients' connections may not always follow the same network path. Unlike TCP, which requires a new connection if the client IP changes, QUIC enables connection migration between networks (e.g., WiFi to 5G) through persistent connection identifiers, without the added delay of a new connection handshake [3].

Work on QUIC began at Google in 2012 [8]. An Internet Draft submission to the IETF followed in 2015, leading to standardization as an RFC in 2021 [3]. HTTP/3, the current latest version of the Hypertext Transfer Protocol (HTTP), uses QUIC at the transport layer. As of December 2025, about 36.8% of websites use HTTP/3, and roughly 8.6% use QUIC on its own [9], [10].

## 2.2. TAPS

Transport Services (TAPS) is an asynchronous and event-driven architecture designed to provide applications with a unified interface for transport protocols. As outlined in RFC 9621, its primary goal is to decouple the API from the underlying protocol. This abstraction enables flexibility in protocol selection without requiring applications to hardcode specific transport mechanisms.

**2.2.1. Architecture and Objects.** The TAPS architecture introduces several objects, defined in RFC 9621 and RFC 9622:

**Preconnection:** The Preconnection object contains the configuration before a connection is established. Applications specify their preferences, requirements, and constraints using Transport Properties. These can influence the selected protocol or path with Selection Properties, protocol-specific requirements used during connection establishment with Connection Properties, or specific details about how messages are transmitted with Message Properties. Security Parameters are used to specify the criteria and parameters for transport security protocols, such as identities or keys [6].

A Preconnection has different methods to establish a Connection or a Listener. The `Listen` method can be used to create a Listener that starts listening for connections. A new outbound Connection can be established using `Initiate`, or one with 0-RTT, if supported, can be established with `InitiateWithSend`. A peer-to-peer Connection can be established using `Rendezvous` [6].

Because the TAPS API is asynchronous, these methods do not block application execution; instead, they initiate a process that progresses through various lifecycle states, eventually emitting an event when the Connection reaches the `Ready` state to indicate successful establishment.

**Connection:** The TAPS architecture keeps the definition of a Connection abstract, as each transport protocol has a different understanding of it. For TCP, it can be an active connection, whereas for UDP, it can be the 5-tuple. A set of connections can be part of a Connection Group, where they use the same Connection Properties and can be multiplexed together. Once established, a Connection can `Send()` and `Receive()` Messages, or be terminated forcefully with `Abort()` or gracefully with `Close()` [6].

**Message:** TAPS defines a Message as the smallest unit of data that will be transported. The Message abstraction allows applications to work with discrete, bounded units of data rather than unstructured byte streams. This closely matches how most modern applications use the transport layer [6]. For example, HTTP builds its own Request and Response objects on top of TCP, where data is sent as a stream of bytes.

**Framer:** A Framer can be added to a Connection to add message boundaries and structure to protocols that use unstructured streams of data to fit in the TAPS concept of Messages [6]. A Framer is required in the TAPS implementation of TCP, where data is sent as a byte stream and often restructured into a defined structure on the application layer. In contrast, UDP datagrams already have clearly defined boundaries, making a Framer unnecessary. Similarly, QUIC streams are byte-oriented and require framing to preserve TAPS Message boundaries, as discussed in Section 4.3.2.

**2.2.2. Connection Initiation and Protocol Selection.** One of the distinguishing features of TAPS is its ability to attempt multiple protocols concurrently during the connection establishment process [11]. For example, if both TCP and QUIC satisfy the application's requirements, the system may initiate both in parallel. QUIC might be unavailable in some environments; in such cases, the system falls back silently to TCP. This approach mitigates ossification by avoiding hardcoded protocol dependencies.

# 3. Mapping QUIC to TAPS

In this section, we will take a look at the draft mapping of QUIC to TAPS by T. Pauly [12].

## 3.1. TAPS Objects

A **Connection** is mapped to a single QUIC stream.

A **ConnectionGroup** is mapped to a single QUIC connection. Security parameters are shared between Connections of a ConnectionGroup, according to the TAPS definition [6]. These are mapped to security properties of QUIC, i. e. TLS parameters, which are also shared inside a single QUIC connection [3], [7].

A **Message** corresponds to data sent in a single `Send()` operation. This data is transmitted using QUIC STREAM frames, though a single message may be fragmented across multiple frames depending on the Maximum Transmission Unit (MTU), as well as QUIC's flow control and congestion control mechanisms.

### 3.2. Connections and Establishment

Calling `Initiate` on a Preconnection will establish a new connection and return a new QUIC stream. If there already was an established connection, only a new stream ID will be allocated for use in a new stream. `InitiateWithSend` does the same, but does the connection establishment with 0-RTT if possible.

A Connection may not reach the `Ready` state if the new stream cannot be created due to various reasons, for example a maximum stream count limit.

Calling `Clone` on a Connection will create a new QUIC stream on the same underlying QUIC connection. The new Connection will be put in a `ConnectionGroup` with the old Connection.

Multiple Connections, or mapped QUIC Streams, received by a `Listener` from the same QUIC Connection will be grouped into one `ConnectionGroup`.

### 3.3. Sending and Receiving

**Send** will send data using a STREAM frame. The ID stored in the Connection object will be used for the Stream ID. No use of a `Framer` is mentioned in the Pauly draft [12]. The need for a framer is discussed in Section 4.3.2.

**Receive** is called by the application to start receiving data.

### 3.4. Closing Connections

**Close** closes a QUIC stream using the `FIN` bit on the stream.

**Abort** closes a QUIC stream forcefully using a `RESET_STREAM` frame.

**CloseGroup** closes a QUIC connection after all streams have been closed, using a `CONNECTION_CLOSE` frame.

**AbortGroup** closes a QUIC connection forcefully using a `CONNECTION_CLOSE` frame.

### 3.5. Connection Properties

RFC 9622 [13] lists a set of Transport Properties for TAPS with explanations on how implementations should interpret them. The Pauly draft does not specify how Transport Properties will be mapped to QUIC. This seems to be a critical flaw in the current draft, as the mapping is essential for TAPS' protocol selection. However, RFC 9623 includes guidelines for implementing various TAPS features for different types of protocols and some examples, including multiplexed and stream based connections [11]. In the following, we evaluate a small subset of these properties and consider how they can be used within a QUIC implementation.

**3.5.1. Reliability.** The `reliability` Transport Property specifies whether the selected protocol should ensure that the messages are sent and received without loss [13]. Standard QUIC can be used if reliability is preferred or required. When an unreliable protocol is needed, the QUIC DATAGRAM extension offers unreliable delivery similar to UDP. However, these datagrams remain subject to the flow and congestion control mechanisms of the underlying QUIC connection. [14].

**3.5.2. Multistream Connections.** The `multistreaming` Transport Property specifies whether multiple Connections in a Connection Group need to be provided by the selected protocol [13]. QUIC provides this by default. If strictly prohibited, this QUIC implementation should not be used. It is possible to make another QUIC mapping that makes this possible, which is discussed in 4.4.

**3.5.3. Direction of Communication.** The `direction` Transport Property specifies the direction of communication between the two hosts [13]. QUIC supports both unidirectional and bidirectional streams. These options can directly be used when initiating a new TAPS Connection (i. e. creating a new QUIC stream.) [3]

## 4. Discussion and Analysis

This section evaluates the mapping drafted by Pauly [12], discusses how QUIC and TAPS benefit from this mapping, and analyzes the decisions made in comparison to an alternative mapping.

### 4.1. How QUIC Benefits from TAPS

TAPS aims to help developers more easily adopt new protocols, such as QUIC. As noted in Section 2.2.2, when TAPS falls back to TCP, it enables developers to experiment with and deploy QUIC without requiring them to rewrite their applications. Developers do not need to create two distinct, tightly coupled implementations; instead, TAPS abstracts the underlying protocol, making integration of QUIC straightforward.

Since QUIC is primarily designed to transport HTTP/3, numerous libraries currently support it. This focus, however, creates a gap for other applications, including online gaming and virtual private networks (VPNs), which could benefit from QUIC's advanced transport features. For instance, online gaming applications can leverage 0-RTT handshakes to reduce initial connection latency, while VPNs can employ connection migration to maintain uninterrupted sessions when a client's underlying IP address changes. The Transport Services (TAPS) interface enables developers to access QUIC through a unified and simplified framework, reducing the complexity of lower-level APIs and facilitating broader adoption across diverse use cases.

Google developed and deployed QUIC, taking advantage of its control over Chromium, which represented more than 78% of the browser market as of September 2025 [15]. This dominant position enabled Google to rapidly implement QUIC support in both its browser and backend infrastructure [2]. In contrast, the process would likely have been much more challenging for a smaller

company with fewer resources and less influence. More broadly, TAPS aims to make protocol integration easier and more accessible to all developers.

## 4.2. How TAPS Benefits from QUIC

QUIC supports the TAPS objectives by providing protocol features unavailable in standard TCP, such as additional connection properties and protocol options (see Section 3.5). These features enable TAPS to expand its protocol options.

A primary benefit of QUIC for TAPS is the greater flexibility it offers after connection establishment, as demonstrated by QUIC's native support for connection migration. Furthermore, QUIC allows TAPS to fulfill the `zeroRttMsg` property, which enables applications to transmit data during connection setup.

TAPS applications instantiate SecurityParameters to request confidentiality and authentication. While legacy transports require a separate security protocol such as TLS, QUIC provides these security properties natively [7]. This enables a TAPS implementation to meet security requirements with a single, integrated protocol stack. As a result, complexity and handshake overhead are reduced.

The TAPS `reliability` parameter can be utilized with the QUIC DATAGRAM extension. This enables a single transport to support both reliable and unreliable message delivery for Transport Services [14]. It enhances flexibility and efficiency in diverse networking scenarios.

## 4.3. Analysis of Mapping Decisions

**4.3.1. Multistreaming.** Mapping one QUIC stream to one TAPS Connection does not utilize QUIC's multistreaming capabilities by default. This design can recreate TCP's HOL-blocking problem unless the application deliberately establishes multiple TAPS Connections. As stated in RFC 9621, the TAPS API should not be used to directly choose one protocol. It is designed to allow the caller to only specify requirements without concern for specific protocol choice [6]. However, to utilize QUIC's independent streams, the developer must recognize these mapping implications and ensure the application uses QUIC efficiently, which contradicts the intended use of the TAPS interface.

**4.3.2. Framer.** The Message-oriented API of TAPS assumes that message boundaries are preserved during transmission, allowing receivers to distinguish where one message ends and another begins. However, unless used with the QUIC DATAGRAM extension, QUIC streams provide a byte stream abstraction without inherent message delimiters, similar to TCP. The Pauly draft does not propose a message framing mechanism, leaving message boundary preservation as a concern for implementation [12].

If TAPS Messages are sufficiently small and each message fits entirely within a single QUIC STREAM frame, frame boundaries could implicitly serve as message boundaries. However, this method is unreliable because QUIC may fragment messages across multiple STREAM frames due to congestion control, maximum packet size, or flow control constraints. Furthermore, the STREAM frame structure is not exposed to the application layer. Relying on frame boundaries would introduce an implicit dependency on QUIC implementation details that are beyond application control.

A lightweight framing protocol is needed to support TAPS Message semantics over QUIC streams. HTTP/3 uses a simple Type-Length-Value frame format [16]. A similar framing structure can be used for the TAPS mapping of QUIC.

## 4.4. Alternative Mapping

An alternative mapping is possible, which maps a single QUIC connection to a TAPS Connection, and a TAPS Message to a single QUIC stream. This approach has benefits but also represents a non-traditional use of QUIC depending on the application's needs.

QUIC streams have well-defined lifetimes. A stream is implicitly opened with its first STREAM frame, carries data, and is closed (typically with the FIN bit), with minimal protocol overhead. Using one stream per message removes the need for a TAPS Framer, because stream boundaries implicitly define message boundaries. A stream can even be opened and closed within a single STREAM frame by setting the FIN bit, making this approach efficient for small messages. RFC 9000 explicitly allows an essentially unlimited number of streams, constrained only by the receiving party's advertised limits [3].

However, this approach has limitations. TAPS has a `preserveOrder` Transport Property, which specifies whether Messages need to arrive in the same order they were sent. The one-message-per-stream mapping cannot fulfill this property, as QUIC only guarantees ordering within individual streams, not across them [3]. Additionally, for workloads involving many small messages, the per-stream state management and flow control accounting may become significant, and applications could exhaust the receiver's maximum concurrent stream limit.

TAPS does not natively expose the stream abstraction that sits between connections and messages in QUIC's design. The choice between Connection-to-Stream and Connection-to-Connection mappings can be guided by TAPS Connection or Message Properties, allowing for the full utilization of QUIC's features in each scenario.

# 5. Conclusion and Future Work

In this paper, we presented QUIC and TAPS, and analyzed an expired draft mapping of QUIC to TAPS proposed by T. Pauly [12].

A central challenge in mapping QUIC to TAPS is preserving message boundaries over QUIC's byte stream abstraction. We identified two viable approaches: using explicit message framing (similar to HTTP/3's frame format) within QUIC streams, or mapping individual TAPS Messages to separate QUIC streams. Each approach presents distinct tradeoffs between implementation complexity, ordering guarantees, and protocol overhead.

As mentioned in Section 4.1, QUIC was primarily designed to transport HTTP/3, which has reduced the immediate need for a generic API implementation. However, as QUIC adoption grows beyond HTTP in other applications, a standardized TAPS mapping becomes increasingly valuable. The availability of a TAPS mapping

democratizes access to QUIC for developers outside of tech giants, and provides the foundation that might allow QUIC to replace TCP in general-purpose applications beyond browsers. The expired draft mapping should be revisited and extended with explicit guidance on message framing to accelerate this process.

## References


[1] G. Fairhurst, B. Trammell, and M. Kühlewind, "Services Provided by IETF Transport Protocols and Congestion Control Mechanisms," RFC 8095, Mar. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8095

[2] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 183–196. [Online]. Available: https://doi.org/10.1145/3098822.3098842

[3] 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

[4] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend TCP?" in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 181–194. [Online]. Available: https://doi.org/10.1145/2068816.2068834

[5] M. Kühlewind and B. Trammell, "Applicability of the QUIC Transport Protocol," RFC 9308, Sep. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9308

[6] T. Pauly, B. Trammell, A. Brunstrom, G. Fairhurst, and C. Perkins, "Architecture and Requirements for Transport Services," RFC 9621, Jan. 2025. [Online]. Available: https://www.rfc-editor.org/info/rfc9621

[7] M. Thomson and S. Turner, "Using TLS to Secure QUIC," RFC 9001, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9001

[8] J. Roskind, "QUIC: Design Document and Specification Rationale," 2013, Google Docs, Accessed: 2025-12-16. [Online]. Available: https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34

[9] w3techs.com, "Usage Statistics of HTTP/3 for Websites, 13 Dec 2025," 2025, accessed: 2025-12-13. [Online]. Available: https://w3techs.com/technologies/details/ce-http3

[10] ——, "Usage Statistics of QUIC for Websites, 6 Dec 2025," 2025, accessed: 2025-12-06. [Online]. Available: https://w3techs.com/technologies/details/ce-quic

[11] A. Brunstrom, T. Pauly, R. Enghardt, P. S. Tiesel, and M. Welzl, "Implementing Interfaces to Transport Services," RFC 9623, Jan. 2025. [Online]. Available: https://www.rfc-editor.org/info/rfc9623

[12] T. Pauly, "A Transport Services Mapping for QUIC," Internet Engineering Task Force, Internet-Draft draft-taps-quic-mapping-00, Mar. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-taps-quic-mapping/00/

[13] B. Trammell, M. Welzl, R. Enghardt, G. Fairhurst, M. Kühlewind, C. Perkins, P. S. Tiesel, and T. Pauly, "An Abstract Application Programming Interface (API) for Transport Services," RFC 9622, Jan. 2025. [Online]. Available: https://www.rfc-editor.org/info/rfc9622

[14] 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

[15] Cloudflare Radar, "Browser Market Share Report for 2025 Q3," Sep. 2025, accessed: 2025-12-20. [Online]. Available: https://radar.cloudflare.com/reports/browser-market-share-2025-q3

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

# Efficient and Accurate Network Modeling with ML

Regina Bayer, Johannes Späth*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: regina.bayer@tum.de, spaethj@net.in.tum.de

***Abstract*—Efficient and accurate network models are critical tools to analyze a wide range of network scenarios, enabling performance estimation, capacity planning, and network optimization. Existing network simulators primarily build on traditional modeling techniques, such as queueing theory, fluid models and Discrete Event Simulation (DES). While DES remains the most commonly used method in practice, as it is valued for its high level of accuracy, network simulators exhibit elevated computational costs, limiting their scalability in large-scale network scenarios. In recent years, machine learning has emerged as a promising alternative to DES-based simulation, offering significantly faster computation. Among various machine learning architectures, Graph Neural Networks (GNNs) have shown particular potential due to their ability to naturally represent network topologies and capture complex dependencies of network elements as graphs. This paper provides a broad literature overview of recent GNN-based network modeling approaches, focusing on RouteNet-Gauss and TwinNet, and discusses their respective strengths and limitations. Additionally, it highlights the challenges of obtaining high-quality datasets for training machine learning models. and outlines complementary techniques, such as transfer learning and machine-learning-based dataset optimization, that aim to bridge the gap between simulated and hardware-measured data.**

***Index Terms*—network modeling, graph neural networks, discrete event simulation, transfer learning, simulation-to-hardware gap**

## 1. Introduction

A high-quality network model is critical for solving optimization problems, conducting network research, and performing capacity planning, as it enables the prediction of key performance metrics such as latency, packet loss and jitter. Therefore, a network model should satisfy two main criteria:

1) It must be computationally efficient, so that many network scenarios can be evaluated within a short period of time.
2) It must be accurate, since effective optimization is only possible when the underlying model reflects real network behavior.

Although techniques like queueing theory offer valuable insights, they often rely on strong assumptions that limit their utility in highly complex modern networks [1]. DES-based network simulators, with commonly used simulators such as OMNeT++ [2] and ns-3 [3], produce more accurate results, but at a high computational cost, limiting their scalability and therefore rendering them unsuitable for large-scale network scenarios [1].

In recent years, several machine-learning-based approaches have been proposed to address these challenges. In particular, GNN-based machine learning methods have demonstrated promising results due to their inherent ability to operate on graph-structured data. Notable recent advances that leverage GNNs, discussed in this paper, include RouteNet-Gauss [4] and TwinNet [1]. While the machine-learning-based network models discussed in this paper significantly improve computational efficiency, they still require large and realistic datasets for training, as their accuracy is largely dependent on a high-quality dataset. However, acquiring real network datasets is costly, poses privacy concerns, and testing scenarios directly in production networks is inherently risky. The lack of high-quality real-world datasets remains one of the key obstacles in developing robust machine-learning-based network models, potentially limiting their ability to generalize over scenarios not seen during training. Training models on simulated data offers a cheaper alternative to hardware-measured datasets. However, those models output less accurate results when applied in real-world scenarios, as DES-based simulators struggle to fully capture hardware-dependent behavior and probabilistic events.

This paper provides a broad overview of recent advances of machine-learning-based network modeling, presented in recent literature. Furthermore, the paper discusses the issues of obtaining meaningful datasets and presents recent approaches that aim to overcome the gap between measured and simulated datasets.

Section 2 introduces the functionality and limitations of DES-based simulation, as well as the fundamental principles of GNNs. Furthermore, it explores the reason why GNN-based network models are more efficient than machine learning models that rely on traditional Neural Networks (NNs). Section 3 presents several GNN-based network modeling approaches and compares their characteristics. Finally, Section 4 discusses the challenges of obtaining meaningful datasets for training machine-learning-based models and presents potential solutions.

## 2. Background

This section provides detailed information on DES and commonly used network simulators. It further introduces the general architecture of GNNs and highlights their advantages when modeling networks compared to traditional Neural Networks NNs.

 doi: 10.2313/NET-2026-05-1_08

## 2.1. The Functionality of DES-based Simulation

The fundamental principles of DES are described by James Gross and Mesut Güneş [5]. DES simulators are widely used for network analysis and performance evaluation, as the simulation paradigm of DES aligns well with the structural characteristics of computer networks. In DES, relevant components, such as packets, are represented as entities, each associated with a set of attributes, such as a source address, destination address and length. From these entities and their interactions, a system is then constructed. The system is then updated in discrete time steps by processing events, such as packet transmission or packet reception. After termination, the final state of the simulation can be evaluated with respect to predefined performance metrics.

Common DES-based simulators, used in both industry and academic research, include for instance OMNeT++ [2] and ns-3 [3]. Both simulators are valued for their packet-level visibility and fine-grained detail, as they model each individual packet throughout the simulation. However, this detailed per-packet simulation also limits the scalability of DES-based network simulators, making them computationally expensive for large-scale network scenarios.

Several approaches attempt to reduce the computational cost of DES-based simulation by introducing parallelization techniques (e. g. DONS [6]) or by incorporating machine learning models (e. g. DQN [7]). Although these approaches demonstrate promising results in reducing simulation time, DES-based simulation methods remain expensive, as shown by Carlos Güemes-Palau et al. [4] in their work.

## 2.2. The Simulation Gap: Idealized Models vs. Real Hardware

While DES simulation is conceptually accurate, multiple researchers have shown that performance metrics produced by DES simulators deviate from measurements obtained on real hardware testbeds [1], [4], [8], [9]. One source of inaccuracy is the limited availability of precise device specifications and implementations due to commercial interests or lack of documentation [4]. Another source stems from the inherent idealization of real network traffic, as the precise representation of real computer networks would require a highly complex simulation design. For example, simulators face difficulties when modeling the network stack or accounting for stochastic processes, such as interrupts [8].

The network simulator ns-3 mitigates some of these issues by designing the model's implementation closer to the actual software implementations they represent, for example by using C++ as a programming language, by incorporating an architecture similar to the Linux Kernel or by emphasizing emulation capabilities [3]. However, these advances do not fully eliminate the inherent challenges associated with network simulation.

## 2.3. The Concept of Graph Neural Networks

GNNs, as presented by Franco Scarselli et al. in their work [10], represent a general class of neural network architectures whose defining characteristics are their ability to operate directly on data given in the form of complex graph structures. A graph is a data structure consisting of a set of objects, referred to as nodes, and the relationships or interactions between them, encoded as edges. Nodes and edges may be associated with a label expressed as a vector of reals which can encode various features or properties of their corresponding concepts. Consequently, each node is characterized by its own feature vector and by the feature vectors of its neighboring nodes, which together define the state of the node.

GNNs impose no assumptions about the specific appearance or the ordering of the input graph and treat isomorphic graphs identically. As a result, GNNs naturally support a broad range of graph types, including cyclic, acyclic, directed and undirected graphs. GNNs' ability to generalize across diverse graph topologies stems from this invariance.

The core functionality of GNNs typically consists of two phases: message-passing and readout.

A) *Message-Passing Phase:* Message passing is an iterative procedure in which the states of two neighboring nodes, optionally combined with the label of their connecting edge, are used to compute a message vector that encodes their relationship. After the computation of all messages, an aggregation function combines the incoming messages for each node into a single vector, ensuring a fixed-size representation that is independent of the number of neighbors. Finally, the node's state is updated by combining the aggregated messages from its neighbors with its previous state.

   In summary, the message-passing mechanism propagates information iteratively over the entire graph for a fixed number of iterations or until a convergence criterion is met.

B) *Readout Phase:* During the readout phase, the updated node states are transformed into the final output of the GNN. A GNN can output node-level predictions by applying a readout function to each node individually, or global graph-level predictions by first aggregating the states of all nodes and then applying the readout function to the resulting global representation.

The message, update and readout functions are typically approximated using smaller neural network architectures, such as Multi-Layer Perceptrons (MLPs), Gated Recurrent Units (GRUs) or Recurrent Neural Networks (RNNs). [1]

As computer networks can be naturally represented as graphs, GNNs are frequently used to model them and to generate meaningful performance-related estimations, including metrics such as latency and packet loss.

## 2.4. Advantages of GNNs Compared to Traditional Neural Networks

As stated by Krzysztof Rusek et al. in their work [11], early approaches that incorporated machine learning into network modeling have primarily relied on traditional NNs, including fully-connected NNs, Convolutional Neural Networks (CNNs) [12], which operate on fixed grids

(e.g. in image processing), RNNs [12], which are designed to process sequential data (e. g. in video, text or speech recognition), as well as Variational Auto Encoders. The reason why those models exhibit lower accuracy than GNN-based models stems from the nature of the problem itself: Computer networks are naturally represented as complex graph structures with circular interdependencies. Since traditional NNs cannot operate directly graph-structured data, they fail to generalize to topologies not seen during training.

For this reason, we focus exclusively on network modeling approaches that leverage GNNs.

## 3. Different Machine Learning Approaches

After introducing the fundamental concepts of DES-based simulation and GNNs, two novel machine learning models for efficient network modelling will be presented. The focus lies on outlining and comparing the different approaches.

### 3.1. TwinNet: A Digital Twin for Network Optimization

TwinNet is a GNN-based network model introduced by Miquel Ferriol-Galmés et al. [1]. As the authors focus on network optimization, TwinNet is designed to capture complex network interrelationships, including different queueing policies, network topologies and routing configurations. TwinNet can be used to predict relevant network performance metrics, including jitter, packet loss and end-to-end path delays, meaning that TwinNet produces flow-level predictions rather than packet-level prediction. However, the authors mainly evaluated the model's ability to estimate path delays in their work, not focusing on the other performance metrics.

TwinNet incorporates the GNN architecture by representing links, queues and source-destination paths as nodes and by assigning feature vectors to each of these elements. The different components exhibit a circular dependency, as shown in Figure 1.

1) The state of each link depends on the states of all queues injecting traffic into the link.
2) The state of each queue depends on the states of all paths injecting traffic into the queue.
3) The state of each path depends on the states of all queues and links it traverses.

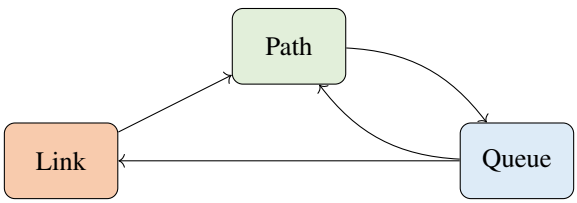



Figure 1: Network components modeled by TwinNet and their dependencies.

To then resolve the interdependencies among links, queues and paths, TwinNet extends the standard GNN framework by implementing a more complex message passing algorithm that updates each node type (link, queue, path) using messages received from the node types that it depends on.

Since the order in which a path traverses links and queues is essential for determining the path's state, RNNs are employed to aggregate link and queue messages, allowing the model to capture the sequential dependencies inherent to these elements. The queues' states are aggregated in a similar way. RNNs are a simpler class of neural network architectures that are well suited for modeling sequential or temporal domains [12].

TwinNet is trained on simulated data produced by a DES-based network simulator (OMNeT++ v5.5). The dataset consists of multiple network scenarios with randomly assigned topology, traffic, routing and queueing configurations and its corresponding per-packet delay measured on each source-destination path.

Limitations of TwinNet lie in both the simulated dataset and the lack of evaluation of jitter and packet loss.

### 3.2. RouteNet-Gauss: Hardware Enhanced Flow-Level Modeling

RouteNet-Gauss is a machine-learning-based network modeling approach introduced by Carlos Güemes-Palau et al. in [13]. It constitutes the most recent iteration of the RouteNet model family [13]–[15]. To improve computational efficiency, RouteNet-Gauss operates at flow-level granularity rather than at packet-level like DES-based simulators. However, the authors argue that the reduced granularity does not diminish the model's ability to facilitate network operation tasks such as capacity planning, Quality of Service (QoS) assurance, topology design, traffic engineering, and performance estimation.

To further address the absence of packet-level visibility, RouteNet-Gauss introduces a mechanism called Temporal Aggregated Performance Estimation (TAPE), designed to capture the temporal dependencies of flows. TAPE works by dividing the network scenario into temporal windows of configurable size, aggregating traffic information of each window and processing every window individually. RouteNet-Gauss takes a network scenario consisting of packet-level traffic information, the network topology and routing configurations as input.

The model represents the complex behavior of computer networks by modeling devices, queues, flows and links as individual entities. The components share a circular dependency, as shown in Figure 2.

1) The devices interact with the queues they are home to.
2) The queues depend on the resources provided by their networking device, the flow they process and the links they serve.
3) The flows interact with a sequence of queues and links on their path.
4) The links depend on the queues that inject traffic into the links.

Each of these elements is implemented using a dedicated NN module as a building block, enabling RouteNet-Gauss to learn the general principles regarding the interactions among network components.

RouteNet-Gauss encodes the initial state of each component using an MLP. During the message passing phase, each component type (device, queue, link, flow) is updated based on the relevant components that they depend on,

TABLE 1: Comparison of TwinNet and RouteNet-Gauss

| Work | Ref | Year | Data Source | Training Topologies | Unseen Test Topologies | Granularity | Target |
|---|---|---|---|---|---|---|---|
| TwinNet | [1] | 2022 | SIM | 2 | 106 | flow-level | jitter, packet loss, delay |
| RouteNet-Gauss | [4] | 2025 | HW | 11 | > 11 | flow-level, temporal window | jitter, packet loss, delay |

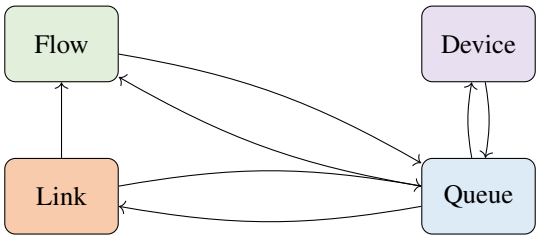



Figure 2: Network components modeled by RouteNet-Gauss and their dependencies.

using GRU cells to model their respective update functions. To incorporate the temporal aspect of RouteNet-Gauss, the message passing algorithm is executed for each time window. The final readout function, which outputs the estimated flow-level performance metrics of the network scenario per temporal window, is approximated using MLPs. The output of the model includes critical performance metrics, such as end-to-end delay, packet loss, and queue occupancy.

RouteNet-Gauss is trained on a dataset captured from a physical testbed consisting of 11 distinct topologies, ranging from five to eight nodes, in order to address the issue of inaccuracies presented in simulated data. However, as the dataset is relatively small, it could hinder RouteNet-Gauss's ability to generalize to modern large-scale networks.

A comparison of the two network models can be seen in Table 1.

## 4. Dealing with Datasets

Machine learning approaches fundamentally rely on meaningful datasets to produce accurate estimations. Accordingly, the training dataset must satisfy three essential criteria, as described in [9]:

1) *Abundance:* Training datasets must contain a sufficiently large number of samples to ensure the model's ability to generalize over scenarios not seen during training.
2) *Diversity:* Training data must be diverse, comprising a broad spectrum of network conditions including rare edge cases that may nonetheless play a significant role in real-world computer networks.
3) *Completeness:* Training data must cover all relevant factors and variables influencing network behavior.

Collecting real-world datasets is often challenging, as measured data tends to be scarce, incomplete and insufficiently diverse, since many edge cases cannot be safely captured on production networks, and acquiring data on hardware testbeds is prohibitively expensive, particularly for large-scale networks. Relying exclusively on simulated data mitigates the issue of data scarcity and can generate a large, diverse and complete dataset. However, for the reasons discussed in Section 2.2, simulated data diverges from real-world conditions, leading to inaccurate results.

This section introduces transfer learning [9] and Sim2HW [8] as potential solutions for these challenges, by benefiting from the advantages of both simulated and hardware-measured datasets.

### 4.1. Transfer Learning: From Simulation to Reality

The concept of transfer learning, as introduced by Carlos Güemes-Palau et al. in [9], is to reuse the weights of a pre-trained donor model to provide a strong starting point for initializing and training a receiver model, thereby mitigating the challenges associated with acquiring meaningful datasets. For network modeling, the authors propose a hybrid approach that combines both simulated and real-world data.

First, they trained RouteNet-Fermi [15], a predecessor of RouteNet-Gauss, solely on a large, diverse and complete simulated dataset acquired by the OMNeT++ network simulator, covering network topologies ranging from five to eight nodes. Afterwards, a significantly smaller real-world dataset was collected by extracting samples from a physical testbed.

The example model, as well as most GNN-based architectures can be divided into three main components: An encoding block, a message passing block and a readout block. This type of fragmentation is essential for the effective use of transfer learning, as it enables the specific configuration of a weight-transfer strategy. According to the authors, the most effective manual transfer learning configuration is to freeze the encoding block, fine-tune the message passing block and to retrain the readout block. Following this approach, only the encoding and message passing blocks of the donor model are transferred to the receiver model.

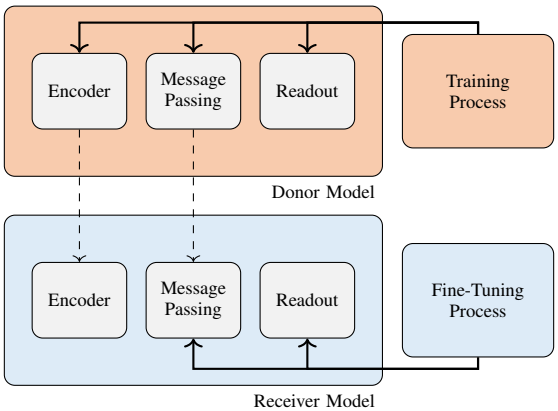



Figure 3: Illustration of the best manual transfer learning configuration for network modeling. The weights of the encoding block and message passing block are transferred from donor model to receiver model. Subsequently, the message passing and readout block of the receiver model are fine-tuned /retrained based on the received initial weights.

The receiver model must then acquire its own readout block through the standard training procedure. During

fine-tuning the receiver's encoding block remains frozen to preserve the transferred initial weights, as modifying the encoding block would invalidate the benefits of transfer learning. Fine-tuning the message passing block proceeds in a similar way to training a model from scratch, but requires substantially less data and training time. Automated transfer strategies, such as Autofreeze [16], L2-SP [17] and GTOT-Tuning [18] were also evaluated and generally outperform the best manual configuration.

In conclusion, transfer learning techniques appear to be a promising approach for improving the accuracy of machine learning based network models while keeping the costs of acquiring real-world datasets low, as transfer learning allow models to benefit from both the abundance and diversity of simulated and the fidelity of real-world datasets simultaneously.

### 4.2. Sim2HW: Correcting Simulation with GNNs

Sim2HW is a machine learning dataset optimization approach introduced by Johannes Späth et al. [8]. It addresses the problem of inaccuracies in simulated datasets by correcting simulation outputs using a GNN-based model. The primary objective of the Sim2HW model is to learn the discrepancy between simulated and hardware-measured datasets. The authors rely on a dataset obtained from a physical hardware testbed by Wiedner et al. [19]. In addition, they replicate the network scenarios present in the physical dataset using the OMNeT++ simulator. Both datasets therefore contain latency values for the same topologies and flow configurations. The flow latency values are then aggregated into percentile metrics, ranging from the minimum percentile to the percentile of 99.999. A percentile $p$ indicates that $p$ percent of the observed latency values are below the given value. For example,the 50th percentile, also referred to as the median, denotes the latency value below which 50% of the flows fall. Using percentiles as target prediction metric allows Sim2HW to capture the distribution of latency values rather than estimating just the mean latency. After converting the network topology, flow properties and latency percentiles from both datasets into a graph representation, Sim2HW's GNN model is trained on this graph. Once trained, the model can be used to infer the latency distribution that would have been obtained on a hardware testbed when provided with simulated latency samples as input.

Sim2HW is particularly effective at estimating performance values for lower percentiles, meaning that the model performs best when predicting flow latencies on lightly used links, where the latency values are small. Its accuracy decreases for high latency values, as such extreme outliers are sparsely represented in both training sets, limiting the model's ability to learn their behavior. Therefore, future iterations of Sim2HW should utilize a more diverse hardware dataset.

## 5. Conclusion

The reviewed GNN-based network models demonstrate significant potential to improve the efficiency of network modeling compared to traditional DES-based approaches. However, the effectiveness of these models remains heavily dependent on the availability of high-quality datasets. While simulated data provides abundance and diversity, it often diverges from real-world measurements, potentially limiting the model's accuracy. Techniques such as transfer learning and Sim2HW offer promising strategies to mitigate these challenges by combining the advantages of simulated datasets with the fidelity of real-world measurements.

## References


[1] M. Ferriol-Galmés, J. Suárez-Varela, J. Paillissé, X. Shi, S. Xiao, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Building a digital twin for network optimization using graph neural networks," *Computer Networks*, vol. 217, p. 109329, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622003681

[2] A. Varga, *A Practical Introduction to the OMNeT++ Simulation Framework*. Cham: Springer International Publishing, 2019, pp. 3–51. [Online]. Available: https://doi.org/10.1007/978-3-030-12842-5_1

[3] G. F. Riley and T. R. Henderson, *The ns-3 Network Simulator*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_2

[4] C. Güemes-Palau, M. Ferriol-Galmés, J. Paillisse-Vilanova, A. López-Brescó, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet-gauss: Hardware-enhanced network modeling with machine learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.08848

[5] J. Gross and M. Güneş, *Introduction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–11. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_1

[6] K. Gao, L. Chen, D. Li, V. Liu, X. Wang, R. Zhang, and L. Lu, "Dons: Fast and affordable discrete event network simulation with automatic parallelization," in *Proceedings of the ACM SIGCOMM 2023 Conference*, ser. ACM SIGCOMM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 167–181. [Online]. Available: https://doi.org/10.1145/3603269.3604844

[7] Q. Yang, X. Peng, L. Chen, L. Liu, J. Zhang, H. Xu, B. Li, and G. Zhang, "Deepqueuenet: towards scalable and generalized network performance estimation with packet-level visibility," in *Proceedings of the ACM SIGCOMM 2022 Conference*, ser. SIGCOMM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 441–457. [Online]. Available: https://doi.org/10.1145/3544216.3544248

[8] J. Späth, M. Helm, B. Jaeger, and G. Carle, "Sim2hw: Modeling latency offset between network simulations and hardware measurements," in *Proceedings of the 3rd GNNet Workshop on Graph Neural Networking Workshop*, ser. GNNet '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 20–26. [Online]. Available: https://doi.org/10.1145/3694811.3697820

[9] C. Güemes-Palau, M. Ferriol-Galmés, J. Paillisse-Vilanova, A. López-Brescó, P. Barlet-Ros, and A. Cabellos-Aparicio, "Bridging the gap between simulated and real network data using transfer learning," 2025. [Online]. Available: https://arxiv.org/abs/2510.00956

[10] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[11] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of graph neural networks for network modeling and optimization in sdn," in *Proceedings of the 2019 ACM Symposium on SDN Research*, ser. SOSR '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 140–151. [Online]. Available: https://doi.org/10.1145/3314148.3314357

[12] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," 2021. [Online]. Available: https://arxiv.org/abs/2104.13478

[13] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet: Leveraging graph neural networks for network modeling and optimization in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260–2270, 2020.

[14] M. Ferriol-Galmés, K. Rusek, J. Suárez-Varela, S. Xiao, X. Shi, X. Cheng, B. Wu, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet-erlang: A graph neural network for network performance evaluation," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 2018–2027.

[15] M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, K. Rusek, S. Xiao, X. Shi, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet-fermi: Network modeling with graph neural networks," *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, p. 3080–3095, Dec. 2023. [Online]. Available: http://dx.doi.org/10.1109/TNET.2023.3269983

[16] Y. Liu, S. Agarwal, and S. Venkataraman, "Autofreeze: Automatically freezing model blocks to accelerate fine-tuning," 2021. [Online]. Available: https://arxiv.org/abs/2102.01386

[17] X. LI, Y. Grandvalet, and F. Davoine, "Explicit inductive bias for transfer learning with convolutional networks," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 2825–2834. [Online]. Available: https://proceedings.mlr.press/v80/li18a.html

[18] J. Zhang, X. Xiao, L.-K. Huang, Y. Rong, and Y. Bian, "Fine-tuning graph neural networks via graph topology induced optimal transport," 2022. [Online]. Available: https://arxiv.org/abs/2203.10453

[19] F. Wiedner, M. Helm, S. Gallenmüller, and G. Carle, "Hvnet: Hardware-assisted virtual networking on a single physical host," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–6.

# Survey of Network Learning and Practicing Tools

Irina Maria Ciocan, Lorenz Lehle*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: irina.maria.ciocan@tum.de, lehle@net.in.tum.de

***Abstract*—Practical exercises and visualization play an important role in the learning process. In networking, they help students understand difficult and abstract concepts, and allow them to observe the behavior of protocols, configurations and devices in different scenarios.**

**This paper presents an analysis of four different tools: Cisco Packet Tracer, Graphical Network Simulator 3 (GNS3), Common Open Research Emulator (CORE) and plain orchestrating service (pos), together with its virtual clone vPos, with respect to their methodologies and three characteristics: realism, scalability and compatibility.**

**The analysis shows that no tool is universally superior, each tool presenting different trade-offs between the aforementioned characteristics.**

***Index Terms*—Cisco Packet Tracer, GNS3, CORE, pos, vPos**

## 1. Introduction

Networking courses often rely on practical exercises to help students understand how different protocols, configurations and network components behave. Since providing physical hardware for a large number of students is expensive, difficult to maintain and thus, unrealistic, many courses and certifications rely on tools that simulate or emulate the behavior of real network components [1]–[3]. As a result, the tools differ in how realistic, scalable and suitable for beginners or advanced users they are.

This paper analyzes several tools commonly used for network learning and practicing. On one hand, it considers tools such as Cisco Packet Tracer (Packet Tracer), GNS3 and CORE, which enable the creation of virtual, emulated or hybrid network environments [1], [3], [4]. On the other hand, it also examines pos, which is used to orchestrate physical hardware, and vpos, which is a virtualized version of the pos testbed [5].

The paper first introduces each tool and their respective methodologies. Subsequently, the analysis is structured according to a set of criteria relevant in an educational context, each accompanied by a brief explanation of its relevance.

## 2. Tools and their methodologies

To understand the functionality of each tool, this section first examines the objectives and use cases they were designed to address.

### 2.1. Cisco Packet Tracer

**Packet Tracer** is the network learning and practicing tool used in the Cisco Certified Network Associate (CCNA) certification. It is used throughout the course to teach topics such as network configuration, routing and the fundamentals of security, automation and programmability. [1], [2], [6], [7]

Packet Tracer uses visualization based learning to teach the basic concepts of network design and configuration, troubleshooting and protocol behavior. The workflow of Packet Tracer is based on the graphical construction of topologies, where users place virtual network devices and configure them [6]. Its GUI is shown in Figure 1a.

In a series of experiments conducted by Vijayalakshmi *et al.*, Packet Tracer was used to teach abstract concepts through visualization. As a result, students were able to better understand protocols such as ICMP, TCP, UDP, and application layer services like DNS, DHCP, HTTP, FTP, and E-mail. [2]

Additionally, Packet Tracer includes the *Activity Wizard* feature, which allows teachers to define a final, target network and an initial configuration which is provided to students. This gives teachers the possibility to automatically assess students' work. [1]

### 2.2. GNS3

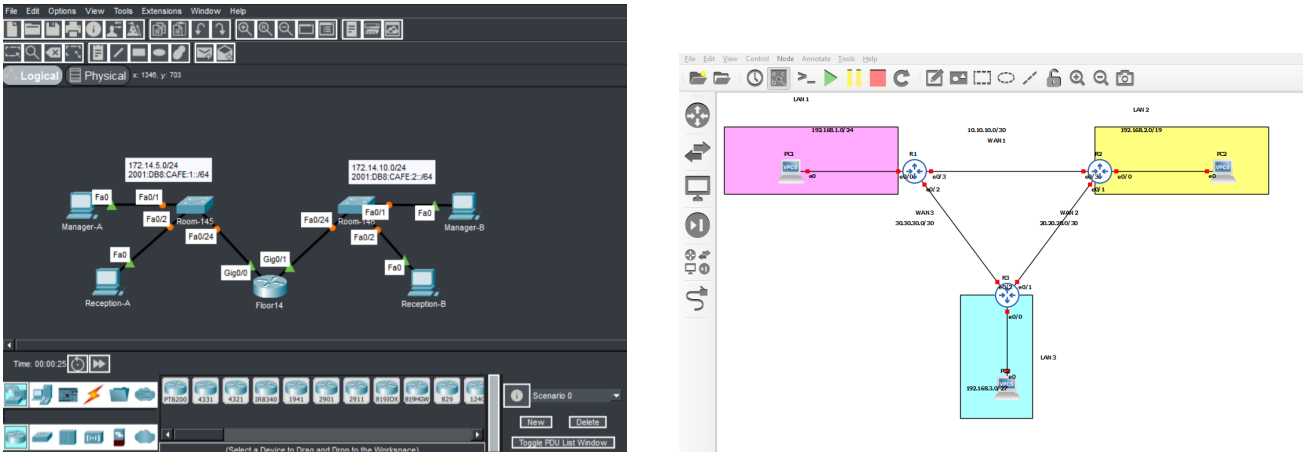
**GNS3** is used to emulate, configure, test and troubleshoot virtual and real networks. GNS3 allows users to run topologies ranging from only a few devices on a laptop, to many devices hosted on multiple servers or even hosted in the cloud. [3], [8]–[10]

The methodology of GNS3 is based on the virtualization of real networks, allowing students to work with emulated network devices and real operating systems in a controlled environment. Similar to Packet Tracer, GNS3 has a visualization oriented workflow in which students draw topologies, as shown in Figure 1b. Students gain knowledge through creating topologies, testing their performance and analyzing their behavior. [8], [9]

### 2.3. CORE

**CORE** is a tool for building virtual networks. It is an efficient and scalable emulator that runs applications and protocols without modification. CORE has many applications, such as network and protocol research, testing, evaluating networking scenarios and increasing the size of physical test networks. [4], [11]

 doi: 10.2313/NET-2026-05-1_09



(a) Cisco Packet Tracer GUI

(b) GNS3 GUI

Figure 1: Side by side view of the graphical user interfaces of Cisco Packet Tracer (left) and GNS3 (right).

CORE is based on interactive experimentation. Similar to the workflows of Packet Tracer and GNS3, the CORE workflow starts with drawing nodes, network devices and links between them. After the user configures and instantiates the topology, the session enters the runtime phase. When the session is stopped, the user has the opportunity to save the log files and node data collected during the runtime phase. [4], [11]

While the graphical workflow is similar to Packet Tracer and GNS3, the underlying methodology differs. CORE is used for experimentation and protocol evaluation rather than visualization based learning. [4], [11]

### 2.4. pos

**pos** is a framework developed at the Chair of Network Architectures and Services, which focuses on incentivizing reproducible network experiments. Reproducibility is defined as the property that allows different people to get the same results on the same setup. The developers' aim is to create a tool that reduces the amount of work required for conducting reproducible experiments. [5]

**vpos** is a virtualized version of the pos framework that replicates the methodology of pos in a virtual environment. It enables the execution of pos experiments without requiring access to the physical testbed. [5]

The methodology of pos and vpos is based on an automated experimental workflow, which is divided into a setup, an experiment and an evaluation phase. [5]

During the setup phase, devices are allocated and booted via live images, to ensure a clean, identical state each time the experiment is executed. The configuration is then done via scripts only. [5]

Depending on which host has access to them and where they are used in the workflow, pos distinguishes between global variables (i.e., accessible from all experiment hosts), local variables (i.e., defined for each experiment host) and loop variables (i.e., shared across all experiment hosts, but continuously changed between different measurement runs). During the measurement phase, the script is executed once for each possible combination of loop variables to ensure full coverage. After the completion of all runs, the collected data and outputs are stored in the result folder. [5]

During the evaluation phase, results are parsed and evaluated using scripts, with the metadata being attached automatically. pos can also generate plots and figures based on the collected data and it has the ability to automatically transform data into a publishable format (e.g. a repository or a website containing all collected information). [5]

This strict experimental methodology enforces reproducibility by design. [5]

## 3. Characteristics

In order to meaningfully compare the presented tools, this section analyzes them based on realism, scalability and compatibility, which are relevant for network learning and practicing.

### 3.1. Realism

Realism measures how accurately a certain tool replicates the behavior of a real network. It is an important trait to consider because it determines how close students can observe the aspects of real networks.

**Packet Tracer** uses software simulation to replicate the behavior of real world network components like routers and switches. Simulation offers many advantages, for example, it allows users to create large-scale topologies that would be difficult or impossible to create with physical hardware or with an emulator. It also facilitates remote access for both students and teachers and it allows students to make mistakes without breaking physical hardware. [1], [2], [6]

However, simulation also has disadvantages. Packet Tracer has limitations caused by the simplified protocol implementation. Some protocols and functions can present different errors, due to the fact that Packet Tracer's simulated devices cannot reproduce exactly how real devices behave. Some advanced features that are present in real devices are missing in Packet Tracer, which causes errors when using protocols like ICMPv6 and DHCPv6. Packet

Tracer is also unable to simulate load and congestion or to measure throughput and latency. [1], [9]

**GNS3** follows a different approach by relying on emulation rather than simplified simulation. It supports all features available on physical routers and can emulate services like httpd, ftpd, telnetd and sshd. This allows it to produce traffic as if it were generated on a real network and to replicate the configuration and behavior of real networks more closely than Packet Tracer. [3], [9], [10]

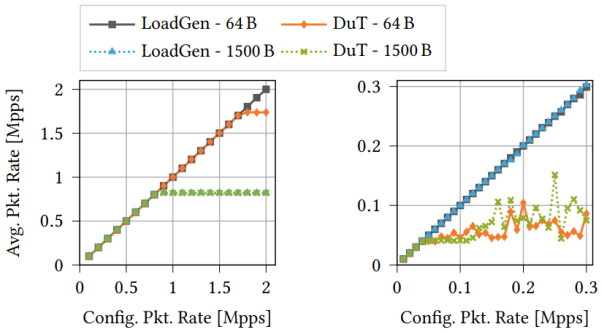
In a study conducted by Gil *et al.*, student participants of the survey stated that GNS3 proved itself to be especially useful in helping them understand topics about communications and TCP/IP. When asked about the similarity to the real laboratory, the students gave it a score of 3/5, while the lecturer estimated the similarity to be up to 90%. [9] Similarly, the effectiveness of Packet Tracer as a learning tool has also been demonstrated in educational studies. Vijayalakshmi *et al.* observed that the use of Packet Tracer in networking exercises improved students' understanding of fundamental concepts, increasing the average quiz score from 65% to 85%. [12]

**CORE** achieves a high level of realism by offering a high fidelity emulation for layers 3 through 7 and a simplified simulation of layers 1 and 2. Similarly to GNS3, CORE can connect a live running emulation to physical hardware with the help of the RJ45 node and the Tunnel tool, allowing users to test protocols and applications in a hybrid environment. Emulated nodes can send and receive traffic from physical hardware, which allows them to interact in real time. [4], [11]

The RJ45 node represents an interface that connects the emulated network to a physical external device through a host network port. The limitations that this approach brings are the host's requirement for a physical port for each external device and the requirement that the host and the external devices be colocated. [11]

The Tunnel tool allows the connection of multiple emulations as well as the connection of an emulation to a physical device. Unlike RJ45, this tool does not require the dedication of a host's port for the connection and allows devices and CORE machines to be connected remotely. [11]

**pos** initializes physical devices with live-boot images and uses direct wiring between hosts to prevent unwanted influence of switches and hubs, which isolates the network from non-investigated devices that could interfere with the experiment. The live images force the OS to repeatedly start from a well-defined state. Thus, pos reflects true hardware behavior. [5]




**(a) Hardware testbed (pos)** **(b) Virtual testbed (vpos)**

Figure 2: Performance comparison between pos and vpos. [5]

In a case study conducted by Gallenmüller *et al.*, the virtual clone of pos, vpos, showed significant performance differences, as shown in Figure 2. For the hardware testbed the forwarded packet rate increases linearly along the generated packet rate until it reaches the system's limits. In contrast, for the virtual testbed the forwarding rate already becomes unstable at relatively low generated rates, resulting in packet loss. This highlights the importance of realism when analyzing network behavior. [5]

However, teaching environments do not require the same level of measurement accuracy as scientific experiments. Therefore, the realism offered by vpos is sufficient for educational purposes.

## 3.2. Scalability and performance

In an educational context, scalability refers to either a tool's ability to handle topologies of different sizes or its ability to support multiple users. Considering scalability is relevant for assessing whether a tool is suitable for a class where collaborative work is required and whether it can effectively substitute hardware based networking laboratories.

As mentioned in Section 3.1, given that **Packet Tracer** is a simulation tool, it can be used to design large and complex topologies, which would be difficult or expensive to implement using physical hardware [2].

The tool is also equipped with a Multiuser feature, which facilitates the creation of a collaborative environment. In a test conducted by Andrew Smith and Colin Bluck, they showed that the Multiuser feature allows a shared topology to be accessed and controlled remotely by more than 40 users, up to 20 being active simultaneously. [7]

Because it can be ran on a single physical computer, **GNS3** maintains the accessibility provided by Packet Tracer, while also creating a more realistic virtual environment. Virtualization also allows for devices to be easily duplicated or moved across the topology. [8], [9]

However, GNS3 emulates real Cisco IOS images, which requires significantly more CPU and memory resources. As a result, its performance is limited by the capabilities of the host system, leading to lower scalability compared to Packet Tracer, while still offering greater scalability than laboratories based on physical hardware. [3], [8]–[10]

The performance of **CORE** depends on several different factors: hardware, OS version, active processes, network traffic and GUI usage. According to the documentation, it is more meaningful to research how much traffic CORE can handle, rather than how many nodes it can emulate, which highly depends on the workload of each node. [11]

On a single-CPU Xeon 3.0GHz, 2GB RAM server running Linux, CORE was able to instantiate more than 100 nodes. CORE nodes share the host's kernel and file system, and require a small amount of system resources, which leads to a high scalability. [11]

TABLE 1: Characteristics comparison between tools

| | **Packet Tracer** | **GNS3** | **CORE** | **pos** |
|---|---|---|---|---|
| **software** | ✓ | ✓ | ✓ | vPos |
| simulation | ✓ | | | |
| emulation | | ✓ | ✓ | |
| **hardware** | | | ✓ | ✓ |
| **realism** | low | moderate | high | very high |
| **scalability** | high | moderate | moderate | low |
| **compatibility** | low | medium | high | high |

On the same server, it was able to handle a traffic of approximately 300.000 packets per second. This number indicates how often the system handled sending or receiving packets. The size of the packets and the number of hops that are emulated do not limit the performance of CORE. [11]

Additionally, CORE allows multiple emulations on different servers to be connected. A large emulation scenario divided across multiple devices can be controlled by the same GUI that can be run either on one of the emulation servers or on a dedicated device. [11]

When using **pos**, topologies have to be physically wired. This prevents them from being automatically created or recreated, and leads to a low topology scalability. [5]

When a multiuser testbed is operated, an integrated calendar system schedules access to experimental nodes and ensures that resources are reserved for the entire duration of an experiment. This enables multiple users to conduct independent experiments in parallel on different nodes while preventing simultaneous use of the same node. [5]

Unlike pos, vpos does not require nodes to be physically wired. As discussed in Section 3.1, vpos offers more scalability through virtualization, although it comes at the cost of reduced reliability compared to pos. [5]

As shown in Figure 2, the maximum forwarding performance of pos reaches approximately 1.750.000 pps for 64 B packets and approximately 800,000 pps for 1500 B packets, both exceeding the 300.000 packets per second achieved by CORE [5], [11]. vpos shows the lowest throughput, reaching approximately 40.000 pps before packet loss occurs, regardless of packet size [5].

### 3.3. Compatibility and flexibility

When designing a network and selecting a network tool, compatibility and flexibility are essential, as they determine which devices and technologies a tool supports.

**Packet Tracer** can simulate multiple routers, switches and wireless devices, but is limited to devices that are part of the Cisco ecosystem. An advantage of this approach is that Packet Tracer does not require users to upload real device images or operating systems. However, this approach brings the disadvantage of reduced device diversity, limiting the range of models a user can introduce in the topology. [1], [6], [13]

One major advantage that **GNS3** holds over Packet Tracer when it comes to compatibility is that it supports the emulation of any Quick Emulator (QEMU) based devices, not only Cisco devices. GNS3 also being compatible with VMware allows virtual machines that operate either Windows or Linux to be integrated into GNS3 networks. [8], [10]

In an experiment conducted by Mohtasin *et al.*, it was concluded that a network realized with GNS3 can also be connected to physical hardware. This was possible both by creating a Loopback network adapter on the physical computer and by connecting it via an Ethernet cable to the machine running the GNS3 network. [8]

**CORE** features three types of nodes: CORE nodes, Docker nodes and Physical nodes [11]. CORE nodes can function as virtual hosts, routers, switches and wireless devices. They originally used the FreeBSD network stack virtualization and are currently backed by Linux network namespaces [4], [11]. Docker nodes allow users to configure nodes using predefined images and filesystems that CORE nodes do not provide [11]. Physical nodes, as discussed in section 3.1, can be connected locally via RJ45 nodes or remotely via Generic Routing Encapsulation (GRE) tunnels [11]. CORE emulates network devices by using Linux namespaces, each having an independent process environment and a private network stack [11]. Compared to Packet Tracer, which relies on predefined device models, CORE offers greater flexibility in configuring nodes and integrating additional software.

**pos** utilizes live-boot images for experiment nodes to ensure that every experiment begins from a consistent initial state. Despite this standardized starting point, the testbed gives users freedom to design and execute experiments using scripts and root access on experiment nodes. [5]

One of the objectives accomplished by pos, referred to as heterogeneity, is the development of a testbed adaptable to different devices. In order to achieve heterogeneity, pos implements two APIs: an initialization and a configuration interface. The initialization interface is used to reset and boot devices, for example via IPMI, while the configuration interface is used to configure devices and execute experiments, typically via SSH. To support devices that rely on APIs other than those already implemented, pos allows the implementation of additional initialization and configuration interfaces. [5]

## 4. Conclusion and future work

This paper shows that there is no single "best" tool for network learning and practicing. Instead, each user has to choose a tool based on the intended use case, the learning objective and their level of expertise. Each of the analyzed tools make different trade-offs between realism, scalability and compatibility, as shown in Table 1.

For beginners, tools like Packet Tracer and GNS3 are more appropriate, as they can be used to learn the fundamentals of networking. CORE and pos are more suitable for advanced users, since they require deeper knowledge and understanding of networking concepts.

The analyzed tools differ significantly in terms of realism. Packet Tracer relies on simulation, which leads to low realism. GNS3 and CORE reach higher levels of realism through emulation. However, none of the three

 doi: 10.2313/NET-2026-05-1_09

tools reach the level of realism provided by pos, as it operates directly on physical hardware.

When the goal is to model large-scale topologies, Packet Tracer and GNS3 provide better scalability than CORE or pos, and are better suited for scenarios in which limited hardware is available. CORE's scalability is limited due to its reliance on the host's resources, while pos offers the lowest scalability, as it depends on the availability of physical hardware. pos prioritizes realism and reproducibility over scalability and accessibility, with scalability being addressed through vPos.

In terms of compatibility, Packet Tracer is the most limited, as it is restricted to devices belonging to Cisco's ecosystem. GNS3 and CORE can integrate a broader spectrum of devices, while pos provides the highest level of compatibility, through its focus on heterogeneity.

Packet Tracer focuses on accessibility rather than realism, making it well suited for introductory courses. GNS3 achieves a higher level of realism while maintaining accessibility, allowing users to move towards more complex and realistic scenarios. CORE offers a strong balance between realism, scalability and compatibility, as it supports a wide range of software and can integrate external devices.

## References


[1] J. Uramová, P. Segeč, and M. Kontšek, "Best practise for creating Packet Tracer activities for distance learning and assessment of practical skills," *2019 International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 2019.

[2] M. Vijayalakshmi, P. Desai, and M. M. Raikar, "Packet Tracer Simulation Tool as Pedagogy to Enhance Learning of Computer Network Concepts," *2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education (MITE)*, 2016.

[3] S. Liu, J. Liu, H. Wang, and M. Xian, "Feasibility analysis of network security teaching platform based on KVM and GNS3," *2019 International Conference on Information Technology and Computer Application (ITCA)*, 2019.

[4] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A real-time network emulator," *2008 IEEE Military Communications Conference (MILCOM)*, 2008.

[5] S. Gallenmüller, J. Naab, I. Adam, and G. Carle, "The pos framework: A methodology and toolchain for reproducible network experiments," *Proceedings of the 17th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '21)*, 2021.

[6] J. Janitor, F. Jakab, and K. Kniewald, "Visual Learning Tools for Teaching/Learning Computer Networks: Cisco Networking Academy and Packet Tracer," *2010 Sixth International Conference on Networking and Services (ICNS)*, 2010.

[7] A. Smith and C. Bluck, "Multiuser Collaborative Practical Learning Using Packet Tracer," *2010 Sixth International Conference on Networking and Services*, 2010.

[8] R. Mohtasin, P. Prasad, A. Alsadoon, G. Zajko, A. Elchouemi, and A. K. Singh, "Development of a Virtualized Networking Lab using GNS3 and VMware Workstation," *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2016.

[9] P. Gil, G. J. Garcia, A. Delgado, R. M. Medina, A. Calderon, and P. Marti, "Computer Networks Virtualization with GNS3: Evaluating a Solution to Optimize Resources and Achieve Distance Learning," *2015 International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2015.

[10] "Gns3 documentation," https://docs.gns3.com/docs/, last accessed: 14.12.2025.

[11] "Core documentation," https://coreemu.github.io/core/index.html, last accessed: 14.12.2025.

[12] M. M. R. Vijayalakshmi M., Padmashree Desai, "Packet Tracer Simulation Tool as Pedagogy to Enhance Learning of Computer Network Concepts ," *2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education*, 2016.

[13] A. Sharma, A. Kumar, and M. Kumar, "Comparison of Packet Tracer and EVE-NG Tools for Efficient Network Design," *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2024.

# MASQUE NADA: QUIC-based proxying with `tokio-quiche`

Alexander Fink, Daniel Petri*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: alexander.fink@tum.de, petriroc@net.in.tum.de

***Abstract*—Multiplexed Application Substrate over QUIC Encryption (MASQUE) is a recent approach to proxying that leverages the QUIC transport protocol and HTTP/3 to tunnel traffic. In this paper, we design and implement a proof-of-concept MASQUE proxy and client, NADA, using Cloudflare's tokio_quiche library in Rust. The prototype establishes UDP tunnels via CONNECT-UDP and forwards datagrams between clients and target hosts through the proxy. We further incorporate the Capsule Protocol to enable reliable data transport between the client and proxy and discuss the implementation constraints regarding payload-size/MTU limits introduced by QUIC and HTTP Datagram encapsulation. We demonstrate that one can easily set up a MASQUE proxy with a moderate amount of code in Rust.**

***Index Terms*—MASQUE, QUIC, HTTP/3, CONNECT-UDP, HTTP Datagrams, Capsule Protocol, tokio-quiche, Rust**

## 1. Introduction

Traditional HTTP Proxies use the HTTP `CONNECT` method to open TCP tunnels to target hosts [1], [2]. While this is suitable for TCP-based protocols, until recently there was no convenient general-purpose way to proxy UDP-based protocols (e.g., QUIC) over an HTTP proxy, aside from protocol-specific workarounds.

MASQUE (Multiplexed Application Substrate over QUIC Encryption) is a recently standardized set of protocols that tackle this shortcoming by enabling the tunneling of non-TCP-based protocols over HTTP [3]. As a relatively new standard, support remains limited and implementations are still emerging. In this work, we present a proof-of-concept MASQUE proxy and client implementation for UDP in Rust using Cloudflare's `tokio-quiche` library. While MASQUE's proxying capabilities have been extended to support the tunneling of arbitrary IP and Ethernet packets, our implementation focuses on UDP tunneling [4], [5].

## 2. Tunneling UDP with MASQUE

**Scenario**: Figure 1 shows the MASQUE proxy scenario used throughout this work. A MASQUE client establishes an HTTP/3 connection over QUIC to a MASQUE proxy (outer connection). The client then requests a UDP tunnel to a target host and port using `CONNECT-UDP`. After the proxy accepts the request, it forwards the tunnel payloads as plain UDP to the target server and vice versa.

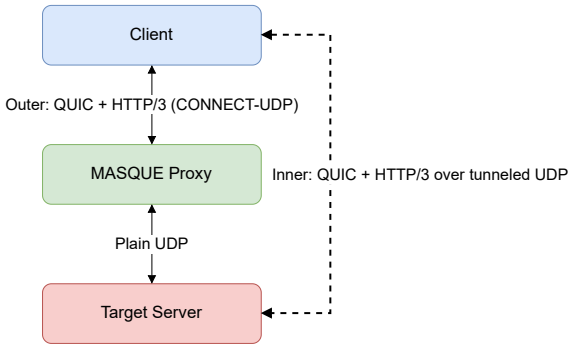



Figure 1: MASQUE proxy scenario used in this work.

`CONNECT-UDP` can carry arbitrary UDP payloads; it does not require QUIC [3]. In our experiments, we tunnel QUIC/HTTP/3 inside the `CONNECT-UDP` tunnel (inner connection) to simplify client-side testing.

At a high level, tunneling UDP with MASQUE works as follows [3]:

1) The client first establishes a secure HTTP/3 connection to the MASQUE proxy.
2) The client sends an extended `CONNECT` request with `:protocol` set to `connect-udp` to instruct the proxy to open a UDP tunnel to a specified target host and port.
3) If the proxy accepts the request, it opens a UDP socket to the target server and sends back a success response, effectively establishing the tunnel.

Once the tunnel is set up, the client and proxy exchange data: the client sends packets to the proxy and the proxy forwards the UDP payloads to the target server. Likewise, UDP responses received by the proxy are relayed back to the client over the outer HTTP/3 connection.

**HTTP Datagrams and the Capsule Protocol**: Once the UDP tunnel is established, MASQUE defines two ways to carry the UDP packet data between client and proxy: HTTP Datagrams and Capsules [3], [6]. When using HTTP Datagrams, MASQUE leverages QUIC `DATAGRAM` frames to transmit the HTTP Datagrams. The HTTP Datagram header itself is rather small, as it only contains the Quarter Stream ID to identify the HTTP stream that the datagram belongs to and is followed by the UDP payload [6]. These HTTP Datagrams are unreliably delivered and unordered, similar to UDP itself. Thus, reliable transmission, when required, is left to the internal connection. Hence, if any datagram is lost in transit be-

 doi: 10.2313/NET-2026-05-1_10

tween client and proxy, the outer QUIC connection will not take care of retransmitting it.

The proxy, upon receiving a QUIC DATAGRAM frame containing an HTTP Datagram from the client, extracts the UDP payload and forwards it to the corresponding target over UDP. Similarly, incoming UDP packets from the target are sent back to the client in HTTP Datagrams.

For certain applications, e.g., when there is strong packet loss between client and proxy, the unreliable behavior is undesired. In those cases, the Capsule Protocol mode is better suited for exchanging the UDP payloads with the server [6].

Unlike pure HTTP Datagrams, capsules are sent over the reliable HTTP stream; hence, they are delivered in order and reliably between client and proxy. The structure of the capsule header is rather simple: The header contains information about the content-type, a length field, and is followed by the payload. For tunneling UDP payloads, there is a special DATAGRAM capsule content-type defined that expects HTTP Datagrams as payload [6]. In our MASQUE proxy scenario (Figure 1), when switching to capsule mode, all exchanges between client and proxy happen through capsules. The server extracts the HTTP Datagram from inside and then proceeds with it as in datagram mode. The responses from the target server are packed into DATAGRAM capsules before sending them back to the client [6].

Both endpoints discover at connection initialization whether the other party supports HTTP/3 datagrams via the QUIC SETTINGS frame at the beginning of the connection, where the SETTINGS_H3_DATAGRAM setting is set to 1 in case it is supported [6], [7]. This is only a hint that HTTP Datagrams are supported, but it is not mandatory for client and proxy to use it. In fact, the Capsule Protocol is always active as soon as the CONNECT-UDP request gets acknowledged by the server [3]. However, it is up to the implementation to decide not to send any packets through capsules; thus, from an external observer's perspective, it looks like the Capsule Protocol is not used.

One trade-off with capsule-based tunneling is reintroducing ordering and head-of-line blocking for that flow [8]. Since all capsules for a given tunnel flow in one HTTP/3 stream, if a packet is lost and retransmitted, subsequent data for the same tunnel must wait. In contrast, in datagram-only mode, each datagram is independent. Loss of one datagram does not delay delivery of later ones; they can arrive out of order. Thus, capsule mode is appropriate when reliability is required, but it may incur latency penalties if loss occurs.

A caveat of MASQUE UDP proxying is that proxies must respect the path MTU of the outer QUIC connection. QUIC DATAGRAM frames cannot be fragmented at the QUIC layer, and QUIC endpoints are required to avoid IP-layer fragmentation [8], [9]. Not only can the MTU on the path between client and proxy be smaller than on the path between proxy and target, but the effective maximum size of inner datagrams is additionally limited by the QUIC configuration of client and proxy. When QUIC DATAGRAM frames are used, the max_datagram_frame_size transport parameter also limits the maximum DATAGRAM frame size, and this limit can be further restricted by max_udp_payload_size [9]. On top of these limits, one has to take into account that the maximum size of inner datagrams on the path between client and proxy is further reduced by the overhead of packing the UDP payload into HTTP Datagrams and QUIC DATAGRAM frames [6], [10].

In our implementation (Section 3), we build a MASQUE proxy and client on top of tokio-quiche that provides a QUIC/HTTP/3 stack in Rust. This reduces boilerplate code and lets us focus on MASQUE-specific logic rather than the details of I/O integration.

## 3. MASQUE NADA Implementation

As mentioned in Section 2, our MASQUE implementation consists of the MASQUE client and the MASQUE proxy. Both components use the tokio-quiche library, an open-source library that builds on top of quiche and provides a high-level, asynchronous interface that integrates Tokio [11].

### 3.1. QUIC / HTTP/3 Stack

Tokio is an asynchronous runtime for Rust that provides an event-driven platform to manage network connections and tasks efficiently [12]. It allows writing networking code using async/await, with the runtime handling the low-level details of scheduling and I/O polling.

The underlying quiche library exposes a relatively low-level API: It requires explicitly managing I/O integration like receiving and sending datagrams and forwarding the payload to quiche [11]. Implementing a MASQUE proxy directly on top of quiche would therefore involve substantial boilerplate and additional overhead, thus we choose tokio-quiche. By delegating low-level UDP socket management and QUIC handshake/events to tokio-quiche, our implementation can focus on HTTP/3 request handling and MASQUE-specific logic.

### 3.2. Tunnel initialization and HTTP Datagrams

As illustrated in Figure 2, in our implementation a user executes the MASQUE client as a CLI application, which establishes an HTTP/3 connection to the MASQUE proxy. The client then issues a CONNECT-UDP request to create a UDP tunnel to a target server. In more detail:

1. **Connection Setup**: The client opens a QUIC connection to the MASQUE proxy [3]. Both client and proxy use the tokio-quiche library, which handles the QUIC handshake and sets up an HTTP/3 session over QUIC. The MASQUE client container opens a QUIC connection to the MASQUE proxy container (Figure 2). Both sides use tokio-quiche to perform the QUIC handshake and establish an HTTP/3 session. This greatly simplifies initialization, because tokio-quiche handles UDP socket I/O and HTTP/3 session setup [11]. Once the connection is established, HTTP/3 streams can be used for sending requests and responses.

2. **Extended CONNECT Request**: Once the HTTP/3 connection is established, the client initiates a MASQUE tunneling request by sending an extended CONNECT request with :protocol = connect-udp. The destination of the UDP tunnel (target IP and port) is encoded in the request :path. In addition, the capsule-protocol = ?1 header

 doi: 10.2313/NET-2026-05-1_10

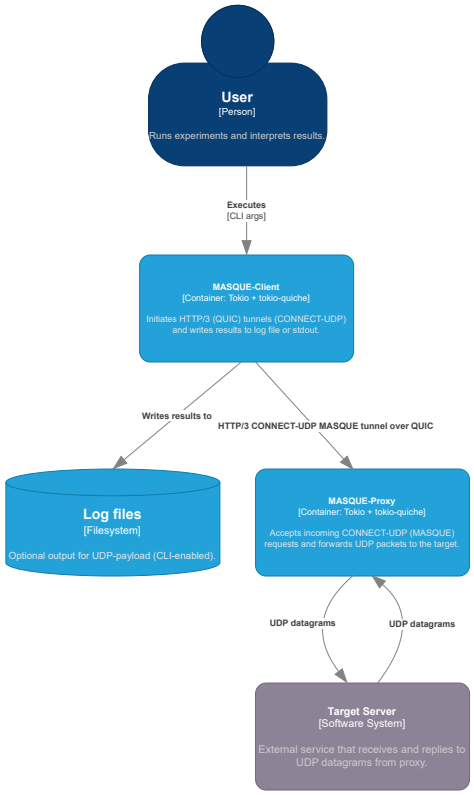



Figure 2: C4 Container Diagram of the MASQUE Rust implementation.

indicates that the client supports exchanging UDP payloads as `DATAGRAM` capsules on the `CONNECT` stream (we send `?1` whenever the header is present).

For example, to reach a UDP endpoint at *target-ip* on *port*, the client sends a request of the following form:

```
:method    = CONNECT
:protocol  = connect-udp
:authority = masque.technology
:scheme    = https
:path      = /.well-known/masque/udp/target-ip/port
capsule-protocol = ?1
```

As outlined in Section 2, after the proxy accepts the `CONNECT-UDP` request, the tunnel is established and the proxy forwards traffic in both directions, carrying client datagrams to the target UDP endpoint and returning UDP packets from the target through the HTTP/3 tunnel.

The MASQUE Client may optionally store the received packet payload to disk. In our implementation, this corresponds to writing output to the "Log files" container (a filesystem) shown in Figure 2.

**QUIC UDP Payload Limits**: We first implemented `CONNECT-UDP` with HTTP Datagrams and tested it with Cloudflare's `async_http3_server` [13]. Testing it with the example server failed, while the same client appeared to work when tunneling to public targets (e.g. google.com). This discrepancy suggested the implementation was not fundamentally broken. We therefore investigated the effective packet budget of the outer QUIC connection and confirmed the issue was caused by faulty QUIC configuration: the inner UDP payload plus HTTP Datagram and QUIC overhead exceeded `tokio-quiche`'s default `max_send_udp_payload_size = 1200 bytes` setting [13]. In the Cloudflare example the `max_send_udp_payload_size` is set to 1350 bytes [13]. This mismatch led to dropped packets on the proxy, as datagrams are not allowed to be fragmented and the received packets from the target server were bigger than what the proxy can send. To solve this situation, we can either increase `max_send_udp_payload_size` on the proxy or decrease the `max_recv_udp_payload_size` QUIC setting in the inner connection between client and target server, as by default it is 65527 bytes. The `max_recv_udp_payload_size` setting value is advertised by the client during connection initialization to the target server, signaling that the client will not process packets with larger payloads [8]. While servers are not strictly required to follow this advertised setting value, most implementations do. That gives us the capability to limit the sender's maximum UDP payload size from the client. After setting the relevant QUIC parameters accordingly, the client–proxy tunnel worked for larger payloads within the configured boundaries. Note that the `max_send_udp_payload_size` setting itself is never advertised by the proxy to the client or target server. We therefore recommend setting `max_send_udp_payload_size` and `max_recv_udp_payload_size` to the same value. The advantage here is that the client can take the payload size setting from the outer connection, decrease it by potential MASQUE overhead and use that value to configure the inner connection's `max_recv_udp_payload_size`.

In our implementation, we configure QUIC's `max_udp_payload_size` transport parameter to 1300 bytes, which is a cap on the size of inner datagrams so that, after adding HTTP Datagram encapsulation and QUIC packet headers, the resulting QUIC packet still fits onto the path. We follow Cloudflare's recommendation to cap `max_udp_payload_size` at 1300 bytes to reduce the risk of exceeding typical path MTUs once encapsulation overhead is included [10]. For typical paths with an MTU of 1500 bytes, they argue that subtracting IP/UDP headers and QUIC `DATAGRAM` frame + HTTP Datagram overhead leaves roughly 1200–1300 bytes for tunneled payloads and that using the fixed cap is a good practice in order to avoid datagrams that are too large for the proxy-client flow while still allowing QUIC clients to send the required initial 1200-byte QUIC packets [8].

### 3.3. Reliable Transport with the Capsule Protocol

After resolving the payload-size limitations observed with QUIC `DATAGRAM` and HTTP Datagram encapsulation, we added Capsule Protocol support to our implementation to enable reliable delivery between client and proxy. As stated above, it is left up to the implementation whether to use HTTP Datagrams or the Capsule Protocol in case both are supported by the implementation, thus we allow choosing between datagram and capsule mode via a command-line interface flag for the client.

When the Capsule Protocol becomes active, the communication changes as follows: Instead of sending UDP payloads in QUIC DATAGRAM frames, the client sends them as HTTP/3 Data frames on the CONNECT request stream, but wrapped in a DATAGRAM capsule, a structure defined by the Capsule Protocol [3], [6]. The proxy reads DATAGRAM capsules from the CONNECT stream, extracts the enclosed UDP payload, and forwards it to the target over UDP. UDP responses from the target are encapsulated into DATAGRAM capsules and written back to the client on the same stream.

Our initial implementation followed the HTTP Datagram approach: We implemented a simple encoder/decoder to wrap each UDP packet with the capsule header (as defined in RFC 9297) [6]. Each outgoing UDP packet was encoded in one DATAGRAM capsule and incoming data was decoded as one capsule. This assumption turned out to be incorrect because capsules are carried over HTTP/3 streams [7]. Unlike QUIC DATAGRAM frames, which preserve message boundaries, HTTP/3 streams are stream-oriented. Thus, reads from them do not preserve message boundaries and can return split or combined data. For example, capsule headers (type/length varints) can span multiple reads, and multiple capsules can be returned back-to-back in a single read. Consequently, our stateless decoder failed for certain payloads.

To correctly handle the capsule mode, we implemented a capsule parser as a state machine. Incoming stream chunks are appended to a buffer, and parsing proceeds in three stages:

1) Decode the capsule type once enough bytes are available
2) Decode the capsule length once enough bytes are available
3) Wait until the buffer contains length bytes to extract the complete capsule payload

Upon full reconstruction of DATAGRAM capsules, we extract their enclosed UDP payload. The server then forwards the payload via UDP to the target server like in HTTP Datagram mode. The client processes received DATAGRAM capsules analogously. The client then extracts the enclosed UDP payload from completed DATAGRAM capsules and processes it exactly as in datagram mode. Any remaining bytes stay in the buffer for the next parsing iteration of the capsule parser. This design makes the implementation independent of how HTTP/3 streams chunk data into QUIC stream frames.

## 4. Conclusion and future work

We implemented a proof-of-concept MASQUE client and proxy in Rust using Cloudflare's tokio-quiche, demonstrating that a functional MASQUE setup can be realized with a reasonably small amount of MASQUE-specific code.

Our prototype supports MASQUE's CONNECT-UDP mechanism to establish a UDP tunnel to a target host and forward traffic between client and target via the proxy. As a result of that, our client is able to build inner HTTP/3 connections to the target through the proxy. We implemented both data transport options relevant to MASQUE UDP tunneling: unreliable delivery using HTTP Datagrams over QUIC DATAGRAM frames, and reliable delivery using the Capsule Protocol by sending DATAGRAM capsules over the CONNECT stream. In addition, we emphasized constraints that matter in interacting with arbitrary target servers, in particular MTU and payload-size limitations imposed by QUIC and HTTP Datagram encapsulation. MASQUE NADA is publicly available [14] for further experimentation and research purposes.

Future improvements for our proof-of-concept could include adding support for proxying arbitrary IP packets using the CONNECT-IP method, implementing proxy authentication and client authorization as well as testing interoperability with other MASQUE implementations.

## References


[1] R. T. Fielding, M. Nottingham, and J. Reschke, "HTTP Semantics," RFC 9110, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9110

[2] J. Wignall. (2025, May) Beneath the masque — a dive into network relay technology on apple platforms. Blog post. [Online]. Available: https://jedda.me/beneath-the-masque-network-relay-on-apple-platforms/

[3] D. Schinazi, "Proxying UDP in HTTP," RFC 9298, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9298

[4] T. Pauly, D. Schinazi, A. Chernyakhovsky, M. Kühlewind, and M. Westerlund, "Proxying IP in HTTP," RFC 9484, Oct. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9484

[5] A. Sedeño, "Proxying Ethernet in HTTP," Internet Engineering Task Force, Internet-Draft draft-ietf-masque-connect-ethernet-08, Oct. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-masque-connect-ethernet/08/

[6] D. Schinazi and L. Pardue, "HTTP Datagrams and the Capsule Protocol," RFC 9297, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9297

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

[8] 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

[9] 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

[10] Cloudflare. (2022, Mar.) Unlocking QUIC's proxying potential with MASQUE. Cloudflare Blog. [Online]. Available: https://blog.cloudflare.com/unlocking-quic-proxying-potential/

[11] Mendes, L. Blöcher, E. Rittenhouse, and F. Darling. (2025, Nov.) Async QUIC and HTTP/3 made easy: tokio-quiche is now open-source. Cloudflare Blog. [Online]. Available: https://blog.cloudflare.com/async-quic-and-http-3-made-easy-tokio-quiche-is-now-open-source/

[12] Tokio Contributors, "Crate tokio 1.48.0 (rust crate documentation)," Docs.rs. [Online]. Available: https://docs.rs/tokio/1.48.0/tokio/

[13] Cloudflare. (2025, Aug.) Github: tokio-quiche. [Online]. Available: https://github.com/cloudflare/quiche/tree/0.24.5/tokio-quiche

[14] D. Petri. Nada, a prototype MASQUE implementation. GitHub repository. [Online]. Available: https://github.com/danielpetri1/nada

# Asynchronous Traffic Shaping in Practice

Fabian Heinrich, Florian Wiedner*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: fabian.heinrich@tum.de, wiedner@net.in.tum.de

***Abstract*—Modern industrial applications impose strict requirements on bandwidth, latency, and jitter that standard Ethernet cannot meet. This is particularly critical for safety-critical systems, which rely on real-time data transmission to function properly. The Time Sensitive Networking (TSN) was introduced to address this problem by enabling real-time data transmission and no loss due to buffer congestion by precisely timing packets using synchronized clocks between all switches. However, not all devices are capable of this precision. Consequently, TSN was expanded by Asynchronous Traffic Shaping (ATS). ATS enables precise packet timing without relying on a globally synchronized clock. While TSN is well-established and utilized in existing commercial network solutions, a research gap exists regarding the practical application of ATS, as the literature is limited in highlighting specific use cases of ATS. This paper examines the capabilities of ATS in various practical use cases and analyzes its viability, concluding that in some cases, ATS is a superior solution to synchronized TSN.**

***Index Terms*—asynchronous traffic shaping, time sensitive networking, latency, jitter, determinism**

## 1. Introduction

Nowadays, industrial control systems require strict guarantees regarding latency, jitter, and bandwidth. However, networks are usually based on best-effort, which attempts to deliver packets as quickly as possible without any guarantees of reliability and robustness [1], [2]. This is an issue when safety-critical applications, like a LIDAR for autonomous driving, are operating in the network that requires packets to be delivered with low latency and no loss [1], [3], [4]. The Time Sensitive Networking (TSN) standard was introduced to address this problem by guaranteeing a limited worst-case latency and zero-congestion loss, ensuring deterministic data transmission. This is achieved by synchronizing the clocks of all bridges and switches within the network with sub-microsecond level precision [1], [5], [6]. However, there are scenarios where synchronizing clocks with this precision might not be feasible, e.g., when geographical distances between nodes are too large [7]. For these scenarios, TSN was expanded with Asynchronous Traffic Shaping (ATS). ATS enables similar features as synchronized TSN, while omitting the dependency on a synchronized clock [2], [8].

TSN is already well-known in the industry and has been implemented in several solutions and devices. However, ATS is not as widespread and has limited literature regarding its use cases and applicability in practice [9], [10]. As most use cases are only for synchronous TSN, this paper analyzes existing literature on how synchronous TSN is used and argues whether ATS is also a valid solution for these, thereby addressing the viability of ATS. We further determine whether ATS offers advantages in certain use cases compared to synchronous TSN.

The paper is organized as follows: Section 2 provides relevant background information. Section 3 presents a literature analysis on the use of TSN in practice, examining academic papers and industrial white papers. For every use case, we determine the requirements and explain why ATS is as well a suitable fit, and analyze the advantages and disadvantages. In Section 4, we discuss the practical relevance of ATS based on the identified use cases. We finally conclude our paper in Section 5.

## 2. Background

This section introduces background information about TSN and ATS. Additionally describes the advantages and disadvantages of each approach.

### 2.1. Time Sensitive Networking

TSN extends standard Ethernet to support both best-effort and real-time traffic on the same link. While best-effort networks do not provide any time or bandwidth guarantees, TSN reserves the required bandwidth in each bridge that participates in transmitting traffic. Each bridge then uses scheduling and shaping to precisely time packets, enabling traffic prioritization. This allows zero packet loss due to buffer congestion and an extremely low packet loss caused by equipment failure. Additionally, the reserved bandwidth guarantees an upper bound on latency, as packet transmission becomes deterministic. To enable this precise timing, all bridges are synchronized with high precision. This results in a global schedule that dictates exactly when which data can be transmitted [1], [11]–[15].

### 2.2. Asynchronous Traffic Shaping

ATS is a shaping and scheduling mechanism within the TSN toolset that does not rely on synchronized clocks but achieves similar features. Instead, ATS uses the local clock on every device for timing, resulting in per-hop shaping that calculates the eligibility time of every packet at each switch. ATS algorithms have higher complexity and computational effort than synchronized TSN algorithms,

 doi: 10.2313/NET-2026-05-1_11

but do not require hardware support for high-precision synchronization. The ATS algorithms are specified in the IEEE 802.1Qcr standard and based on algorithms such as Paternoster and Length-Rate Quotient [2], [8], [13], [16], [17].

## 3. Analysis of Practical Use Cases

This section examines the various practical applications of ATS and assesses the benefits and drawbacks of utilizing ATS for these use cases. As there is limited availability of literature about applied ATS, we analyze the use cases of TSN in practice and assess the suitability of ATS for these use cases.

IEEE defines six profiles of use cases: Aerospace, Industrial Automation, Automotive, Fronthaul, Service Provider Networks, and Audio-Video-Bridging [18]–[23].

### 3.1. Aerospace onboard Ethernet Communication

The IEEE P802.1DP is a profile intended for use in aerospace applications ranging from commercial and military aircraft to the space domain. It describes certain functions that are needed to meet the constraints of aerospace networks [18], [24], [25]. These constraints range from the capability of handling very low data flows of 10 Mbit/s, up to very high bandwidths of >10 Gbit/s [24], [25].

The main challenge for aerospace networks is the demand for varying determinism, real-time transmission, and reliability, depending on the type of communication, while meeting specific safety requirements, such as fault tolerance. Traditionally, aerospace networks rely on legacy protocols like SpaceWire, MIL-STD-1553, or ARINC-664, which often fail to meet the increasing demand for bandwidth, as modern architectures require higher data throughput due to the installation of more sensors and devices [3], [24], [25].

**3.1.1. Commercial Aircraft.** One use case of TSN lies in commercial aircraft. Commercial aircraft typically consist of three distinct types of networks: the Aircraft Control Domain (ACD), the Airline Information Service Domain (AISD), and the Passenger Entertainment and Network Service Domain (PIESD). ACD consists of systems and networks to ensure the safety of the flight. This includes, for example, flight control systems like engine control, autopilot. This network requires a low bandwidth, while determinism, bounded-latency, and real-time transmission are crucial. The AISD covers data that is not directly necessary for safety but is still important, such as safety information provided to passengers. This has medium requirements on determinism. The PIESD handles network and entertainment services for passengers. It requires low determinism but a high bandwidth [3], [18].

TSN is able to converge these three networks into a single network while still being able to meet the requirements of the respective bandwidth, determinism, synchronization, and real-time transmission, due to its capability of flow prioritization and policing [3], [14], [18].

As ATS offers similar features to TSN, it can be applied as well. Additionally, ATS offers the advantage of reduced configuration complexity and increased robustness due to its lack of global clock dependency. However, ATS has a higher computational overhead than TSN for high data rates, e.g., when large amounts of passengers stream videos [3], [26], [27].

**3.1.2. Military Aircraft.** Another use case for TSN is military aircraft. They typically deploy two different types of networks. The first domain is the Air Vehicle System, which is responsible for the safe control and monitoring of the aircraft. The second domain is the Mission System, which can be used for various applications, including weapon systems, radar, and other sensors. Both domains require real-time data transmission, high bandwidth, and high determinism, which again can be achieved by TSN [3], [18], [28].

In contrast to synchronized TSN, ATS has reduced configuration complexity, as it does not rely on synchronized clocks. Hence, the used network devices do not need hardware support for high-precision synchronization. However, mission-critical systems require high reliability, which might not be given by ATS, as it doesn't guarantee ultra-low jitter like TSN [2], [3], [28].

**3.1.3. Satellites and Space Travel.** The last major use case for TSN in the aerospace industry is satellites and space travel. Satellites usually consist of two domains: the flight control system, which requires high determinism, and the payload network that is responsible for communications, transponders, cameras, and scientific research instruments, which need a high bandwidth [3].

Sanchez et al. [29] have successfully integrated and validated TSN for the avionics of the Miura 1 sounding rocket, a suborbital research rocket, and verified its applicability to similar scenarios. This means that the feasibility of using TSN for building avionics networks has been illustrative shown. Compared to other space bus and interface standards, such as SpaceWire, TSN offers more promising results in terms of flexibility, determinism, reliability, and bandwidth capability. Additionally, convergence of networks allows for cable reduction, leading to a lower weight [3], [14], [18], [29], [30].

While ATS inherits most features of TSN, it offers the additional advantage of being less prone to faults caused by radiation than TSN. The reason behind this is that for synchronized TSN, the timing of packets relies on a global schedule and synchronized clock. If radiation causes a fault in the schedule, by flipping bits, the properties of TSN cannot be guaranteed. However, ATS uses a local clock on every device, hence it is less prone to this kind of failure as only a single device would suffer a fault when bits are flipped by radiation, not the entire transmission [1]–[3], [24], [27].

### 3.2. Industrial Automation

Networks in industrial automation usually consist of two parts: the Industrial Ethernet (IE) and a network for non-critical traffic. IE is used to transport critical traffic between devices, such as the interaction between programmable logic controllers and robots. It usually requires special-purpose standards and solutions, such as EtherCat [31]–[33]. Additional non-critical networks are

required, e.g., to facilitate data exchange between the enterprise and operations to optimize production processes and increase efficiency. IEs are expensive and hinder interoperation between these networks [31], [34].

These constraints result in another use case of TSN as described in the IEEE/IEC 60802: TSN Profile for Industrial Automation. Industrial control applications require consistent and deterministic delivery of data from sensors to controllers and actuators. Additionally, data is needed for information and analytical systems, which require a high bandwidth [4], [19], [32], [35], [36]. TSN enables both critical and non-critical traffic to share the same physical network, ensuring that time-critical data is delivered on time by securing bandwidth [4], [32], [34], [37]. It also has the potential to connect the industrial world in real-time to other systems, enabling the retrieval of massive amounts of information from factories and migrating production control capabilities from local controllers to more central enterprise servers [4], [32], [34].

Multiple automation manufacturers have already integrated TSN into their standards and automation devices, and it is expected to replace IE in the future. Additionally, some industrial protocols, such as CC-Link IE TSN or PROFINET, have adopted communication using the TSN mechanism [32].

The challenge with synchronized TSN, however, is that it requires a centralized, global schedule for the whole factory. When adding new devices, the schedule has to be recalculated by a Centralized Network Controller. Adding a new device results in recalculating the network's whole schedule. ATS enables that new devices can be used immediately without a setup change, which is desirable for the flexible Industry 4.0. This is enabled by class-based shaping, where only the configuration for the specific class need to be updated [2], [4], [32]. Additionally it might be desired to connect devices using Fifth Generation (5G) to maintain high flexibility in the setup. For this technology, high-precision clock synchronization is not feasible, resulting in ATS being a superior solution [31], [33], [38].

## 3.3. Automotive In-Vehicle Communications

In-vehicle networks support various types of traffic that require different levels of determinism and bandwidth. Some systems require hard real-time performance for functional safety, such as power trains and vehicle body control systems. Additional bandwidth with high determinism is required for automatic driving features to transfer data from sensors, while simultaneously delivering audio and video streams to individual devices, to provide glitch-free infotainment [5], [6], [37].

As most of the traffic in the in-vehicle network is predictable, TSN is a suitable candidate for the network, as the predictability of traffic enables it to provide low latency and zero loss of critical data while transmitting it in real-time. Additionally, it enables the implementation of different bandwidth and determinism requirements for in-vehicle networks within a single physical network. TSN enables high-speed Ethernet and communication to enter multiple segments of modern vehicles, supporting future autonomous driving technology up to the highest level of driving automation [6], [14], [35], [36], [39].

As ATS provides similar functionality to synchronized TSN, it initially appears to be a good candidate, offering real-time support while not relying on clock synchronization. ATS is capable of achieving lower end-to-end latencies than TSN for some scenarios, as TSN allows traffic bursts to accumulate, whereas ATS smooths out traffic bursts due to its per-hop shaping property. Hence, ATS is a suitable choice for shaping bursty traffic generated by radar and LIDAR sensors [6], [39], [40].

However, ATS can cause unbounded latencies in networks providing redundancy as well as in non-FIFO networks, which are often used for in-vehicle networks. ATS is challenging for these kinds of networks as ATS guarantees real-time delivery of critical data and requires careful design. Failure to do so could result in non-conformity to automotive functional safety as outlined in ISO 26262 [6], [39], [40].

## 3.4. Fronthaul

The IEEE P802.1CM is a TSN Profile for Fronthaul. It describes the network segment that connects cellular radios to their base-band controllers using deterministic Ethernet. Traditionally, this connection requires dedicated, expensive point-to-point fiber cables [21], [37].

As 5G networks demand ultra-reliable, low-latency, and high-bandwidth, TSN is an appealing technology to meet these constraints. 5G requires real-time transmission for interconnecting 5G network functions and services [31], [37], [38].

Additionally, it facilitates data transfer between the cellular radio and the base-band controllers, allowing mobile traffic to coexist with critical services without relying on dedicated hardware [31], [37].

Due to its lower complexity and higher scalability, Prados et al. [31] suggest using ATS over TSN for this type of application, as fronthaul networks often cover large geographical distances, making precise clock synchronization challenging. However, using ATS might result in higher jitter compared to TSN, which is additionally amplified by wireless transmissions [31], [41].

## 3.5. Service Provider Networks

Service Provider Networks are large-scale telecommunication infrastructures. Their profile is defined in IEEE P802.1DF. These networks include 5G access, backhaul, and multi-access edge computing systems. They provide connectivity services to vertical industries such as industrial automation. They facilitate complex scenarios, such as the remote control of cranes [5], [22], [41].

Service Provider Networks have special requirements regarding bandwidth and reliability. Currently, service provider networks often operate on a best-effort basis, which does not meet current industry demands. The industry requires determinism and real-time data transmission while ensuring low packet loss [5], [41].

TSN is able to provide the required bandwidth while guaranteeing reliability and determinism. However, due to hardware compatibility, it can be beneficial to use ATS, as it does not rely on clock synchronization and can therefore be implemented on existing quality of service devices,

which often do not provide hardware support for high-precision synchronization. However, ATS might not offer the same precision as TSN regarding jitter [15], [41].

### 3.6. Utility Operational Network

Another use case is utility operational networks, which are used to manage the distribution of electricity on the transmission and distribution grid. This involves substations and various protective and control devices that communicate over an additional network. Special-purpose digital connections are used to separate these networks [7].

These networks require low latency and real-time performance on the network to support critical applications, such as teleprotection, substation automation, and Distributed Energy Resource management. E.g., teleprotection requires a very low latency of $<$10 ms and highly consistent end-to-end latency. Additionally, these networks need the possibility to integrate new devices [7].

TSN fits this use case, as it provides bounded latency and zero congestion loss. It offers a converged architecture to support both real-time applications and regular communication, providing the required reliability [7].

However, TSN has the disadvantage that the integration of new devices requires recalculating the global schedule, and clock synchronization over long distances is difficult. Therefore, ATS would offer some benefit over TSN, as it does not have these disadvantages [7], [31].

## 4. Discussion

The analysis of use cases in Section 3 shows that ATS is not just a fallback for TSN, but a practical solution with distinct advantages in specific domains. While the industry currently favors TSN, our analysis highlights scenarios where ATS offers superior viability.

### 4.1. Scalability and Robustness

The most significant advantage of ATS compared to synchronized TSN is that it eliminates the dependency on a synchronized clock. As seen in the *Fronthaul* and *Utility operational network* use cases, achieving a sub-microsecond precision clock synchronization over large geographical distances or in large networks is technically challenging due to propagation delay and jitter [7], [31].

Omitting global clock synchronization also increases the robustness for certain scenarios. As seen in the *Satellites and Space Travel*, a radiation-induced fault in the grandmaster clock can desynchronize the entire network. As ATS uses per-hop shaping and relies on the local clock of every device, this is not an issue. ATS even smooths out bursts, further increasing the robustness [3].

### 4.2. Flexibility

Another key advantage of ATS over synchronized TSN is that ATS provides greater flexibility when the network is reconfigured. Adding a new device to a network with synchronized TSN requires recalculating the global schedule. This plays a major role in the *Industrial Automation*, *Utility Operational Network*, and *Service Provider Networks* use cases. ATS enables a "plug-and-play" approach, due to per-class shaping, which allows adding new devices without reconfiguring the entire network, as long as bandwidth is available within the specific class [2], [41].

### 4.3. Further Advantages

While synchronized TSN often requires new specialized hardware for clock synchronization, ATS is capable of running on hardware that doesn't support high-precision synchronization but is often already used in industry. This can be advantageous in the *Fronthaul*, *Industrial Automation*, *Service Provider Networks*, *Automotive In-Vehicle Communications*, and *Aerospace Onboard Ethernet Communication*. Finally, synchronous TSN consumes link capacity more than asynchronous approaches due to the need for synchronization communication [31].

### 4.4. Limitations

However, ATS is not a universal replacement for synchronized TSN. While ATS guarantees bounded latency and zero congestion loss, it is not capable of achieving the ultra-low jitter that synchronized TSN can. Because of this, for applications requiring precise periodic data arrival, synchronized TSN remains superior. Finally, as discussed for the *Automotive In-Vehicle Communications* use case, ATS results in high latencies in non-FIFO networks and TSN should be preferred [40].

### 4.5. Practical Comparison of Synchronized TSN and ATS

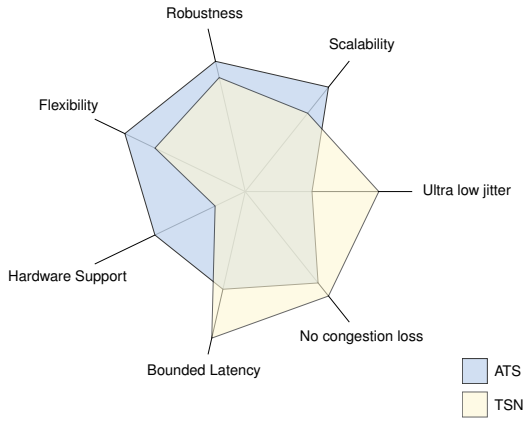



Figure 1: Practical Comparison of TSN and ATS

Overall, we can summarize our findings that ATS and synchronized TSN both have superior properties depending on the use case. Figure 1 summarizes our findings and shows for which properties how suitable TSN and ATS are. For use cases that require scalability and flexibility, ATS is a more suitable candidate as it does not rely on a global schedule. If robustness is required, ATS and TSN can be chosen equally, while for space applications ATS is superior as it is less prone to failures induced by radiation. ATS should also be used for use cases, where hardware does not support high precision clocks and synchronization. However for use cases that require ultra low jitter,

no congestion loss and bounded latency, TSN should be preferred to ATS, as TSN guarantees these features more reliable due to its determinism in transmitting packets, while ATS can lead to unbounded latency and hence to congestion loss and higher jitter for certain scenarios.

## 5. Conclusion

To conclude, we can state that ATS has some advantages over TSN, which is already widely used and implemented in several devices. These use cases span over aircraft networks, industrial automation, automotive in-vehicle communications, fronthaul, audio and video streaming, service provider networks, and many more. We pointed out that ATS can be applied to most of the presented use cases, because ATS offers advantages for some of these use cases compared to TSN.

While TSN currently dominates the industry, we have shown that ATS addresses several limitations regarding scalability, robustness, and flexibility, underlining its practical viability. As ATS does not require global clock synchronization with sub-microsecond precision, it is a more suitable solution when there are large geographical distances between two devices, providing greater robustness. Additionally, it requires orchestration overhead to synchronize the clocks in large networks, which is not necessary for ATS, thereby providing better scalability. Another benefit of not relying on the global clock is increased flexibility, as devices can be added or removed more easily using ATS. However, even though ATS offers advantages for these scenarios, it is still not used in practice. One reason might be that ATS is still experimental and lacks widespread implementations. Future work should focus on the experimental validation of ATS's performance in real-world scenarios.

## References


[1] N. Finn, "Introduction to time-sensitive networking," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 22–28, 2018.

[2] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, 2016, pp. 75–85.

[3] "Time Sensitive Networking for Aerospace," *Fraunhofer IPMS, 0625011WP DCC TSN*, 2023, White Paper.

[4] "A Practice of TSN over 5G for Industry," *Intel Corporation, Comba Telecom, WP-633530-0.9*, 2021, White Paper.

[5] D. A. Schwarz, F. Wagner, J. Bierschenk, D.-I. G. Stöger, and D.-I. J. Wohlmuths, "Software-Defined Vehicles (SDV) with Time-Sensitive Networking (TSN) and Software-Defined Networking (SDN)," *ETAS GmbH, TTTech Computertechnik AG, and Robert Bosch GmbH*, 2025, White Paper.

[6] M. Kim, Y. Do, J. Kim, and J. Jeon, "Asynchronous traffic handling in time-sensitive in-vehicle network," in *2023 17th International Conference on Ubiquitous Information Management and Communication (IMCOM)*. IEEE, 2023, pp. 1–6.

[7] R. Salazar, T. Godfrey, N. Finn, C. Powell, B. Rolfe, and M. Seewald, "Utility applications of time sensitive networking white paper," *Utility Applications of Time Sensitive Networking White Paper*, pp. 1–19, 2019.

[8] Z. Zhou, Y. Yan, M. Berger, and S. Ruepp, "Analysis and modeling of asynchronous traffic shaping in time sensitive networks," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2018, pp. 1–4.

[9] "Time-Sensitive Networking (TSN) for Industrial Automation Using Open Platform Communications Unified Architecture (OPC UA)," *Microchip Technology Inc., DS50003952A*, 2025, White Paper.

[10] D. K. Weber, "EtherCAT and TSN – Best Practices for Industrial Ethernet System Architectures," *EtherCAT Technology Group*, 2018, White Paper.

[11] Z. Zhou, M. S. Berger, S. R. Ruepp, and Y. Yan, "Insight into the IEEE 802.1 Qcr asynchronous traffic shaping in time sensitive network," *Advances in Science, Technology and Engineering Systems Journal*, vol. 4, no. 1, pp. 292–301, 2019.

[12] J. Schopohl, F. Wiedner, and C. Schwarzenberg, "Time-Sensitive Networking – Cyclic Queuing and Forwarding Shaper on Linux," 2022, Master's Thesis.

[13] "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks - Amendment 34: Asynchronous Traffic Shaping," *IEEE Std 802.1Qcr-2020 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, IEEE Std 802.1Qcy-2019, and IEEE Std 802.1Qcx-2020)*, pp. 1–151, 2020.

[14] M. Hegarty, "A Deterministic Ethernet Solution based on Time Sensitive Networking (TSN)," *Data Device Corporation*, 2018, White Paper.

[15] "Fragen und Antworten zu TSN," *Siemens, 109757263*, 2020, White Paper.

[16] J. Schopohl, F. Wiedner, and C. Schwarzenberg, "Asynchronous Traffic Shaping with Paternoster," 2022, Interdisciplinary Project.

[17] F. Heinrich, F. Wiedner, and M. Helm, "Asynchronous Traffic Shaping with UBS/LRQ," 2025, Bachelor's Thesis.

[18] "IEEE Draft Standard for Local and Metropolitan Area Networks - Time-Sensitive Networking for Aerospace Onboard Ethernet Communications," *IEEE P802.1DP/D3.2, SAE AS6675-2025, June 2025*, pp. 1–66, 2025.

[19] "IEC/IEEE Draft International Standard Time-Sensitive Networking Profile for Industrial Automation," *IEC/IEEE P60802/D3.4, May 2025*, pp. 1–218, 2025.

[20] "IEEE Standard for Local and metropolitan area networks - Time-Sensitive Networking Profile for Automotive In-Vehicle Ethernet Communications," *IEEE Std 802.1DG-2025*, pp. 1–62, 2025.

[21] "IEEE Standard for Local and metropolitan area networks – Time-Sensitive Networking for Fronthaul," *IEEE Std 802.1CM-2018*, pp. 1–62, 2018.

[22] T. Wang, "P802. 1DF-TSN profile for service provider networks," 2020.

[23] "IEEE Standard for Local and metropolitan area networks–Audio Video Bridging (AVB) Systems," *IEEE Std 802.1BA-2011*, pp. 1–45, 2011.

[24] "Enabling TSN Ethernet for Space Applications," *Microchip Technology Inc., DS00005495B*, 2024, White Paper.

[25] T. Fiori, F. G. Lavacca, F. Valente, and V. Eramo, "Proposal and investigation of a lite time sensitive networking solution for the support of real time services in space launcher networks," *IEEE Access*, vol. 12, pp. 10 664–10 680, 2024.

[26] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of IEEE 802.1 TSN time aware shaper (TAS) and asynchronous traffic shaper (ATS)," *IEEe Access*, vol. 7, pp. 44 165–44 181, 2019.

[27] S. F. Bush, "Toward efficient time-sensitive network scheduling," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 3, pp. 1830–1842, 2022.

[28] C. Turner, D. Zage, K. Avery, D. Walsh, S. Rodriguez, and M. Chabroux, "Hitting the Moving Target with Time Sensitive Networking -An Industry Collaboration on Risk Mitigation," *Aptiv PLC, Intel Corporation, Lockheed Martin Corporation, Parry Labs, TTTech Auto, Eind River Systems Inc.*, 2020, White Paper.

[29] J. Sanchez-Garrido, B. Aparicio, J. G. Ramírez, R. Rodriguez, M. Melara, L. Cercós, E. Ros, and J. Diaz, "Implementation of a time-sensitive networking (TSN) Ethernet bus for microlaunchers," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 5, pp. 2743–2758, 2021.

[30] F. Ekstedt Karpers, "Combining proprietary real-time Ethernet protocols with Time-Sensitive Networking for avionics: A simulation study in OMNeT++ with INET 4.4," 2023.

[31] J. Prados-Garzon and T. Taleb, "Asynchronous time-sensitive networking for 5G backhauling," *IEEE Network*, vol. 35, no. 2, pp. 144–151, 2021.

[32] J. Wiberg and T. Bieber, "Time Sensitive Networking (TSN)," *HMS Networks*, 2021, White Paper.

[33] F. Fazel and D. Cavalcanti, "Wireless TSN: Tests and Use Case Demonstrations with Wi-Fi," *Avnu Alliance*, 2024, White Paper.

[34] S. Brooks and E. Uludag, "Time-Sensitive Networking: From Theory to Implementation in Industrial Automation," *TTTech Computertechnik AG, Intel Corporation, WP-01279-1.0*, 2019, White Paper.

[35] P. Varis and T. Leyrer, "Time-sensitive networking for industrial automation," *Texas Instruments, Texas, SPRY316*, 2023, White Paper.

[36] A. Regev, "Asynchronous Traffic Shaper (802.1Qcr) and its applicability to Automotive use-cases," 2023, Presentation.

[37] N. Finn, "Time-sensitive and Deterministic Networking Whitepaper," *Huawei Technologies Co. Ltd*, 2017, White Paper.

[38] J. Caleya-Sanchez, J. Prados-Garzon, P. Muñoz, J. M. Lopez-Soler, and P. Ameigeiras, "Flow Prioritization in asynchronous TSN with multiple ATS instances," *IEEE Transactions on Mobile Computing*, 2025.

[39] S. Samii and H. Zinner, "Level 5 by layer 2: Time-sensitive networking for autonomous vehicles," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 62–68, 2018.

[40] T. Lübeck, P. Meyer, T. Häckel, F. Korf, and T. C. Schmidt, "Asynchronous traffic shaping and redundancy: Avoiding unbounded latencies in in-car networks," in *2025 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2025, pp. 1–8.

[41] X. Jia and T. Wang, "Practice of "TSN-IP" over 5G carrier networks," 2022, Presentation.

# Analyzing Real-World DoH and ECH Adoption Using Mozilla Telemetry

Matthias Kirstein, Lion Steger*, Christian Dietze*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: kirm@net.in.tum.de, stegerl@net.in.tum.de, diec@net.in.tum.de

***Abstract*—DNS over HTTPS (DoH) and Encrypted Client Hello (ECH) are two of the most critical recent developments for closing remaining privacy gaps in Internet protocols. This paper analyzes the adoption of these technologies by leveraging Mozilla Firefox's public telemetry data. We find that global DoH usage has remained nearly constant throughout 2025, at around 13.91 % of handshakes. In regions where the protocol is rolled out, network heuristics leave DoH active for approximately 64 % of clients. A significant portion of the remaining share is attributed to enterprise-controlled browsers. Among users who manually configure their DoH settings, the choice of provider is notably distributed: custom resolvers (37 %) are the most frequent selection, surpassing both Cloudflare (34 %) and NextDNS (29 %). In contrast, ECH adoption remains at a marginal 0.17 % of handshakes. While only 38 % of ECH-enabled handshakes used DoH, this is primarily due to the limited regional rollout of DoH.**

***Index Terms*—DNS, DoH, ECH, measurement, evaluation**

## 1. Introduction

Modern Internet protocols have largely secured the confidentiality of payload data through encryption mechanisms. However, critical gaps remain regarding the privacy of metadata. Specifically, the domain names a user visits continue to be exposed during traditional cleartext Domain Name System (DNS) resolution and within the unencrypted Server Name Indication (SNI) of the Transport Layer Security (TLS) handshake. The plaintext visibility of these domains enables network observers to reconstruct sensitive user profiles and track activity, even when the application's content itself is encrypted. To address these vulnerabilities, the Internet Engineering Task Force (IETF) has standardized DoH [1] and is currently standardizing the TLS ECH extension [2].

While the theoretical benefits of these protocols are well-established, their real-world impact depends on widespread deployment and correct configuration. Mozilla has been an early adopter of these technologies, enabling DoH by default in specific regions and implementing support for the draft ECH specification in Firefox [3], [4]. However, the extent to which these features are successfully utilized by clients in the wild remains an open question.

In this paper, we analyze the deployment progress of these privacy technologies. By leveraging Mozilla's public telemetry data, we assess the adoption rates of DoH and ECH directly from the client's perspective. This approach allows us to quantify how frequently these protections are actually used in real-world connections.

## 2. DoH

Conventional DNS resolution was historically designed to prioritize speed, relying primarily on the connectionless User Datagram Protocol (UDP) for transport. While avoiding connection overhead makes this design lightweight and efficient, the reliance on cleartext transmission without cryptographic authentication provides no inherent security. Consequently, DNS traffic is fully visible to on-path observers, and the stateless nature of UDP allows responses to be easily spoofed or modified. This visibility allows a local network or Internet Service Provider (ISP) to track visited domains, while the lack of authentication enables active attacks, such as redirecting users to malicious destinations.

To mitigate these vulnerabilities, DoH, standardized in RFC 8484 [1], encapsulates DNS traffic within HTTPS. By leveraging the underlying TLS [5], [6], this approach replaces unauthenticated, cleartext exchanges with a secure channel that guarantees confidentiality, integrity, and authentication of all DNS queries. In addition to encrypting queries and responses to prevent eavesdropping, server authentication protects against active manipulation by on-path attackers. An additional benefit is traffic obfuscation: by encapsulating DNS queries within HTTPS packets sent on the default HTTPS port (port 443), DoH traffic can be nearly indistinguishable from standard web activity. This complicates traffic analysis and filtering.

Rather than treating HTTP as a mere transport tunnel, DoH can directly benefit from its application-layer features, including authentication mechanisms, compression, and caching. Moreover, by using HTTP as the protocol, DoH enables direct DNS access for web applications, bypassing the operating system's local resolver.

Encapsulating DNS in HTTPS introduces significant overhead compared to UDP, driven largely by TLS handshake delays and HTTP payload requirements. Nevertheless, the overhead introduced by DoH has been shown to have only a marginal impact on actual webpage load times [7]. Modern protocols like HTTP/2 or HTTP/3 offset these costs by reusing connections to amortize handshake delays and using header compression and multiplexing to reduce data footprints and eliminate head-of-line blocking.

While DoH resolvers are typically configured out-of-band using URI templates, RFC 9462 [8] enables the automated discovery of DoH endpoints via Service Binding (SVCB) records. These specialized DNS entries

 doi: 10.2313/NET-2026-05-1_12

supply clients with the necessary connection details, such as supported protocols and URIs. This allows clients to seamlessly and securely upgrade from unencrypted DNS to DoH.

Finally, DoH should not be confused with DNS Security Extensions (DNSSEC). While DoH secures the transport channel between the client and the resolver, DNSSEC provides cryptographic validation of the DNS data itself. These mechanisms address different threat models and are most effective when used together to provide comprehensive security.

## 3. ECH

Although TLS version 1.3 significantly enhances the protocol's privacy by encrypting the handshake immediately after the initial key parameters are exchanged, the initial `ClientHello` message remains partially visible. Critically, the SNI extension, which identifies the target domain, is transmitted in plaintext within the initial message. This allows on-path attackers to track user activity, even when the subsequent application data is encrypted. To close the remaining privacy gap, the IETF is currently standardizing the ECH extension [2].

ECH secures the TLS handshake by encrypting sensitive parameters, most notably the SNI. The encrypted payload is encapsulated within a standard, unencrypted `ClientHello`. Upon successful decryption, the server proceeds with the connection using the concealed parameters. Otherwise, if decryption fails, often due to stale ECH configurations, the server either safely falls back to the unencrypted message or triggers a connection retry to provide the client with updated cryptographic keys.

By encrypting the SNI, ECH prevents observers from distinguishing which specific site on a shared IP address is being requested. As a result, the architecture offers significant privacy advantages, particularly in environments where multiple sites are hosted behind a common reverse proxy or Content Delivery Network (CDN). To initiate an ECH-enabled connection, the client must first obtain the server's ECH configuration, typically distributed via DNS HTTPS resource records [9], [10]. Crucially, relying on conventional plaintext DNS to fetch these records, or to resolve the domain itself, exposes the target domain name and leaves the ECH configuration vulnerable to spoofing. Consequently, ECH achieves its full privacy potential only when paired with secure DNS transport like DoH, which protects the confidentiality of the domain query and the integrity of the retrieved ECH keys.

To prevent network middleboxes from targeting and filtering ECH traffic, the protocol introduces a GREASE mechanism. Even if the destination server lacks ECH support and no valid configuration is available, clients still include the extension using a randomized, non-functional payload. This ensures all modern TLS handshakes appear uniform, preventing genuine ECH connections from standing out and discouraging networks from selectively blocking privacy-enhancing technologies.

## 4. Related Work

The real-world impact of DoH and ECH depends on their deployment, which presents different challenges for each protocol. While DoH deployment primarily involves a client-side transition to a compatible resolver, ECH requires individual server operators to actively configure and maintain cryptographic keys. Previous research has primarily measured protocol adoption through active server-side scans and passive network monitoring, which contrasts with our client-centric, telemetry-based analysis.

Zirngibl *et al.* [11] scanned over 400 million domains and found that while 10.5 million had adopted DNS HTTPS records, deployment was almost entirely driven by Cloudflare. Additionally, they found that only a negligible 20 domains possessed active ECH configurations in 2023.

Dong *et al.* [12] conducted a longitudinal study of server-side DNS HTTPS record deployment using the Tranco top 1 million domains. They found that over 27 % of domains published HTTPS records by March 2024, a trend largely driven by Cloudflare's default configurations. However, active ECH adoption dropped to nearly zero in late 2023 after Cloudflare temporarily disabled the feature.

Merlach *et al.* [13] analyzed a month-long campus dataset to evaluate ECH adoption. They found that while 59 % of QUIC flows included an ECH extension, these extensions consisted almost entirely of randomized GREASE values. This discrepancy shows that while modern clients are actively trying to use ECH, server-side adoption has yet to catch up.

García *et al.* [14] conducted Internet-wide scans of the entire IPv4 address space to detect and verify public DoH resolvers. Their analysis revealed a rapid expansion of the ecosystem, reporting that between 2021 and 2022, the number of DoH resolvers increased by 4.8 times and the number of organizations providing DoH doubled.

## 5. Mozilla's Telemetry

To measure protocol adoption directly from the user's perspective, our analysis leverages public telemetry data from Mozilla Firefox. The browser's telemetry infrastructure collects performance and protocol-related metrics across its desktop and mobile platforms. This system is centered around a publicly documented registry of metrics and events that ensures data consistency between the different platforms [15], [16].

Telemetry data is transmitted in discrete packets called pings [17], which contain a primary payload of counters, histograms, and events, alongside environment metadata. These raw pings are processed and aggregated daily into derived datasets. Mozilla restricts raw telemetry access to protect user privacy, only sharing data after strict aggregation and review [18], [19]. Public access is provided through two primary channels: the Glean Aggregated Metrics (GLAM) dashboard and curated public datasets.

### 5.1. GLAM

The GLAM dashboard serves as Mozilla's primary high-level public analysis tool [20]. The dashboard visualizes heavily aggregated telemetry data, enabling the tracking of trends for Firefox Desktop and Firefox for Android across releases. However, not all metric types are available in the tool, and the strict aggregation of data into percentiles makes it impossible to perform precise analysis of certain metrics without the raw data.

### 5.2. Public Datasets

For more granular data than GLAM provides, researchers can request datasets through Mozilla's telemetry data publishing process [18], [19]. To prevent user reidentification, these requests undergo strict privacy and security reviews to enforce aggregation constraints, such as minimum bucket sizes. Approved datasets are published and regularly updated on Mozilla's public data HTTP endpoint [21], [22].

## 6. Adoption

We analyze the real-world adoption of both DoH and ECH using the Mozilla Firefox telemetry datasets introduced in section 5.2. The public availability of this data provides a practical, client-side perspective on the usage of these privacy-enhancing protocols. However, Firefox accounts for a relatively small share of the global browser market [23], [24], which must be considered when generalizing these metrics to the broader web ecosystem.

### 6.1. DoH Adoption

To date, Mozilla's default DoH rollout remains limited to the United States, Canada, Russia, and Ukraine [3]. Although Mozilla has expressed intentions to expand to additional regions, no further geographic deployments have occurred since March 2022.

We analyze DoH adoption within these specific regions using the public `doh_adoption_rate v1` dataset [22], [25]. This dataset provides the count of distinct clients reporting a given metric by country code, beginning January 1, 2025. Notably, this dataset is restricted to Firefox Desktop, meaning mobile telemetry falls outside the scope of this analysis.

**6.1.1. DoH Heuristics.** While DoH significantly improves user privacy, it bypasses local DNS resolvers, potentially disrupting split-horizon DNS and network-level security policies like enterprise filtering or parental controls [3]. Consequently, Firefox performs a series of heuristic checks to validate the network environment before enabling DoH by default [26], [27]. The primary metric for evaluating these checks is `networking.doh_heuristics_result`, which provides insight into both the proportion of clients with DoH enabled and disabled, and the specific reasons for each outcome. Notably, these heuristics are only executed and reported within the rollout regions.

An analysis of the 2025 data reveals that the distribution of heuristic results, as well as the total number of reporting clients, has remained stable throughout the year. This stability is expected, as reporting is limited to the few selected DoH rollout regions and no new geographic deployments occurred in 2025.

Table 1 details the distribution of the individual DoH heuristic results:

- Across the rollout regions, approximately 64 % of clients successfully pass the heuristics check, resulting in DoH being enabled by default.

TABLE 1: Aggregated DoH heuristic results for Firefox Desktop (release channel), January 1, 2025 – November 1, 2025. Based on the sum of daily unique client reports per result value for the `networking.doh_heuristics_result` event.

| DoH Heuristic Result | DoH State | Reports ($10^6$) | Share |
|---|---|---|---|
| `pass` | ✓ | 1363.12 | 64.08 % |
| `enterprise-present` | × | 390.63 | 18.36 % |
| `canary` | × | 235.61 | 11.08 % |
| `vpn` | × | 89.54 | 4.21 % |
| `google` | × | 20.82 | 0.98 % |
| `enterprise-disabled` | × | 11.41 | 0.54 % |
| `nrpt` | × | 5.82 | 0.27 % |
| `parental` | × | 5.79 | 0.27 % |
| `youtube` | × | 1.64 | 0.08 % |
| `manually-disabled` | × | 1.25 | 0.06 % |
| `manually-enabled` | ✓ | 0.96 | 0.05 % |
| `enterprise-enabled` | ✓ | 0.54 | 0.03 % |
| `OTHER` | - | 0.05 | <0.01 % |
| Total | | 2127.18 | 100.00 % |

- If Firefox has enterprise policies set, the decision whether DoH is enabled is determined by the `DNSOverHTTPS` policy rather than the DoH heuristics [26]. Approximately 18 % of heuristic runs detected enterprise policies but lacked an explicit `DNSOverHTTPS` configuration, resulting in DoH being disabled. When configured, results are categorized as `enterprise-enabled` or `enterprise-disabled`.
- Approximately 11 % of clients disable DoH upon resolving the `use-application-dns.net` canary domain. This domain enables network administrators to notify Firefox that the local DNS resolver offers special features, which make the network unsuitable for DoH [28].
- In 4 % of cases, Firefox detects an active VPN on Windows, which prevents the automatic use of DoH [27].
- The remaining results can be attributed to enabled parental controls (`parental`), forced SafeSearch (`google`, `youtube`), the presence of the Name Resolution Policy Table (NRPT) on Windows (`nrpt`), or manual DoH configurations (`manually-enabled`, `manually-disabled`) [27].

In summary, the heuristics successfully validate the network environment for the majority of users, allowing Firefox to enable DoH by default for approximately 64 % of client networks in the rollout regions.

**6.1.2. Global DoH Usage.** Because the DoH heuristic metric is only reported in the rollout regions, it cannot be used to assess global DoH adoption. However, the `ssl_handshake.privacy` metric provides a broader perspective. This metric tracks the usage of several privacy features for every handshake, including whether TLS 1.3 is negotiated, whether the connection's DNS records were fetched over DoH, and whether the server accepted an offered ECH configuration. As this metric is currently only available within the ECH dataset, detailed further in section 6.2, publicly accessible data is limited to the last six months.

TABLE 2: Aggregated DoH provider choices for Firefox Desktop (release channel), January 1, 2025 – November 1, 2025. Based on the sum of daily unique client reports per choice for the `security.doh.settings.provider_choice_value` event.

| DoH Provider Choice | Reports ($10^3$) | Share |
|---|---|---|
| `custom` | 274.61 | 37.12 % |
| `mozilla.cloudflare-dns.com` | 250.64 | 33.88 % |
| `firefox.dns.nextdns.io` | 211.67 | 28.61 % |
| `private.canadianshield.cira.ca` | 2.86 | 0.39 % |
| `OTHER` | 0.02 | 0.00 % |
| Total | 739.80 | 100.00 % |

The data indicates that the proportional usage of DoH remained stable over this six-month period. Of the $9.54 \times 10^{12}$ handshakes recorded globally in Firefox Desktop, approximately 13.91 % utilized DoH.

When breaking down this figure on a per-country basis, the disparity between rollout and non-rollout regions becomes clearly evident. Across the four DoH rollout countries, 43.78 % of handshakes utilized DoH, whereas across all other countries, this figure drops to just 0.79 %.

**6.1.3. DoH Provider Choice.** Mozilla's Trusted Recursive Resolver (TRR) program allows resolver operators to become default or non-default DoH options in Firefox [29]. Participating resolvers must contractually meet strict data privacy, transparency, and operational standards. Mozilla categorizes TRR partners as follows:

- Regional default partners: Cloudflare (US, Russia, Ukraine) and CIRA (Canada).
- Global non-defaults: Cloudflare and NextDNS.
- ISP steering partners: Comcast (US) and Shaw (Canada), which take precedence in their networks unless manually overridden.

When a user modifies their DoH settings in Firefox, the `security.doh.settings.provider_choice_value` event records the DoH resolver selection. This event captures the URL for Mozilla's partner resolvers or reports `custom` for any other user-defined URL. An analysis of the telemetry data shows that the distribution of reported values has remained relatively stable throughout 2025. As shown in table 2, approximately 37.12 % of clients configuring their DoH provider opted for a `custom` resolver, followed by Cloudflare at 33.88 % and NextDNS at 28.61 %.

## 6.2. ECH Adoption

To analyze the ECH adoption, we use Mozilla's `ech_adoption_rate v1` dataset [22], [30], which at the time of research contained ECH-related telemetry for Firefox Desktop from May 1, 2025, to November 1, 2025.

As previously discussed in section 6.1.2, the `ssl_handshake.privacy` metric records the privacy features used during each handshake, including whether the server accepted an ECH configuration. The telemetry indicates that the percentage of handshakes using ECH is negligible and has not significantly changed over the observed six-month period. As shown in table 3, ECH accounts for only 0.17 % of total handshakes and 0.47 % of handshakes using DoH. This negligible adoption indicates that only a handful of servers currently have ECH configured.

TABLE 3: Aggregated ECH usage in Firefox Desktop (release channel) handshakes, May 1, 2025 – November 1, 2025. Based on the total number of handshakes reported in the `ssl_handshake.privacy` metric. Percentages are calculated relative to the total handshakes within each respective table section. Anomalies omitted.

| ECH Usage | Handshakes ($10^9$) | Share |
|---|---|---|
| Without ECH | 9519.60 | 99.83 % |
| With ECH | 16.29 | 0.17 % |
| Using DoH, without ECH | 1320.51 | 99.53 % |
| Using DoH, with ECH | 6.19 | 0.47 % |
| Using ECH, without DoH | 10.10 | 61.98 % |
| Using ECH, with DoH | 6.19 | 38.02 % |

ECH negotiation in Firefox was initially tied to the internal DoH implementation, as the privacy value of ECH is severely diminished if the browser uses unencrypted DNS to fetch the necessary configurations [31]. Since version 129, however, Firefox allows delegating these lookups to the native operating system DNS resolver to broaden ECH availability when DoH is disabled [4], [31]. This shift is reflected in the telemetry: only 38.02 % of ECH-enabled handshakes used Firefox's DoH, with the majority relying on native DNS infrastructure.

This observation is largely a consequence of Mozilla's geographically limited DoH rollout. A country-level analysis confirms the regional disparity: in all four DoH rollout countries, over 70 % of ECH handshakes used DoH. In contrast, in all other countries, the share remains below 30 %, with the majority falling well under 10 %.

# 7. Conclusion

While DoH and ECH represent critical advancements in closing the remaining privacy gaps in web communication, our analysis of Mozilla Firefox telemetry indicates that real-world deployment has stagnated. Although DoH is enabled by default in select regions, the rollout has seen no geographic expansion since March 2022, leaving global adoption at approximately 13.91 % of Firefox Desktop handshakes. Our findings show that users who manually configure their DoH provider frequently opt for custom resolvers, representing approximately 37 % of selections. Meanwhile, ECH adoption remains marginal, accounting for only 0.17 % of total handshakes with no significant upward trend observed in 2025.

The effectiveness of these protocols is interdependent, as ECH provides limited privacy if the preceding DNS query is unencrypted. We found that over 60 % of ECH handshakes in Firefox still rely on native DNS, potentially leaking domain names via cleartext queries. Furthermore, preventing domain leaks during the handshake relies on widespread ECH support from CDN and server operators. Therefore, these technologies will only provide robust privacy guarantees when the client-side transition to encrypted DNS is matched by broader ECH adoption across the server landscape.

# References


[1] P. E. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," RFC 8484, Oct. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8484

[2] 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/

[3] Mozilla, "DNS over HTTPS (DoH) FAQs," https://support.mozilla.org/en-US/kb/dns-over-https-doh-faqs, Feb. 2024, accessed: 2025-12-02.

[4] ——, "Encrypted Client Hello (ECH) - Frequently asked questions," https://support.mozilla.org/en-US/kb/faq-encrypted-client-hello, Aug. 2024, accessed: 2025-12-02.

[5] E. Rescorla, "HTTP Over TLS," RFC 2818, May 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2818

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

[7] Böttger, Timm and Cuadrado, Felix and Antichi, Gianni and Fernandes, Eder Leão and Tyson, Gareth and Castro, Ignacio and Uhlig, Steve, "An Empirical Study of the Cost of DNS-over-HTTPS," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 15–21. [Online]. Available: https://doi.org/10.1145/3355369.3355575

[8] T. Pauly, E. Kinnear, C. A. Wood, P. McManus, and T. Jensen, "Discovery of Designated Resolvers," RFC 9462, Nov. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9462

[9] B. M. 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, Jun. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-svcb-ech/08/

[10] ——, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)," RFC 9460, Nov. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9460

[11] J. Zirngibl, P. Sattler, and G. Carle, "A First Look at SVCB and HTTPS DNS Resource Records in the Wild," in *2023 IEEE European Symposium on Security and Privacy Workshops*, July 2023, pp. 470–474.

[12] H. Dong, Y. Zhang, H. Lee, S. Huque, and Y. Sun, "Exploring the Ecosystem of DNS HTTPS Resource Records: An End-to-End Perspective," in *Proceedings of the 2024 ACM on Internet Measurement Conference*, ser. IMC '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 423–440. [Online]. Available: https://doi.org/10.1145/3646547.3688410

[13] G. Merlach, M. Trevisan, and D. Giordano, "Encrypted Client Hello Is Coming: A View from Passive Measurements," *Network*, vol. 5, no. 3, 2025. [Online]. Available: https://www.mdpi.com/2673-8732/5/3/29

[14] García, Sebastián and Bogado, Joaquín and Hynek, Karel and Vekshin, Dmitrii and Čejka, Tomáš and Wasicek, Armin, "Large Scale Analysis of DoH Deployment on the Internet," in *Computer Security – ESORICS 2022*, Atluri, Vijayalakshmi and Di Pietro, Roberto and Jensen, Christian D. and Meng, Weizhi, Ed. Cham: Springer Nature Switzerland, 2022, pp. 145–165.

[15] Mozilla, "Glean," https://docs.telemetry.mozilla.org/concepts/glean/glean, Oct. 2025, accessed: 2025-12-02.

[16] ——, "Glean Dictionary," https://dictionary.telemetry.mozilla.org/, accessed: 2025-12-02.

[17] ——, "What Data does Mozilla Collect?" https://docs.telemetry.mozilla.org/introduction/what_data, Aug. 2021, accessed: 2025-12-02.

[18] ——, "Data Publishing @ Mozilla," https://blog.mozilla.org/data/2020/09/25/data-publishing-mozilla/, Sep. 2020, accessed: 2025-12-02.

[19] ——, "Data Publishing - MozillaWiki," https://wiki.mozilla.org/Data_Publishing, Feb. 2021, accessed: 2025-12-02.

[20] ——, "GLAM: Glean Aggregated Metrics Explorer," https://glam.telemetry.mozilla.org/, accessed: 2025-12-02.

[21] ——, "Accessing Public Data," https://docs.telemetry.mozilla.org/cookbooks/public_data, Feb. 2021, accessed: 2025-12-02.

[22] ——, "Mozilla Public Datasets," https://public-data.telemetry.mozilla.org/all-datasets.json, accessed: 2025-12-02.

[23] StatCounter, "Browser Market Share Worldwide | Statcounter Global Stats," https://gs.statcounter.com/browser-market-share, accessed: 2025-12-20.

[24] Cloudflare, "Adoption and Usage Worldwide | Cloudflare Radar," https://radar.cloudflare.com/adoption-and-usage, accessed: 2025-12-20.

[25] Mozilla, "Bug 1982897 - Data Publishing Request," https://bugzilla.mozilla.org/show_bug.cgi?id=1982897, accessed: 2025-12-02.

[26] ——, "Security/DNS Over HTTPS - MozillaWiki," https://wiki.mozilla.org/Security/DNS_Over_HTTPS, accessed: 2025-12-02.

[27] ——, "Security/DNS Over HTTPS/Heuristics - MozillaWiki," https://wiki.mozilla.org/Security/DNS_Over_HTTPS/Heuristics, accessed: 2025-12-18.

[28] ——, "Canary domain - use-application-dns.net," https://support.mozilla.org/en-US/kb/canary-domain-use-application-dnsnet, accessed: 2025-12-02.

[29] ——, "Security/DOH-resolver-policy - MozillaWiki," https://wiki.mozilla.org/Security/DOH-resolver-policy, accessed: 2025-12-14.

[30] ——, "Bug 1957082 - Data Publishing Request," https://bugzilla.mozilla.org/show_bug.cgi?id=1957082, accessed: 2025-12-02.

[31] ——, "Security/Encrypted Client Hello - MozillaWiki," https://wiki.mozilla.org/Security/Encrypted_Client_Hello, accessed: 2025-12-20.

# Optimization for Secure Two-Party Inference with Homomorphic Encryption

Christopher Knop, Christopher Harth-Kitzerow*, Nina Schwanke*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: christopher.knop@tum.de, christopher.harth-kitzerow@tum.de, nina.schwanke@tum.de

***Abstract*—In this paper, we build upon a previous project where we used Homomorphic Encryption (HE) to improve non-linear and linear operations for Secure Two-Party Computation (2PC) and integrated them into a Secure Multi-Party Computation (MPC)-framework. We used beaver triples [1] to put most computation in the offline phase. However, we realized that our algorithms require more Boolean and arithmetic triples than anticipated. Hence, we decided to install further optimizations, especially in the LAN setting. We optimized ReLU, Convolution, and Batch Normalization. We show that our optimizations improve performance, particularly for LAN. On top of that, only about 14.89% of our computation is done online, which significantly improves on the implementations of alternative frameworks, such as SecONNds, which have about 65.77% of the total computation in the online phase.**

***Index Terms*—secure multiparty computation, homomorphic encryption, optimization, benchmark**

## 1. Introduction

Secure Multi-Party Computation (MPC) solves the problem of collaboratively computing any function while also preserving the privacy of the parties' inputs. This is often required for private communication as for instance, in satellite communications [2]. As neural networks have gained widespread adoption, they can be used for private inference as for example, for cancer detection on confidential data. This way, the patients' images and the trained network remain private [3]. In this paper, we examine protocols that use Homomorphic Encryption (HE) for MPC and aim to optimize communication and end-to-end times.

MPC enables multiple parties to collaboratively evaluate a function while preserving the privacy of their respective inputs. This capability is essential in scenarios where sensitive information must be processed jointly without disclosure. For example, MPC can facilitate secure communication and coordination between satellites while protecting confidential operational data [2]. Another important application is privacy-preserving neural network inference, where medical images can be analyzed for tasks such as cancer detection without revealing either the patients' data or the model parameters [3]. In this paper, we investigate MPC protocols based on HE and explore optimizations to reduce communication overhead and end-to-end execution time.

This work extends a prior implementation effort based on OpenCheetah [4], an open-source framework that leverages HE to optimize Secure Two-Party Computation (2PC) operations for MPC. However, we found that additional optimizations were necessary to minimize online execution time while maintaining a reasonable offline time in both LAN and WAN environments. Here, the offline phase refers to all computation that can be done without any input. Only in the online phase do both parties have to provide their inputs.

First, we will give a short introduction to the topic, give an overview of our optimizations, and finally evaluate the results by comparing our implementation against well-established HE-frameworks OpenCheetah and SecONNds

## 2. Background

### 2.1. Secure Two-Party Computation

In general, MPC solves the problem where $n$ parties try to compute a common function, and each party might hold private input. In the special case with only two parties ($n = 2$; commonly abbreviated as 2PC), there are implications that allow for optimizations, as discussed in the following sections. Typical approaches include: a) a trusted third party which is honest and all parties can rely on, b) Oblivious Transfer (OT) (see Section 4.6). However, the latter usually requires more communication for arithmetic operations, as it operates at the bit level [1]. Therefore, HE has recently attracted attention because it reduces communication requirements, but at a higher computational cost, especially in the two-party case, where only one party needs to encrypt their input, while the other can use plaintext-ciphertext operations.

### 2.2. Homomorphic Encryption

Homomorphic Encryption allows for computation on ciphertexts, such as addition or multiplication, usually in a ring modulo a prime or a power of two [4]. Regarding MPC, previous work has primarily focused on Ring Learning with Errors (RLWE) because its operations are cheap, and with appropriately chosen parameters, the error terms can be circumvented [4]. For the operations OpenCheetah implements, the multiplicative depth is low, and as the layers are independent, the error term does not increase and can be neglected [4].

### 2.3. Oblivious Transfer

We use OT to generate Boolean triples for non-linear operations. OT solves the problem where one party has a

 doi: 10.2313/NET-2026-05-1_13

choice bit $b$ and another party holds two messages $m_0, m_1$ in the end the party holding $b$ wants to receive message $m_b$ while the other party must not learn $b$. This must be efficient, as our protocols require several hundred million Boolean triples. OpenCheetah uses the three different implementations, which we briefly discuss in Section 4.6 [4].

### 2.4. High-Throughput MPC

For our implementations, we used the open-source framework HPMPC, which supports various algorithms and schemes for different security settings in MPC [5]. We focus on the 2PC protocols and optimizing the 2D convolutions, Batch Normalization, and ReLU operations. For this, we defined two protocols denoted *AB* and *AB2* (see Appendix A.1). The main difference is that, for *AB2* party $P_0$ does not have shares of input $B$ (the filters for convolution see Sec. A.1), which simplifies the process.

## 3. Related work

This work is based on the projects OpenCheetah [4] and SecONNds [6]. The former is an online-only phase framework that implements several linear and nonlinear operations optimized for 2PC using RLWE. SecONNds optimizes OpenCheetah by using Number Theoretic Transform (NTT) and Random OT (ROT) for ReLU. As a result, they were able to put approximately 20 % of the runtime in the offline phase. We build two 2PC protocols that utilize their algorithms but make use of beaver triples to put most computation in the offline phase.

Other closely related projects include BumbleBee [7] and Secretflow [8], which have further optimized the packing methods by OpenCheetah to reduce the size of ciphertexts relevant for matrix multiplication using HE and Troy-Nova by Liu et al. [9], who implemented SEAL (a HE library by Microsoft) on GPU and implemented the algorithms introduced by OpenCheetah and Bumblebee for 2D-Convolution and matrix-multiplication.

## 4. Optimizations

In this context, we will often refer to the inputs $A$ and $B$ as images and filters, respectively (see Section A.1), since that is how the inputs are interpreted in the context of convolutional neural networks.

### 4.1. Security Definitions

Our protocols are secure in the semi-honest adversary setting (also known as the passive setting), where an adversary executes the protocols as defined but may use any information given to them to derive additional information [1]. We are using HE for 128 bit security, hence, an attacker requires at least an order of $2^{128}$ operations to obtain information [10, 11].

### 4.2. Implementation

OpenCheetah combines serialization, sending, and deserialization. However, this adds superfluous delays to our *AB* protocol (see Section A.1), as it implies that while one party is serializing and sending, the other must wait and can start serializing and sending only after the first party finishes. Profiling has shown that serialization itself can be a bottleneck. Thus, to reduce the receiver's waiting time, we introduce a method that combines sending and receiving by first having both parties serialize their encrypted shares, exchange them, and finally deserialize the received messages simultaneously, thereby minimizing the time each party has to wait.

Additionally, by default, OpenCheetah flushes its packets immediately, which generates unnecessary traffic. Since, when sending a ciphertext, it is also important to include the packet size. Instead of sending the size and shares separately, send them in a single packet. Therefore, we use buffering and only flush when all messages to send have been registered, ensuring everything arrives. This way, we can fill the sending buffers appropriately, maximizing our throughput while minimizing communication.

When performing the benchmarks, we noticed that because OpenCheetah is not perfectly parallelized, it is more efficient to use it in single-threaded mode when introducing a batch size parameter. This is because OpenCheetah assumes the batch size to be equal to one for most operations. This currently comes with the disadvantage that we have to encrypt the input filters multiple times. We note that it is possible to precompute the encryption of the filters once and then share it across all processing threads.

### 4.3. Initial Setup

As we have to close and reopen connections to perform operations when required by HPMPC. To prevent the two frameworks (HPMPC/OpenCheetah) from interfering with each other and to reduce the number of required ports, we initially had to exchange the public keys and seeds required for OT multiple times. To reduce overhead, we introduced a static singleton that, as soon as it is created, exchanges all required keys and random seeds. This is particularly helpful for Correlated OT (COT), as setting it up takes about one second on average. This improved architecture enables us to repeatedly call the Boolean triple Generator with minimal overhead compared to running it once and generating all the required shares in only a single call. For this, we had to implement additional disconnect and connect methods, as the input-output channels by EMP-OT[1] must be consistent and cannot change. This was not an issue for OpenCheetah, as they do not need to free the ports because they do not have two interfering frameworks. We thus minimize the overhead associated with the connection build-up between the two parties.

### 4.4. Convolution

Firstly, we introduce a parameter for the batch size for our protocols that allows us to encode our filters once.

Additionally, we used SecONNds optimizations for Convolution and used NTT to encode the filters. Since we only encode the filters once, we can save time for subsequent batches (see Section 5). This optimization

1. OpenCheetah uses this framework for OT

 doi: 10.2313/NET-2026-05-1_13

also improves the multiplication operation for HE, saving execution time [6](see Section 5).

#### 4.4.1. GPU

Based on the idea of SecONNds, we also added an option to use a GPU version of SEAL. However, instead of using Troy, we implemented Convolutions using Troy-Nova, which is a more modern version of Troy. Similarly to SecONNds, they optimize for batch sizes greater than one using NTT and only encode the filters once.

Additionally, Troy-Nova supports what they call reverse convolutions, which enables us to encrypt the filters rather than the images. This reduces the communication required for multiple batches, as the filters need to be sent only once, whereas, with the other implementation the images must be sent separately for each batch.

Another main difference between our implementation and SecONNds is that we use the GPU for encoding/decoding and encryption/decryption, whereas SecONNds uses SEAL for these operations and only uses the GPU for the convolution itself [6].

### 4.5. Batch Normalization

OpenCheetah has two methods for batch normalization and switches between them based on which has the more compact packing. Their first method is just using elementwise multiplication, while their second method makes some optimizations to reduce the size of the packed ciphertexts. However, when first using this, we used only a single thread for their HPMPC implementation. We decided to split the work onto multiple threads, similar to what OpenCheetah does.

Nonetheless, we also fused subsequent batch normalizations and convolutions into a single convolution call. Therefore, we were able to remove all calls to batch normalization, which has only a minor impact on convolutions but greatly improves runtime and communication (see Section 5).

OpenCheetah also supports fused Batch Normalization (BN). However, they still require a couple of calls to BN (see Section 5).

### 4.6. Arithmetic and Boolean triples

ReLU in MPC consists of two phases: first, solve the millionaire's problem (a < b) and second, multiply the result by the input shares. This fulfills the definition of ReLU (1).

$$\mathrm{ReLU}(x) = \max(x, 0) \tag{1}$$

Initially, we attempted to solve the second phase using HE and implemented it using OpenCheetah's elementwise multiplication, which it uses for BN. However, even after parallelizing this operation, it still took too long in the WAN setting as it requires about 750 MiB for ResNet50. Which itself given a network with a bandwidth of 200 Mbit $s^{-1}$ takes about 52.5 s. Therefore, we decided to use OpenCheetah's COT approach instead, which leverages the optimized OT scheme that utilizes seeded randomness to significantly reduce communication [12]. This takes about half a second longer to compute but drastically reduces communication (see Section 5).

The first phase, however, requires numerous Boolean shares. OpenCheetah supports three schemes to generate these. The first uses ROT. However, this approach, uses random data without taking any input, which makes it hard to use for us, since we have given input shares. A similar problem arises with their second approach, which uses 1-out-of-8 OT, but it also changes the input values. Thus, the only approach we can use without making changes to HPMPC is to use their 1-out-of-16 OT scheme. This requires more communication as it does not utilize COT. However, to further optimize it, we simply spread the workload over multiple threads, which initially did not improve our runtime, as we assumed we would require fewer shares.

### 4.7. LAN and WAN

First of all, unlike previous work, all our communication is spread across multiple ports and threads. Thus, the ciphertexts can be quickly shared, serialized, and deserialized.

To maximize the time on the CPU and reduce time spent waiting for packets to arrive, we implemented a thread-safe queue that allows use to to have two separate threads for sending and receiving the ciphertexts. Here, the send thread waits for the computing threads to encrypt a layer, then pushes them to the send thread, which then sends the number of ciphertexts, the total package size, and all the ciphertexts. The receive thread then receives the packages, deserializes them, and pushes them onto the receive queue for the computing threads to pull from as needed. By using thread-safe queues and using only one thread for sending and receiving, respectively, we ensure that the order in which we send/receive packets does not change without implementing a more complex mechanism. This enables the computing threads to run continuously without waiting for all packets to arrive, but only the ones they need for the next step, which is critical in slower networks or over WAN.

Finally, we added an option to execute only the offline phase and store all generated triples in a file, which can then be used to compute the online phase later. We achieved this by implementing a first-in, first-out queue, which allows us to store multiple offline runs in advance. However, this has not been optimized for multiple runs and may require significant copying when reading the data, as the remaining triples for subsequent runs must be moved to the front after the first set of triples is read and erased. A more efficient way to solve this could be to store a pointer to the start of unread data and delete the file once every byte has been read.

## 5. Evaluation

We evaluate our implementation by comparing it to OpenCheetah and SecONNds. All evaluations were distributed and parallelized with 32 threads on the same CPU (AMD EPYC 7543). Additionally, we compare the results in LAN and WAN settings. We used OpenCheetah's *ResNet50* architecture for benchmarking and implemented an equivalent architecture for HPMPC, ensuring that all

 doi: 10.2313/NET-2026-05-1_13

TABLE 1: Data sent by $P_0$ (upper) and $P_1$ (lower) in MiB using the *AB2* protocols.

| | Conv-2D | BN | FC | Triples | Total |
|---|---|---|---|---|---|
| Cheetah | 834.17 | 162.11 | 0.72 | 282.57 | 1279.57 |
| | 167.77 | 189.91 | 1.19 | 282.54 | 451.50 |
| SecONNds | 834.16 | 162.11 | 0.72 | 282.57 | 1279.56 |
| | 167.77 | 189.91 | 1.19 | 282.54 | 451.50 |
| Our impl. (87) | 834.11 | 0 | 0.72 | 378.83 | 1213.66 |
| | 167.77 | 0 | 1.19 | 344.49 | 513.45 |
| Our impl. (187) | 834.11 | 0 | 0.72 | 935.61 | 1770.44 |
| | 167.77 | 0 | 1.19 | 901.80 | 1070.76 |
| Our impl. (287) | 834.11 | 0 | 0.72 | 1741.18 | 2576.01 |
| | 167.77 | 0 | 1.19 | 1707.37 | 1876.33 |

input dimensions are identical. HPMPC also supports different adders here denoted as 87 for ripple carry adder, 187 for parallel prefix adder, and 287 for 4-way parallel prefix adder. For this project, the difference between these approaches lies in the number of triples they require, and thus, less communication may be required in the online phase. To evaluate our implementation, we will consider only the server-side perspective ($P_0$), since the client-side runtimes are nearly identical, as they have to wait for the results of $P_0$ and would complicate our visualization.

## 5.1. Communication

Table 1 shows how much data the server and client have to send to one another ($P_0$ upper value and $P_1$ lower value). The most interesting operation for communication is ReLU, as its implementation differs significantly from that of other frameworks. We see that using a ripple-carry adder results in the least communication, as it requires the fewest Boolean triples. For all frameworks, we utilized the available option to fuse batch normalization with convolution operations. Nonetheless, OpenCheetah and SecONNds still required certain batch normalization steps. Additionally, we excluded the communication required for exchanging public keys and the data necessary for COT.

We note that the numbers provided for our protocols do not include the online-phase communication, but only the data required for the operations defined in the previous sections. The parties send about 150-200 MB during the online phase.

For our *AB* protocol, one must sum up the data sent by $P_0$ and $P_1$ for FC, BN and 2D-convolutions to determine the total amount each party must send, as both parties now perform the same computation and communication.

## 5.2. Local Area Network

For LAN, we averaged the runtimes over 10 interferences and display the averages as well as the minimum and maximum values.

Figure 1 compares the end-to-end times. Note that SecONNds does not support optimization to precompute triples in the offline phase for bit length 32 and ResNet50. Hence, the offline time here is just the filter encoding using NTT. This way, about 65.77% of the total computation is done in the online phase. Although our approach with the 4-way parallel-prefix-adder achieves the best online time ($\approx 4.86\%$ online phase), its offline phase lags behind because it requires the most triples and thus has the worst

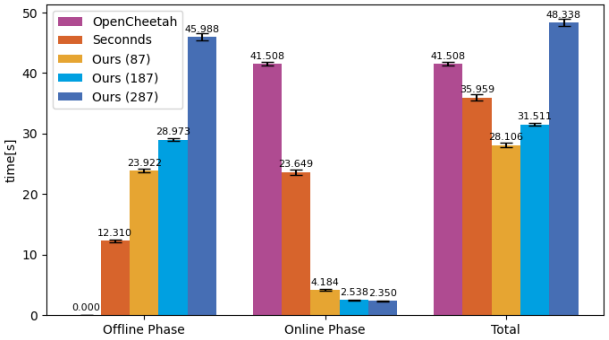



Figure 1: Online and offline times for the different frameworks in seconds (includes NTT).

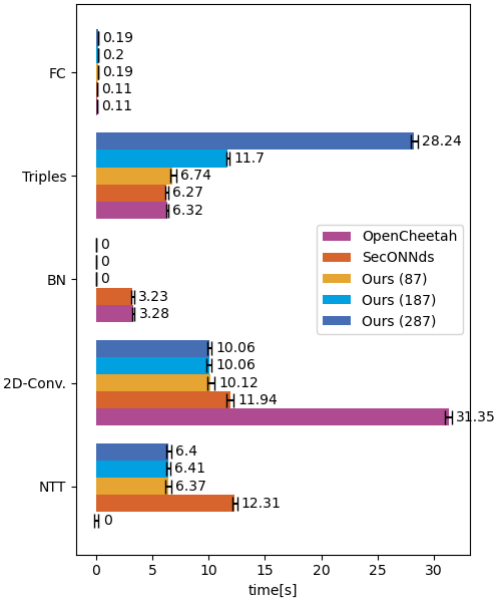



Figure 2: Comparing the runtime of different operations.

end-to-end time. The best overall time is achieved by using the ripple-carry adder approach. Here, only about 14.89% is done in the online phase.

Figure 2 illustrates how the operations perform across different frameworks.

We see that SecONNds and OpenCheetah still require some seconds for batch normalization. For fully connected layers, our implementation is slower by a couple of milliseconds, because we have to perform several conversions between 32-bit and 64-bit integers and also re-establish the connection between the two parties. Our three approaches primarily differ in the number of Boolean triples required and therefore, the time required to generate them. Finally, when it comes to convolution, we save about 6 s for NTT and only about 1s for the computation. This seems strange as we have not modified the NTT-conversation at all. We suspect this may be due to measurement inaccuracies, as by default we do not distinguish between the NTT-phase and the computation phase. To investigate this, we increased the batch size to see how long convolutions take when there is no need for encoding the filters. With batch size set to two convolutions we measured about 6.82 s for NTT and about 23.35 s for the computation and with batch size set to ten

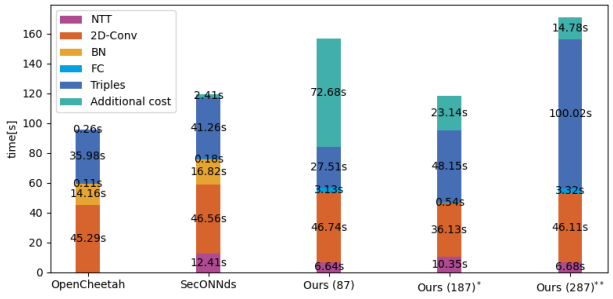



Figure 3: Evaluation of each operation and end-to-end times in the WAN setting (*uses separate send/receive threads; **uses online-phase optimizations).

we still got about 6.5 s for NTT and 128.83 s assuming that the first batch took about 10 s any subsequent batch takes about 13 s.

Finally, when making use of SIMD (vectors of 8 32 bit numbers) and parallelization via 32 processes, the last process takes about 1863.49 s, thus for an overall batchsize of 256 (8 (because of SIMD) $\cdot$ 32 (number of processes) $= 256$), each inference takes about $\frac{1863.49\,\mathrm{s}}{256} = 7.28\,\mathrm{s}$. This shows that utilizing parallelization improves the end-to-end times.

### 5.3. Wide Area Network

For these experiments, we set the network to a bandwidth of 200 Mbit s$^{-1}$ and a latency of 20 ms.

Fig. 3 shows the runtime for each operation for ResNet50. With *"Additional Cost"*, we refer to the time required for additional operations beyond the operations. This way, we can properly compare the end-to-end times. In our implementations, this also includes the online phase. We observe that our implementation with the ripple-carry adder, although it takes only approximately 90 s in its offline phase, requires about the same time in the online phase, diminishing the benefit. We notice that simple operations, such as FC, take slightly longer because we have to reestablish a connection between 32 threads, which in itself takes about 2 s but for function 187 we used a updated version where connections are established in parallel and reduced the only the connections required as for example FC only requires one port.

To (287) we applied some online phase optimizations outside of the scope of this paper. This can also be applied to (187) for only a minor performance loss in the offline phase of about 1.62 s but reduces the online-phase to about 21.49 s).

Finally, our results for the optimized version as introduced in Section 4.7, reduce the total time required for convolutions to about 46 s where the first $\approx 10$ s are used for NTT. In total, we arrive at $\approx 102$ s in the offline phase (including the key exchange $\approx 5.91$ s). This greatly improves upon the work of SecONNds who require approx. 58.97 s and is about even with OpenCheetah (see Fig. 3). However, since NTT is only required for the first batch, our implementation has the edge when the main bottleneck is network speed. Nonetheless, this has no impact on our online phase, which adds an additional 23.14 s (using the parallel-prefix-adder). Note that for our function 187 in Fig. 3 we used this version, whereas for 87 and 287 we used to sequential unoptimized implementation.

## 6. Conclusion and future work

We showed that using HE can be beneficial for more complex operations, and that most CPU-intensive operations can be performed in the offline phase using beaver triples. This way, the online phase accounts for only about 14.89 % of the total computation.

However, common approaches still need to be optimized, as the ciphertext quickly grows with the number of operations and is strongly dependent on the packing. Therefore, when creating an algorithm, one has to avoid sparse packing. Otherwise, the communication increases, but using less sparse encoding might require more operations, such as error correction or multiplications. Also, secret-sharing approaches become more efficient, as schemes like COT greatly reduce the required communication [12]. However, projects such as BumbleBee show that packing can still be optimized [7].

## List of Acronyms

| | |
|---|---|
| **2PC** | Secure Two-Party Computation |
| **BN** | Batch Normalization |
| **COT** | Correlated OT |
| **CPU** | Central Processing Unit |
| **FC** | Fully Connected Layer |
| **GPU** | Graphics Processing Unit |
| **HE** | Homomorphic Encryption |
| **HPMPC** | High-Throughput MPC |
| **LAN** | Local Area Network |
| **MPC** | Secure Multi-Party Computation |
| **NTT** | Number Theoretic Transform |
| **OT** | Oblivious Transfer |
| **ReLU** | Rectified Linear Unit |
| **RLWE** | Ring Learning with Errors |
| **ROT** | Random OT |
| **SEAL** | Simple Encrypted Arithmetic Library |
| **SecONNds** | Secure Outsourced Neural Network Interference |
| **SIMD** | Single Instruction, Multiple Data |
| **WAN** | Wide Area Network |

## References

[1] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2.0: Improved Mixed-Protocol secure Two-Party computation," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 2165–2182. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/patra

[2] C. Fedele, K. Butler, C. Petersen, and T. Lovelly, *Protecting Satellite Proximity Operations via Secure Multi-Party Computation*, 2024. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2024-0271

[3] Y. Son, K. Han, Y. S. Lee, J. Yu, Y.-H. Im, and S.-Y. Shin, "Privacy-preserving breast cancer recurrence prediction based on homomorphic encryption and secure two party computation," *PLoS ONE*, vol. 16, no. 12, p. e0260681, 2021.

[4] Z. Huang, W. jie Lu, C. Hong, and J. Ding, "Cheetah: Lean and fast secure Two-Party deep neural network inference," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 809–826. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/huang-zhicong

[5] C. Harth-Kitzerow, A. Suresh, Y. Wang, H. Yalame, G. Carle, and M. Annavaram, "High-throughput secure multiparty computation with an honest majority in various network settings," *Proceedings on Privacy Enhancing Technologies*, vol. 2025, no. 1, pp. 250–272, 2025.

[6] S. Balla, "SecONNds: Secure outsourced neural network inference on imagenet," 2025. [Online]. Available: https://arxiv.org/abs/2506.11586

[7] W. Lu, Z. Huang, Z. Gu, J. Li, J. Liu, C. Hong, K. Ren, T. Wei, and W. Chen, "BumbleBee: Secure two-party inference framework for large transformers," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/bumblebee-secure-two-party-inference-framework-for-large-transformers/

[8] J. Ma, Y. Zheng, J. Feng, D. Zhao, H. Wu, W. Fang, J. Tan, C. Yu, B. Zhang, and L. Wang, "SecretFlow-SPU: A performant and User-Friendly framework for Privacy-Preserving machine learning," in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. Boston, MA: USENIX Association, Jul. 2023, pp. 17–33. [Online]. Available: https://www.usenix.org/conference/atc23/presentation/ma

[9] X. Liu, Z. Liu, Q. L. 0002, K. X. 0002, and M. Xu, "Pencil: Private and extensible collaborative learning without the non-colluding assumption," in *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*. The Internet Society, 2024. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/pencil-private-and-extensible-collaborative-learning-without-the-non-colluding-assumption/

[10] E. Barker, "Recommendation for key management: Part 1 - general," National Institute of Standards and Technology (NIST), Tech. Rep. SP 800-57 Part 1 Rev. 5, May 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-57pt1r5

[11] "Microsoft SEAL (release 4.1)," https://github.com/Microsoft/SEAL, Jan. 2023, microsoft Research, Redmond, WA.

[12] K. Yang, C. Weng, X. Lan, J. Zhang, and X. Wang, "Ferret: Fast extension for correlated ot with small communication," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1607–1626. [Online]. Available: https://doi.org/10.1145/3372297.3417276

# Appendix

## A.1 Our protocols

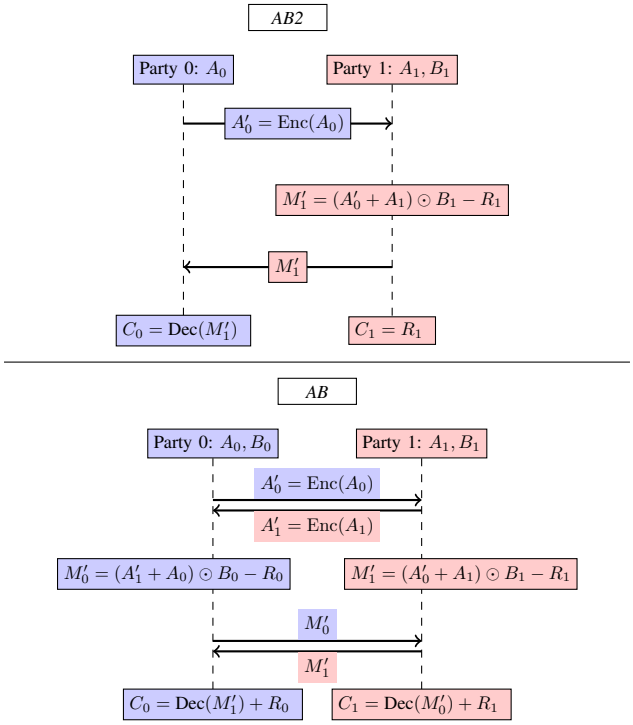



Figure 4: Protocols for beaver triple generation

Our protocols for beaver triple generation are depicted in Fig. 4. Red represents all operations done by party $P_1$ and blue, the operations by party $P_0$ has to to do and all encrypted values are denoted with a single quote. Note, that *AB* generates the standard beaver triples, whereas *AB2* assumes only one party has the filters. The resulting triple $A_p, B_p, C_p \quad \text{for } p \in \{0, 1\}$ fulfill Equation (2).

$$(C_0 + C_1) = (A_0 + A_1) \odot (B_0 + B_1) \tag{2}$$

# Usage Statistics for a Scientific Testbed

Marcel Kornegger, Kilian Holzinger*, Sebastian Gallenmüller*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: marcel.kornegger@tum.de, holzinger@net.in.tum.de, gallenmu@net.in.tum.de

***Abstract*—The plain orchestrating service (pos) has been developed to manage and orchestrate experiments on various scientific testbeds. While pos provides a structured experiment workflow and automation features to enhance reproducibility, it currently lacks the capability to analyze historical testbed usage. For testbed operators, it would be particularly useful to know to what extent individual nodes are being utilized.**

**To address this limitation, we present the design and implementation of a dedicated usage statistics system for pos. This system leverages the existing calendar-based reservation system to generate approximated node utilization statistics based on historical booking data. The primary purpose of the tool is the generation of usage statistics. In addition, it enables users to further filter and aggregate these statistics based on various criteria, such as access groups and time ranges.**

**We demonstrate the applicability of our tool through a case study on the *Blockchain* testbed, showcasing its ability to provide insights into resource usage.**

***Index Terms*—usage statistics, testbeds, pos**

## 1. Introduction

Scientific testbeds play a crucial role in advancing research in computer networking and distributed systems by providing researchers with access to infrastructure to create reproducible experiments [1]. For this purpose, the plain orchestrating service (pos) has been developed at the *Chair of Network Architectures and Services* to manage and orchestrate experiments on various testbeds [2]. Pos enables a structured experiment workflow with fully scripted experiments and support for automated collection and publication of experiment artifacts [2]. By design, this approach reduces manual work and minimizes sources of nondeterminism. Therefore, pos enhances the reproducibility and repeatability of experiments.

Pos currently lacks the capability to analyze historical testbed usage. In particular, pos does not offer statistics on how test nodes are utilized over time. However, this information would be beneficial for testbed operators. For example, in heterogeneous testbeds with different types of nodes, such as GPU-equipped machines, usage statistics can help to identify popular resources and usage patterns. These insights help operators make informed decisions regarding resource allocation and future infrastructure investments.

In this paper, we present the design and implementation of a usage statistics system for the plain orchestrating service to address this issue. Our proposed system leverages the existing calendar-based reservation system of pos to approximate node utilization statistics based on historical booking data. The implemented tool allows users to filter and aggregate usage statistics based on various criteria, such as access groups and time ranges. In addition, we demonstrate the applicability of our tool through a case study on a testbed of the *Chair of Network Architectures and Services*, showcasing its ability to offer resource usage insights.

The remainder of this paper is structured as follows. In Section 2, we investigate related work on usage statistics for scientific testbeds. Section 3 provides background information on the plain orchestrating service (pos). In Section 4, we present the design of our usage statistics system for pos, followed by implementation details in Section 5. We demonstrate our implementation in a case study in Section 6 and conclude the paper with an outlook on future work in Section 7.

## 2. Related Work

To contextualize our work, we review related scientific testbeds and their approach to monitoring and publishing resource utilization. Our analysis is based on publicly available information, as we do not have access to internal usage statistics or monitoring systems of these testbeds.

Several scientific testbeds for research in computer networking and distributed systems have been developed, e.g., Emulab [3], CloudLab [4], Grid'5000 [5], Chameleon [6], FABRIC [7], PlanetLab [8] and Fed4FIRE+ [9]. These differ in their targeted research domain, provided resources, architecture, and user communities [1], [10]. Available usage statistics of academic testbeds typically fall into two categories: (i) on-demand statistics available through a user interface [11]–[14], and (ii) published statistics that appear in papers or reports but are not accessible in real time [4], [15], [16]. On-demand statistics are either generated upon request or continuously updated, thereby providing a real-time view of current and historical resource utilization.

Emulab offers node usage statistics publicly via a web interface [11]. This includes graphs indicating the average number of free nodes, clustered by different node types, with historical data over both short- and long-term periods. CloudLab provides basic live statistics about the number of active experiments and available nodes on its website [12]. Historical usage statistics were published in the past but are not available on demand [4]. Grid'5000 has a web interface for usage statistics. However, it is not publicly

 doi: 10.2313/NET-2026-05-1_14

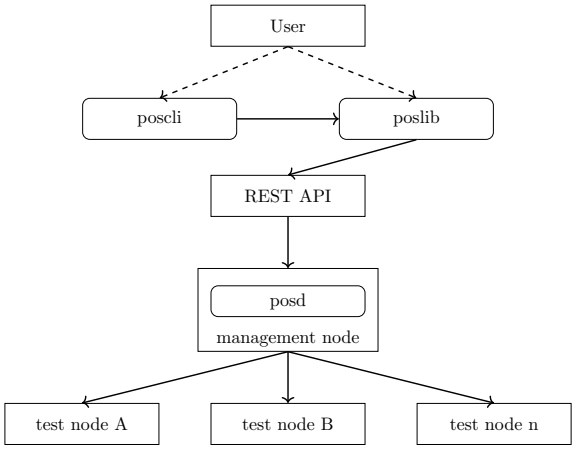



Figure 1: Overview of pos testbed architecture.

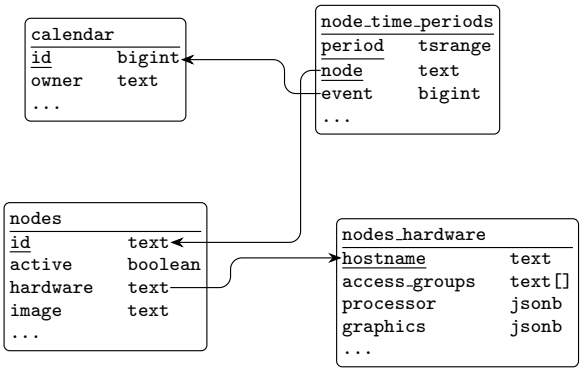



Figure 2: Relevant tables of pos database schema.

available [13]. FABRIC offers a dashboard providing an overview of the currently available resources but does not include any historical usage data [14]. Fed4FIRE+ published usage statistics about users and experiments in its EU project deliverables [15], but did not offer a usage interface at the federation level. Chameleon has published statistics, such as available nodes and used node-hours over time, but does not provide on-demand statistics publicly [16].

## 3. Background

The testbeds of the *Chair of Network Architectures and Services* are powered by the plain orchestrating service (pos). In the following, we give a brief overview of pos, its workflows, and its architecture, mainly based on the documentation in [17].

Figure 1 shows an overview of the architecture of a pos-based testbed. A testbed consists of multiple physical machines, called test nodes, which are connected to a management node via a dedicated management network. The management node runs the pos daemon (posd), which is responsible for managing the test nodes, orchestrating experiments, and collecting experiment data.

**Typical experiment workflow.** In order to run an experiment on a pos-based testbed, a user must first create a calendar entry that reserves the required test nodes for a specific time period. The calendar system is an optional feature, but it is used in all testbeds of the chair. Within this time period, an allocation has to be created to gain access to the reserved test nodes. Then, an experiment can be conducted by configuring image, variables and scripts for the test nodes. After the experiment has finished, the user can optionally free the allocation. Users may intentionally leave the allocation active, allowing the experiment to run on a best-effort basis until someone else frees it.

**Daemon.** The pos daemon is the core component of a pos-based testbed. The daemon maintains the state of the testbed and manages test nodes, calendar bookings, allocations, users and more. The daemon exposes a REST API to allow users to interact with the testbed.

**Database.** The daemon stores persistent state of the testbed in its PostgreSQL database. This includes test nodes, users, calendar, allocations and other relevant information. The database schema reflects the operational aspects of the testbed. Figure 2 shows an excerpt of the database schema with the relevant tables for our matter. Underlined attributes indicate primary keys and arrows indicate foreign key relationships. The `nodes` table contains information about all test nodes, such as the currently running image. Additionally, nodes can be deactivated using the `active` flag. For each test node, more specific details about the hardware of a node is stored in the `nodes_hardware` table. This table also includes the access groups of a node, allowing access to nodes to be restricted to a certain user group. For example, access to the GPU-equipped machines of the chair requires the user to have the corresponding access group. When a calendar booking is created, an event is recorded in the `calendar` table, while the `node_time_periods` table links calendar events to test nodes and the reserved time period.

**Interacting with a testbed.** As seen in Figure 1, users can interact with a testbed using different tools. Through the REST API of the daemon, users can create calendar bookings, allocate nodes, and conduct experiments. The poslib Python library provides a high-level abstraction of this API. The poscli command-line tool is built on top of poslib and offers commands to interact with the testbed from the terminal. In addition, pos provides a web interface for users to manage their calendar bookings.

## 4. Design

In this section, we present the design of our usage statistics system for pos.

**Requirements.** To guide the design of the usage statistics system for pos, we derive the following requirements.
*Hardware-aware aggregation.* The system must support clustering and aggregation of usage statistics based on node hardware characteristics (e.g., GPU-equipped nodes) in order to enable informed resource planning.
*Configurable time ranges.* The system must provide utilization statistics for configurable time intervals.
*Fine-grained filtering.* The system must allow filtering based on access groups and node IDs to enable focused analysis of specific subsets of the testbed.
*Absolute and relative metrics.* The system must report utilization both in absolute terms (e.g., hours used) and relative terms (e.g., share of total usage).

**Statistics generation.** In order to assess node usage, we can leverage the calendar system of pos, which is

 doi: 10.2313/NET-2026-05-1_14

used to reserve nodes for experiments. A calendar booking does not directly imply that the reserved nodes are actively used in this time period. In practice, however, users typically create calendar bookings only when they intend to conduct experiments, making bookings a reasonable proxy for node usage. Therefore, we can use calendar bookings as a proxy for node usage. To generate usage statistics, we use the `node_time_periods` table of the pos database (cf. Figure 2), which contains the reserved time periods for each node. In combination with the `nodes` and `nodes_hardware` tables, we can filter the usage statistics based on node IDs and access groups. To support clustering based on different node hardware, the access group is used as a proxy: Some access groups correspond to hardware features, e.g., GPU-equipped nodes belong to a special access group. We cannot rely on the actual hardware columns(e.g., `processor` or `graphics`) because they are optional and are not consistently populated with data. The `calendar` table would only be needed if we wanted to filter based on user IDs, which is currently out of scope and can be addressed in the future.

**Software architecture.** To ensure consistency with the existing pos architecture and maintain a modular system design, the usage statistics functionality is integrated following the established action grouping logic. In the existing pos architecture, all available actions are logically grouped. Each group of actions has a dedicated endpoint handler in posd, exposing functionality via the REST API. If a group of actions requires persistent data storage, the corresponding database queries are implemented in a dedicated database handler. In poscli, each group of actions has a corresponding command module, which implements a new subcommand for the poscli command-line tool. Following this architecture, we introduce a new endpoint handler in posd for statistics-related actions with the route `/stats/node_utilization`. This handler implements the logic to process incoming requests, validate parameters, and call the database handler to retrieve the required data. To start off with the usage statistics, we want to make integration as easy as possible. Therefore, we do not create new tables for statistics and no new database handler is required. Instead, we implement the query logic in the existing database handler for the `nodes` table. In poscli, we implement a new command module `stats` with a subcommand `node_utilization` to allow users to request usage statistics from the command line. The command accepts parameters for time range, clustering, and filtering options.

# 5. Implementation

In the following, we describe the implementation of our usage statistics system for pos, following the design presented in Section 4.

## 5.1. REST API

To maintain architectural consistency and separation of concerns, we introduce a new endpoint handler in posd for statistics-related actions covering the `/stats/` routes. The handler is implemented in the `StatsRouteHandler` class, which extends the `EndpointRouteHandler` class to reuse existing request validation and response handling logic. The new route `/stats/node_utilization` returns usages statistics via a GET request as the operation is read-only and stateless. The endpoint accepts the following query parameters:

- `start_date` and `end_date` (required): Define the time range for the statistics.
- `filter_nodes`: Only include the defined node ids in the statistics.
- `filter_access_groups`: Only include nodes belonging to the defined access groups in the statistics.
- `group_by_access_groups`: Group and aggregate results by access groups.

The handler processes incoming requests, validates parameters and calls the `NodesDBHandler` database handler to retrieve the required data. The API returns the usage statistics in a structured JSON format, including absolute usage in booked hours and relative usage for each node id. If grouping by access groups is requested, the handler returns statistics for each access group instead of individual nodes. Similar to other testbeds (cf. Section 2), usage statistics are computed on-demand when the endpoint is called.

## 5.2. Database Query

The query logic is implemented in the existing `NodesDBHandler` database handler, which is responsible for queries related to the `nodes` table. We use parameterized SQL queries and structure the query with Common Table Expressions (CTEs) for better readability and maintainability. The query is structured into the following parts: (i) Join required tables, (ii) Filtering, (iii) Aggregation and (iv) Calculation of relative usage. We describe the subqueries in a simplified manner, leaving out type casts and null checks for clarity.

Listing 1: SQL subquery (i): Join required tables

```
1  WITH base AS (
2      SELECT n.id AS node_id, n.active, hw.access_groups,
        ntp.period,
3      CASE WHEN :GROUP_BY_AG THEN ag.access_group ELSE n.id
        END AS id
4      FROM nodes AS n
5      LEFT JOIN nodes_hardware AS hw ON n.hardware = hw.
        hostname
6
7      LEFT JOIN LATERAL (
8          SELECT unnest(hw.access_groups) AS access_group
9      ) AS ag ON :GROUP_BY_AG = true
10
11     LEFT JOIN node_time_periods AS ntp ON ntp.node = n.id
12 )
```

In Listing 1, we join the required tables based on their foreign key relations to retrieve the required data for the statistics. The resulting `id` column contains the final ID which is either the node ID or the access group, depending on whether grouping by access groups is requested (L3). Since `access_groups` is an array, we use a lateral join with the `unnest()` function to create a row for each access group of a node (L7–L9), if grouping is requested.

 doi: 10.2313/NET-2026-05-1_14

Listing 2: SQL subquery (ii): Filtering

```
1  WITH filtered AS (
2      SELECT node_id, id, period FROM base
3      WHERE active = true
4          AND (:FILTER_NODES IS NULL OR node_id = ANY(:
       FILTER_NODES))
5          AND (:FILTER_AG IS NULL OR access_groups && :
       FILTER_AG)
6          AND (
7              period IS NULL OR period
8              && tsrange(:START_DATE, :END_DATE, '[)')
9          )
10 )
```

In Listing 2, we filter the joined data based on the provided parameters. We only include active nodes (L3) and apply filters for node IDs (L4), access groups (L5) and `period` to be overlapping with the request time range (L6–L9).

Listing 3: SQL subquery (iii): Aggregation

```
1  WITH per_node AS (
2      SELECT id, node_id, COUNT(period) AS booking_count,
3          SUM(EXTRACT(EPOCH FROM (
4              LEAST(upper(period), :END_DATE)
5              - GREATEST(lower(period), :START_DATE)
6          ))) AS booked_seconds
7      FROM filtered
8      GROUP BY id, node_id
9  ),
10
11 per_id AS (
12     SELECT id, SUM(booking_count) AS booking_count, SUM(
       booked_seconds) AS booked_seconds
13     FROM per_node
14     GROUP BY id
15 )
```

In Listing 3, we first aggregate the filtered data per node ID (L1–L11) to calculate the number of bookings and total booked seconds within the specified time range. We use the `LEAST` and `GREATEST` functions to correctly handle bookings that partially overlap with the requested time range (L4–L5). In the second CTE (L11–L15), we further aggregate the data per final ID (either node ID or access group) to get the total bookings and booked seconds.

Listing 4: SQL subquery (iv): Calculation of relative usage

```
1  SELECT id, booking_count,
2      ROUND((booked_seconds / 3600), 4) AS booked_hours,
3      ROUND(booked_seconds / SUM(booked_seconds), 4) AS
       relative_utilization
4  FROM per_id;
```

In Listing 4, we calculate the final usage statistics by converting booked seconds to booked hours (L2) and calculating the relative utilization as the fraction of booked seconds over the total booked seconds (L3). The final result includes the ID, booking count, booked hours and relative utilization for each node ID or access group.

### 5.3. Command Line Interface

In poscli, we implement a new subcommand `node_utilization` of the `stats` command module using the Click library [18], which allows users to request usage statistics from the command line. The command accepts the following parameters:

TABLE 1: Top 10 most-used nodes in Blockchain testbed (November 2025)

| Node ID | Bookings | Booked hours | Rel. Utilization |
|---|---|---|---|
| smartcash | 23 | 491.4 | 9.24% |
| ada | 21 | 483.9 | 9.09% |
| chrysos | 8 | 369.0 | 6.94% |
| litecoin | 13 | 251.8 | 4.73% |
| litecoincash | 4 | 235.7 | 4.43% |
| vmexp1 | 39 | 208.6 | 3.92% |
| bitcoin | 63 | 192.2 | 3.61% |
| idex | 51 | 175.8 | 3.30% |
| amdexp0 | 74 | 165.9 | 3.12% |
| algofi | 41 | 164.8 | 3.10% |
| stoi | 41 | 149.1 | 2.80% |

- `--start-date DATE` and `--end-date DATE`: Define the time range for the statistics (required).
- `--filter-nodes NODES`: Only include the defined node IDs in the statistics.
- `--filter-access-groups ACCESS_GROUPS`: Only include nodes belonging to the defined access groups in the statistics.
- `--group-by-access-groups`: Group and aggregate results by access groups.
- `--order-by FIELD`: Order the results by the specified field.
- `--desc`: Order the results in descending order.
- `--json`: Print results as json. By default, results are printed as a table.

In poslib, a high-level abstraction of the new daemon API route is implemented in the `node_utilization()` method of the `stats` module. This method accepts the same parameters as the API route and is called by the poscli command module to request the statistics from the daemon. Ordering is performed by the poscli command module.

## 6. Case Study

In this section, we present a case study in which we apply the usage statistics to a real-world scenario as a demonstration.

**Setup.** The usage statistics tool is applied to the *Blockchain* testbed operated by the Chair of Network Architectures and Services at TUM [19]. The testbed includes heterogeneous hardware resources, such as special test nodes equipped with GPUs, P4-switches, and compute nodes, as well as multiple access groups. The statistics are generated on-demand from the historical booking data stored in the pos database of the testbed.

**Results.** Using the implemented usage statistics tool, we generate node utilization statistics for the Blockchain testbed for the month of November 2025. Table 1 shows the 10 most-used nodes in terms of booked hours, along with their booking counts and relative utilization. For example, the node `smartcash` was booked for a total of 491.4 hours across 23 bookings, accounting for 9.24% of the total usage in the testbed during this period.

Figure 3 shows the distribution of booking times across the individual access groups, with each segment representing an access group and its booked hours, and its relative utilization. This visualization is based on data

 doi: 10.2313/NET-2026-05-1_14

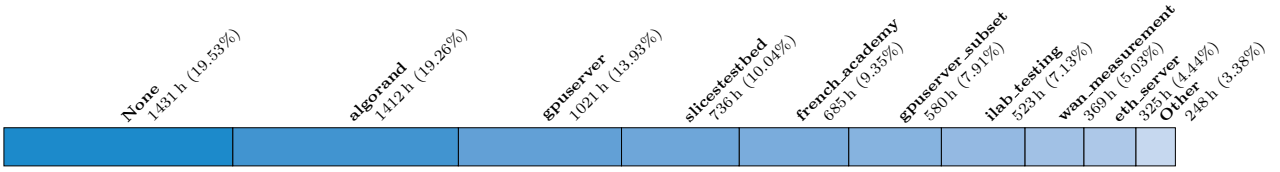



Figure 3: Node usage statistics for the Blockchain testbed, grouped by access groups (November 2025)

generated with the `--group-by-access-groups` option. A node can belong to multiple access groups. In this case, its booked hours contribute to multiple access groups in the aggregated view. The `None` access group represents nodes that do not belong to any access group. We can see that certain access groups, such as `algorand` and `gpuserver`, have a significant share of the total usage, indicating their popularity among users. The `algorand` nodes are the newest test nodes in the testbed and provide higher performance, making them attractive for computationally demanding experiments. Similarly, the `gpuserver` group consists of nodes equipped with GPUs, which are required for workloads involving machine learning, parallel computing, or hardware acceleration.

## 7. Conclusion and Future Work

In this paper, we presented the design and implementation of a usage statistics system for the plain orchestrating service (pos). By leveraging the existing calendar system of pos, we are able to approximate node utilization statistics based on historical booking data. The implemented tool allows users to filter and aggregate usage statistics based on various criteria, such as access groups and time ranges. We demonstrated the applicability of our tool through a case study on the *Blockchain* testbed, showcasing its ability to provide insights into resource usage.

For future work, the usage statistics can be extended in several ways. Further filtering options could be added, for example, based on hardware features or users. A dedicated web interface could be developed to visualize node usage or other statistics. For this purpose, it would be beneficial to pre-compute statistics or cache them, reducing the load on the database for repeated requests.

## References

[1] L. Nussbaum, "Testbeds Support for Reproducible Research," in *Proceedings of Reproducibility'17*. Los Angeles, CA, USA: ACM, Aug. 2017, pp. 24–26.

[2] S. Gallenmüller, D. Scholz, H. Stubbe, and G. Carle, "The pos framework: A methodology and toolchain for reproducible network experiments," in *Proceedings of the 17th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '21. New York, NY, USA: ACM, 2021, pp. 259–266.

[3] F. Hermenier and R. Ricci, "How to Build a Better Testbed: Lessons from a Decade of Network Experiments on Emulab," in *Testbeds and Research Infrastructure. Development of Networks and Communities*, T. Korakis, M. Zink, and M. Ott, Eds. Berlin, Heidelberg: Springer, 2012, vol. 44, pp. 287–304.

[4] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb, A. Akella, K. Wang, G. Ricart, L. Landweber, C. Elliott, M. Zink, E. Cecchet, S. Kar, and P. Mishra, "The Design and Operation of CloudLab," in *Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference*, ser. USENIX ATC '19. USA: USENIX Association, Jul. 2019, pp. 1–14.

[5] "Grid5000," https://www.grid5000.fr/, accessed: 2025-11-30.

[6] "Chameleon," https://chameleoncloud.org/, accessed: 2025-12-02.

[7] "FABRIC Testbed," https://fabric-testbed.net/, accessed: 2025-12-02.

[8] "Planetlab Europe," https://www.planet-lab.eu/, accessed: 2025-12-02.

[9] "Fed4FIRE+," https://www.fed4fire.eu/, accessed: 2025-12-13.

[10] J. Gomez, E. F. Kfoury, J. Crichigno, and G. Srivastava, "A survey on network simulators, emulators, and testbeds used for research and education," *Computer Networks*, vol. 237, p. 110054, Dec. 2023.

[11] "Emulab – testbed node usage stats," https://www.emulab.net/node_usage/, accessed: 2025-11-30.

[12] "CloudLab," https://cloudlab.us/, accessed: 2025-11-30.

[13] "Grid5000 – status," https://www.grid5000.fr/w/Status, accessed: 2025-11-30.

[14] "FABRIC – resources overview," https://portal.fabric-testbed.net/resources/overview, accessed: 2025-12-02.

[15] "Fed4FIRE+ – D2.7: Federation Operations, Tools and Support," https://www.fed4fire.eu/wp-content/uploads/sites/7/d2-07-federation-operations-tools-and-support-update-1.pdf, 2020, accessed: 2025-12-13.

[16] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, and J. Stubbs, "Lessons learned from the chameleon testbed," in *Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference*, ser. USENIX ATC'20. USA: USENIX Association, 2020.

[17] "Plain Orchestrating Service (pos) – Documentation," https://i8-testbeds.pages.gitlab.lrz.de/pos/cli, accessed: 2025-11-15.

[18] "Click – Python Library – Documentation," https://click.palletsprojects.com/, accessed: 2025-12-17.

[19] "i8 – Blockchain Testbed – Wiki," https://gitlab.lrz.de/I8-testbeds/wiki/-/wikis/testbeds/blockchain, accessed: 2025-12-11.

# Automated Network Management for Large Networks

Nguyen Le, Florian Wiedner*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: nguyen.le@tum.de, wiedner@net.in.tum.de

***Abstract*—The management of large-scale networks remains complex and largely manual despite advances in network management solutions. This paper examines the key challenges in automating network management at scale, focusing on complexity arising from heterogeneous and legacy environments, the lack of reliable pre-deployment validation for configuration changes, and scalability challenges related to telemetry overhead and bandwidth fairness. It then assesses established approaches, namely Software-Defined Networking and script-based automation, and discusses how they reduce operator effort and improve traffic control. Next, the paper analyzes emerging paradigms, namely Network Digital Twin (NDT) and Intent-Based Networking (IBN), as promising directions for addressing the remaining challenges. NDT enables safer pre-deployment validation and can support predictive analysis. IBN raises the abstraction level of operations by translating high-level intents into device-level configurations and supporting closed-loop automation. Building on this comparison, the paper argues that combining complementary paradigms can address a broader set of challenges than any single approach in isolation. Overall, the analysis indicates that all challenges identified in this work, except the lack of standardized, vendor-agnostic data models, are being actively addressed by current research and approaches discussed.**

***Index Terms*—Network management automation, large-scale networks, network digital twin, intent-based networking, software-defined networking**

## 1. Introduction

Despite advances in network management tools, operating large-scale networks remains largely manual. As networks grow in size and heterogeneity (i.e., diversity across device types, vendors and application requirements), this manual operation leads to inefficiency and scalability challenges. Collecting and storing operational data at scale can require extremely large storage [1]. At the same time, legacy infrastructure often lacks the interfaces required for automation [2]. Multiple management systems coexist, each tied to a specific management protocol [3], and effective automation requires deep expertise in both networking and the tools themselves [4].

These limitations have motivated ongoing research into more advanced network management paradigms that aim to reduce manual effort, improve scalability and efficiency. This paper begins by identifying the key challenges associated with automating network management in large-scale networks. It then systematically analyzes widely adopted network management approaches, including Software-Defined Networking (SDN) and script-based automation (SBA), highlighting their limitations and the key challenges they aim to address. Subsequently, emerging paradigms such as Network Digital Twin (NDT) and Intent-Based Networking (IBN) are discussed as promising candidates for addressing the unresolved challenges.

Building on this analysis, the paper then examines how hybrid combinations of these approaches can help resolve key management challenges more effectively. Selected research works are also reviewed to illustrate potential directions for addressing the remaining challenges identified in this paper. Overall, the paper aims to assess whether any single approach provides a sufficiently complete solution for large-scale environments, or whether combining multiple paradigms is necessary to achieve scalable and reliable automation.

The remainder of this paper is organized as follows. Section 2 presents the key challenges in automating large-scale networks. Section 3 and Section 4 respectively review established and emerging approaches to network management and evaluate, for each approach, the challenges it addresses and its limitations. Section 5 explores how combining the discussed approaches can mitigate key challenges. Section 6 reviews related work, and Section 7 concludes the paper.

## 2. Challenges

This section identifies the key challenges that motivate the network automation paradigms discussed in the subsequent sections.

**High expertise requirements for network management systems:** Heterogeneous networks typically use different management protocols that are not mutually compatible. Therefore, network operators are required to master multiple network management tools, often at least one for each device type or network domain [3].

**Pre-deployment validation of configuration changes:** In large-scale networks, operators often cannot safely verify the impact of configuration updates (e.g., routing or firewall policy changes) on reachability and performance by virtually testing them in an isolated environment before rollout, so errors may only surface after the change is applied [1], [5].

**Lack of standardized data models within the network:** According to a survey conducted by Enterprise Management Associates (EMA) [2], limited standardization results in a smaller scale of network automation, as

 doi: 10.2313/NET-2026-05-1_15

operators must individually configure each device. For example, inconsistent vendor data representations for inventory and configuration make it difficult to build reusable automation workflows. A network source of truth (NSoT) has emerged as a key concept for enabling consistent and standardized representation of network state and network intent. It serves as a centralized repository that contains all related data that network administrators need for network configuration.

**Limited automation interfaces in legacy infrastructure:** Large-scale enterprise networks are often existing production environments in which older devices constrain automation. In practice, this can mean that vendor automation solutions do not fully support older products, forcing operators to fall back to manual or device-specific workflows and slowing down consistent rollout at scale [2].

**Bandwidth unfairness arising from heterogeneous traffic:** Different traffic classes commonly traverse the same physical medium to maximize bandwidth utilization. For example, surveillance video streams, regular data traffic and multicast discovery traffic often share the same links, which can cause contention when cameras continuously generate high-definition video frames to capture events inside a building [6]. Managing this contention is a network management challenge, since network operators need to define and enforce quality of service (QoS) policies to prevent one traffic class from starving others during congestion [7].

**Collection and storage of data:** With the increasing use of NDT to simulate real-world scenarios, the collection of telemetry plays a significant role, as these network modeling tools utilize machine learning (ML) concepts, such as neural networks (NNs), to generate insights. Moreover, in large-scale data centers, the majority of traffic consists of short flows, and storing fine-grained statistics for all flows could result in significant data volume [1].

For continuous monitoring of a single aggregate (e.g., the percentage of links above 50 % utilization), it is unrealistic to achieve maximum accuracy due to the huge amount of fine-grained statistics that need to be collected and processed. Instead, monitoring protocols could be designed to provide estimates with low overhead, small delay, and a high degree of robustness [8].

## 3. Current Approaches

This section reviews the currently adopted approaches for network management and discusses their limitations, as well as the challenges they aim to address. It focuses on SDN and SBA.

### 3.1. Software-Defined Networking

SDN is an approach that decouples the control plane from the data plane, thereby increasing the programmability of network devices and facilitating dynamic network management through software. The control plane denotes the software logic that provides traffic control functionality, whereas the data plane primarily performs packet switching and forwarding [9].

**3.1.1. Architecture.** The architecture of SDN is illustrated in Fig. 1, consisting of three layers [9].

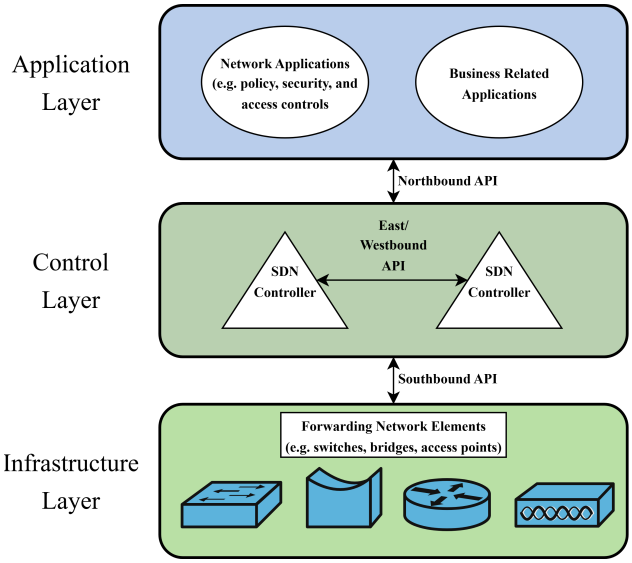



Figure 1: SDN Architecture (adapted from [9])

**Application Layer:** The Application Layer typically contains network applications as well as related business applications [9]. A relevant use case would be replacing a hardware firewall or load balancer with a software-based solution [10].

**Control Layer:** The Control Layer consists of the SDN controller software, which manages policies and supervises packet flow behavior [9]. In comparison, the control layer makes network decisions, while the application layer expresses resource needs [10]. It is a central component that uses three interfaces (northbound, southbound, and east/westbound) to communicate with its peers and the other two layers.

**Infrastructure Layer:** The Infrastructure Layer comprises physical switches or other devices within the network [10]. These devices perform packet switching and forwarding based on the entries inside the flow table. They may be populated in a proactive, reactive, or hybrid manner. When the proactive mode is used, the controller loads the traffic rules that match a particular switch in advance. In the reactive mode, the rules are loaded on demand [11].

**3.1.2. Limitations.** Based on the multi-domain emulation experiments in [9], latency in the legacy scenario is lower than in the SDN scenarios under the tested conditions. In particular, the legacy setup reports latencies mostly below 2.5 ms, whereas the SDN scenarios exhibit minimum latencies of 6.5 ms (SDN with Border Gateway Protocol) and 8 ms (SDN without Border Gateway Protocol), largely due to additional control-plane interaction and overhead from populating flow table entries (e.g., reactive flow setup adds delay for first packets in a flow).

**3.1.3. Addressed Challenges.** By decoupling the control and data planes, network switches become simple packet-forwarding devices, while the control logic is moved to a logically centralized controller, thereby simplifying policy enforcement and network configuration and evolution [11].

SDN provides network-wide data-flow control, enabling administrators to define flows that meet specific requirements of different traffic classes [12]. This facilitates more effective traffic engineering and fairer bandwidth allocation across heterogeneous traffic types.

### 3.2. Script-based automation

SBA utilizes programming languages such as Python, Java, or Bash to automate repetitive networking tasks. A script might rely on protocols such as SSH and SNMP, or use command-line interfaces (CLIs) [13], and utilize available binaries on the remote target device to execute tasks such as configuring interfaces, editing routing tables, and adjusting firewall rules. These scripts are highly customizable and operate at a higher level of abstraction, enabling network professionals to perform ad hoc configuration across multiple devices without having to deal with low-level or platform-specific details.

**3.2.1. Common languages.** As discussed in [14], Python is a powerful language that comes with numerous libraries for network configuration, such as Netmiko or Paramiko. The authors report that automating the configuration of 36 Cisco routers takes, on average, only 6 s when the code includes multithreading and queueing. In contrast, other authors report that performing the same task manually takes 5797 s [15].

Another popular tool among network engineers is Ansible, which Red Hat characterizes as the de facto language for enterprise network automation, with broad adoption across diverse network domains. This is largely due to the extensive ecosystems of certified content and integrations that cover multi-domain and multi-vendor architectures [16].

**3.2.2. Limitations.** Network administrators often use scripts to handle one-off or small automation tasks, which is feasible in small-scale network environments. Nevertheless, once a system scales to hundreds or thousands of servers, the use of scripts without adequate error handling or documentation could lead to system-wide misconfigurations, since such scripts are typically applied uniformly across a large number of machines [13].

**3.2.3. Addressed Challenges.** Script-based approaches can be adopted at low cost using freely available scripting languages and libraries. Some libraries, such as Netmiko and NAPALM (Network Automation Programmability Abstraction Layer with Multivendor support), abstract underlying connection and device-specific details, enabling network administrators to automate tasks without handling low-level protocols directly. As a result, network management tasks can be performed more easily and efficiently [17].

Overall, SDN and SBA can reduce operator effort and improve traffic control, thereby addressing the high expertise requirements and bandwidth unfairness challenges to a significant extent. However, several challenges remain open in large-scale environments. Accordingly, the next section reviews emerging approaches such as NDT and IBN and analyzes which of these remaining challenges they address.

## 4. Emerging Approaches

This section reviews NDT and IBN and evaluates which remaining challenges they address.

### 4.1. Network Digital Twin

A digital twin is a virtual representation of a physical object, system, or process within a digital environment. In this work, an NDT is a virtualized replica of the essential components of the network infrastructure, enabling operators to perform what-if analyses, troubleshoot issues, and plan network upgrades in a safe and controlled environment [1].

The NDT is a data-driven way to represent the current state of the network through network-related data. This data could come from various sources, including real-world traffic and simulated traffic from testbeds or tools [1].

Currently, the market adoption of NDT is mostly limited to simpler capabilities such as policy validation, migration to cloud, or single-vendor what-if analyses. More advanced, autonomous, multi-vendor deployments that support complex what-if scenarios remain mostly aspirational because they are expensive to implement and still lack the ability to reliably predict outcomes in dynamic and complex networks [18].

**4.1.1. Architecture.** At present, there is no standardized definition of the characteristics of an NDT. The IETF draft in [5] proposes five key elements (data, models, mapping, interface, and logic) to characterize it. Fig. 2 illustrates these elements.

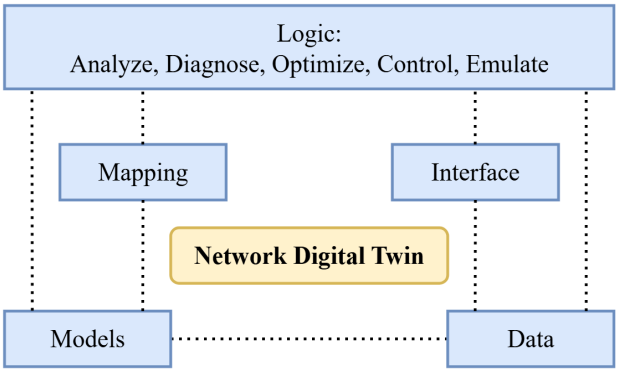



Figure 2: Key Elements of NDT (adapted from [5])

**Data:** Data is considered the single source of truth and is stored in a data repository. It contains historical and/or real-time data (e.g., configuration, metric data) of the real-world twin that are required by the models to understand real-world behaviors.

**Models:** Models enable the emulation of changes in the configuration, state, and resource usage by capturing the behaviour of a real network and generating reasoning information that may be useful in operational decision-making.

**Interfaces:** There are two types of interfaces. The first interface is between the NDT platform and the real network infrastructure, enabling the collection of real-time data. The second interface connects the NDT platform

 doi: 10.2313/NET-2026-05-1_15

to the network management applications (e.g., network optimization, visualization tools [19]), which enables bidirectional communication, exposing numerous platform functionalities to applications, as well as transmitting application requests to the NDT platform.

**Mapping:** Mapping defines the relationship between the NDT and the underlying network entities, enabling synchronization between the real network and its virtual counterpart, or among multiple NDT instances, which is relevant in multi-domain deployments where separate NDT instances are maintained per administrative domain. The observability provided by this mapping allows the NDT to correctly reflect and analyze the state of the real network.

**Logic:** Logic is the decision-making component of the NDT, responsible for tasks such as optimizing resource usage and diagnosing issues based on information produced by the models. Additionally, it is a key component in planning, as it enables a dry run to be performed and anticipates the end result before implementation.

**4.1.2. Addressed Challenges.** NDT enables network operators to perform what-if analyses on the virtualized twin environment without having to do it directly in the production environment, which would be expensive and difficult to reverse [1]. This capability directly addresses the challenge of pre-deployment validation of configuration updates (Section 2). Previously, simulation platforms (e.g., NS-3) were used for such purposes. However, they suffer from the problem of low fidelity and a lack of real-time interaction with the real network [5].

Recent advances in ML are key enablers for NDT, enabling faster execution with lower resource overhead compared to traditional network simulation technologies [1]. Moreover, their strong predictive capabilities enable NDTs to extend beyond reactive service deployment and management, focusing on predictive maintenance (e.g., anticipating link congestion before service degradation occurs) [5].

## 4.2. Intent-based Networking

IBN is a software-based automation paradigm in which network operators specify high-level intents, and the network translates them into a set of configurations that can be understood by the underlying network components, thereby achieving the desired outcome without requiring the low-level execution of individual tasks [20].

Recent progress in Large Language Models has made these models particularly suitable for intent extraction and translation in next-generation networks, thanks to their ability to generate fast and consistent output in text, code, or other forms of writing [21].

**4.2.1. Architecture.** The authors in [22] present a reference three-layered architecture for integrating intent-based technology into existing network management systems (see Fig. 3). The Business Layer captures high-level intents based on Key Performance Indicators that align with the Service Level Agreements objectives. The Intent Layer continuously evaluates the current network state after every execution step based on the available data. The novel aspect of this approach is that all actions are not executed blindly in a single operation; instead, the system dynamically replans after every step [22]. At the lowest layer is the Network Layer, which contains the physical infrastructure and is responsible for translating raw network data into a format comprehensible to the Intent Layer.

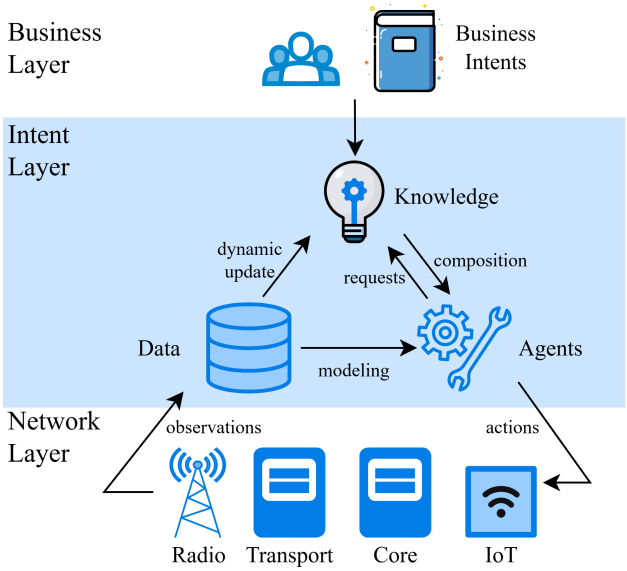



Figure 3: IBN Architecture (adapted from [22])

**4.2.2. Addressed Challenges.** IBN enables network automation with minimal or no human intervention, a concept commonly referred to as Zero-Touch Network and Service Management. By accepting high-level intents as input from network operators, either through voice, text, or any other form of information, the operators do not need to specify the low-level configuration details imperatively [21]. This not only reduces the likelihood of human-induced errors, but also lowers tooling proficiency required to operate large-scale networks reliably.

If an issue appears during a deployment push, network operators can immediately roll back using the previously defined versioned intent specification. Moreover, because the system is aware of the expected outcomes of a configuration, it can still achieve those outcomes even if a commonly used method is unsuccessful, by employing a different mechanism to achieve the same result [20].

Overall, NDT and IBN address additional open challenges in large-scale network management. In particular, NDT supports safer pre-deployment validation by enabling what-if analyses and planning without direct interaction with the production network [1]. IBN further reduces expertise requirements for different management systems by allowing operators to express high-level intents that are automatically translated into device-level configurations [21].

# 5. Hybrid Approaches

This section examines how combining the approaches discussed above helps address key network management challenges.

 doi: 10.2313/NET-2026-05-1_15

### 5.1. Combination of SDN and SBA

Legacy network devices often lack the necessary Application Programming Interfaces to be controlled by SDN systems and only support management through a CLI [23]. In situations where network operating systems do not support RESTCONF or NETCONF, the Python library Netmiko can be used to manage these older devices through SSH [24]. Alongside legacy hardware, organizations could deploy SDN-enabled devices that act as intermediaries between the legacy infrastructure and the SDN controller, allowing the SDN controller to indirectly control legacy devices [23]. This combination helps mitigate the challenge of limited automation interfaces in legacy infrastructure identified in Section 2.

### 5.2. Combination of SDN and IBN

IBN is often associated with SDN and relies on SDN monitoring capabilities to continuously assure that declared intents are met [25]. This combination, together with ML, can achieve a closed-loop automation cycle [25]. The authors in [26] proposed an intent-based SDN network slicing framework that continuously observes the current state of the slices to ensure correct behavior based on the objectives throughout the intent lifecycle.

Intents are also used as the means for defining northbound interfaces of SDN controllers. These intent-driven interfaces reduce the need for deep networking expertise when interacting with SDN controllers and make SDN systems easier to integrate with other systems [27].

### 5.3. Combination of NDT and IBN

When IBN is used, applying the generated configurations directly to a production network carries inherent risk. By using an NDT, intents can be tested and verified before real deployment. For example, Juniper Apstra's integration with Digital Twin Lab is a powerful combination that combines both NDT and IBN to eliminate this gap between testing and production [28].

## 6. Related Work

This section reviews related work that addresses additional open challenges in automated network management, complementing the approaches discussed earlier.

As discussed in Section 2, collecting exhaustive telemetry in large-scale networks does not scale efficiently. Without appropriate pre-processing or edge-based summarization, the resulting data volumes may overwhelm artificial intelligence models [29]. To address this issue, ML inference can be offloaded onto the programmable data plane (PDP), as demonstrated in [30]. This work uses model compression to facilitate convolutional neural network inference on the PDP, achieving 97.3 % accuracy on an anomaly detection task while using only 22.7 % of the SRAM available on the resource-limited Intel Tofino ASIC.

The challenge of bandwidth unfairness arising from heterogeneous traffic has not been solved universally. However, within the context of industrial private 5G networks, whose adoption is increasing as factories evolve toward Industry 5.0, the QoS-Aware Proportional Fairness scheduling algorithm proposed in [31] shows strong potential. The results demonstrate that the algorithm improves fairness while achieving better deadline adherence compared to Maximum Carrier-to-Interference and Static Priority scheduling algorithms.

As discussed in Section 2, a NSoT has been proposed as a solution to address the challenge of standardization. However, the NSoT market remains in its early stages [32]. According to EMA, an NSoT should document network intent data rather than network state data. The actual network state can instead be retrieved from network systems and used to validate alignment between the intended configuration and the operational state [32].

## 7. Conclusion

This paper examined the key challenges in automating network management for large-scale networks, and systematically reviewed both established and emerging approaches that aim to address them. The analysis indicates that all identified challenges, with the exception of the lack of standardized data models within the network, are actively being addressed by current research or by approaches including SDN, SBA, NDT, and IBN. In particular, the emerging approaches such as NDT and IBN introduce new capabilities for pre-deployment validation, predictive maintenance and closed-loop automation.

Evaluating the approaches individually and in combination suggests that no single approach provides a sufficiently complete solution. Instead, hybrid designs are preferable, because each approach targets a different subset of challenges.

Future work should focus in particular on advancing NDT toward more autonomous, multi-vendor deployments for complex what-if analyses. In addition, further research on standardized, vendor-agnostic data models is needed, since such models are a key enabler of a higher degree of automation in large-scale networks.

## References


[1] P. Almasan, M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, D. Perino, D. López, A. A. P. Perales, P. Harvey, L. Ciavaglia, L. Wong, V. Ram, S. Xiao, X. Shi, X. Cheng, A. Cabellos-Aparicio, and P. Barlet-Ros, "Network digital twin: Context, enabling technologies, and opportunities," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 22–27, 2022.

[2] S. McGillicuddy, "Enterprise network automation: Emerging from the dark ages and reaching toward netdevops," 2024, available online at https://efficientip.com/wp-content/uploads/2024/04/EMA7235-Network-Automation-RR-Summary-EfficientIP.pdf; last accessed on 2026/02/22.

[3] R. Hassan, R. Razali, S. Mohseni, O. Mohamad, and Z. Ismail, "Architecture of network management tools for heterogeneous system," 2010. [Online]. Available: https://arxiv.org/abs/1001.1967

[4] U. H. Rao and S. Mohapatra, "Deploying network management solutions in enterprises," in *INC2010: 6th International Conference on Networked Computing*, 2010, pp. 1–6.

[5] C. Zhou, H. Yang, X. Duan, D. Lopez, A. Pastor, Q. Wu, M. Boucadair, and C. Jacquenet, "Network Digital Twin: Concepts and Reference Architecture," Internet Engineering Task Force, Internet-Draft draft-irtf-nmrg-network-digital-twin-arch-11, Jul. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/11/

[6] T. Shanmugam and B. Malarkodi, "Analysis of campus network management challenges and solutions," in *2019 TEQIP III Sponsored International Conference on Microwave Integrated Circuits, Photonics and Wireless Networks (IMICPW)*, 2019, pp. 312–316.

[7] Cisco Systems, Inc., "Qos modular qos command-line interface configuration guide, cisco ios xe fuji 16.8.x," 2018, available online at https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_mqc/configuration/xe-16-8/qos-mqc-xe-16-8-book/qos-scheduling.html; last accessed on 2026/02/22.

[8] A. Pras, J. Schönwälder, M. Burgess, O. Festor, G. Martinez Perez, R. Stadler, and B. Stiller, "Key research challenges in network management," *Communications Magazine, IEEE*, vol. 45, pp. 104 – 110, 11 2007.

[9] F. X. A. Wibowo and M. A. Gregory, "Software defined networking properties in multi-domain networks," in *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*, 2016, pp. 95–100.

[10] R. Awati, L. Rosencrance, and J. English, "What is software-defined networking (sdn)?" 2025, available online at https://www.techtarget.com/searchnetworking/definition/software-defined-networking-SDN; last accessed on 2025/12/12.

[11] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[12] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74–98, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128614002588

[13] Selector Software Inc., "Network automation in 2025: Technologies, challenges, and solutions," 2025, available online at https://www.selector.ai/learning-center/network-automation-in-2025-technologies-challenges-and-solutions/; last accessed on 2025/12/12.

[14] O. W. J. Altalebi and A. A. Ibrahim, "Optimization of elapsed time of automation for large-scale traditional networks and proposing new automation scripts," in *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2022, pp. 1–10.

[15] A. M. Mazin, R. A. Rahman, M. Kassim, and A. R. Mahmud, "Performance analysis on network automation interaction with network devices using python," in *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2021, pp. 360–366.

[16] D. Mendoza, "Accelerating your network automation journey with ansible network validated content," 2023, available online at https://www.redhat.com/en/blog/accelerating-your-network-automation-journey-ansible-network-validated-content; last accessed on 2025/12/13.

[17] T. Slattery, "Compare paramiko, netmiko and napalm network automation," 2024, available online at https://www.techtarget.com/searchnetworking/tip/Network-automation-with-Python-Paramiko-Netmiko-and-NAPALM; last accessed on 2026/01/08.

[18] K. Rohrecker, "Maximizing the impact of network digital twins through automated network assurance," 2024, available online at https://ipfabric.io/blog/maximizing-network-digital-twins-with-assurance/; last accessed on 2025/12/13.

[19] D. Chen, H. Yang, and C. Zhou, "Requirements and Design for Interfaces of Network Digital Twin," Internet Engineering Task Force, Internet-Draft draft-chen-nmrg-dtn-interface-03, Mar. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-chen-nmrg-dtn-interface/03/

[20] Hewlett Packard Enterprise Development LP, "What is intent-based networking?" 2025, available online at https://www.hpe.com/us/en/what-is/intent-based-networking.html; last accessed on 2025/12/13.

[21] D. M. Manias, A. Chouman, and A. Shami, "Towards intent-based network management: Large language models for intent extraction in 5g core networks," in *2024 20th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2024, pp. 1–6.

[22] E. Zeydan and Y. Turk, "Recent advances in intent-based networking: A survey," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–5.

[23] S. Cooper, "Sdn and legacy system integration: Challenges & solutions," 2025, available online at https://www.comparitech.com/net-admin/sdn-legacy-integration/; last accessed on 2026/01/14.

[24] C. Uneze, "How to use network automation to ease cloud integration," 2024, available online at https://www.techtarget.com/searchnetworking/tip/How-to-use-network-automation-to-ease-cloud-integration; last accessed on 2026/01/14.

[25] X. Zheng and A. Leivadeas, "Network assurance in intent-based networking data centers with machine learning techniques," in *2021 17th International Conference on Network and Service Management (CNSM)*, 2021, pp. 14–20.

[26] B. Martini, M. Gharbaoui, and P. Castoldi, "Intent-based network slicing for sdn vertical services with assurance: Context, design and preliminary experiments," *Future Generation Computer Systems*, vol. 142, pp. 101–116, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X2200437X

[27] P. H. Gomes, M. Buhrgard, J. Harmatos, S. K. Mohalik, D. Roeland, and J. Niemöller, "Intent-driven closed loops for autonomous networks," *Journal of ICT Standardization*, vol. 9, no. 2, pp. 257–290, 2021.

[28] SynerComm and Juniper Networks, Inc., "Digital twin lab and intent-based networking: Building trust through network validation," 2025, available online at https://20503945.fs1.hubspotusercontent-na1.net/hubfs/20503945/eBooks/Digital%20Twin%20Lab%20and%20Intent-Based%20Networking%20eBook.pdf; last accessed on 2026/01/14.

[29] O. Aramide, "Advanced network telemetry for ai-driven network optimization in ultra ethernet and infiniband interconnects," *SAMRIDDHI A Journal of Physical Sciences Engineering and Technology*, vol. 17, p. 2025, 01 2025.

[30] M. Zhang, L. Cui, X. Zhang, F. P. Tso, Z. Zhen, Y. Deng, and Z. Li, "Quark: Implementing convolutional neural networks entirely on programmable data plane," in *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications*, 2025, pp. 1–10.

[31] M. Seliem, U. Roedig, C. Sreenan, and D. Pesch, "Qos-aware proportional fairness scheduling for multi-flow 5g ues: A smart factory perspective," 2025. [Online]. Available: https://arxiv.org/abs/2508.21783

[32] S. McGillicuddy, "Getting started with a network source of truth," 2025, available online at https://blog.enterprisemanagement.com/getting-started-with-a-network-source-of-truth; last accessed on 2026/01/14.

# Multipath QUIC Schedulers under the Microscope

Andreea Lupu, Daniel Petri*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: andreea.lupu@tum.de, petriroc@net.in.tum.de

***Abstract*—Multi-homed devices, such as phones, can use WiFi and cellular networks on the same device. Most of the HTTP/2 traffic comes from mobile devices, through video streaming and web pages. Efficient traffic prioritization is crucial for loading web content. The high latency can negatively impact user experience and the revenue of companies. Improving performance requires reducing the transmission delays across heterogeneous network paths, which are caused by inefficient scheduling decisions and retransmission.**

**Multipath QUIC (MPQUIC) is the solution researched in this paper. Depending on its application, MPQUIC is used to either mitigate the inter- and intra-stream Head-of-Line blocking by leveraging different schedulers, path, stream, uplink, and downlink schedulers, or to improve the Quality of Experience in video streaming by modifying the congestion control mechanisms. Our study compares different implementations of MPQUIC in mobile environments, exposing similarities, differences, and their research gaps.**

***Index Terms*—MPQUIC, MPTCP, head-of-line blocking, mobile networks**

## 1. Introduction

Latest developments in mobile networks and web technologies have increased the volume and the complexity of the Internet traffic, especially for video streaming and web browsing. Efficient transmission is critical, as it can damage the revenue and the user satisfaction. Google reports that an additional 500ms at the Page Load Time (PLT) decreases up to 20% of the user engagement, while Bing reports that a similar delay can lead to a 1.2% decrease in their revenue [1]–[3]. The extra delays can arise from the enlarged complexity of the web pages, including the increased number of resources, as well as the size.

Video streaming also suffers from similar challenges. Delays lead to buffering, interruptions, and maybe never seeing a segment again [4]. Although the network technologies are performant for end-to-end devices, it would be advantageous to extend these technologies to a multipath strategy due to the diversity of the network interface cards (NICs) in mobile devices.

The most common protocol currently used on the Internet is HTTP over the Transmission Control Protocol (TCP). Multipath TCP (MPTCP) was developed to enable a single connection to use different paths simultaneously [5]. However, both TCP and HTTP suffer from Head-Of-Line (HOL) blocking; HTTP due to scheduling and TCP due to retransmission, both leading to high latencies in web page completion times [6].

To address the issues with TCP, the QUIC protocol has been developed. It enables multiple byte streams to be sent independently over a single session, preventing congested streams from blocking each other [6]. As with TCP, QUIC should support a diversity of NICs, and therefore, *Multipath QUIC (MPQUIC)* was introduced.

## 2. Background

This section explains the concepts used in MPQUIC implementations. We start with the problem that MPQUIC tries to mitigate, moving to MPTCP, and lastly explaining the original MPQUIC protocol.

**HOL Blocking.** The HOL blocking challenge is categorized into two parts: *inter-stream* and *intra-stream* [6]. The inter-stream HOL blocking occurs when processing one resource is delayed by another resource. HTTP/1.1 over TCP allows only one resource to be transmitted at a time, which can cause blocking at the application layer in case of retransmissions or delays. Although HTTP/2 reduces this issue through stream multiplexing, blocking still occurs due to the in-order transmission of TCP. QUIC allows the usage of multiple independent streams over a single session to mitigate the transport-level HOL blocking, but it may suffer from application-layer blocking when the stream prioritization and scheduling are handled poorly. The intra-stream HOL blocking occurs in QUIC when the data within one stream must be delivered in order, and one packet is lost, and subsequent packets are blocked.

**MPTCP.** Lim et al. [7] highlight that alternating between WiFi and cellular networks makes it difficult for TCP to utilize both interfaces. MPTCP divides a single connection into multiple subflows across all end-to-end interfaces, enabling aggregated bandwidth and allowing WiFi and cellular connections to be combined. The scheduler uses the minimal Round Trip Time (minRTT) estimation, but it lacks the path heterogeneity examination, which is common in mobile networks. MPTCP selects the path with the lowest RTT and an available congestion window, and ensures in-order delivery at the connection level, not only at the subflow level. To cover the packet loss, MPTCP includes a retransmission method that passes lost packets from a slow to a fast path.

**ECF.** The original MPTCP fails to take advantage of heterogeneous paths with different bandwidths and delays.

 doi: 10.2313/NET-2026-05-1_16

MinRTT transmits data over the lowest RTT path only when its congestion window allows, without accounting for the impact on the flow completion time (FCT). Lim et al. [7] introduce the *Earliest Completion First (ECF)* scheduler for MPTCP to minimize the FCT by selectively scheduling data across available paths based on the estimated completion time. Unlike minRTT, ECF considers both the bandwidth and the number of bytes to be sent at the connection level to determine whether the FCT is reduced or increased if it would send data on a specific subflow.

**MPQUIC.** To enhance QUIC for multi-homed devices, MPQUIC was developed. De Coninck et al. [8] present two main goals: adapting to connection errors and coordinating resources on different paths. The authors include a Path ID in the header of the packet so that the path can be identified. An additional scope is to help MPQUIC maintain information such as RTT, congestion state, and packets lost in case the address changes over a path. MPQUIC enables different paths to have different packet number spaces so that a receiver can acknowledge packets from different paths. The path manager is used to create and delete paths; a path is created over each interface during the handshake. If one or more paths are active, the sender must select the path to send data on. The packet scheduler does the selection. MPQUIC uses the same scheduler as MPTCP, namely minRTT. The difference is that MPQUIC also decides which type of frame is sent on a particular path. Moreover, MPQUIC enables a flexible retransmission strategy by sending a window update frame on all paths. This way, it avoids retransmission on the same path. MPQUIC allows a fair resource distribution. Implementations may integrate the Opportunistic Linked Increases Algorithm (OLIA) congestion control mechanism [9]. If the subflows are not sharing a bottleneck, all congestion windows are treated separately; otherwise, they are grouped, and the congestion control operates in multipath mode [4]. The use of minRTT as a scheduler created the need to develop different scheduling mechanisms, such as those described in the next section.

## 3. MPQUIC-Scheduler Design

This section details the MPQUIC schedulers: SA-ECF [2], HBES [10], CAPS [1], MPQUIC-SBD [4]. Its scope is to highlight their different methods.

### 3.1. SA-ECF

To overcome the asymmetry in bandwidth and delay, Rabitsch et al. [2] propose an MPQUIC algorithm, called *Stream-Aware ECF (SA-ECF)*. This scheduler is based on ECF and ensures a fair allocation of the aggregated bandwidth, utilizing the prioritization provided by HTTP/2 through the dependency tree. The algorithm has two main components: a *stream scheduler* and a *path scheduler*, as highlighted in Figure 1. First, the stream scheduler assigns a fair share of the aggregated bandwidth to active streams based on the received HTTP/2 information about the resources. Next, the path scheduler decides which path each stream should utilize to transmit data. The results show a lower stream completion time (SCT) and well handling of heterogeneous paths [2].

**Fair Stream Scheduler.** As input, it takes the HTTP/2 dependency tree, where each node is a stream. On its input, it uses the Weighted Round Robin (WRR) algorithm as follows: each parent node shares sending opportunities with its children according to the weights received from the dependency tree. An opportunity is a share of aggregated bandwidth that a stream can send data on. If a node is active, it will take the opportunity to send data. If a node is inactive, meaning it has no data to share, it passes its opportunity further to its children. Each stream receives a concrete number of opportunities. If one stream is unable to send its data, SA-ECF does not give the stream another opportunity to send its share until it is its turn again. The retransmission and control messages have a higher priority and are therefore sent before stream frames, ensuring fairness within each scheduling round [2].

**Earliest Completion First.** ECF determines which path the data should be sent on [2]. SA-ECF implements a per-stream ECF mechanism aiming to minimize the FCT. The scheduler determines for each stream whether it is better to wait for a faster path that is currently occupied or to send the data through a slower path. The decision depends on the remaining amount of data to be sent and the available bandwidth of each path. The transmission waits for the faster path if and only if the FCT would be lower than the transmission over the slower path. This decision significantly reduces the completion time.

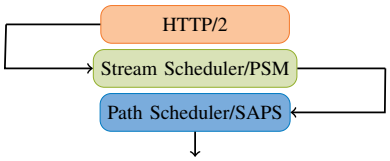



Figure 1: Abstract design of SA-ECF and HBES [10].

### 3.2. HBES

Xing et al. [10] discuss two challenges of MPQUIC: the transmission of out-of-order (OFO) packets over multiple paths and the prioritization of streams. The authors propose a solution, the *HoL Blocking Eliminating Scheduler (HBES)*, to address inter-stream and to avoid intra-stream HOL blocking. The solution consists of two components: the *Priority-based Stream Manager (PSM)* and the *Stream-aware Arrival-time-based Path Selector (SAPS)*, as displayed in Figure 1. First, the PSM is used to determine the stream that would send data and the number of packets to send. Next, SAPS is responsible for selecting the path that would result in the lowest arrival time. The results show a reduction in the SCT and a good send buffer optimization [10].

**Priority-Based Stream Manager.** PSM handles the HTTP/2 prioritization mechanism for the streams based on the dependency tree [10]. First, the parent node must complete its transmission before the children can send. The client usually sets priorities for each stream in the

 doi: 10.2313/NET-2026-05-1_16

dependency tree. To schedule the stream, the authors have developed a scattered WRR (SWRR), which limits the number of packets a stream can send in each round, thereby preventing exhaustion of the sending window. Afterwards, PSM determines the number of packets a stream is able to send in a round and sets a token number for each stream accordingly. When the parent node has finished, all the children are visited in the order of establishment, and the token number of packets is sent to each child. PSM also ensures that each stream receives its opportunities, thereby eliminating the inter-stream HOL blocking based on those opportunities.

**Stream-Aware Arrival-Time-Based Path Selector.** SAPS determines the path for the sent packets [10]. The decision is based on the actual RTT, throughput, and the send buffer capacity. It calculates the total arrival time (queuing time + flight time) for each packet on each path and chooses the shortest one, even if it involves queuing. It constantly observes the RTT and throughput, enabling it to adapt to changes. It implements a send buffer at the path-level to hold the payload scheduled for paths that are occupied, allowing complete packets to be formed at a later time and for the payload to be sent in a concrete order. SAPS will use the path with the lowest RTT until its buffer starts to fill up, then it will continue with the next smallest. In case all congestion windows are complete, SAPS will hold, but only the scheduled packets will be affected.

## 3.3. Context-aware MPQUIC

Wang et al. [1] address the suboptimal PLT issue that arises using heterogeneous paths in mobile networks. MPTCP sends the acknowledgement (ACK) frames back on the same path on which the data arrived, which can lead to OFO packets. Moreover, the original MPQUIC does not consider the delays and dependencies introduced by inter-object downloading. The solution proposed to combat these challenges is the *Context-Aware MPQUIC Packet Scheduler (CAPS)*. CAPS consists of two schedulers, one for the uplink (client to server) and one for the downlink (server to client), and a mechanism that determines the order of stream transmissions based on priority and size, thereby improving the SCT accordingly. Figure 2 illustrates the context-aware MPQUIC architecture at an abstract level. The results show a decrease of up to 8.5% in the PLT and up to 12.9% in the SCT [1].

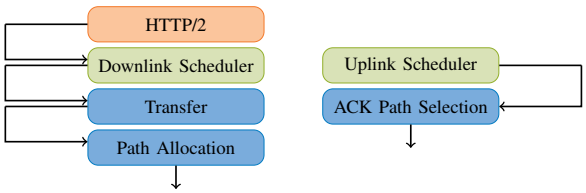



Figure 2: CAPS Architecture. Two separate schedulers, uplink and downlink, improve the PLT.

**Stream-Aware Downlink Scheduler.** The scheduler takes the path characteristics and stream priority into account [1]. To optimize the delivery of the resources, the scheduler utilizes the dependency tree. There is a parent-child relation between the nodes, where each parent applies a scheduler based on a modified WRR (MWRR) on its children. It calculates the completion time of streams using a fair WRR on the stream length and dependency tree to ensure fairness and avoid starvation. The priority scheme of Firefox was used, which introduces empty, non-resource nodes to gather other nodes and gives them different weights [6]. After the completion time is calculated and the transfer order is determined, it will send the streams exclusively. The following step is to ensure that all paths complete a stream data transfer at once. The SCT is calculated based on the bandwidth, RTT, and queuing time. The paths are taken in RTT order and transmit packets until they are full. Then, it selects streams based on priority with sending limits, and at last, it updates the stream state.

**Latency-aware Uplink Scheduler.** The context-aware mechanism sends the ACK frames on the path that has the lowest delay, reducing the RTT [1]. To determine the fastest route for ACK, the protocol sends regularly duplicated ACK packets on all uplink paths. The server monitors the identical ACKs, and it maintains the information as valid. Based on this information, it chooses the path for the ACK frame. The authors inserted a new control frame, called CHANGE_RETURN_PATH_FRAME, to elect an ACK route for each path. The ACK's Path ID does not play any role in the congestion window. This way, MPQUIC removes the HOL blocking and ensures retransmission on rapid paths.

## 3.4. MPQUIC-SBD

In the paper [4], Paiva et al. discuss the problem encountered in video streaming environments. They propose *MPQUIC Shared Bottleneck Detection (MPQUIC-SBD)*, which makes the congestion control adapt dynamically when multiple subflows share a bottleneck. The MPQUIC-SBD scope is to increase the fairness and to balance the bandwidth utilization of video streams. The solution handles the stream multiplexing on different subflows. The results of the research show a 13% increase in the throughput for the non-shared bottlenecks scenarios and 90% accuracy in the SBD [4].

**SBD Decision.** The SBD protocol is implemented at the receiver side, although both the client and the server must be able to use SBD [4]. The receiver handles the metric relying on the timestamp provided by the sender. SBD considers only those packets from the same connection that have this timestamp to perform the One-Way-Delay (OWD) calculations. Based on the calculated metrics, SBD groups the subflows into sharing or not sharing a bottleneck and makes its decision based on the majority vote of the last 10 observations.

**New QUIC Frames and Congestion Control.** MPQUIC-SBD extends the QUIC frames by adding the following attributes for stream frames: Path ID, Loss Count, and Timestamp, while it adds the SBD Epoch ID and Group fields in the ACK frames [4]. The new fields of the stream frames are required for computing the OWD and the loss

metrics without creating overhead on the network. The fields are used for congestion control to scale the congestion window of each subflow. The OLIA algorithm is responsible for coupling subflows with a shared bottleneck and decoupling subflows without, enabling the congestion control adaptation.

**The Sparsity of Video Transmissions.** The video transmission can have a sparse density and a pause period between the fragments, leading to noise introduction [4]. Paiva et al. propose a solution to eliminate the noise issue that uses the timer received from the sender to calculate the time elapsed, the OWD, and the SBD metrics. To further eliminate the noise, they implement a filter to take out the noise caused by sparsity.

## 4. Comparison

In this section, we will compare the implementations. Table 1 summarizes the main similarities and differences between the four implementations.

TABLE 1: MPQUIC Scheduler Comparison

| **Features** | **SA-ECF** [2] | **HBES** [10] | **CAPS** [1] | **SBD** [4] |
|---|---|---|---|---|
| *Algorithm* | WRR | SWRR | MWRR | minRTT |
| *OFO* | Medium | Strong | Low | Low |
| *D. Tree* | ✓ | ✓ | ✓ | ✓ |
| *Stream-aware* | ✓ | ✓ | ✓ | × |
| *Real-World* | × | ✓ | ✓ | × |

From an architectural design perspective, HBES and SA-ECF have a similar structure. Both implementations first select the stream based on the HTTP/2 priorities, and then the path is decided. SA-ECF decides the path at the stream level, while HBES decides at the packet level [2], [10]. Both implementations also used the HTTP/2 dependency tree mechanism, where the parents are responsible for resource allocation, and the children are allowed to send data only when the parent node is inactive [2], [10]. Moreover, both implementations give a concrete number of opportunities to each stream to send its fair share of data.

A significant difference is the scheduling algorithm that was used. SA-ECF uses the WRR, while HBES uses a scattered WRR, which limits the number of packets each stream can send in each round. Another difference lies in how they handle OFO packets. SA-ECF focuses on avoiding slow paths. HBES instead tries to eliminate OFO packets by estimating the total completion time. As a result, HBES mostly maintains the in-order arrival of the data, leading to better management of the OFO data and eliminating HOL [10]. At the same time, SA-ECF utilizes its buffer to the maximum in path asymmetry scenarios [10].

Despite their differences, Xing et al. [10] show that both implementations have a similar result for FCT when using small streams. HBES performs better than SA-ECF because it is more successful in eliminating the HOL delays. Both protocols aim for a low maximal FCT through the fair share of its stream. In a serial experiment, SA-ECF and HBES outperform other protocols, such as RR or Lowest-RTT-First, by avoiding high-latency paths for small streams. By doing so, the FCT is reduced. In a parallel experiment, HBES successfully reduces the FCT of high-priority streams, but the tradeoff between ordering and throughput gets highlighted because the entire throughput is reduced.

The paper [10] also compares the overall performance, and its results show that both protocols have the best performance, achieving a reduction of 70% in scenarios with a lot of asymmetry, with HBES performing slightly better than SA-ECF. SA-ECF waits for the lowest preferred path, while HBES is able to schedule data for unavailable paths due to the buffer for sending; therefore, HBES is able to achieve faster transmission at the stream level.

Neither SA-ECF nor HBES addresses dynamic switching of scheduling mechanisms to make it more adaptable in the case of different scenarios [2], [10]. SA-ECF focuses on stable networks, while HBES focuses on mobile networks, but the two scenarios are not always exclusive. Plus, neither SA-ECF nor HBES schedulers are testing the interaction with HTTP/3. Also, SA-ECF does not evaluate its implementation in a real-world scenario; it is on an emulated network. HBES is using a small testbed. All these would be good points to research further.

HBES and SA-ECF are not the only implementations making MPQUIC aware of the context. Wang et al. [1] develop CAPS, a solution to make MPQUIC context-aware for the HTTP/2 mobile traffic. The solution uses the HTTP/2 dependency tree as well as the stream priorities. The goal is to eliminate HOL and reduce PLT in mobile networks.

Wang et al. [1] derive a WRR-based algorithm, but they modify it. First, the decision is taken based on the stream priority and length. Afterwards, it sends the stream exclusively, improving the SCT across multiple paths. This is different from HBES and SA-ECF, which focus on fair-based multiplexing scheduling.

Moreover, CAPS enables different schedulers for up- and downlink [1]. The downlink ensures the order and the shares for the data that is sent, while the uplink scheduler ensures that ACK frames take the fastest way to the destination. This is a unique difference from HBES and SA-ECF because it reduces the RTT. HBES and SA-ECF send the ACK on the same path that it came from [2], [10].

Similar to SA-ECF, CAPS is making the scheduling decision per stream. Wang et al. [1] highlight the importance of the real-world scenario of the user in motion. Despite their research, CAPS does not explicitly measure the OFO packets and their improvement for their implementation, as the OFO packets play an important role in HOL improvement.

Paiva et al. [4] take a different approach to handling the multipath improvement for QUIC. Their solution focuses on the coupled congestion control of separate paths. By integrating SBD in MPQUIC, the approach decides whether or not to couple subflows based on the OWD metrics. MPQUIC-SBD does not consider the stream and path scheduling, nor where to send the ACK frames. The scheduler maintains the minRTT path scheduler.

MPQUIC-SBD is testing its implementation in an emulated network environment. Paiva et al. [4] use video streaming to highlight the importance and the results of the created algorithm. Although the results of MPQUIC-SBD are significant, the authors do not consider the HTTP/2

prioritization. The stream selection is not handled at all, while the path selection is not improved. This results in a research gap. Their primary focus is to improve the Quality of Experience and the throughput rather than the multipath diversity. Another research gap is that the MPQUIC-SBD is not evaluated under motion.

## 5. Conclusion and future work

This paper presents a detailed comparison between four MPQUIC schedulers. All the implementations have proven good results in mitigating the initial problem and are proof that a further improvement of the original MPQUIC is needed.

While some of the implementations focus on making MPQUIC aware of streams and mitigating the HOL problem, others try to improve the congestion control of the subflows. Both of them are required in the mobile networks and must be researched further.

As Google and Bing demonstrated, HOL blocking and additional delays have a huge negative impact on the business and must be improved. Although this paper is tackling the improvement of the traffic in mobile networks, it only discusses HTTP/2. For QUIC, there is a new HTTP version created, HTTP/3, that replaces the dependency tree prioritization of HTTP/2 with the Extensible Prioritization Scheme [11]. Future work should research how the MPQUIC schedulers interact with HTTP/3.

## References


[1] J. Wang, Y. Gao, and C. Xu, "A multipath quic scheduler for mobile http/2," in *Proceedings of the 3rd Asia-Pacific Workshop on Networking*, ser. APNet '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 43–49. [Online]. Available: https://doi.org/10.1145/3343180.3343185

[2] A. Rabitsch, P. Hurtig, and A. Brunstrom, "A stream-aware multipath quic scheduler for heterogeneous paths," in *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, ser. EPIQ'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 29–35. [Online]. Available: https://doi.org/10.1145/3284850.3284855

[3] E. Schurman and J. Brutlag, "Performance related changes and their user impact," in *velocity web performance and operations conference*, 2009.

[4] T. W. d. P. Paiva, S. Ferlin, A. Brunstrom, O. Alay, and B. Y. L. Kimura, "A first look at adaptive video streaming over multipath quic with shared bottleneck detection," in *Proceedings of the 14th ACM Multimedia Systems Conference*, ser. MMSys '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 161–172. [Online]. Available: https://doi.org/10.1145/3587819.3590982

[5] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 8684, Mar. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8684/

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

[7] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "Ecf: An mptcp path scheduler to manage heterogeneous paths," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 147–159. [Online]. Available: https://doi.org/10.1145/3143361.3143376

[8] Q. De Coninck and O. Bonaventure, "Multipath quic: Design and evaluation," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 160–166. [Online]. Available: https://doi.org/10.1145/3143361.3143370

[9] Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, "Managing multiple paths for a QUIC connection," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-multipath-18, Dec. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/18/

[10] Y. Xing, K. Xue, Y. Zhang, J. Han, J. Li, D. S. L. Wei, R. Li, Q. Sun, and J. Lu, "A stream-aware mpquic scheduler for http traffic in mobile networks," *Trans. Wireless. Comm.*, vol. 22, no. 4, p. 2775–2788, Apr. 2023. [Online]. Available: https://doi.org/10.1109/TWC.2022.3213638

[11] K. Oku and L. Pardue, "Extensible Prioritization Scheme for HTTP," RFC 9218, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9218

# Alternate Service Mechanisms

Theodor Ogeday Özuluca, Tim Betzer*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: theodor.oezuluca@tum.de, betzer@net.in.tum.de

***Abstract*—Alternative service mechanisms allow servers to advertise authoritative endpoints and protocol configurations to improve performance and security. This paper compares HTTP-based Alt-Svc and DNS-based SVCB and HTTPS resource records, analyzes connection establishment latency, protocol negotiation, fallback behavior, security and real-world deployment. It shows that Alt-Svc enables incremental and backwards compatible deployment but cannot optimize initial connections, while SVCB and HTTPS records allow earlier and more authoritative service discovery with more complex deployment. The evaluation highlights that both mechanisms serve complementary roles in incremental deployment and early connection optimization, and discusses the current adoption in practice.**

***Index Terms*—alternative services, alt-svc, svcb, https records, dns, protocol negotiation, http/3, quic**

## 1. Introduction

Modern web communication relies on efficient connection establishment and the negotiation of appropriate protocols for both security and performance. Clients must be able to determine not only how to reach the service of an origin, but also which protocols, endpoints, and transport options are supported by the server. Without the ability to discover this information, the client may connect via a slower or less secure protocol, even if better alternatives are available [1], [2]. The absence of standardized mechanisms for discovering alternative services delays protocol upgrades, underutilizes performance optimizations, and may negatively affect security. Therefore, mechanisms are required that enable the client to discover alternative services while still preserving backwards compatibility with existing web infrastructure.

To address these problems, the Internet Engineering Task Force has standardized multiple alternative service mechanisms. These allow servers to advertise alternative ways of access to the same resource. An alternative service is identified by the protocol name, host and port [2].

This paper focuses on two such alternative service mechanisms. The first approach uses the Alt-Svc HTTP header field, which enables the server to advertise alternative services in its HTTP response header after the client has established its initial connection to the origin's resource [2]. The second approach is DNS-based service discovery using Service Binding (SVCB) and HTTPS DNS resource records, which enable clients to learn of alternative services before establishing the initial connection [3].

This paper describes and compares these two alternative service mechanisms regarding performance, protocol negotiation, security, and deployment status. Furthermore, advantages and disadvantages of each approach are discussed.

## 2. Related work

HTTP-based alternative service discovery is standardized in RFC 7838 by Nottingham et al. [2]. It enables incremental deployment of new protocols, and early studies on QUIC deployment by Zirngibl et al. [1] and Matsuzawa and Ichikawa [4] show that it helps with protocol transition while keeping backwards compatibility. DNS-based alternative service discovery was standardized more recently in RFC 9460 by Schwartz et al. [3], introducing the SVCB and HTTPS resource records. This allows for more authoritative service advertisements. Measurement studies by Dong et al. [5] and Zirngibl et al. [6] show that these records are primarily deployed by large providers and commonly used for QUIC-capable endpoints granting performance improvements, but face operational problems and misconfiguration during deployment.

## 3. Alternate Service Mechanisms

Alternative service mechanisms enable an origin to authoritatively advertise access to its resources via alternative network locations, potentially employing different protocol configurations. The goal is to reduce connection establishment overhead, improve overall performance and security while maintaining compatibility with existing web infrastructure [2]. This section introduces two standardized approaches to alternative service discovery and outlines their fundamental design principles.

### 3.1. HTTP Alt-Svc Header

The HTTP Alternative Services mechanism, defined in RFC 7838 [2], allows servers to advertise alternative services through the Alt-Svc HTTP header field. An alternative service represents an authoritative way of accessing the same origin's resource. Each alternative service is identified by an Application-Layer Protocol Negotiation (ALPN) protocol name, a host, and a port.

When a client receives an Alt-Svc advertisement, it may open a new connection to the advertised alternative service if the alternative presents a TLS certificate that is valid for the original site. Once the connection is established, the client can use it for subsequent requests. The

 doi: 10.2313/NET-2026-05-1_17

use of alternative services is transparent to the application layer. This means that, if the client uses an alternative, it may internally change the protocol, host, or port but these changes must not be propagated to the application.

The Alt-Svc mechanism allows servers to suggest a different protocol or network location without changing the original URL. Typical use cases include offering access to a new protocol, such as HTTP/2, redirecting clients to servers that are under less load or geographically closer to the client, or segmenting clients based on capability support. For incremental deployment, using an alternative service is optional for clients and fallback to the original service remains possible if the alternative service is unavailable or unresponsive.

Clients cache alternative services for usage. The freshness parameter (ma) dictates, how long the advertised alternative service is considered valid. Clients should use fresh/valid alternative services but can continue using an alternative service after the freshness has expired. Furthermore, a persist flag can be used to manage, if an entry for an alternative service should be kept in the clients cache after the network configuration has changed. If not set, the clients should delete the entry in the cache.

Despite its advantages like flexibility and backwards compatibility, the Alt-Svc mechanism has inherent limitations [2]. Because this mechanism advertises alternative services via HTTP, clients can only learn about them after establishing an initial connection to the origin. Thus, only subsequent connections can benefit from potential performance and security improvements. Moreover, the server cannot force the client to use a certain connection. This results in unpredictable behavior of clients which can complicate load balancing or lead to clients continuing using a less secure or less efficient protocol. There are also other potential security and privacy considerations, such as downgrade attacks and the possibility of tracking clients by advertising a unique host name to each [2].

Alt-Svc provides a flexible and backwards compatible mechanism for alternative service discovery but it is also constrained by being based on HTTP. This motivates mechanisms that enable the discovery of alternative services before connection establishment. In the next section, such a mechanism will be discussed.

### 3.2. SVCB and HTTPS DNS Resource Records

SVCB and HTTPS DNS resource records, defined in RFC 9460 [3], provide a DNS-based mechanism for advertising and discovering alternative services for an origin. While the SVCB record type is designed for general service binding, HTTPS records offer a specialization tailored to HTTPS services. Both record types use the same encoding, structure, and high-level semantics.

SVCB and HTTPS resource records enable domain owners, via authoritative DNS servers, to advertise service-related information during DNS resolution. In contrast to HTTP-based mechanisms, this allows clients to learn about alternative endpoints and supported protocol configurations before establishing a connection. The primary goal of these records is to enable clients to obtain required information for establishing a connection to an authoritative endpoint with at most one additional DNS query. While in practice, typically no additional round trip is needed [3], [5].

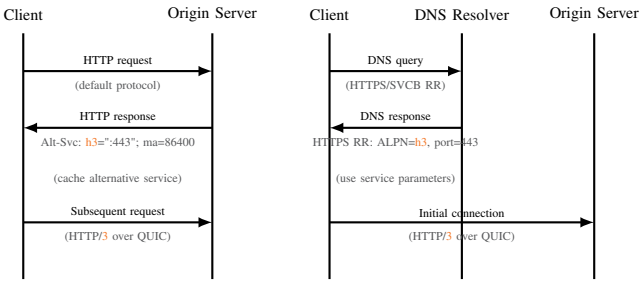
SVCB records support two modes of operation. In AliasMode, the record delegates a service name to another domain name. This provides CNAME-like functionality with the benefit that it can also be used at the zone apex where CNAME records are not permitted. In ServiceMode, the record specifies alternative endpoints together with sets of service parameters that describe how the endpoints should be used [3].

For HTTPS origins, HTTPS resource records provide special handling [3]. They can advertise support for newer protocols such as HTTP/3 over QUIC, signal a preference for HTTPS and enable automatic upgrade where possible when an HTTPS record is present. Additionally, they can enable the usage of non-default port numbers for TCP and UDP, provided no port restricting firewall is between the client and the service endpoint. HTTPS records can also advertise configuration required for Encrypted Client Hello (ECH). This enables encryption of sensitive parts of the TLS handshake.

While DNS-based service discovery offers advantages, such as discovering endpoints prior to connection, compared to HTTP-based Alt-Svc, it introduces new aspects to consider, including DNS caching and interaction with DNS security mechanisms. This architectural difference forms the basis of the comparative evaluation of the two mechanisms in the next sections.

## 4. Comparative Evaluation

The alternative service mechanisms described in the previous section differ fundamentally in the stage at which service information becomes available to the client and in the protocol used for service advertisement. These differences have a direct impact on performance, protocol negotiation, security, privacy, and deployment. Figure 1 illustrates the workflow of both mechanisms and highlights the difference in the timing of service discovery.




(a) Alt-Svc (b) SVCB/HTTPS

Figure 1: Alternative service discovery mechanisms: (a) HTTP Alt-Svc advertisement after the initial HTTP connection [2]; (b) DNS-based discovery via SVCB/HTTPS records during DNS resolution [3].

This structured evaluation aims to provide a systematic comparison of Alt-Svc and SVCB and HTTPS DNS resource records regarding performance, protocol negotiation, security, privacy, and deployment, and to clarify the benefits and trade-offs of HTTP-based and DNS-based service discovery. Additionally, at the end complementary use of both mechanisms is discussed.

## 4.1. Performance

The primary performance goal of alternative service mechanisms is to decrease connection establishment time (latency) and to minimize the number of round trips needed before a client can use the best possible transport and protocol configuration for a given request. The performance of these mechanisms, therefore, significantly depends on when the client receives the appropriate service information for an alternative service and how this information is integrated into the existing connection establishment process.

Because HTTP Alt-Svc advertises alternative services after the initial HTTP connection to the origin server has been established, the first connection uses the default protocol and endpoint. Thus, any potential performance improvements due to using a different protocol or a closer server can only occur on subsequent connections. Although Alt-Svc supports both protocol upgrade and endpoint migration without changing the URL of the requested origin's resource, the initial connection latency is consequently not reduced. When connections are relatively short-lived or clients frequently access new origins, this limits the performance benefits that can be achieved by using alternative service mechanisms.

On the other hand, SVCB and HTTPS DNS resource records enable clients to discover alternative endpoints and the protocol capabilities during DNS resolution, before the actual connection is established. By utilizing the provided service parameters of the record, such as supported application-layer protocols and alternative transport endpoints, DNS-based service discovery can enable the client to establish a direct connection to the server using an optimized configuration for each specific request. This can prevent additional round trips that would have been needed to negotiate the protocol and/or redirect the client to an alternative endpoint after the initial connection has been established. Measurement studies [6], [7] have shown that the early availability of service information can reduce the connection establishment latency, particularly if the client supports protocols like HTTP/3 over QUIC, which can reduce round trips significantly by cutting out the TCP and TLS handshake.

Empirical analyses by Zirngibl et al. [6] show that these records are often used to advertise QUIC-capable endpoints and to direct clients to alternative services operated by content delivery networks (CDNs). Their analysis further shows that HTTPS resource records are primarily provided by large providers and CDNs, where the performance benefits of faster connection establishment and optimized protocol selection are greatest. When supported by resolvers and clients, SVCB and HTTPS records can both reduce the number of DNS queries required and speed up the connection establishment process by providing protocol information and, in case address records are not available, supplying IP address hints via the ipv4hint and ipv6hint parameters, allowing clients to connect without issuing another DNS query [3].

Despite the above advantages, the performance benefits of DNS-based service discovery are dependent on the behavior of resolvers and client support. While SVCB-aware DNS resolvers can include SVCB results in the additional section of DNS responses, legacy resolvers may force clients to request additional queries or to fallback to traditional DNS resolution. To mitigate the negative effect of legacy resolvers, it is recommended that clients perform the SVCB resolution in parallel to usual A and AAAA lookups [3]. This is confirmed by measurement studies [5] and ensures that DNS-based service discovery does not negatively affect the connection establishment time.

Ultimately, the performance comparison illustrates the trade-off between the two approaches. While Alt-Svc improves performance for subsequent connections without affecting the initial connection establishment latency, SVCB and HTTPS DNS resource records aim to optimize the first connection by moving the service discovery to an earlier point in the connection process. How much the theoretical benefits of each mechanism affect the real-world performance depends on the supported protocols and the resolvers' capabilities.

## 4.2. Protocol Negotiation and Fallback Behavior

Alternative service mechanisms not only differ in the timing of when the service information is discovered, but also in the handling of protocol negotiation, upgrades, downgrades, and fallback.

Alt-Svc enables the server to advertise alternative services to the client after the initial connection has been established. With the information advertised in the Alt-Svc HTTP header field, the client can use the ALPN protocol name, host, and port to connect to the alternative resource. The client dictates, whether and when to use an alternative service. Fallback is straightforward: If the client cannot connect to the alternative service, it uses the original connection or a different cached alternative service. This grants backwards compatibility [4] but limits the server's capability to influence protocol selection. This also has the risk of protocol downgrade attacks which are discussed in Section 4.3 [2].

In contrast, with SVCB and HTTPS DNS resource records, the alternative service discovery is moved to the DNS resolution phase. Through the advertised service parameters, the client can learn about supported application-layer protocols before the connection is established. Multiple protocol options can be advertised, the client chooses one and directly connects to it. Additionally, an HTTPS resource record indicates that the service is only available over HTTPS, which prevents clients from using HTTP. Fallback is more complex: If resolution via AliasMode terminates without finding a ServiceMode record, the client treats the final target name as a valid endpoint and tries to connect to it using conventional address resolution. For compatibility, clients for pre-existing protocols such as HTTP should be SVCB-optional. This means, that a client is able to connect to a resource without SVCB via the traditional DNS lookup with A/AAAA records in case the SVCB resolution is not possible. However, when DNS resolution is cryptographically protected and SVCB resolution fails, the client must not fallback to non-SVCB resolution, as it could be a downgrade attack [3]. Measurement studies confirm that clients often implement parallel resolution via SVCB and traditional address resolution [5].

Overall, SVCB and HTTPS records allow earlier and more authoritative protocol negotiation but have more

complex fallback handling. Protocol negotiation of Alt-Svc is less authoritative and leaves more control to clients while having simpler fallback behavior.

### 4.3. Security and Privacy Implications

The security and privacy of the alternative service mechanisms are primarily influenced by the channel through which the services are advertised.

Alt-Svc advertisements are delivered via an already established HTTP connection. Therefore, Alt-Svc inherits the security properties of the underlying transport, typically TLS over TCP or QUIC [8]. Clients have to confirm that an alternative service is authoritative for the origin's resource by validating that it has a TLS certificate that is valid for the origin. Because Alt-Svc advertisements are optional, the client may ignore them and fallback to the original service with a potentially less secure protocol configuration or a cached alternative endpoint. Downgrade attacks are possible by blocking access to a more secure alternative due to the fallback behavior described earlier. Additionally, from a privacy point of view, Alt-Svc enables servers to advertise unique host names to clients which can be used for tracking them.

As DNS responses are not inherently authenticated, clients must treat SVCB information as advisory, unless the SVCB information is secured, for example using DNSSEC [3]. To mitigate downgrade attacks, clients must not fallback to non-SVCB resolution when the DNS resolution is cryptographically protected. HTTPS RRs enforce HTTPS usage instead of HTTP and enable features like ECH to reduce the leakage of sensitive TLS handshake data.

As discussed, measurements show that SVCB and HTTPS records are primarily used by large providers and CDNs so far. They often use DNSSEC and modern TLS configurations to increase security. On the other hand, DNS-based alternative service discovery exposes service metadata stored in SVCB and HTTPS records to recursive resolvers during name resolution. While this information might not be very sensitive, it introduces a privacy concern that is not present on HTTP-based alternative service mechanisms.

In summary, Alt-Svc uses strong encryption via TLS but is more prone to downgrade attacks, has difficulty of enforcing secure protocols and poses the risk of clients being tracked when a server is under control of an attacker. SVCB and HTTPS records enable a stricter protocol enforcement and security guarantees while being more reliant on DNS and exposing more metadata.

### 4.4. Deployment Status, Real-World Adoption, and Discussion

Real-world adoption of alternative service mechanisms varies significantly between HTTP-based Alt-Svc and DNS-based SVCB and HTTPS resource records because they differ fundamentally in their operational requirements and deployment complexity.

Alt-Svc is widely adopted into major browsers and servers because it is implemented solely at the HTTP layer and does not require any updates to the underlying DNS infrastructure or resolvers. This means it can be adopted incrementally for each origin server individually. There are reports showing Alt-Svc has been primarily used to add new protocol versions such as HTTP/3 over QUIC in a backwards compatible way [1], [4].

SVCB and HTTPS resource records are at an earlier stage of deployment. Large-scale measurements show that most HTTPS records are served by a small number of large CDNs, especially Cloudflare [6]. This concentration reflects the need for control over the DNS infrastructure, resolver behavior and service configuration which is not given for broad ecosystem deployment.

The primary barrier to widespread deployment of DNS-based service discovery is its deployment complexity [5]. Empirical studies report that SVCB and HTTPS resource records may have misconfigured ALPN values, IP hints and ECH key management [5], [9]. Additionally, client support is not widespread yet and resolver behavior varies.

However, despite the difficulties associated with deploying DNS-based service discovery, it is becoming increasingly adopted in scenarios where early protocol negotiation, privacy enhancements, and close integration with CDN routing are important.

In conclusion, the design of Alt-Svc [2] and observations from real-world deployments [6] indicate that HTTP Alt-Svc and SVCB and HTTPS Resource Records are likely to be used in a complementary fashion. As such, HTTP Alt-Svc will probably remain for backwards compatibility and to enable incremental adoption of new protocols [4], while SVCB and HTTPS Resource Records are likely to be used primarily by larger organizations to enable early optimizations of connections and stronger security guarantees. The success of DNS-based service discovery will depend on the support of resolvers and clients for it, as well as the proper and consistent configuration of the SVCB and HTTPS records.

## 5. Conclusion and Future Work

This paper presents a comparative analysis of HTTP-based Alt-Svc and DNS-based SVCB and HTTPS resource records for alternative service discovery. The evaluation concludes that both approaches have a similar goal of enabling clients to access an origin's resource via an authoritative alternative service. However, the two mechanisms vary significantly in several aspects, including the timing of discovery, the difficulty of deployment and security guarantees [2], [3].

Alt-Svc is backwards compatible, relatively easily deployed, and widely supported. Servers can only advertise alternative services to the client after the initial connection has been made. This limits the possible performance and security improvements to subsequent connections. In contrast, with SVCB and HTTPS resource records alternative endpoints can be discovered during DNS resolution, prior to connecting. Thus, performance and security improvements can apply to the initial connection [3], [6]. SVCB and HTTPS records also allow for a more authoritative protocol negotiation and have stronger security guarantees but are more complex to deploy and depend heavily on DNS infrastructure, resolver behavior, and client support [3], [6].

Real-world measurements indicate that both mechanisms are used. Alt-Svc is useful for incremental rollout of new protocols [1], while SVCB and HTTPS records are provided by large CDNs and enable early connection improvements regarding performance and security.

Ongoing work is necessary to evaluate the adoption of SVCB and HTTPS RRs as client and resolver support improves [5]. Additional future work is needed to create best practice guidelines for the deployment of SVCB and HTTPS RRs to help avoid misconfigurations and to allow for a broader adoption across the web [5].

## References


[1] J. Zirngibl, P. Buschmann, P. Sattler, B. Jaeger, J. Aulbach, and G. Carle, "It's over 9000: analyzing early QUIC deployments with the standardization on the horizon," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 261–275, [Online]. Available: https://doi.org/10.1145/3487552.3487826.

[2] M. Nottingham, P. McManus, and J. Reschke, "HTTP Alternative Services," RFC 7838, April 2016, [Online]. Available: https://www.rfc-editor.org/info/rfc7838.

[3] B. Schwartz, M. Bishop, and E. Nygren, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)," RFC 9460, November 2023, [Online]. Available: https://www.rfc-editor.org/info/rfc9460.

[4] T. Matsuzawa and K. Ichikawa, "Implementation and Evaluation of HTTP/3 Connectivity Check Using Happy Eyeballs Algorithm," *Network*, vol. 2, pp. 389–397, 2022. [Online]. Available: https://doi.org/10.3390/network2030024

[5] H. Dong, Y. Zhang, H. Lee, S. Huque, and Y. Sun, "Exploring the Ecosystem of DNS HTTPS Resource Records: An End-to-End Perspective," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 423–440, [Online]. Available: https://doi.org/10.1145/3646547.3688410.

[6] J. Zirngibl, P. Sattler, and G. Carle, "A First Look at SVCB and HTTPS DNS Resource Records in the Wild," in *International Workshop on Traffic Measurements for Cybersecurity 2023*, ser. WTMC, 2023.

[7] 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.

[8] P. Hoffmann and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA," RFC 6698, August 2012, [Online]. Available: https://www.rfc-editor.org/info/rfc6698.

[9] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu, "When https meets cdn: A case of authentication in delegated service," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 67–82.


ISBN 978-3-937201-86-3
DOI 10.2313/NET-2026-05-1
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)