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

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

Summer Semester 2024

Februar 2, 2024 – August 18, 2024

Munich, Germany

Editors

Georg Carle, Benedikt Jaeger, Marcel Kempf, Leander Seidlitz

Publisher

Chair of Network Architectures and Services

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

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

Summer Semester 2024

Munich, Februar 2, 2024 – August 18, 2024

Editors: Georg Carle, Benedikt Jaeger, Marcel Kempf, Leander Seidlitz

Network Architectures
and Services
NET 2024-09-1

Proceedings of the Seminar
Innovative Internet Technologies and Mobile Communications (IITM)
Summer Semester 2024

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: kepmfm@net.in.tum.de
Internet: `https://net.in.tum.de/~kempfm/`

Benedikt Jaeger
Chair of Network Architectures and Services (I8)
E-mail: jaeger@net.in.tum.de
Internet: `https://net.in.tum.de/~jaeger/`

Leander Seidlitz
Chair of Network Architectures and Services (I8)
E-mail: seidlitz@net.in.tum.de
Internet: `https://net.in.tum.de/~seidlitz/`

Cataloging-in-Publication Data

Seminar IITM SS 24
Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)
Munich, Germany, Februar 2, 2024 – August 18, 2024
ISBN: 978-3-937201-80-1

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2024-09-1
Innovative Internet Technologies and Mobile Communications (IITM) NET 2024-09-1
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 Summer Semester 2024. 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 Dimitar Vasilev with the paper *Improving MassDNS: Adding CNAME Resolution Output Information* and Nils Lorentzen with the paper *Evolution of Wireless Security*.

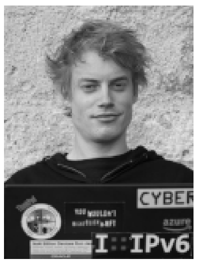
We hope that you appreciate the contributions of these seminars. If you are interested in further information about our work, please visit our homepage `https://net.in.tum.de`.

Munich, September 2024

Georg Carle

Marcel Kempf

Benedikt Jaeger



Leander Seidlitz

Seminar Organization

Chair Holder

Georg Carle, Technical University of Munich, Germany

Technical Program Committee

Marcel Kempf, Technical University of Munich, Germany
Benedikt Jaeger, Technical University of Munich, Germany
Leander Seidlitz, Technical University of Munich, Germany

Advisors

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

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

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

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

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

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

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

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

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

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

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

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

Johannes Zirngibl (zirngibl@net.in.tum.de)
Technical University of Munich

Richard von Seck (seck@net.in.tum.de)
Technical University of Munich

Seminar Homepage

`https://net.in.tum.de/teaching/ss24/seminars/`

Contents

Analysis of LEO-Satellite Fronthaul Schedulers

Yiran Duan, Eric Hauser*, Leander Seidlitz*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: yiran.duan@tum.de, hauser@net.in.tum.de, seidlitz@net.in.tum.de

Abstract—**Low Earth Orbit (LEO) satellite networks (LSNs), such as Starlink, OneWeb, and Kuiper, are developing rapidly, providing a new and more geographically equal possibility for internet access. LSNs consist of two parts: fronthaul is the connection between users and satellite constellations; backhaul links the constellations to the core network. The large scale of the constellations and high movement speed of LEO satellites make the scheduling problem distinct from that of geostationary networks. This paper focuses on the design concept of fronthaul scheduling algorithms for LSNs. As objectives of the scheduling algorithms, low latency, high capacity, wide coverage, energy efficiency, and fault tolerance are considered; the impact of various satellite parameters on these goals are discussed, including Angle of Elevation, direction, launch date and sunlit status. Since these factors have both positive and negative impacts on the objectives of scheduling, the design of scheduling algorithms involves trade-offs among different goals.**

Index Terms—**LEO satellite networks, LSNs, scheduling, Starlink**

1. Introduction

Low Earth Orbit (LEO) satellite networks, abbreviated as LSNs, are rapidly developing network technologies that provide commercial civilian network services. Currently, the three largest-scaled LSN projects include Starlink, OneWeb, and Kuiper [1], among which Starlink is already serving over 2.6 million users [2].

The altitude of LEO is below 2,000 km, compared to geostationary orbit (GEO) with 35,786 km. [3] Its comparatively short distance to ground stations allows a significant lower latency of ground-satellite communication, and less cost of satellite deployment. Relative to terrestrial networks, the service coverage provided by LSNs is wider and more independent of the geographic environment. Especially for maritime and remote areas, where ground-based stations struggle to cover, LSNs can offer much more affordable internet services than GEO-satellite based communication.

On the other hand, lower orbits lead to a limited terrestrial coverage by each single LEO satellite, as well as a higher velocity. The orbital period of LEO satellites typically ranges from 10 to 50 minutes [4], corresponds to a travel speed of ~27,000 km/h [3], which is over 30 times of the cruising speed of modern airliners. The scheduling between ground-stations and LEO satellite must thus be designed specially to cape with the fast and constant changes of the network topology. With these limitations caused by the low orbital altitude, one single LEO satellite is not helpful to provide network serviced. LSNs require thousands of LEO satellites to cooperate, in order to achieve efficiency and wide service area.

Table 1 summarizes the current and planned status of LEO satellite constellations of Starlink, OneWeb and Kuiper.

TABLE 1: Constellation design and status of Starlink, OneWeb and Kuiper

Project	**Number of Satellites**		**Orbit altitude**
	In operation	Planned	
Starlink	5564 [5]	42000 [6]	340-550 km [7]
OneWeb	632 [8]	6372 [9]	~1200 km [10]
Kuiper	2 [11]	3236 [12]	590-639 km [12]

This paper first provides an overview of the structure of LSNs, then focuses on the scheduler for user-satellite connections, highlighting the differences between LSNs scheduling and the relatively static terrestrial and GEO-satellite-based networks.

2. Architecture of LSNs

With the LEO satellites playing a central role, LSNs can be divided into two main parts: fronthaul, which is the connection between users and the LEO satellite constellation, and backhaul, through which the constellation connects to the core network. The backhaul involves the connection between LEO satellites and the ground stations, and from these ground stations to the core network through Points of Presence (PoPs).

Figure 1 illustrates the basic architecture of LSNs. More details about each component are described in Section 2.1; Section 2.2 expands on the different topology structures of connections between LEO satellites.

2.1. Composition of LSNs

As shown in Figure 1, components of an LSN include user terminals, LEO satellites, ground stations, PoPs and the core network. The functionality and technical configuration of each component can vary across different commercial projects and user group characteristics (e.g. users accessing the network services at sea versus those from remote areas on land). This paper primarily considers the configurations of Starlink for terrestrial network users [13], which is the most common scenario in current practical applications.

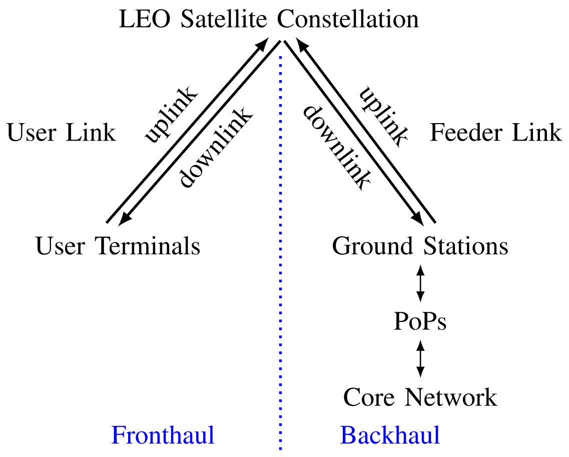


Figure 1: The Architecture of LSNs

In an LSN, user terminals, or dishes according to their common shape, communicate with LEO satellites, enabling individual or household networks to connect to the LSN. Each user terminal only has sight of a limited range of Angle of Elevation (AoE) of the satellites, and connects to only one of the LEO satellites in sight at the same time.

In contrast, each LEO satellite often connects to multiple user terminals simultaneously. This is reasonable considering their quantities: Starlink for instance utilizes less than 5.6×10^3 LEO satellites to serve over 2.6×10^6 users, as mentioned in the introduction.

Ground stations are also referred to as gateways. Their geographical distribution is designed strategically to ensure a high global coverage, while taking into account the different regulation of various countries and regions. The distribution of ground stations for commercial LSNs is usually not published, but according to an unofficial statistical report [14], Starlink now utilizes at least 150 ground stations, distributed across the globe.

LEO satellites connect to user terminals and ground stations through radio waves, with the frequency varying for different projects and applications [15]. The link between ground stations and satellites is called feeder link; the link between user terminals and satellites is referred to as user link. Uplink refers to the channels for sending signals from user terminals or ground stations to LEO constellations; downlink refers to the link from satellites to ground components. The connections between ground stations and PoPs, and those between PoPs and the core network, are mostly via optical fiber cables.

This paper focuses on fronthaul scheduling of LSNs, i.e. at any given moment, which satellite should each user terminal be connected to. This is one of the most complicated and distinctive processes of the scheduling within LSNs compared to other networks and influences user experience directly.

2.2. Topology Structure of Connections between Satellites

Inter-satellite connectivity is a key variable of LSNs' topology structure. The links within a LEO satellite constellation can either be set directly via laser or radio connection, or relayed by a ground station. The former is called inter-satellite links (ISLs). The scenario without ISLs and purely relies on ground station relayed links is called bent-pipe (BP) connectivity.

It is possible to provide a low-latency internet connection without ISLs [16], [17]. However, compared to BP-based LSNs, incorporation of ISLs can bring even lower latency, increase the network throughput and the resilience against bad weather, as well as provide more equitable network services with the same number of available ground stations [17].

Due to regulatory issues, ISLs have not been adequately integrated in existing LSN projects: ISLs via radio spectrum require licenses from the authorities, while laser based ISLs require the use of silicon-carbide components with a high melting point, thus possibly violating the "burn on reentry" requirement for LEO satellites [17]. Currently, Starlink is increasingly adopting ISLs in the constellations, and numerous researches on traffic scheduling algorithm for ISLs are carried on.

3. Fronthaul Scheduling for LSNs

Although the fronthaul scheduling algorithms of the existing and planned commercial LSNs are not open-source, researches have been conducted on the factors that should be considered in the design of scheduling methods [1], [13], the development of specific algorithms [18], [19], and the properties of currently operational scheduling algorithms of Starlink [13], [20].

Scheduling in LSNs includes many aspects, and the system models vary significantly depending on the extent to which ISLs are utilized. This paper focuses on the scheduling of satellites as a scarce resource for user link, which is one of the most critical parts of scheduling under the BP model. This section is organized as two Subsections: in Subsection 3.1, the objectives of scheduling algorithms are listed; Subsection 3.2 introduces various factors that should be considered in the fronthaul scheduling algorithms, and their impact on the goals considered in the previous subsection.

3.1. Goals of Scheduling Algorithms

The objectives of scheduling algorithms for LSNs include:

Low latency. Latency refers to the round-trip time (RTT) of a packet from the sender to the receiver. It measures how responsive the network connection is, and influences user experience directly. Low latency is a critical goal of network services, especially for real-time scenarios, such as online gaming and video conferencing. Starlink sets its goal to stable 20-millisecond median latency and considers the fronthaul scheduling as the major focus to improve the response speed of its service [2]. As of March 2024, Starlink provides most terrestrial regions an RTT between 25 to 75 milliseconds [2], [21], corresponding to that within North America for geostationary-based network services with distances ranging from 1000 km to 4000 km, according to a linear regression result based on observational data [22].

High capacity. The capacity of LSNs is limited by the number of satellites and the design of the constellations

[23], but for operational LSNs such as Starlink, this is generally not the bottleneck of the service, but rather an abundant and underutilized resource [24]. According to an estimation model [23], the capacity of LSNs can be seen as linearly proportional to the number of satellites in operation, with each satellite providing a data rate no larger than 10 Gbps. Based on this, with currently 5564 LEO satellites in use, Starlink's capacity is limited to approximately 55.6 Tbps, compared to global internet bandwidth at 1217 Tbps in September 2023 [25]. To utilize the capacity and achieve higher bandwidth, the scheduling algorithm should balance the dataflow in the LSNs, and pay special attention to the single-point bottleneck of bandwidth on the feeder link [26].

Wide coverage. The coverage of LSNs is mostly determined by the constellation designing, but should also be considered in the scheduling process, especially for regions lacking ground stations.

Energy efficiency. Energy efficiency is especially important to LEO satellites, not only from an environmental-friendly point of view, but also due to the limited energy resources acquirable in orbit, and that the charging and discharging process reduces the satellites' life time [27]. The energy for LEO satellites is mostly solar energy, and thus whether a satellite is being sunlit can make a difference for the scheduling algorithms.

Fault tolerance. As the deployment of satellites and ground stations is more time consuming than terrestrial networking, and due to the limited number of them, LSNs are more fragile than network services. Under various types of cyber attacks, LEO satellites and ground stations can be overloaded, isolated, or put into outage, causing a cascading failure [28]. Consequently, fault tolerance should be carefully considered in the design of scheduling algorithms.

3.2. Influence Factors of Fronthaul Scheduling

With the position of satellites changing constantly, the scheduling procedure of ground-satellite links must be decided frequently. An empirical research [13] on the behavior of Starlink shows that Starlink likely utilizes a global scheduler that plans the connections between user terminals and satellites every 15 seconds. Accordingly, the scheduling for user links should be simple enough to be executed swiftly while taking the goals mentioned formerly into account in order to achieve a high quality of service (QoS).

In the following, the mechanisms of four important influence factors are discussed. These factors are analysed in the experiment by Tanveer *et al.* [13], but this paper focuses more on how they are manifested at the Starlink user end, whereas this Subsection dives deeper into the rationale and related researches of them.

3.2.1. AoE (Angle of Elevation).

AoE of a satellite from some point on the earth refers to the angle between the horizontal line at that point and the line of sight pointing directly upwards to the satellite. Figure 2 illustrates the definition of it.

For satellites of the same orbit altitude, AoE decides its distance to points on the ground. From the law of cosines, the geometric relationship between them can be represented by formula 1:

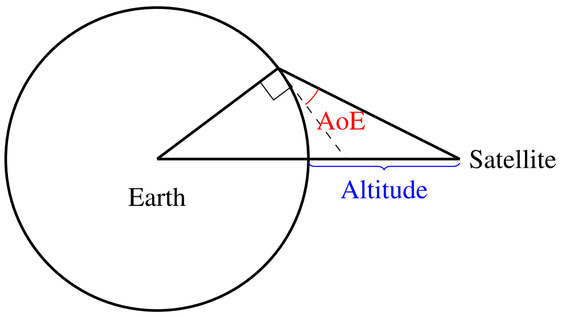


Figure 2: Angel of Elevation

$$(R+h)^2 = R^2 + d^2 - 2Rd\cos(\frac{\pi}{2}+\theta) \qquad (1)$$

Where θ represents AoE in the range of $(0, \frac{\pi}{2}]$, R is the radius of the earth (approximated as a sphere), h is the height or altitude of the satellite, and d is the distance between the observation point on the earth and the satellite. Through mathematical derivation, the expression for d as a function of AoE θ can be obtained as following:

$$d = \sqrt{h^2 + 2Rh + R^2\sin^2\theta} - R\sin\theta \qquad (2)$$

This function is strictly monotonically decreasing over $(0, \frac{\pi}{2}]$. This means that the distance between a point on the earth and a LEO satellite is always shorter for a greater AoE, and the minimum is met when $\theta = \frac{\pi}{2}$, i.e. when the satellite is directly above this point, and then the distance would be equal to the altitude of the orbit. The shorter distance brings lower latency and energy consumption, thus for user links, satellites with greater AoE for user terminal should be favoured by the scheduling algorithm.

3.2.2. Being sunlit.

The current operating LEO satellites are powered by solar energy [29]. When being sunlit, they use solar energy directly, and charge the on-board batteries if there is surplus energy; during the eclipse period, their only power source is the batteries. Battery lifetime is the bottleneck of the lifespan of the LEO satellites and is very sensitive to the depth of discharge (DoD), which describes the percentage of energy consumed during discharge relative to the total capacity of the battery. Quantitatively, by carefully designing the routing methods of ISLs, a reduction of 11% - 16% of DoD can be achieved, leading to a doubled battery lifespan [30]. Consequentially, the usage of satellites not being sunlit is to be avoided in order to reduce DoD as much as possible and, in turn, to increase the lifespan of the satellites' batteries.

3.2.3. Satellite Age.

LEO satellites have a life span of around 5 years [31] [13], which is very short compared to GEO satellites and terrestrial network infrastructure. Furthermore, LSNs are still in the early stages of development, with more and more satellites being deployed. Updates to both software and hardware of the satellites that could influence their functionalities and performance significantly are still taking place frequently, e.g. the satellites launched for Starlink constellation after September 2023 are equipped with more advanced optical space laser hardware than before [32], which is likely to improve the efficiency of ISLs. This being considered, the newer satellites should be favoured in the scheduling of user link.

3.2.4. Exclusion Zones. LEO satellites, as part of the non-geostationary-satellite system (NGSO), shall not cause unacceptable interference to GEO satellite networks, according to Article 22 of the ITU Radio Regulations [33]. As GEO satellites remains above a fixed point of the earth and communicates with the same ground stations, this regulation leads to several exclusion zones for LEO satellites, within which the satellites should reduce radio contact with ground stations or other satellites, in order to keep their Equivalent Power Flux Density (EPFD) under the regulated limitation.

This regulation is currently under controversy. Technical analyses [34], [35] point out that Starlink has likely been violating the EPFD limitations in some exclusion zones, leading to possible interference to specific GEO satellites. On the other hand, there are criticisms [36] that this regulation is outdated, not considering the progresses in the technologies of satellite communication, and reducing the economic benefits of LSNs. However, this opinion is opposed by Bazelon *et al.* [37]. Despite the controversy on the policy, the scheduling algorithm should avoid frequent allocation of links to LEO satellites within the exclusion zones.

3.2.5. Summary. In this subsection, four factors that significantly influence LSNs services are discussed. According to the experiment by Tanveer *et al.* [13], all of these factors are likely considered in the scheduling algorithms of Starlink. The impact of these factors on the goals of the scheduling algorithms mentioned in the previous subsection is complex: each factor can affect multiple design goals, some positively and some negatively. Thus, the design of scheduling algorithms needs to balance the trade-offs of different goals.

Table 2 summarises the overall influence of different factors to be considered in the scheduling algorithm, and their general impact on the goals mentioned in Section 3.1. Positive impacts are indicated with a "+", "-" is for non-favorable impacts.

TABLE 2: The Influence of each Factor on the Goals of Scheduling Algorithms for LSNs

	Higher AoE	**Being Sunlit**	**Newer Satellite**	**Avoid Exclusion Zones**
Latency	+	-	+	-
Capacity		-		-
Coverage				-
Energy	+	+	+	-

The preference for satellites with higher AoE will lead to a lower average distance of user links, and thus brings positive influence on the low latency and energy efficiency goals. The inclination to satellites being sunlit helps to improve energy efficiency, but generally leads to a suboptimal choice towards lower latency and underutilization of total capacity. The favoring for newer satellites can enhance energy efficiency, but it impacts latency in two different directions: on the one hand, it may lead to selecting satellites that are further from the user terminals, thus extending the latency; on the other hand, choosing satellites that equipped with more updated ISL hardware over the older ones allows better utilization of ISLs, thereby reducing latency. The algorithm is responsible to balance these factors and ensure a positive impact on reducing latency overall. In order to avoid radio transmissions within the exclusion zones, satellites on orbits crossing the exclusion zones cannot work full-time during the orbital periods, which leads to negative affections to all of these four goals.

4. Conclusion, Limitations, and Future Work

LSNs, such as Starlink, are providing network services worldwide, especially to ocean and remote areas, where ground-based networks struggle to cover within a reasonable budget. This paper first introduces the current commercial applications of LSNs, including Starlink, OneWeb and Kuiper, highlighting the difficulties and distinction of LSNs' scheduling problem.

Then, in Section 2, the basic composition and topology structure of LSNs are discussed. LSNs can be divided into front- and backhaul. LEO satellite constellations connect to user terminals and ground stations through user and feeder links. The topology structure within satellite constellations differs on whether ISLs are involved, which use laser or radio for direct communication between satellites, improving the transmission efficiency. The topology structure without ISLs is called BP structure, and it has been deployed in commercial LSNs. Currently, ISLs are not yet widely applied, but commercial LSNs like Starlink are paying great attention to them, relative researches and integration are conducted continuously.

Section 3 focuses on fronthaul scheduling. Its objectives and influence factors are analyzed, and summarizes their intertwined relationship, revealing the complexity of designing fronthaul scheduling algorithms. The discussion on fronthaul scheduling in this paper has the following limitations:

- Only ground components and LEO constellations are considered, flying vehicles and GEO satellites as users and relays are not included in the simplified model of the architecture of LSNs. However, studies [38], [39] show that they also have impact on the scheduling for LSNs.
- The application of LSNs as backhaul is not considered. Using LSNs as backhaul of mobile network operators, especially in remote areas, is being considered and studied [40], [41]. If the backhaul services share the same constellations with wideband services, they should be also considered in the scheduling algorithms, to balance the resources between different services.
- Scheduling for backhaul and ISLs is not included. To integrate fronthaul scheduling into the whole scheduling system for LSNs, the cooperation and cross influences of the scheduling for other parts also need to be analyzed.

As for future work, a simple fronthaul scheduling algorithm can be designed based on the concepts of this paper and then tested on simulation platforms such as

Hypatia [42]. The design of algorithms could be improved by taking the situations mentioned in the limitations into consideration with the system.

References

[1] W. Zhang, Z. Xu, and S. A. Jyothi, "An in-depth investigation of leo satellite topology design parameters," 2024.

[2] SpaceX, "Improving Starlink's Latency," https://api.starlink.com/public-files/StarlinkLatency.pdf, 2024, [Online; accessed 23-March-2024].

[3] D. Bhattacherjee and A. Singla, "Network topology design at 27,000 km/hour," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, ser. CoNEXT '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 341–354. [Online]. Available: https://doi.org/10.1145/3359989.3365407

[4] P. Yue, j. an, J. Zhang, J. Ye, S. Wang, G. Pan, P. Xiao, and L. Hanzo, "Low earth orbit satellite security and reliability: Issues, solutions, and the road ahead," *IEEE Communications Surveys & Tutorials*, vol. 25, pp. 1604 – 1652, 07 2023.

[5] J. McDowell, "Starlink Launch Statistics," https://planet4589.org/space/con/star/stats.html, [Online; Last updated 22-March-2024; accessed 23-March-2024].

[6] C. Henry, "SpaceX submits paperwork for 30,000 more Starlink satellites," https://spacenews.com/spacex-submits-paperwork-for-30000-more-starlink-satellites/, 10 2019, [Online; accessed 23-March-2024].

[7] M. Williams, "Starlink's satellites will be orbiting at a much lower altitude, reducing the risks of space junk," https://phys.org/news/2019-05-starlink-satellites-orbiting-altitude-space.html, 05 2019, [Online; accessed 24-March-2024].

[8] J. McDowell, "OneWeb Launch Statistics," https://planet4589.org/space/con/ow/stats.html, [Online; Last updated 22-March-2024; accessed 23-March-2024].

[9] OneWeb, "OneWeb Streamlines Constellation: OneWeb reduces constellation plans from 47,884 to 6,372 satellites," https://oneweb.net/resources/oneweb-streamlines-constellation, 01 2021, [Online; accessed 23-March-2024].

[10] ——, "First 5G end-to-end link recorded at ESA 5G hub using OneWeb constellation," https://oneweb.net/resources/first-5g-end-end-link-recorded-esa-5g-hub-using-oneweb-constellation, 06 2022, [Online; accessed 23-March-2024].

[11] J. McDowell, "Kuiper Launch Statistics," https://planet4589.org/space/con/kp/stats.html, [Online; Last updated 22-March-2024; accessed 23-March-2024].

[12] T. Kohnstamm, "Everything you need to know about Project Kuiper, Amazon's satellite broadband network," https://www.aboutamazon.com/news/innovation-at-amazon/what-is-amazon-project-kuiper, [Online; Last updated 30-October-2023; accessed 23-March-2024].

[13] H. B. Tanveer, M. Puchol, R. Singh, A. Bianchi, and R. Nithyanand, "Making sense of constellations: Methodologies for understanding starlink's scheduling algorithms," 2023.

[14] Starlink Insider, "Starlink Ground Station Locations: An Overview," https://starlinkinsider.com/starlink-gateway-locations/, 2024, [Online; accessed 31-March-2024].

[15] P. Yue, J. An, J. Zhang, J. Ye, G. Pan, S. Wang, P. Xiao, and L. Hanzo, "Low earth orbit satellite security and reliability: Issues, solutions, and the road ahead," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1604–1652, 2023.

[16] M. Handley, "Using ground relays for low-latency wide-area routing in megaconstellations," in *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, ser. HotNets '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 125–132. [Online]. Available: https://doi.org/10.1145/3365609.3365859

[17] Y. Hauri, D. Bhattacherjee, M. Grossmann, and A. Singla, ""internet from space" without inter-satellite links," in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, ser. HotNets '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 205–211. [Online]. Available: https://doi.org/10.1145/3422604.3425938

[18] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "Scheduling for ground-assisted federated learning in leo satellite constellations," in *2022 30th European Signal Processing Conference (EUSIPCO)*, 2022, pp. 1102–1106.

[19] Z. Yang, H. Liu, J. Jin, and F. Tian, "A cooperative routing algorithm for data downloading in leo satellite network," in *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, 2021, pp. 1386–1391.

[20] J. Garcia, S. Sundberg, G. Caso, and A. Brunstrom, "Multi-timescale evaluation of starlink throughput," in *Proceedings of the 1st ACM Workshop on LEO Networking and Communication*, ser. LEO-NET '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 31–36. [Online]. Available: https://doi.org/10.1145/3614204.3616108

[21] Starlink, "Availability & Speeds Map," https://www.starlink.com/map, 3 2024, [Online; accessed 31-March-2024].

[22] G. Martinez, J. A. Hernandez, P. Reviriego, and P. Reinheimer, "Round trip time (rtt) delay in the internet: Analysis and trends," *IEEE Network*, pp. 1–6, 2023.

[23] D. Shulakova, "Estimation of the starlink global satellite system capacity," in *Proceedings of International Conference on Applied Innovation in IT*, vol. 11, no. 1. Anhalt University of Applied Sciences, 2023, pp. 55–59.

[24] L. Liu, W. Liu, Y. Li, and H. Li, "Enabling 6g and beyond network functions from space: Challenges and opportunities," *IEEE Internet Computing*, pp. 1–9, 2024.

[25] P. Brodsky, "Total International Internet Bandwidth Now Stands at 1,217 Tbps," https://blog.telegeography.com/total-international-bandwidth-now-stands-at-1217-tbps, 9 2023, [Online; accessed 1-April-2024].

[26] L. Liu, H. Li, Y. Li, Z. Lai, Y. Deng, Y. Chen, W. Liu, and Q. Wu, "Geographic low-earth-orbit networking without qos bottlenecks from infrastructure mobility," in *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, 2022, pp. 1–10.

[27] C. Westphal, L. Han, and R. Li, "Leo satellite networking relaunched: Survey and current research challenges," *arXiv preprint arXiv:2310.07646*, 2023.

[28] L. Zhang, Y. Du, and Z. Sun, "Modeling and analysis of cascading failures in leo satellite networks," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 1, pp. 807–822, 2024.

[29] R. do N. Mota Macambira, C. B. Carvalho, and J. F. de Rezende, "Energy-efficient routing in leo satellite networks for extending satellites lifetime," *Computer Communications*, vol. 195, pp. 463–475, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422003486

[30] M. Hussein, G. Jakllari, and B. Paillassa, "On routing for extending satellite service life in leo satellite networks," in *2014 IEEE Global Communications Conference*, 2014, pp. 2832–2837.

[31] O. B. Osoro and E. J. Oughton, "A techno-economic framework for satellite networks applied to low earth orbit constellations: Assessing starlink, oneweb and kuiper," *IEEE Access*, vol. 9, pp. 141 611–141 625, 2021.

[32] E. J. Arevalo, "SpaceX Reveals Operation of Over 8,000 'Space Lasers' Across Starlink Satellite Constellation – Enabling Faster Internet," https://www.tesmanian.com/blogs/tesmanian-blog/starlink-video, 9 2023, [Online; accessed 1-April-2024].

[33] International Telecommunication Union, "Radio Regulations," http://handle.itu.int/11.1002/pub/814b0c44-en, 2020, [Online; accessed 2-May-2024].

[34] Viasat, "Technical Analysis: Starlink Violations of EPFD Limits (Fuchsstadt, Germany)," https://www.viasat.com/content/dam/us-site/corporate/documents/EPFD%20Results%20Germany%2020221029.pdf, 10 2022, [Online; accessed 2-May-2024].

[35] ——, "Technical Analysis: Starlink Violations of EPFD Limits (Chandigarh, India)," https://www.viasat.com/content/dam/us-site/corporate/documents/EPFD%20Results%20India%2020230409.pdf, 4 2023, [Online; accessed 2-May-2024].

[36] H. Furchtgott-Roth, "The economic benefits of updating regulations that unnecessarily limit non-geostationary satellite orbit systems," 2023.

[37] C. Bazelon and P. Sanyal, "Unacceptable interference: Economic analysis does not support degrading protections for gso networks," *https://dx.doi.org/10.2139/ssrn.4634764*, 2023.

[38] V. P. Kafle, M. Sekiguchi, H. Asaeda, and H. Harai, "Integrated network control architecture for terrestrial and non-terrestrial network convergence in beyond 5g systems," in *2022 ITU Kaleidoscope-Extended reality – How to boost quality of experience and interoperability*, 2022, pp. 1–9.

[39] J. Chen, M. Ozger, and C. Cavdar, "Nash soft actor-critic leo satellite handover management algorithm for flying vehicles," 2024.

[40] Vodafone, "World's First 5G Backhaul Demo over LEO Satellite," https://www.vodafone.com/news/technology/first-5g-backhaul-demo, 7 2019, [Online; accessed 3-April-2024].

[41] Z. Abdullah, E. Lagunas, S. Kisseleff, F. Zeppenfeldt, and S. Chatzinotas, "Integrated access and backhaul via leo satellites with inter-satellite links," 2023.

[42] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla, "Exploring the "Internet from space" with Hypatia," in *ACM IMC*, 2020.

B.A.T.M.A.N Unpacked: A Guide to the Protocol's Fundamental Concepts

José Afonso Gastaldi de Araújo Teixeira, Leander Seidlitz*
Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: joseafonsogat@gmail.com, seidlitz@net.in.tum.de

***Abstract*—B.A.T.M.A.N is a routing protocol designed for ad-hoc mesh networks. Throughout its development, various versions of the core algorithm were released that addressed issues of previous ones. This paper explains the core ideas behind the B.A.T.M.A.N protocol and highlights the differences between its different versions.**

***Index Terms*—B.A.T.M.A.N, ad-hoc mesh networks**

1. Introduction

Mobile ad hoc networks are often used as a replacement for traditional structured wireless networks where a client has to communicate directly to an access point that alone holds all the routing knowledge. In a mobile ad hoc network, each device (usually called a node) that is part of the network follows a protocol that allows it to relay information (packets/frames) to its local-link neighbours, which comprise all nodes that are close enough for direct communication. This creates a path across various nodes that ultimately allows two mutually distant nodes to communicate.

This type of network was first developed for military applications in the early 1970s [1] but as technology advanced and commercial devices such as cellphones and laptops got smaller, cheaper and more powerful, there was also more motivation to develop protocols that handle self-creating, self-organizing and self-administering networks that do not rely on fixed structures to function.

The B.A.T.M.A.N (Better Approach to Mobile Ad-hoc Networks) was developed by the Freifunk community in Berlin and since the release of kernel version 2.6.38 it is part of the official Linux kernel. This paper aims at explaining the core components of the B.A.T.M.A.N protocol and summarize the improvements that have come along with every generation of the protocol.

2. B.A.T.M.A.N

The development of the B.A.T.M.A.N protocol started around 2006 and, as of completion of this paper, includes 5 major versions/generations. One can think of generations I to V as the abstract ideas, the blueprints of the protocol, whereas B.A.T.M.A.N Deamon (batmand) and B.A.T.M.A.N Advanced (batmanadv) are the actual implementations/programms that turn the idea into reality.

B.A.T.M.A.N is a mobile ad hoc protocol well suited for mesh networks with unstable links. Naturally, other protocols such as the OLSR (Optimized Link State Routing) protocol exist, but what makes B.A.T.M.A.N stand out is that in this environment there is no need for nodes to gather knowledge about the state or topology of the network any time. Even though the algorithms of different generations differ from one another, the basic idea remains the same. Each node must only rely on the metadata received (or the lack of it) by Originator Messages (OGMs) that are broadcast in the network to decide about the best next hop to forward packets. How the nodes use the information from the OGMs varies from generation to generation because of the neighbour ranking system used in each one.

Next, we dissect the Originator Message and formally present each field contained in such packets used in B.A.T.M.A.N III. Later versions of B.A.T.M.A.N introduced new fields, but the ones from B.A.T.M.A.N III continue to be the backbone of newer OGM versions.

```
                 Originator Message (OGM) format.

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |U|D|           |      TTL      |    GWFlags    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Sequence Number        |            GW Port            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Layout of an Originator Message [1]

- **Version**: Value defined by the protocol; if a packet has a different `VERSION` number than expected, the packet is ignored.
- **Is-direct-link Flag**: Indicates whether the neighbour node is also a direct neighbour of the Originator or not.
- **Unidirectional Flag**: Indicates whether the neighbor node is bidirectional or not.
- **Time to Live (TTL)**: The maximum number of hops an OGM can undergo before it is dropped.
- **Gateway Flags (GWFlags)**: Set to 0 if the node is not an internet gateway; otherwise it contains the connection bandwidth (upload/download).
- **Gateway Port (GWPort)**: Set to 0 if the node is not an internet gateway; otherwise, it specifies the tunneling port to use.
- **Sequence Number**: The number that uniquely identifies the OGM in a given timeframe.
- **Originator Address**: The IPv4 (MAC address in B.A.T.M.A.N Advanced) of the B.A.T.M.A.N interface that generated the OGM.

These data are essentially all that is required by the protocols to establish the correct and efficient paths among nodes in the mesh, allowing them to make well-informed and timely routing decisions. (Fields related to internet gateways are explained in more detail in 6).

In the next sections of this paper, each neighbour ranking system and the differences in the format of the OGMs are explained in detail. It is also important to note that generations I and II are mentioned in B.A.T.M.A.N III's section since they are considered to be initial flawed prototypes in the early development of the protocol [2].

3. B.A.T.M.A.N III

In this section, a comprehensive explanation of how Originator Messages flood the network precedes the description of the neighbour ranking system of the third generation of B.A.T.M.A.N. [1], to facilitate a better understanding of the protocol.

3.1. How Does the Protocol Operate?

Every node in the network broadcasts Originator Messages periodically, with the purpose of announcing its existence to other nodes. Every OGM contains a sequence number which uniquely identifies it, enabling the distinction between new and duplicates OGMs. That way an OGM is only counted once.

An Originator (an interface of a node) will create an OGM with all the metadata necessary, as listed in section 2, and broadcasts it to all its local-link neighbours. These neighbours will again process the OGM and foward it to all of their neighbours. It is easy to see that the network will be flooded with OGMs until the packets are lost or the Time To Live (TTL) reaches 0 which will make a receiving node silently drop the packet.

A neighbour Ranking System was implemented based on the number of unique OGMs a neighbour has relayed from a specific Originator, where the neighbour with the most relayed OGMs is considered the current Best Link to the Originator. This ranking process is very dynamic and can change path as new OGMs are broadcasted.

Later when a node receives a standard non-OGM packet, it will know immediately which neighbour is the next best hop towards the destination and will forward the packet to that chosen neighbour, ultimately completing the purpose of the protocol.

3.2. Neighbour Ranking System

Each node maintains a list of Originators [1] it knows of, with some basic data collected about them from OGMs received. For each known Originator, an entry is created that contains all important information about this Originator, such as address, gateway capabilities, Host Network Announcement (HNA) list, and the current sequence number. For each Originator there is also a frame called a NBRF (Neighbour Ranking Sequence Frame) [2], nicknamed "sliding window", in which a fixed number of the last OGMs received from that specific Originator is stored (as an ordered list).

When a new OGM arrives from that specific Originator, a new entry is posted in the sliding window along with the identification of the neighbour from which the OGM came. If the window is full, the last entry is dropped. If the OGM's sequence number is greater than the current sequence number (current greatest), this field is updated with the new incoming sequence number.

The best ranking neighbour for that Originator is constantly re-evaluated, selecting the one with the most entries (OGMs received from) in the sliding window, accredited as the best next hop for communication with that specific Originator.

3.3. Generation I and II Design Flaws

Generations I and II are considered to be initial prototypes in the early development of the protocol [2]. Generation III is a direct upgrade that fixed the design flaws. Below are the main flaws of these versions.

3.3.1. Generation I. B.A.T.M.A.N does not check if a link is bidirectional or not. This is a design flaw because B.A.T.M.A.N only uses bidirectional nodes; this means that if a node receives packets from another node, it must also be able to send packets back. Unidirectional links are useless and a burden to the network under Generation I.

The simple fact of receiving a self-originated packet from a neighbour already suggests that that link is bidirectional, but to confirm this, a node will keep a record of the sequence number of its self-originated OGMs when broadcasting it to its neighbours. When the node possibly receives a self-originated OGM from one of its neighbours, it will check whether the sequence number is within a certain range (BI_LINK_TIMEOUT) of the node's current sequence number. If it is, and the Unidirectional Flag (UDF) is not set, then the check is successful. In Generation II a bidirectional link check was implemented.

3.3.2. Generation II. The inherent flaw with this version is that a node will retransmit all OGMs it receives from its neighboring nodes. This might not look like an issue at first, but when a node forwards OGMs from all its direct neighbours, it may create an illusion of quality that is not real, and can even create loops. Here is an example:

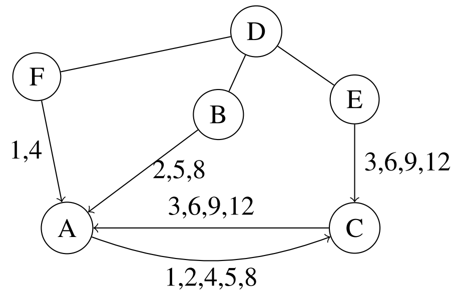


Figure 2: Not Best-Ranking Neighbour Problem [2]

The numbers next to the arrows correspond to the sequence numbers of OGMs originated from D. If a packet has to be forwarded from node C to D, it is easy to see that the best path towards D is through E. That is because E has relayed the most amount of OGMs from D, 4 in this case, whereas nodes B and F have just relayed 3 and 2 OGMs, respectively, which indicates packet loss through these paths. The issue is at node A, because it relays OGMs coming from both nodes, B and F, to C, resulting in A relaying 5 OGMs in total to C, which creates an illusion of it being a better path than it actually is from C's point of view. Generation II chooses A as the next best hop towards D, even though A is connected to weaker links.

To solve the issue a node must, upon receiving an OGM, only foward it if the neighbour the packet came from is the current best next hop towards the Originator, otherwise the packet is processed and then dropped.

4. B.A.T.M.A.N IV

B.A.T.M.A.N III's main problem concerns the asymmetric links between nodes. An asymmetric link between two bidirectional interfaces occurs when packet loss in one direction is different from that in the other direction, which may lead to a false analysis of the perfect next hop towards an Originator.

An example would be a network with nodes A, B, and C where all nodes are connected. OGMs from Node A propagate to B and to C. Since the links are asymmetric arbitrary packet loss values are chosen. B receives 100% of packets from A but A receives only 5% of the packets from B. C receives 90% of A's packets, which in turn also receives 90% of C's packets back. Finally, B and C receive 80% of the packets sent to each other. If the network is using B.A.T.M.A.N III, node B would think it has the perfect link towards A, which would be a single hop, but in fact this is not the best choice to make, as there is a huge packet loss from B directly to A. The better choice would be to hop to C and only then jump to A.

B.A.T.M.A.N IV [3] introduces a new concept of how to determine the best next hop towards an Originator, the quality of transmission, i.e., how likely it is a neighbour node will actually receive the packets sent. The key variables are Receive Quality (RQ), Echo Quality (EQ) and, of course, Transmit Quality (TQ) [4]. These metrics are always between a node and a specific neighbour.

- RQ: number of OGMs received from a specific neighbour (generated by this neighbour) divided by the number of expected OGMs in a given time frame.
- EQ: number of self-originated OGMs received back from a specific neighbour divided by the total number of self-originated OGMs in a given time frame.
- TQ: the node calculates this value by dividing the Echo Quality by the Receive Quality from its neighbour $\rightarrow EQ \div RQ$. It estimates the reliability of a transmission to a neighbour.

The Originator Message is also updated to carry a 1-byte long TQ value. When a new OGM is created the TQ value is set to the maximum and with each hop, this value is readjusted and tends to decrease. Three main aspects influence the adjustments of the OGM's TQ:

1) The local transmit quality of the node. When a node receives an OGM, it will multiply its local TQ towards the previous sender by the incoming TQ, which is the value found in the OGM. This way, each node receiving an OGM packet knows the TQ towards the Originator of the message. When forwarding the packet to other neighbours the updated OGM's TQ should be updated to $TQ = TQ_{\text{incoming}} \times TQ_{\text{local}}$
2) Penalties due to asymmetric links. Networks that use Wi-Fi, for example, use acknowledgment messages (ACKs) when a packet is sent, so the sender gets the confirmation that the message actually arrived, otherwise the sender will continue resending the message until a timeout occurs. If a node knows that the Receive Quality of packets towards a neighbour is low, it will adjust the TQ of the OGM before sending it to that specific node, as to: $TQ_{\text{new}} = TQ \times \text{asym}$ where $\text{asym} = (100\% - (100\% - RQ)^3)$
3) Penalties due to the number of hops. Even in a perfect scenario where every path from A to B holds a TQ of 100% it is still advantageous to choose the path with fewest hops, since more hops usually translate into more latency. BATMAN IV makes sure that at every hop the node receiving the OGM will decrease its TQ value by a fixed value, regardless of any other penalty before forwarding the packet.

Each node keeps "sliding windows" of types local and global. A node will have a local sliding window for each of its neighbours so it can track the last $TQ_{\text{LOCAL\_WINDOW\_SIZE}}$ (the amount of entries that the window can hold) OGMs received from that neighbour to calculate the current link quality. A node will also have a global sliding window for each originator that the node has knowledge of. Within this window there will be the average TQ values of each distinct neighbour leading to that specific originator.

In summary, nodes will use these sliding windows to make a competent analysis of which neighbour is best suited for the next hop towards the destination.

5. B.A.T.M.A.N V

B.A.T.M.A.N V [5] is the latest generation of the protocol by the time this paper is completed. It deviates from the packet loss metric implemented in the previous generation, as this time the focus is on packet throughput. It was observed that B.A.T.M.A.N IV is not optimal in dealing with meshes with nodes that present little to no packet loss, but diverse throughput capabilities. Furthermore, to diminish the nodes' overhead due to processing so many OGMs with such short intervals between them, a "divide and conquer" approach was used. The original OGM, used until generation IV, had the purpose of finding neighboring nodes through bidirectional checks and, most importantly, to flood the network with link quality information so that nodes could make better routing decisions. B.A.T.M.A.N V divides these functionalities into two distinct types of packets, ELP (Echo Location Protocol) packets and OGMv2 packets. Each of these two types of packets will be described in the following subsections.

5.1. ELP

These packets, similarly to OGMs, possess version numbers, sequence numbers, and the Originator's address. Once a neighbour receives such a packet, it will process it by first performing the necessary checks to ensure the validity of the packet, and then updating its Neighbours List with new information. If the neighbour is new and is not yet in the list, the list gains a new entry; otherwise, the `Last Time Seen` and `Last Sequence Number` fields of the existing entry for this neighbour are updated.

The key distinction with ELP packets is that after their processing, they will not be re-broadcast as an OGM would; their path ends after one single hop. (It is only used to keep the receiving node's Neighbours List updated.)

5.2. OGMv2

It is similar to the previous generation OGMs, but instead of carrying the TQ value, there is a new field for the throughput measurement. When a node interface receives this type of packet, it will perform the usual validity checks on version, source, destination and self-originated message. If the OGMv2 passes the initial checks, its sequence number is checked if it is within the range of a "protection window". The packet is dropped if the sequence number is too old or unexpectedly new.

The interface will then update the information it has about the Originator, such as the sequence number and last seen timestamps. Because the throughput of the path towards an Originator is as high as that of its weakest link, the throughput value may need to be updated if the throughput of the neighbour that receives the OGMv2 is lower than the throughput value in the OGMv2. There is also a 5.8% hop penalty applied to the throughput value in order to create a decreasing metric over multiple hops.

If the OGMv2 already comes from the current best neighbour towards the Originator, the packet is just re-broadcast. If it does not come from the best neighbour, but the OGMv2 throughput is higher than that of the best neighbour, the best neighbour is updated and the packet is re-broadcast. If the packet neither comes from the best neighbour nor has a higher throughput, it is not forwarded.

6. Connection to Outside Networks

Using the B.A.T.M.A.N protocol, standard nodes are able to communicate with distant nodes that are also inside the mesh and use the same protocol. That is, the protocol allows the creation of an isolated network, a bubble with no outside communication. Creating an isolated network is an accomplishment in itself; however, a scenario may arise where a client node desires to communicate with a server or another client located outside the mesh, for instance, on the internet. To allow a B.A.T.M.A.N mesh network to interact with other networks, there can be special nodes put in place in the network to provide additional functionality. These nodes can also function as an internet gateway, which is basically a bridge between two separate networks. To announce that a node is also a gateway, the node must declare this functionality when creating its OGMs and add the correct information in the following fields:

- **GWFlag:** Specifies the upload and download speeds in kbit per second;
- **GWPort:** Specifies the port number for tunnel communication with the gateway node;
- **HNA Extension Messages:** These are appended after the OGM and contain the (outside) network address and a netmask.

A client node can receive OGMs from multiple gateway nodes and decide which one they want to establish a connection with, based on connection quality or speed, or even a mixture of both parameters. It is common for the client to set his/her preferences manually, but an auto-selection mechanism is also available if needed.

For B.A.T.M.A.N generations that work on the layer 3 of the OSI model, a client node needs to encapsulate the Internet data in an UDP/IP datagram. This means that packets destined for the outside network must be wrapped into UDP packets when forwarding the data to the gateway node. As mentioned earlier, gateway node OGMs will specify which port number should be used for gateway functionality. Upon receiving a packet, the gateway node will read the port number from the outer UDP header and if the port number is correct, it will “decapsulate” the packet, keeping only the original IP packet to forward it to its final destination.

7. B.A.T.M.A.N Advanced

One can think of generations I to V of B.A.T.M.A.N as the abstract ideas, the blueprints of the protocol. B.A.T.M.A.N Deamon and Advanced are the actual implementations/programms that turn the idea into reality. B.A.T.M.A.N Deamon is the name of the older implementation of the protocol and operates on layer 3 of the OSI model. In contrast, B.A.T.M.A.N Advanced currently uses B.A.T.M.A.N IVbut also operates on layer 2. This means that instead of using IPs and packets, it encapsulates data and fowards them as raw Ethernet frames using MAC (Media Access Control) addresses to route them across the mesh until they reach their intended destination node. Since this implementation uses MAC addresses, one can make the comparison between the mesh network and a virtual switch. Each node can be interpreted as a port of this switch, so when non-mesh nodes connect to a mesh node, they get the illusion of having connected to a switch port, thus being in a local network. The underlying network topology invisible to them.

7.1. Data Forwarding in B.A.T.M.A.N Advanced

B.A.T.M.A.N Advanced excels at using different network interface types to foward data. Based on the link qualities defined by the algorithm B.A.T.M.A.N Advanced can choose which hard interface of a node is best suited for communication (e.g. Wi-Fi or Ethernet) in order to ensure the best path. This also means another interface outside the mesh can be bridged to a mesh mode with the bat0 interface which will then seamlessly foward the data. B.A.T.M.A.N Advanced uses raw Ethernet frames so it is not possible to just send that type of data over Wi-Fi because Wi-Fi and Ethernet have different formats and protocols. Here is an example of what a communication through a Wi-Fi connection:

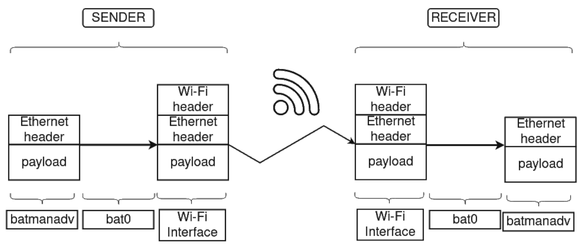


Figure 3: B.A.T.M.A.N Advanced encapsulation process

In the figure above, node A is the sender and B the receiver. B.A.T.M.A.N Advanced in node A will craft the raw Ethernet frame and then the Wi-Fi driver will encapsulate the data into a Wi-Fi frame. The data will then be sent from A’s Wi-Fi card to B’s. The Wi-Fi interface will receive the data, the driver will decapsulate it and deliver it to the bat0 interface. After that the B.A.T.M.A.N Advanced protocol from B will take over and process the data in its desired format.

7.2. Distributed ARP Tables

A mesh network using B.A.T.M.A.N Advanced can be used as an intermediary that connects non-mesh clients. The problem is that these clients will only know each other's IP address, but not their MAC addresses, which are the key data necessary for communication on B.A.T.M.A.N Advanced networks.

In the traditional case, when a non-mesh client linked to mesh node A wants to communicate with one linked to mesh node B, it send an ARP request to A. A broadcasts the request until it eventually reaches B, which would respond with the MAC, or else the packets are lost, which will eventually result in A having to wait for a timeout before requesting again.

The advantage of using a distributed ARP Table [6] is that after the initial misses, a cache memory of IP to MAC entries is stored in the nodes. This means that groups of nodes are able to cache subsets of entries (IP & MAC) of the non-mesh clients they are linked to. Thanks to a distributed hash fuction, even if a node does not have the needed entry, it will know exacly where the entry could be found. Given an IP address to any node, it will apply the hash function and forward the request to a group of nodes where the key-value pair (MAC,IP) is stored. This allows unicast of the ARP requests, thus the likelihood of packet losses is significantly lower.

8. Conclusion

In conclusion, the B.A.T.M.A.N protocol and its current implementation in B.A.T.M.A.N Advanced as a Linux Kernel driver is a viable option for networks where a fixed infrastructure is not trusted, too expensive or unreliable such as in military operations scenarios or in areas of natural disasters. The protocol has good application in these areas because the protocol is suited for unreliable nodes that can unexpectedly go offline since the protocol will automatically find a substitute route from point A to B without severing the connection, something that makes it an invaluable tool for creating self-healing networks. One of the most relevant examples of its practical and successful application is the Freifunk Community in Germany, which has been using the B.A.T.M.A.N protocol for its mesh networks throughout many German cities in order to provide free Wi-Fi, demonstrating the protocol's potential to facilitate community-driven, decentralized networking solutions on a larger scale.

References

[1] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, "Better approach to mobile ad-hoc networking (b.a.t.m.a.n.)," Internet Engineering Task Force, Internet-Draft draft-openmesh-b-a-t-m-a-n-00, Mar. 2008, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-openmesh-b-a-t-m-a-n-00.txt

[2] A. Neumann, C. E. Aichele, and M. Lindner, "B.a.t.m.a.n. status report," Tech. Rep., Jun. 2007. [Online]. Available: https://downloads.open-mesh.org/batman/papers/batman-status.pdf

[3] "Batman-iv," https://www.open-mesh.org/projects/batman-adv/wiki/BATMAN_IV, 2024, accessed: 26.03.2024.

[4] A. Quartulli and R. Lo Cigno, "Client announcement and fast roaming in a layer-2 mesh network," *Technical Report DISI-11-472, Università degli Studi di Brescia*, October 2011, version 1.0. [Online]. Available: https://www.researchgate.net/publication/265010299_Client_announcement_and_Fast_roaming_in_a_Layer-2_mesh_network

[5] "Batman-v," https://www.open-mesh.org/projects/batman-adv/wiki/BATMAN_V, 2024, accessed: 26.03.2024.

[6] "Distributed ARP Tables," https://www.open-mesh.org/projects/batman-adv/wiki/DistributedARPTable, 2024, accessed: 26.03.2024.

Increasing Throughput & Redundancy with Multipath QUIC

Hyeon Su Kim, Marcel Kempf*, Kilian Holzinger*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: hs.kim@tum.de, kempfm@net.in.tum.de, holzinger@net.in.tum.de

Abstract—**With the increasing amount of connected devices over the years, existing internet protocols have been changed and new protocols have emerged. One of the most recent protocols is the QUIC protocol. QUIC is a transport protocol designed to overcome the shortcomings of the more commonly used Transmission Control Protocol (TCP).**

Although QUICs functionalities serve as a solid foundation for a connection, dedicated working groups have wondered about the establishment of a Multipath QUIC connection. In this paper various versions of the IETF draft as well as practical implementations of Multipath QUIC will be analyzed. First the mechanisms that the current IETF draft specifies will be summarized and important keypoints during the development will be pointed out and analyzed. And lastly the MPQUIC extension will be evaluated based on potential use cases and the resulting benefits as well as concerns that arise.

Index Terms—**QUIC, Multipath Transport Protocol, Multipath TCP**

1. Introduction

Although TCP has been the most commonly used reliable transport protocol for a long time, many points of improvement have been noticed. QUIC is a reliable transport protocol with additional security functionality that addresses these points and deploys different mechanisms to improve on these points [1, section 1].

Development on QUIC began in 2012 and was finally standardized in 2021 by the IETF [2]. The protocol offers a reliable connection between two hosts that claims to be faster and more secure than the TCP and TLS stack. While QUIC offers a reliable connection with one path, without modification it lacks the ability to utilize more than one interface concurrently. Therefore a dedicated group of researchers have been working on an extension that provides the ability to form a QUIC connection between two hosts with the concurrent use of multiple paths [3, section 1]. MPQUIC aims to utilize up to all network interfaces of a hosts machine in order to establish a multipath connection by mostly reusing the QUIC protocol and avoiding changes to the protocol. This concept has been a work in progress since 2017 and is heavily worked on by the dedicated group.

The anticipated potential for an even higher throughput of data and even higher resistence against network location changes awakes interest in this extension.

In this paper, we will observe the past and present drafts [3] in order to achieve an understanding of the operations that take place with the deployment of the extension and the problems and disagreements that occured during the development of the draft. Afterwards we will compare MPQUIC to other multipath transport protocols such as MPTCP [4]. The last section this paper evaluates the current MPQUIC draft by analyzing use cases in terms of the benefits and stating concerns about the extension in its current form.

This paper offers insight into similar works such as the archived MPQUIC IETF drafts [3] and the original paper proposing the MPQUIC draft [5].

2. Background

This section aims to establish background knowledge about MPQUICs multipath operations and additionally QUICs mechanisms, since these are vital in order to understand MPQUICs procedures.

2.1. QUIC

One of the interesting features of QUIC is connection migration. A devices' network location can change due to middleboxes assigning new network parameters or the device changing the physical location to another network. These circumstances can cause issues to TCP. Connections between machines are classified by the 4-tuple, consisting of the ip-address and the port number of both hosts. TCP can classify the managed connection only by these 4 values [6, Section 3.1] and a change in one of those will identify the previous connection as a new one. The connection will be discarded and eventually the disconnected host will have to reconnect to the service. This handling of the changed parameters is highly inefficient and QUIC provides a better way of handling such an event.

QUIC uses the Connection ID parameter in order to identify the connection and the 4-tuple to identify the path [1, Section 3.1]. In a situation where the 4-tuple changes, the protocol can identify the changed connection as a migrated one. After validating the new path with a simple challenge and response procedure the modified connection can be used again [1, Section 9]. This validation procedure is called Path Validation and it is mandatory for any migrated connection that needs to be used again [1, Section 9]. The packets that contain these frames are classified as probing packets.

QUIC defines a unique packet structure that consists of packet headers which carry a variable amount of frames. One packet consists of a UDP header that carries one or more headers. There are two types of headers defined in QUIC, the short header and long header [1, Section 17]. Frames serve a single function or carry one parameter that are utilized by the transport protocol for certain functionalities. The most relevant frames in this paper are the ACK frame, the PATH_CHALLENGE, PATH_RESPONSE, and the RETIRE_CONNECTION_ID frame. The ACK frame acknowledges the last received frame, PATH_CHALLENGE and PATH_RESPONSE frames carry the data for the Path Validation procedure and RETIRE_CONNECTION_ID informs the peer that the sending host will no longer use the connection id that was agreed upon during the handshake [1, Section 19].

A unique feature that QUIC deploys is packet protection. With the assistance of TLS, packets are encrypted with Authenticated Encryption with Associated Data (AEAD) [7, Section 5]. Version Negotiation packets are the exception to this procedure and will not be encrypted.

QUIC also implements features that reduce latency [1, Section 7, 13]. Since these features are unaffected by the extension, it is sufficient to know solely about the existence of the features.

2.2. Multipath QUIC

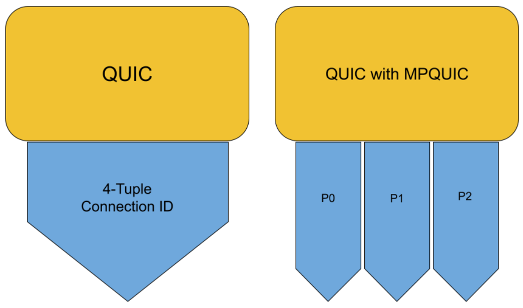


Figure 1: Structure of QUIC and MPQUIC

The Multipath extension aims to establish a connection between two hosts with multiple paths. Path in this context is the connection between one network interface to another [3, Section 1]. This is realized by instantiating paths that are different from each other in their respective 4-tuple, Destination Connection ID and packet number space [3, Section 4.4]. With more than one path being utilized, it requires operations such as RTT-measurement to be processed on each path individually.

2.3. Multipath Connection Establishment

The Multipath QUIC extension is only usable after successful negotiation during the handshake phase. Note that during this phase, the deployed transport protocol does not use any mechanisms of MPQUIC. The following conditions have to be met during the handshake phase for a successful negotiation:

- each of the hosts provide a set value for the transport parameter enable_multipath
- each of the hosts use a non-zero length connection id for source and destination
- negotiated cipher suites are at least 12 bytes long

If either of these conditions are not met, the extension can not be used [3, Section 3] and it is highly likely that a single path QUIC connection will be established.

Additionally during the handshake the parameter active_connection_id_limit dictates how many paths can be established.

2.4. Multipath Operations

Path instantiation is done by sending a non-probing packet to the server from unused path. The server then detects this as an attempt to create a new path and sends a packet with a path challenge in order to validate the new path. Only after successfully responding to the challenge the newly founded path can be used.

Each of the hosts can notify the peer on which path they would like to receive data from. Hosts can notify the paths preference with the frames PATH_AVAILABLE and PATH_STANDBY. The first frame signals that the specified path can be used to send data and the second frame signals that the peer would preferably not like to receive data on the path [3, Section 4.2].

It is also possible to close a path that no data transmission takes place on the specified path. The conventional way to close a path is to send a PATH_ABANDON frame. After receiving this frame, the host has to wait for three times the probe timeout interval for potential packets inflight that are related to the closed path. After receiving the acknowledgement for the PATH_ABANDON frame a RETIRE_CONNECTION_ID frame is sent to finally release the resources for the closed path [3, Section 4.3].

Other events that close a path are closure of the entire connection through CONNECTION_CLOSE frame or a stateless reset [3, Section 4.3].

A special case of closure is the sole usage of RETIRE_CONNECTION_ID frame which renders the specified connection id unusable. The associated path can't be used anymore, unless there is another connection id to be assigned. If this frame is used the sending host should consider inflight packets related to the retired connection id and a waiting period should be considered [3, Section 4.3.3].

Another reason for a path closure would be for an idle timeout. This is detected by the lack of non-probing packets received or lack of acknowledgements from sent packets [3, Section 4.3.4].

Each path has to be managed separately and operations such as path RTT-measurements and congestion control must be processed for each individually path. As previously mentioned, each path has its own packet number space that packets with number N can be observed on each path. This complicates the identification process of packets since packet numbers can not be used as a unique identifier anymore. Therefore MPQUIC uses additionally the Destination Connection ID in order to identify packets [3, Section 5].

In total, there are 4 different states that a path can be classified with. A path with the Validating state refers to a path that has been just instantiated by either a sent or received PATH_CHALLENGE frame. The path then transitions over to the Active state when a response has been received. These paths can be used and preferences can be set to them just as mentioned before. A path then transitions to the Closing state when the host sends a PATH_ABANDON frame. And at last, according to the IETF draft [3, Section 4.4] there are 3 different events that lead to the Closed state:

- Path Validation process failure
- Timeout during the Active state
- Sending a RETIRE_CONNECTION_ID frame

An important note to MPQUIC that one must keep in mind is that the ACK_MP frame does not have to follow the same path that it acknowledges [3, Section 5.1].

2.5. Packet Number Spaces

The draft states that each path has their own packet number space. With the change from single to multiple Packet Number Spaces, some of the mechanisms that are dependent on the Packet Number are modified.

MPQUIC packets carry the Destination Connection ID in addition to the packet number for association to the correct path [3, Section 5]. With the addition of Destination Connection ID the ACK_MP frame has to include the Destination Connection ID sequence number in order to acknowledge packets for the specified path [3, Section 5.1].

The AEAD for the packet protection normally requires the packet number for the calculation of the nonce. This changes when using MPQUIC since packet numbers can occur multiple times per path meaning the nonce is not unique for the processed packet. Therefore AEAD mechanism under MPQUIC calculates the nonce additionally with the destination connection ID sequence number [3, Section 5.2].

2.6. Scheduling

The draft vaguely specifies the necessity of a scheduler with the usage of Multipath. The general scheduling procedure consists of manipulating congestion windows of the active paths and distributing the packets accordingly. The distributing logic of the scheduler does not affect control frames due to their urgency.

An important note is that the choice of scheduler algorithm depends on the application and potentially the role of the hosts as either the client or the server [3, section 7.4].

3. Development

In order to understand the design choices during development this section will cover aspects of MPQUIC that were especially difficult to handle during development of the several versions of the ietf drafts and the versions of QUIC implementing the extension. Work on an IETF draft document started in 2017 by Quentin De Coninck and Olivier Bonaventure. In 2020 two additional versions covering the Multipath QUIC extension were created. One authored by Christian Huitema and Mirja Kühlewind, the third draft document by Yanmei Liu and Yunfei Ma from the Alibaba coorperation. Those 3 drafts are different in the technicalities in how the multipath connection is created. In 2021 the three drafts were merged into one with ideas from each of the drafts influencing the final version.

All of the created drafts share some common ground:

- negotiation during handshake with the enabling parameter
- validation of a path before usage
- notifying preferences for data reception on certain paths
- a communication flow is not limited to one path
- consideration for a scheduler

Even though all three groups goal were equal, there were significant differences between the drafts. Finally in 2021 all three branches of drafts were merged into one with clear implementation instructions that compromise the ideas of each draft. During the development, two features of the extension brought up considerable amount of discussion: Packet Number Spaces and Path Identifier.

3.1. Single or Multiple Packet Number Spaces

During the development period the 3 branches of drafts each implemented different mechanisms:

- one single Packet Number Space for all paths
- separate Packet Number Spaces for each path
- option to use both Packet Number Spaces

The reason for experimentation with the two methodologies is due to the different advantages that these offer. In [8, Section 1] it is stated that the advantage of a unified Packet Number Space is the ability to hide the Connection ID and using a zero length Connection ID. Another advantage that Christian Huitema brought up was that technically the default QUIC was already aware of multiple paths with the connection migration feature and also used a single Packet Number Space for processing packets [9]. Therefore the implementation of multiple Packet Number Spaces would have seemed like a severe modification to the QUIC protocol.

Unfortunately acknowledgements with a shared Packet Number Space are more difficult to handle since the in-flight packets can not be expected to be received in order due to the different paths that the packets can travel through. Even with several countermeasure options stated in the draft [8, section 7.1.1], 4 implementation difficulties were found and stated in an Email by Yunfei Ma [10]:

1) Inaccuracy with RTT measurements
2) ACK range with holes of significant size
3) degrading speed with increasing data size
4) ACK size can be suppressed

With this technical report the support for multiple Packet Number Spaces increased significantly since the issues could not be ignored and further development on the extension was slowed.

Fortunately, separated Packet Number Spaces did not present with the same issue. Due to the addition of identifying parameters of the paths, the ACK_MP frames have to include the identifier in order to adopt the same acknowledgement procedure as the default QUIC protocol [3, Section 5.1]. Although the disadvantage of exposing the connection ID and the path identifier is significant, the latest draft has adopted multiple Packet Number Spaces and offers no option to use a single one [3].

The adaption of either single or multiple Packet Number Spaces was heavily discussed among the QUIC working group and due to disagreements the decision was put to a poll [11] and due to overwhelming support for the removal of single packet number space, it was removed in 2023.

3.2. Path Identifier

During development the Path Identifier was created as a parameter identifying the path. In 2023 path identifiers were removed from the draft and implementations that implement the ietf MPQUIC draft. The reasons for the removal were the added complications when considering the usage of connection IDs and Path IDs [12]. More accurately the usage of path ID relies on the 4-tuple which is unreliable as an identifier in the current network structure. Additionally it is difficult to differenciate the events where the 4-tuple changes with a path ID [12]. Therefore the working group decided to use the Destination Connection ID sequence number for path identification.

However the use of Destination Connection ID sequence number seems to be a short-term solution. This parameter is not static for a long period of time and changes frequently due to e.g. Connection ID rotations. Moreover since the Packet Number Space is bound to the Connection ID, changes to the Connection ID would also affect the Packet Number Spaces. Therefore in [13] Marten Seemann proposes an explicit Path Identifier. The new Path ID is an independent parameter that is solely bound to the path that it is related to. This would also allow the calculation of Packet Number Spaces without the concern of involuntary changes. are unrelated to the paths directly. This explicit Path Identifier was eventually tested and presented on the 119 IETF meeting [14] and it is highly likely that the IETF draft will implement the new parameter in the future.

4. Evaluation

This section will cover the evaluation of MPQUIC. This includes comparison with MPTCP, existing implementations and use cases.

4.1. Comparison with MPTCP

Many similarities between the two extensions are easy to notice. Both extensions can not be used without a successful negotiation during their respective handshake phase. Additionally both extensions map the sent packets to the different paths in order to acknowledge packets according to the paths [4].

While both MPQUIC and Multipath TCP (MPTCP) establish multiple paths with their respective transport protocols, the structure is slightly different.

As mentioned in the previous section the MPQUIC extension enhances the default QUIC connection to be mapped over more than one path. This means that there is only one instance of a reliable transport protocol running at both hosts devices [3]. MPTCP manages multiple so called subflows that are on surface a whole TCP connection. Therefore the MPTCP extension is an additional protocol that governs over several TCP instances [4].

4.2. Implemetations

In total there are 3 open-source QUIC libraries that actively follow the IETF draft and implement the specified mechanisms: picoquic, quiche and xquic.

The MPQUIC extensions are implemented in each of these repositories.

All repositories implement the current version of the MPQUIC draft with the base functionalities such as modified handshake, initializing new paths, path management. It should be noted that xquic is the only version implementing various schedulers [15]. In fact xquic has the shortest development cycle from all three versions. This could be attributed to the funding that the repository receives from Alibaba inc [16].

4.3. Use Cases

With the availability of the Multipath QUIC extension various applications would make use of the extended transport protocol. Applications in need of high throughput will highly favor the protocol since servers could make use of its multiple network interfaces in order to deliver data at a faster rate. Especially services that benefit greatly from a consistently high data rate such as video streaming services are potential users of MPQUIC.

Another use case that can come to mind is to build dedicated communication flows. Since MPQUIC allows to notify about the preferences of paths in terms of reception of data, one can build an application that uses a dedicated path for receiving data. This means with dedicated control the dataflow of sending and receiving packets do rarely collide leading to a smoother operation.

Applications that are dependent on the constant availability of the service will benefit greatly from the use of MPQUIC. Due to the awareness of past open paths hosts can react faster to a change in network addresses. For example, an application on a smart phone would be able to detect a departure from the local wireless network. From this deduction the application could open a second path utilizing the LTE capability. Outside of the wireless networks range the application would still have an active connection and can maintain a data exchange without any delay.

4.4. Concerns

The main concern for MPQUIC would be its security. Moreover using MPQUIC could leave the protocol more vunerable to spoofing atempts. In the default QUIC protocol there are several countermeasures against spoofing atempts such as the limit on a newly migrated connection and ability to detect such an attempt [1, section 9.3]. In the current IETF draft [3] there is neither a way to detect a spoofing attempt nor is there a countermeasure to protect against such attacks. Especially concerning is the path initialization procedure that only requires a packet to be sent [3, section 4.1]. This means that a malicious thirdparty could potentially open a new path and the server would have difficulties to distinguish between the legitimate client and the third party.

Another concern that should be mentioned is the increase of congestion through the use of MPQUIC. A simple scenario would be a popular video streaming service that uses the MPQUIC extension. While the service would utilize the increased throughput of the transport protocol, it would also mean that the immediate network connections would be more congested. Especially at times when the streaming service would see increased usage the network would experience a huge increase of data flow and would therefore be more congested.

5. Conclusion and future work

The Multipath QUIC extension realizes the aim for high throughput and redundancy successfully. Especially promising are the ongoing findings in the IETF MPQUIC drafts [3] and discussions in the IETF mailing list [17]. These ongoing improvements build confidence that concerns and issues will be addressed soon. With this speed of development one can imagine that the stanardization will take place in the near future. With the standardization many application developers that either are already using QUIC or are in search of a realiable Multipath Transport Protocol will benefit greatly from the extension and common users of services that require transportation of data will be able to experience better services going forward.

References

[1] 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

[2] Wikipedia contributors, “Quic — Wikipedia, the free encyclopedia,” 2024, [Online; accessed 28-March-2024]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=QUIC&oldid=1215763552

[3] Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, “Multipath Extension for QUIC,” Internet Engineering Task Force, Internet-Draft draft-ietf-quic-multipath-06, Oct. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/06/

[4] A. Ford, C. Raiciu, M. J. 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

[5] 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

[6] W. Eddy, “Transmission Control Protocol (TCP),” RFC 9293, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9293

[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] Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, “Multipath Extension for QUIC,” Internet Engineering Task Force, Internet-Draft draft-lmbdhk-quic-multipath-00, Oct. 2021, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-lmbdhk-quic-multipath/00/

[9] QUIC IETF Mailinglist. (2020) Re: Preparing for discussion on what to do about the multipath extension milestone. [Online]. Available: https://mailarchive.ietf.org/arch/msg/quic/licsYSzeigfuLxnizDDv-r0dlVE/

[10] ——. (2022) Alibaba’s Technical Report of single packet number space implementation (SPNS) for multi-path QUIC. [Online]. Available: https://mailarchive.ietf.org/arch/msg/quic/LiEYJ2k8ldbz-SWeiXygeRAGHdk/

[11] ——. (2023) Consensus call: Removing 0-length CIDs (Single Packet Number Space) From MPQUIC. [Online]. Available: https://mailarchive.ietf.org/arch/msg/quic/_Dsr-cwm055bRzRKU-hxsVWjYqI/

[12] IETF MPQUIC Working Group. (2023) draft-ietf-quic-multipath-04 presentation slides. [Online]. Available: https://datatracker.ietf.org/meeting/116/materials/slides-116-quic-multipath-quic-00

[13] ——. (2023) separate Path IDs from Connection IDs. [Online]. Available: https://github.com/quicwg/multipath/issues/214

[14] ——. (2024) Draft-ietf-quic-multipath-ietf199. [Online]. Available: https://datatracker.ietf.org/meeting/119/materials/slides-119-quic-multipath-quic-00

[15] Alibaba inc. (2023) xquic schedulers. [Online]. Available: https://github.com/alibaba/xquic/tree/main/src/transport/scheduler

[16] ——. (2023) xquic schedulers. [Online]. Available: https://github.com/alibaba/xquic

[17] IETF MPQUIC Working Group. (2023) MPQUIC Mailinglist. [Online]. Available: https://mailarchive.ietf.org/arch/browse/quic/?q=multipath

ISP networks - Differences and Challenges

Johannes Rief, Florian Wiedner*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: johannes.rief@tum.de, wiedner@net.in.tum.de

***Abstract*—This paper presents information about Internet Service Provider (ISP) Networks and their challenges. It takes a closer look at the network structure and covers routing technology such as Internet Protocol Addresses and Multiprotocol Label Switching. Furthermore the paper presents Content Delivery Networks and Data Center Networks and how they work together with ISP networks to offer services over the internet to users. Finally we will present general difficulties that can occur when operating an ISP Network and general attributes of the Networks that are worth optimising. These include network security and internet connection in rural areas.**

***Index Terms*—ISP: Internet Service Provider; IP: Internet Protocol; MPLS: Multiprotocol Label Switching; CDN: Content Delivery Network; DCN: Data Center Network**

1. Introduction

Internet Service Providers (ISP) are essential for todays structure of the Internet because they provide households as well as companies an access to communication via the Internet. Without them, the Internet as we know it today would not exist. They are normally supervised by companies like Telekom and Telefonica in Germany or AT&T and Comcast in the United States and form the "Internet Backbone", a large scale network connecting most parts of the world [1]. Because of the importance of consistent communication especially in the business sector, ISPs have to ensure stability in their network and provide high speed connections. In this paper we will first try to give a structural overview of ISP networks and routing technology like Multiprotocol Label Switching (MPLS) and Internet Protocol (IP) Addresses. Additionally we will look at Content Deliver Networks and Data Center Networks and how they work together with ISP networks. In Section 4 we address the main challenges of ISP networks like network failure and challenges in network security. We conclude with providing solutions to these challenges.

2. Related work

One important reference of this paper is chapter 2 of Rober D. Doverspikes et. al. book about ISP networks [2]. This chapter goes more in depth about the structure of ISP networks and its different network layers. It also addresses network failures and possible fixes that can be applied. Furthermore the MPLS is explained in his work, which

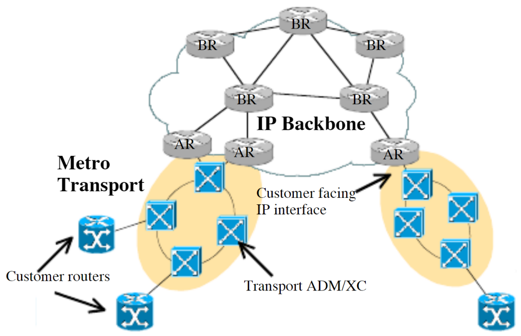


Figure 1: ISP network structure [3]

is also mentioned in this work. While Doverspikes book about ISP networks describes the networks architecture and network protocols in great detail, this paper tries to give a very broad overview of all aspects of ISP networks. Our focus lies more on different challenges of ISP networks and possible solutions to them.

3. Background

In this part of the paper we will introduce ISPs and how they can be categorized into different tiers. This is followed by an overview of the structure of ISP networks. Additionally, we will look at IP Addresses and MPLS and how they help to perform efficient routing in ISP networks. Finally we present different networks like Data Center Networks and how they work together with ISP networks to offer services to customers over the Internet.

3.1. ISPs

ISPs are companies that provide routing infrastructure to transport data over the Internet. Additionally, they provide services like email hosting and virtual private networks. ISPs can be roughly categorized into three different tiers depending on their provided services and size. Tier 1 ISPs are the largest and maintain wide ranging networks and are connected to other Tier 1 ISP to transport Data over far distances. Well known examples are ISPs like Deutsche Telekom or AT&T [4]. While Tier 2 ISPs still operate their own national networks, they rely on Tier 1 ISPs for wide Internet access. Tier 3 ISPs offer Internet in regional areas and depend on Tier 1 and 2 networks for Internet access [5].

Also the models of different ISPs can vary. A Wholesale ISP offers services to different ISPs as well as businesses, rather than end-users like private households. Customers of these ISP present a middle man who sells these services to their own customers. The most commonly known ISPs are Retail ISPs. They provide Internet access to end-users and typically package it into bundles. These include a certain tier of network speed and different services depending on the amount the customers are willing to pay [5]. The Managed Service Providers model includes selling broad IT services to companies like network infrastructure management and network security. There are even more types of models for ISP networks [5].

3.2. Structure of ISP networks

ISP networks are a main part of todays Internet and have the task to enable Internet communication for households and businesses. An ISP network consists of many routers located in different areas that control network traffic and route packets to their correct destinations. As seen in figure 1 customers are connected to an ISP network through their own router which is normally provided by the ISP itself. This router is indirectly connected to an access router (AR) of the ISP network which is part of the the IP-Backbone. The IP Backbone is a network of Backbone routers (BR) that are connected with each other over long distances. This makes communication all over the world possible. Access routers are the entry point for data from customers into the wide ISP network [2].

In between the customer and access router the packets are transported via metropolitan transport networks [3]. The metropolitan network, also called access network, is more local compared to the wide spanning Backbone network. It connects multiple households and businesses to the access routers of the an ISP network. There are multiple technolgies used in metro networks to transport data. A typical contender is Digital Subscrible Line (DSL) which makes use of telephone lines and is quite limited in terms of speed and bandwidth of the Internet access. Fiber-Optic cables are the much superior alternative compared to DSL if they are available in the area. Also very commonly used is a wireless connection between the end-user and an ISP. This includes mobile connection like 5G which is transmitted over radio waves to a mobile device. Another possibility is satellites which can cover huge areas but lack bandwidth and speed [5].

Network layers play a significant role in ISP networks as they organise network hardware into different layers, depending on the functionality. Each layer of ISP networks consits of nodes and links. These together can form connections for packets to travel through. The different network layers are affected by each other. For example each connection in a higher level network layer is represented by one or multiple connections in the lower layer. Therefore if network failures in lower layers occur this directly affects connections in higher layers of the network. A single connection failure in a lower network layer can disrupt multiple connections in a higher layers which makes the lower level network design very important [2].

One of the layers in ISP networks is the network Layer which is also represented in Figure 1 as the IP Backbone. In that layer, nodes are represented as routers that have the task to route packets, sent out by customers, through the network and finally to their correct destination. Transportation itself is performed over lower layers of the network that interact with the network layer inorder to fulfill their task of transportation [2].

The layer at the bottom of the network architecture is the Physical layer. This layer represents the physical optical fiber cables layed mostly underground around the world. They connect all the central offices of the ISP network, which are central Hubs connecting multiple households in one area to a singular network node [2].

At the very top of these layers ISP networks can also offer services like VPN to connect multiple local networks or Voice over IP. Voice over IP is a service to transmit phone calls not via the traditional way but rather as data packets that are routed over the IP layer of the ISP network [6].

3.3. OSI model

Not only the architecture of ISP networks can be structured into different layers. Also important is the protocol layering of Internet communication over ISP networks. Each message that is sent over the Internet is modified by many protocols that turn the message into packets which are fit to be transported over the network. A standardization of this protocol layer modell for package transport via Internet and therefore using ISP networks is the OSI modell defined by ISO, an Organisation for Standardisation. It defines seven different layers that all execute a specific part of the task to make Internet communication possible [7]. An important protocol layer of the OSI model in ISP networks is third layer called network layer. This layers job is to ensure that the data packets send by one source arrive at their correct destination in the network. It includes routing protocols as well as logical addressing of packets [8].

3.4. IP Addresses

As discussed in section 3.3 about the OSI model, the network layer has to correctly address the source and destination of data packets. In most computer networks including ISP networks this is accomplished with IP addresses. Each IP address uniquely identifies every device in the network and there are two different types of IP addresses. IPv4 is the older IP address version and ist represented by a 32-bit Integer while the newer version IPv6 is 128 bits long. To ensure the correct delivery of data packets, the packets receive a source and destination IP Address header in the network layer. This allows routers in the network to correctly pass on the packets [8]. As IPv4 only allows 4.3 billion different addresses, it is by far not enough to address every device of the Internet. Thats why IPv6 usage is increasing in ISP network architecture [5]. Network Address Translation (NAT) is a method which maps a public IP address to multiple private ones and therefore can increase the IPv4 addressing possibilities. This however isn't sufficient in all cases as it can't be applied to applications that use end to end communication [9]. In the case of IPv6 addressing NAT isn't necessary as the amount of addresses cover more than is necessary in the forseeable future. That's why transitioning from IPv4 to IPv6 is important. However the transition is costly, especially in developing countries, as they are missing the financial opportunities and correct training [9].

3.5. MPLS

Routing data packets is the main purpose of ISP networks. Therefore the routing has to be especially efficient. Two main goals of routing are to determine the shortest path between two endpoints and balancing the load of different connections [10]. Multiprotocol Layer Switching (MPLS) is a technology, used in ISP networks to efficiently route packets through the network. When MPLS is used as the routing protocol in the network, edge routers of the network attach certain labels to incoming packets. After inner routers receive packets with a label, they only have to examine the label to determine its next destination. When forwarding the packet, the router also swaps out the label of the packets for a new one for the next router to examine. The paths that are enforced by the labels are determined by a separate protocol [10]. Using MPLS in an ISP network helps reduce the network latency and keep up with expected quality standards. Therefore it is beneficial to the network [11].

3.6. CDNs

A way to significantly improve latency and performance of services offered over the Internet is through Content Delivery Networks (CDN). They have become especially useful due to rising demand of video on demand or huge download sizes [12]. CDNs are networks of servers that cache data that is frequently used by users. This improves the connection speed because the servers with the cached data are optimally located closer to the user than the actual Data Centers of the content provider. CDNs can be operated by ISPs themselves or seperate companies and in this case have to work together to optimise network speeds [12]. In general CDNs are benefitial to ISPs as they reduce traffic in their network and also improve user experience [5].

3.7. Data Center Networks

One of the Internets main benefits is the variety of services it offers. The services include email, web search and online games. These services would not be possible without Data Centers, that have the task to store and process Data that is relevant for the respective service. Data Centers include computing resources and storage resources which are interconnected with and intra Data Center Network [13].

A possible design for intra Data Center Networks is the cluster network design. In this design a cluster is a basic part of the network. A cluster connects multiple server racks through cluster switches. Devices inside a Server Rack are connected via a Top-of-rack switch. The cluster switch aggregators connect multiple cluster switches and makes communication between cluster in the network possible. A data center also includes core network devices which are the access point for data transport between different data centers and the internet [13].

Data Center providers like Facebook, Google or Amazon typically operate multiple Data Centers. These also have to be connected via an inter Data Center Network to efficiently exchange Data. Not only do providers use their infrastructure for their own services but they also sell it for other companies to offer their own services. Compared to ISP networks Data Center Networks (DCN) do not transport Information between third parties over their network but rather exchange Data between their own Services that are running in Data Centers of their network. Inter DCNs consist of edge nodes that rout traffic in the backbone network. This is very similar to Backbone routers in ISP networks. If users want to use a service provided by the DCN they connect to one of the edge routers over an ISP network. To connect to the correct DCN users use the Domain Name System which can be offered by the users ISP which links Internet Domains to Server locations [13].

3.8. Peering and Transit

As discussed earlier Tier 1 ISP networks can span over huge distances, also connecting multiple countries. On their own however they can only transmit communication between members of their network, which would exclude households or businesses connected to a different ISP. Therefore all ISPs have to agree on contracts with other ISPs to make data transit between their networks possible and provide their customers access to the entire internet. There are two main types of agreements that ISPs can engage in which are Peering and Transit Agreements. A Peering Agreement defines a contract between two ISPs for network traffic between their networks without financial compensation. This is a viable option if both ISPs are similar in size, have similar amounts of customers and around the same amount of network capacity. Peering relationships are not transitive, which means that if the second ISP has another peering relationship to a different ISP, traffic from the first ISP to the additional ISP would not be transported over the second one. In case of a Transit Agreement one ISP has to pay the other for network access depending on the amount of network traffic between them. In this case the agreement is transitive and would allow traffic over the second ISP to another network [14]. This type of agreement is typically used between ISPs of different tiers where lower tier ISPs have to pay higher tier ISPs for their broader access [15].

4. Challenges of ISP networks

In this section of the paper we will address some challenges that occur when operating large ISP networks. For each challenge possible solutions are mentioned that can be applied by ISPs. We will also include some research about making ISP networks more efficient and therefore reducing their substantial contribution to carbon emissions.

4.1. Network Restoration

A main challenge ISP networks have to overcome are network disruptions and outages. This can be caused by the failure of devices which are part of the network or due to maintenance work and can occur in different layers of the network. In the situation of a network disruption, availability of the Internet can be heavily affected. This is why ISPs have to ensure to always to provide a certain standard of quality, which is defined in the Service Level Agreements (SLA). SLAs are guarantees to the customers which define metrics like latency and packet loss [2].

When it comes to network restoration there are different procedures to apply, depending on the network layer the disruption happens in. For example if a cable in the fiber layer is damaged between two network nodes, the connection has to be manually reestablished over different cables and can take hours [2].

In case of IP layer failures, ISP networks can use the MPLS Fast Reroute to overcome the issue. This method makes use of backup paths for links between network nodes. These alternative paths can reroute incoming packets over functioning connections in case of a failure. It is also quite performant. In case of a connection failure between two network nodes, Fast Rerout is applied in less than 100 ms [2].

4.2. Network Security

Another main challenge ISPs are facing is cyber security in their networks. Their task is to protect customer data as well as protecting network infrastructure. Possible attacks that ISP networks have to defend against include infiltration of malware or Distributed Denial of Serivce (DDoS) attacks. DDoS attacks make use of botnets that overwhelm services, that are part of the Internet, with requests. This hinders users from accessing the service as the servers are overloaded [16]. There are ways for the ISP networks as well as connected servers to protect themselves. Methods include packet filtering to drop packets with suspicious header information. An additional method is load balancing where the ISP temporarily grants the attacked service a higher bandwith to keep the service alive during the attack [17].

4.3. Internet connection in rural areas

In 2019 around 86% of the population used mobile connection to the Internet [18]. The ISPs goals include covering all parts of the world with proper access to the Internet. This is still not the case as many rural areas struggle with a bad quality wireless connection or no connection at all. It stems from the fact that ISPs face geographical challenges when deploying wired or wireless connection in rural areas. Futhermore it is in many cases not worth it to deploy any access in rural areas at all. Compared to cities, rural areas have a much lower population density which means less customers for a similar expense. Eventhough many Internet infrastructure projects are financially supported by the government, the issues still remain [18]. One possible solution to this problem is Satellite connection as Sattelites are not affected by geographical challenges of the area. End-users can directly connect to a Satellite with an Antenna to access the Internet. Satellite deployment still comes with significant costs for the ISP but is a valid option for covering Internet connection of remote areas [18].

4.4. Reducing carbon emissions of ISP networks

The Internet is a a huge carbon emitter. According to Statista, if the Internet itself was a country it would take up the sixth spot on the carbon emission leaderboard [19]. Obviously ISP networks are only a part of the Internet but they do contribute to that number with network routing and operating data centers. A possible solution to that problem is to minimize the amount of active routing devices in the network that are necessary to keep up with the current demand. This is done while still keeping up with certain quality standards for network communications [20]. An approach like this is especially useful, because the normal network design mostly focuses on handling the most amount of traffic during peak-hours. Therefore the energy efficiency falls short when choosing such a design [20]. The theory and approach of this type of network design is explained more in detail in [20].

5. Conclusion and future work

As the paper has shown ISP networks are a very broad topic. Research areas include network architecture, network protocols, network security and many more. Thats why it can be difficult to describe all aspects of ISP networks in just one paper. This paper tried to cover the most important aspects of ISP networks like architecture and routing technologies like IP Addresses and MPLS. We also compared Data Center Networks to ISP networks and how they work together to offer services to customers over the internet. A main aspect of the paper were the challenges that ISP networks are facing. We covered challenges like network security which included DDoS attacks and how to prevent them. The still existing problem of rural internet connection and how satellite internet might be a solution was also addressed. As carbon emission still presents a problem we also mentioned how ISP networks can be optimized to reduce their emissions.

This paper presented more of a broad overview of ISP network challenges. A possible future work could tackle a concrete challenge and go more in depth on the solution to it. The paper could offer a concrete solution to a problem and describe it in detail which was not possible in this paper. Another important topic to present in a future work is user privacy and how ISPs might violate them. As ISPs have huge control over network flow and can easily aggregate data about user behavior, user privacy can easily be violated, and private data misused. A paper could analyze, how ISPs treat their sensitive user data and if any obvious privacy violations exist.

References

[1] A. M. T. Mitchell L. Moss, “The internet backbone and the american metropolis,” *The Information Society*, vol. 16, no. 1, pp. 35–47, 2000. [Online]. Available: https://doi.org/10.1080/019722400128310

[2] R. D. Doverspike, K. K. Ramakrishnan, and C. Chase, *Structural Overview of ISP Networks*. London: Springer London, 2010, pp. 19–93. [Online]. Available: https://doi.org/10.1007/978-1-84882-828-5_2

[3] P. Sebos, J. Yates, G. Li, D. Rubenstein, and M. Lazer, “An integrated ip/optical approach for efficient access router failure recovery,” 03 2004.

[4] GTT Editorial Team, " Global Tier 1 IP Networks: Everything You Need To Know," 2023, available online at https://www.gtt.net/de-de/resources/blog/global-tier-1-ip-networks-everything-you-need-to-know/; last accessed on 2024/03/18.

[5] PShaun Allen, " ISP Networks: Understanding the Fundamentals," 2023, available online at https://www.shaunallen.co.uk/blog/isp-networks/; last accessed on 2024/03/18.

[6] U. Varshney, A. Snow, M. McGivern, and C. Howard, "Voice over ip," *Communications of the ACM*, vol. 45, no. 1, pp. 89–96, 2002.

[7] F. Becher and J. Steitz, "Iso/osi-referenzmodell."

[8] P. Jasud, "The osi model: Overview on the seven layers of computer networks," *International Journal for Innovative Research in Science & Technology*, vol. 4, no. 3, pp. 116–124, 2017.

[9] B. R. Dawadi, S. R. Joshi, and A. R. Khanal, "Service provider ipv4 to ipv6 network migration strategies," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 6, no. 10, 2015.

[10] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni, "Traffic engineering with mpls in the internet," *IEEE network*, vol. 14, no. 2, pp. 28–33, 2000.

[11] M. A. Ridwan, N. A. M. Radzi, W. S. H. M. Wan Ahmad, F. Abdullah, M. Z. Jamaludin, and M. N. Zakaria, "Recent trends in mpls networks: technologies, applications and challenges," *IET Communications*, vol. 14, no. 2, pp. 177–185, 2020.

[12] N. Kamiyama, T. Mori, R. Kawahara, and H. Hasegawa, "Optimally designing isp-operated cdn," *IEICE transactions on communications*, vol. 96, no. 3, pp. 790–801, 2013.

[13] J. Meza, T. Xu, K. Veeraraghavan, and O. Mutlu, "A large scale study of data center network reliability," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 393–407.

[14] R. Kariyawasam, "Telecoms regulation: Peering and transit over tcp/ip networks," *Computer Law & Security Review*, vol. 17, no. 1, pp. 36–40, 2001.

[15] W. B. Norton, "Internet service providers and peering," in *Proceedings of NANOG*, vol. 19, 2001, pp. 1–17.

[16] N. N. Tuan, P. H. Hung, N. D. Nghia, N. V. Tho, T. V. Phan, and N. H. Thanh, "A ddos attack mitigation scheme in isp networks using machine learning based on sdn," *Electronics*, vol. 9, no. 3, p. 413, 2020.

[17] D. Mahajan and M. Sachdeva, "Ddos attack prevention and mitigation techniques-a review," *International Journal of Computer Applications*, vol. 67, no. 19, 2013.

[18] M. Bhuiyan, "Solutions for wireless internet connectivity in remote and rural areas," Master's thesis, M. Bhuiyan, 2020.

[19] Peter Schmidt-Feneberg, " Infografik: So viel Energie verbraucht das Internet," 2023, available online at https://de.statista.com/infografik/26873/co2-vergleich-dsl-und-glasfasernetz/; last accessed on 2024/03/16.

[20] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing isp network energy cost: Formulation and solutions," *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, pp. 463–476, 2012.

The State of Middleboxes Inside Containers

Josef Schönberger, Florian Wiedner*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: josef.schoenberger@tum.de, wiedner@net.in.tum.de

***Abstract*—Due to the reduced cost and increased flexibility, Network Function Virtualization (NFV) is a lucrative alternative for middleboxes compared to dedicated hardware. However, current NFV implementations are mostly VM-based despite the current trend toward containers for software deployments in the industry.**

This paper analyzes the current state of research on container-based NFV for middleboxes. We identify several issues that will need to be solved for widespread deployment, especially for low-latency applications. Finally, we discuss for which service level requirements container-based NFV might be viable for middleboxes.

***Index Terms*—containers, nfv, cnf, middleboxes**

1. Introduction

In recent years, *Network Function Virtualization* (NFV) has become a hot topic both in research and the industry. It promises to replace expensive dedicated hardware for specific network functionality with inexpensive off-the-shelf server hardware that provides these functions using software in virtual machines or containers. Especially for middleboxes, this presents the advantages of increased flexibility, scalability, updateability, and maintainability. So far, most implementations are based on system virtual machines.

At the same time, more and more software developers have started deploying their products in containers. Containers are a lightweight alternative to full-system virtual machines, which come with little overhead compared to bare-metal installations while providing a degree of isolation that is only slightly below virtual machines. Combined with automated container orchestration software such as Kubernetes[1], they offer automated scaling depending on the load and automated failover should one container fail. Finally, because these containers all run on a single kernel, no static resource allocation, such as CPU pinning, is necessary for containers, which allows for overprovisioning of resources, i.e., for more services per machine as long as the combined load of all services remains within limits.

While VM-based NFV has proven to be practical, using containers instead of virtual machines might also be lucrative for middlebox operators. However, this presents new challenges, as service level agreements might define strict limits on metrics like bandwidth, latency, reliability, and jitter. For example, the 5G *ultra-reliable low-latency communication* (URLLC) service category specifies an overall round-trip time of at most 1 ms and a reliability of at least 99.999% [1]. This requires a reevaluation of the viability and practicability of container-based network functions—also called *Cloud-Native Network Functions* (CNFs)—for middleboxes.

In this paper, we first provide an overview of some related work on this topic. We then discuss several potential issues, study the current state of research on them, and analyze the proposed solutions. We also offer potential approaches for solving some of these issues. Finally, we conclude with a discussion on whether containers might be viable for middleboxes.

2. Background

In this section, we define several terms that we will use throughout this paper.

2.1. Middleboxes

The term "*middlebox*" refers to all packet processing units that provide functionality beyond basic routing. Typical examples of middleboxes are firewalls, intrusion detection and prevention systems, and *Network Address Translations* (NATs).

2.2. Network Function Virtualization

The classical approach to middleboxes was using dedicated hardware for each function. Due to their low-volume nature, these devices are very costly. Furthermore, they are inflexible, with no possibility of customizing their behavior. Additionally, it is often impossible to install updates; and even if updates are possible, they have to be provided by the hardware vendor.

NFV solves these problems. It typically comprises virtual machines running on regular off-the-shelf server hardware, which can provide the same functionality in software [2]. These typically only offer simple functions and are composed together (in a process called "*Service Chaining*") to provide complex functionality.

Apart from solving the aforementioned problems with dedicated hardware, NFV also offers a few additional advantages. As the functions are implemented in simple virtual machines, they can be migrated from one server to another, e.g., for maintenance. They can also be replicated to dynamically adjust to increased load or changes in service level agreements. Finally, multiple *Virtual Network Functions* (VNF) can be run per machine, further reducing the overall hardware cost.

1. https://kubernetes.io/

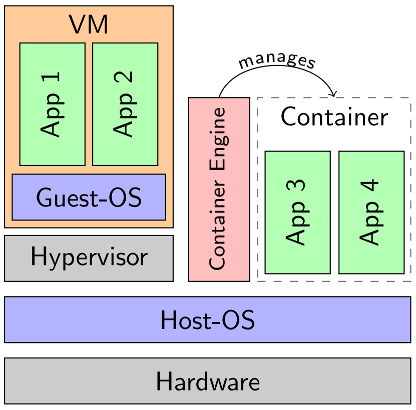


Figure 1: Comparison between VMs and Containers. Containers bundle processes and isolate resources such as the file system at the Host-OS level, thereby removing the Hypervisor and Guest-OS overhead.

2.3. Containers

Most NFV solutions are currently based on system virtual machines. Containers are a lightweight alternative; their architecture is shown in Figure 1. Instead of virtualizing hardware components and resources, they rely on mechanisms of the underlying operating system's kernel to provide an isolated view of system resources. This results in a more efficient architecture in which neither a guest operating system nor a hypervisor is required, but processes exist directly on the host operating system instead. These are bundled into groups called "containers" with a common view on resources such as the file system (and, thereby, system libraries) and network devices. The container engine (which sets up, monitors, and destroys the containers) and isolation mechanisms themselves introduce very little overhead [3] and are comparatively cheap to create and destroy [4].

Due to their architecture, resource allocation for containers is far simpler than for regular virtual machines. For example, containers do not generally need to statically allocate CPUs, memory, or storage [5].

However, the overall attack surface is increased since all containers operate on a common host operating system. A vulnerability in the host kernel, such as a privilege escalation, can be used to compromise the kernel not only for the malicious container but also for all containers and services on the system [6].

3. Related Work

Several studies identify hindering factors for commercial deployments of CNFs for middleboxes [7]. This paper focuses on container-based network functions specifically and will therefore not discuss the general problems of NFV for middleboxes.

Some previous work has already identified and discussed several problems. This section overviews attempts to develop complete CNF platforms that can be used for middleboxes. We will later discuss relevant literature concerning each issue in the relevant sections.

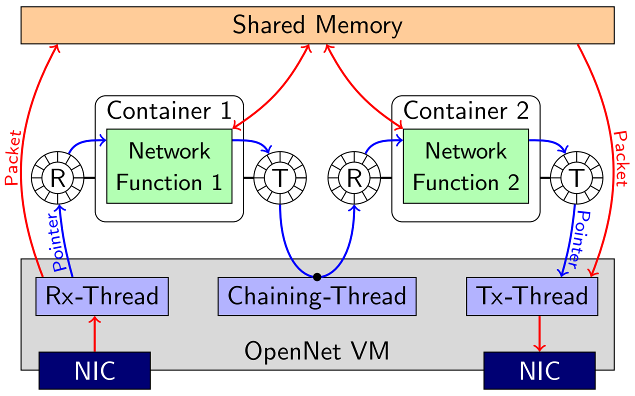


Figure 2: Architecture of OpenNetVM [8].

In 2016, Zhang et al. [8] presented *OpenNetVM*[2]. It executes the CNFs in separate Docker containers. The architecture is shown in Figure 2. Packets are exchanged in a shared memory region to avoid repeatedly copying packet data. Only the addresses of the packets within the shared memory region are transferred; the network functions read and write them in ring buffers and separate chaining threads on the host forward them from one network function to the next. Rx- and Tx-Threads transfer the packets between the shared memory and the *Network Interface Cards* (NICs). OpenNetVM uses the *Data Plane Development Kit*[3] (DPDK), a high-performance userspace network driver and application framework, for communication with the NICs.

Zheng et al. [9] implemented *MVMP*, which extends OpenNetVM's architecture by sharing a ring buffer between chained functions to eliminate the chaining threads. In their evaluation with a simple forwarding application, they measure a near-identical throughput to a reference DPDK application that runs on the host system directly.

Dzeparoska et al. [10] made an effort to develop CNFs for *Cloudify*, "an open source cloud orchestration framework" [11]. As opposed to [8] and [9], their VNFs exclusively operate in an *Software Defined Networking* (SDN) environment based on *OpenStack*[4] and interact with the network using Linux bridges and virtual ethernet devices. Dzeparoska et al. [10] compared the latency of their container-based solution to Cloudify's existing VM-based solution and consistently measured a significantly lower latency for containers.

4. Potential Issues with CNF Middleboxes

In this section, we identify several potential problems that might hinder the widespread usage of CNFs for middleboxes and discuss their current state in research.

4.1. Lack of Data for Realistic Scenarios

Plenty of literature exists regarding container-based network functions and their performance advantages compared to VM-based NFV. Some studies develop and evaluate real-world applications with multiple chained VNFs

2. https://sdnfv.github.io/onvm/
3. https://www.dpdk.org/
4. https://www.openstack.org/

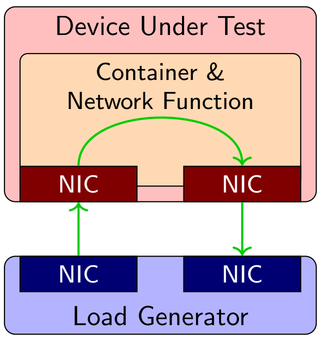


Figure 3: Typical evaluation setup with forwarding function.

with promising results [12]. However, most of them do not study realistic middlebox scenarios. For example, much work focuses on VNFs in SDN that are part of a control plane, which do not have the same bandwidth and latency requirements as middlebox VNFs on the data plane. While some have analyzed the performance of middlebox applications (such as intrusion detection systems) in particular, their experiments' deployment scenarios are still unrealistic.

In the end, there are several aspects that realistic experiments should consider:

4.1.1. Insufficient System Load. Many studies on container-based network functions analyze their performance with simple forwarding or acknowledging applications, shown in Figure 3. While these are great for reducing noise in the measured data and isolating the performance overhead of the container environment, they do not provide realistic scenarios, as resource contention might remain an unidentified problem. For example, none of the studies mentioned in Section 3 use realistic resource-intensive workloads [8]–[10]. As a result, the true impact of a high system load on the bandwidth and especially the latency is still unknown, particularly with high CPU, memory, and/or I/O bandwidth usage.

This becomes especially problematic for scenarios with multiple (possibly chained) VNFs per host: In 2023, Attaoui et al. [13] conducted a study on existing work regarding the placement of VNFs. In section IV subsection A specifically, they mention that "containers fight for the same resources in the system" [13]. Because existing container engines do not provide enough fairness guarantees for resource distribution, this can result in contentions on system resources. The authors also explicitly recommend using multiple virtual machines and strategically distributing VNF containers within those to alleviate the problem despite the reintroduction of additional virtualization overhead.

4.1.2. "Clean" Traffic. For most publications on CNFs, the performance evaluation is based on "clean" traffic: the packets were of constant size, perfectly paced (i.e., spread equally on the timeline), and had no short bursts. When handling real-world traffic, the VNFs are expected to deal with imperfect traffic without significant performance implications. Yet, the behavior of container-based VNFs in these scenarios has yet to be analyzed.

4.1.3. Insufficient Data on (Tail) Latencies. Modern systems can have stringent requirements on (tail) latencies. For example, the 5G URLLC scenario specifies an overall end-to-end round-trip time limit of as little as 1 ms [1]. Combined with the high-reliability requirement of 99.999% [1], this imposes significant challenges for VNFs on the 5G URLLC data plane.

All publications mentioned in Section 3 do not analyze latency at high-percentile tails. Additionally, most publications on CNFs only analyze the average latency, if they analyze the latency at all.

Only a few publications directly address very low tail latency requirements. Gallenmüller et al. [14], [15] analyzed the difficulties in archiving low high-percentile tail latencies for bare-metal network functions on Linux. Gallenmüller et al. [16] also successfully reduced the latency of a real-world application (in particular, the intrusion prevention system *Snort 3*[5]) significantly. While their work proves that Linux can generally be a suitable platform for low-latency network functions, several deployment obstacles must be overcome. Most notably, a specific realtime-optimized Linux kernel with several kernel parameters and CPU pinning (i.e., reserving a specific CPU core and restricting a task to it) is required.

Wiedner et al. [17] extended this work to containers and proved that CNFs are generally a viable option for very low (tail) latency applications. Wiedner et al. [18] also analyzed the impact of cgroups v2 (a fundamental building block for Linux containers responsible for resource isolation) on tail latencies in container-based VNFs. However, these publications also only evaluate with a single forwarding network function.

In summary, while it has been shown that very low tail latencies are possible in principle, practical container network function implementations for these scenarios have yet to be developed.

4.1.4. Container Interferences. Multiple container-based VNFs on a single host might interfere with each other. For example, as previously mentioned, containers fighting for a common set of resources can decrease performance [13]. The effects of these interferences have yet to be analyzed.

One potential problem stems from synchronization. Since most experiments only use simple forwarding functions for evaluation, the impact of thread synchronization is never measured. However, since all threads operate on the host kernel directly (c.f. Figure 1), all thread synchronization is centered on it. This may have implications on the (tail) latency for middlebox VNFs that extensively use thread synchronization, e.g., for concurrent accesses on data structures.

A similar problem arises from in-kernel contention. In VM-based NFV, many kernel tasks, such as memory management, are distributed across all VNFs. In contrast, for container-based network functions, it is plausible that in-kernel contention (e.g., on the memory allocator or scheduler) could further increase the (tail) latencies.

Finally, TLB shootdowns cause latency spikes [14], [15], [19]. Each core has an independent *Translation Lookaside Buffer* (TLB), which effectively caches address

5. https://www.snort.org/snort3

translations. Since the CPU does not enforce TLB consistency, modifications or invalidations of address mappings are not immediately visible to all cores. Therefore, the issuing processor must send a so-called TLB shootdown—a broadcasted *Inter Processor Interrupt* (IPI) that disrupts all other CPUs' execution in order to flush all TLBs. On VM-based NFV, these TLB shootdowns are restricted within the guest operating system and the virtual machine; in contrast, on container-based network functions, they are broadcasted to all cores and thereby disrupt all VNFs.

4.2. Resource Contention from Overprovisioning

VM-based NFV naturally does not suffer from resource contention due to static resource allocation; since the hypervisor typically assigns each VM a (mainly) static share of CPUs, memory, and I/O devices, different VNFs are independent regarding resource distribution. However, one of the major advantages of containers is that static resource allocation similar to VMs is not necessary—the host operating system allocates the individual resources on demand. But this can cause resource contention in container-based network functions with a significant performance impact [13]. Additionally, Wiedner et al. [17] and Gallenmüller et al. [14], [15] have shown that CPU pinning is essential for low tail latencies.

This is directly at odds with another expectation on containers: the ability to overprovision the system resources such as CPU time and memory. In this context, overprovisioning means creating more containers than the system resources allow under full load. In other words, if all containers were to use all of their available resources simultaneously, the operating system would not be able to fulfill the requirements of all containers. This can make sense for containers as it is rare that all containers simultaneously require all resources and because it is possible to migrate and thereby offload VNFs to other hosts in cases of high resource pressure [5].

Additionally, resource overprovisioning is essential for self-healing container replication as implemented in Kubernetes, for example. Spawning multiple instances of the same VNF not only allows for dynamic scaling based on load but also allows the orchestrator to spawn fallback containers that do not yet accept any work but are ready to take over at any point should another active container fail. Such backup containers are idle for most of their lifetime and would thereby waste resources under static resource allocation.

In the end, there is a lack of an analysis of the implications of resource overprovisioning beyond the studies that reveal general problems with resource contentions in CNFs. However, data on this matter would be critical since software developers should be able to decide whether static resource allocation is unavoidable under a given set of service level requirements.

4.3. Latency Spikes at CPU Migration

CPU migration (i.e., the kernel moving a task from one core to another) needs to temporarily stop the execution of the VNF, which causes a latency spike. The new core also does not have the data ready in its caches, further negatively impacting performance. For this reason, Gallenmüller et al. [14], [15] recommend isolating the container threads to a single core. However, this effectively results in static resource distribution for CPU cores, as this core is now reserved for this single thread, which hinders overprovisioning. As a result, newer approaches to thread migration are necessary for low-latency applications.

We propose that this problem is solvable. Instead of leaving the migration to the kernel's dispatcher, CPU core migration should only be done by replacing the application thread. In other words, the application should spawn a new worker thread on the new core, request the existing threads to stop accepting packets, and finally terminate the previous threads.

Of course, this approach also has drawbacks. The new application has not yet established its memory working set, so the caches are not filled with relevant data. Similarly, on NUMA systems, moving the relevant data from one node to the other might be necessary, which is difficult considering that the thread on the previous node might still actively use this data. Additionally, creating processes and threads involves TLB shootdowns on all cores, which could result in latency spikes in other independent VNFs. Finally, this approach requires both the old and new cores to be allocatable at the same time. As a result, if a system wants to remain able to move a thread from one core to another, it always needs to keep a spare core unoccupied.

In the end, this problem will need to be addressed by future work.

4.4. Service Chaining

Depending on the service level requirements, different approaches to service chaining might be necessary. Most research on service chaining of CNFs focuses on throughput [20], with some measuring the average latency as well [21].

If the overall end-to-end latency of the complete chain is not critical, regular approaches using default Linux networking mechanisms such as bridges and virtual ethernet (`veth`) devices [20] or SDN mechanisms such as *Open vSwitch*[6] might be sufficient [10], [21] and potentially preferable due to their simplicity and flexibility. On the other hand, if high throughput and low (tail) latency are essential, more sophisticated approaches based on shared memory might be necessary [8], [9]. The current approaches could probably still be optimized by further reducing the amount of threads touching the packets, e.g., by placing NICs into the container directly using *Single Root I/O-Virtualization* (SR-IOV), which could, for ,example eliminate the dedicated Rx- and Tx-Threads in OpenNetVM and thereby potentially lead to improved cache locality. Additionally, the existing approaches are still comparatively inflexible.

5. Conclusion

There is no scientific consensus on whether CNF-based middleboxes are viable. While several papers successfully implemented CNFs [8]–[10], others are more

6. https://www.openvswitch.org/

skeptical and actively encourage the use of VMs instead [13], [16]. We showed that there are several aspects in which we lack sufficient data to come to a conclusive answer. In particular, the behavior with real-world applications and data, tail latencies in realistic scenarios, and interferences of concurrent CNFs are still largely unknown. We also demonstrated that resource overprovisioning can cause resource contention and that a VNF CPU migration could result in latency spikes, for which we proposed a new approach to reduce the impact. Finally, we also showed that while service chaining for CNFs has been explored, we believe there to be room for improvement.

Ultimately, we also conclude that the suitability of CNFs for middleboxes heavily depends on the service level requirements. Assuming that the security benefits of virtual machines over containers are negligible for the use case, if only high bandwidth and efficiency are required, CNFs are well-tested and a good choice. If a low average latency is specified, container-based middlebox VNFs could still be viable. However, if very low tail latencies should be guaranteed, several pitfalls remain. Nevertheless, we believe that all of the aforementioned problems can be solved, even though much future work is still required.

References

[1] European Telecommunications Standards Institute (ETSI), "Study on scenarios and requirements for next generation access technologies," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 38.913, 08 2017, version 14.3.0.

[2] ——, "Network functions virtualisation – an introduction, benefits, enablers, challenges & call for action," *SDN and OpenFlow World Congress*, Oct. 2012, accessed on 2024-04-07. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf

[3] A. M. Joy, "Performance comparison between Linux containers and virtual machines," in *2015 International Conference on Advances in Computer Engineering and Applications*, 2015, pp. 342–346.

[4] R.-S. Schmoll, T. Fischer, H. Salah, and F. H. P. Fitzek, "Comparing and evaluating application-specific boot times of virtualized instances," in *2019 IEEE 2nd 5G World Forum (5GWF)*, 2019, pp. 602–606.

[5] S. K. Tesfatsion, C. Klein, and J. Tordsson, "Virtualization techniques compared: Performance, resource, and power usage overheads in clouds," in *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 145–156. [Online]. Available: https://doi.org/10.1145/3184407.3184414

[6] S. Sultan, I. Ahmad, and T. Dimitriou, "Container security: Issues, challenges, and the road ahead," *IEEE Access*, vol. 7, pp. 52 976–52 996, 2019.

[7] A. U. Rehman, R. L. Aguiar, and J. P. Barraca, "Network functions virtualization: The long road to commercial deployments," *IEEE Access*, vol. 7, pp. 60 439–60 464, 2019.

[8] W. Zhang, G. Liu, W. Zhang, N. Shah, P. Lopreiato, G. Todeschi, K. Ramakrishnan, and T. Wood, "OpenNetVM: A platform for high performance network service chains," in *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ser. HotMIddlebox '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 26–31. [Online]. Available: https://doi.org/10.1145/2940147.2940155

[9] C. Zheng, Q. Lu, J. Li, Q. Liu, and B. Fang, "A flexible and efficient container-based NFV platform for middlebox networking," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 989–995. [Online]. Available: https://doi.org/10.1145/3167132.3167240

[10] K. Dzeparoska, A. Nasiri, and B. Najafi, "Deploying container-based NFV within SDN environment," 03 2017, accessed on 2024-04-04. [Online]. Available: https://doi.org/10.13140/RG.2.2.29961.88166

[11] Cloudify Documentation Center – About Cloudify. Accessed on 2024-04-05. [Online]. Available: https://docs.cloudify.co/latest/about/

[12] D. T. Nguyen, N. L. Dao, V. T. Tran, K. T. Lang, T. T. Pham, P. H. Nguyen, C. D. Pham, T. A. Pham, D. H. Nguyen, and H. T. Nguyen, "Enhancing CNF performance for 5G core network using SR-IOV in Kubernetes," in *2022 24th International Conference on Advanced Communication Technology (ICACT)*, 2022, pp. 501–506.

[13] W. Attaoui, E. Sabir, H. Elbiaze, and M. Guizani, "VNF and CNF placement in 5G: Recent advances and future trends," *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 4698–4733, 2023.

[14] S. Gallenmüller, F. Wiedner, J. Naab, and G. Carle, "Ducked Tails: Trimming the tail latency of(f) packet processing systems," in *3rd International Workshop on High-Precision, Predictable, and Low-Latency Networking (HiPNet 2021)*, Izmir, Turkey, Oct. 2021.

[15] S. Gallenmüller, F. Wiedner, J. Naab, and G. Carle, "How low can you go? a limbo dance for low-latency network functions," *Journal of Network and Systems Management*, vol. 31, no. 20, Dec. 2022. [Online]. Available: https://doi.org/10.1007/s10922-022-09710-3

[16] S. Gallenmüller, J. Naab, I. Adam, and G. Carle, "5G URLLC: A case study on low-latency intrusion prevention," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 35–41, Oct. 2020.

[17] F. Wiedner, M. Helm, A. Daichendt, J. Andre, and G. Carle, "Containing Low Tail-Latencies in Packet Processing Using Lightweight Virtualization," in *2023 35rd International Teletraffic Congress (ITC-35)*, Oct. 2023.

[18] F. Wiedner, A. Daichendt, J. Andre, and G. Carle, "Control Groups Added Latency in NFVs: An Update Needed?" in *2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2023.

[19] E. Rigtorp. (2020, Mar.) Low latency tuning guide. Accessed on 2024-04-02. [Online]. Available: https://rigtorp.se/low-latency-guide/

[20] S. Livi, Q. Jacquemart, D. L. Pacheco, and G. Urvoy-Keller, "Container-based service chaining: A performance perspective," in *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, 2016, pp. 176–181.

[21] R. Cziva, S. Jouet, K. J. S. White, and D. P. Pezaros, "Container-based network function virtualization for software-defined networks," in *2015 IEEE Symposium on Computers and Communication (ISCC)*, 2015, pp. 415–420.

Cryptic Contracts: The State of Smart Contract Transparency

Amine Smaoui, Richard von Seck*, Filip Rezabek*, and Kilian Glas*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: amine.smaoui@tum.de, seck@net.in.tum.de, rezabeck@net.in.tum.de, glask@net.in.tum.de

Abstract—**Blockchain technology has seen the emergence of new cryptocurrencies that use smart contracts. A smart contract is a self-executing code that, when deployed, is compiled into lower-level bytecode, which is usually more complex to grasp from a human perspective than high-level language. This complexity can hide several errors or vulnerabilities, potentially leading to stolen assets and undermining the network's stability. Consequently, there is a need for tools and decompilers to reverse engineer the process, gain an understanding of their logic, and enhance their security. With this paper, we seek to understand and compare various techniques for enhancing smart contract transparency on the major Blockchains of Ethereum, Algorand, and Dfinity's Internet Computer. We collect recent literature mainly focused on smart contract deployment, decompilation, and analysis tools. These tools and their approaches are examined and evaluated based on the transparency level they provide. Subsequently, we conclude that Ethereum has the best decompilation support, positioning it as a trustworthy and transparent Blockchain to build decentralized applications.**

Index Terms—**Smart Contracts, Ethereum, Algorand, Dfinity, Decompilation, Reverse Engineering**

1. Introduction

Blockchain has gained significant popularity over the last few years, and even financial institutions are adopting it for their transactions [1]. Several Peer-to-Peer systems, such as Ethereum (introduced in Subsection 3.1), support smart contracts [2] that act as trustworthy, self-executing middlemen. Ethereum compiles the smart contract's high-level code into less readable bytecode to be processed by the distributed network. This procedure complicates the possibilities of error and vulnerability detection.

One of the significant factors that intensified the scientific community's interest in smart contracts decompilation was the "DAO Attack" [3]. This happened in 2016 due to a vulnerability in smart contract code; an attacker succeeded in controlling around 60 million US dollars worth of Blockchain tokens, called Ether, on Ethereum. Researchers have since started focusing more on methods that analyze, debug, and potentially decompile the bytecode to understand the vulnerabilities and prevent the recurrence of such attacks.

Our goal in this paper is to examine the efforts deployed in reverse engineering the process of smart contracts on Ethereum, Algorand, and Dfinity's Internet Computer, aiming to provide an overview of the effectiveness and usability of these tools.

1.1. Blockchain

Blockchain [4] was unveiled to the world in 2009 with the introduction of Bitcoin [5]. It was presented as a tool enabling decentralized, immutable, and transparent digital transactions. However, these properties may vary depending on the type of Blockchain. We differentiate three types of Blockchains [6]: Public, private, and hybrid Blockchain. In this paper, our work mainly focuses on public Blockchains. This type of Blockchain is an open, distributed ledger system that is available to everyone.

Decentralized systems, such as Blockchain, are designed to eliminate the need for a central authority to monitor the network traffic and approve transactions. These tasks are performed by network users, also known as nodes. The immutability property of this architecture means that all data recorded on the public ledger can no longer be modified after the approval of the network nodes [7]. This property is ensured thanks to "a consensus mechanism, cryptography, and back-referencing blocks." The transparency of this technology stems from the fact that all transactions on the Blockchain are recorded on a public ledger. These actions are visible and verifiable by all participants [4].

1.2. Smart contracts

Nick Szabo [8] introduced the term *smart contract* and presented it as "a set of promises, specified in digital form, including protocols within which the parties perform on these promises." [9].

Nowadays, smart contracts use Blockchains as their underlying platform [10]. After agreeing on the contract details, these are translated into computer code that can be executed automatically. To replicate the decision-making process, smart contracts usually rely on programming structures, such as "if-else statements," and every action taken is recorded on the ledger and cannot be changed. A penalty can also be predetermined if a contract's party does not honor the agreement. This penalty is automatically subtracted from the violator's deposit.

A smart contract life cycle [10] starts from written code in a programming language supported by the Blockchain. This code is compiled and stored in the network; henceforth, it cannot be altered. Additionally, the funds of participating members are frozen in their digital wallets. The execution of smart contracts is similar to buying from a vending machine [11]. In contrast to buying from a supermarket, where you need to interact with a cashier, the process of purchasing from a machine is fully automated; The buyer throws in enough coins, presses the button, and gets the product. The final step involves unlocking parties' funds and updating the states of all contract parties.

Since our main focus is reverse engineering smart contracts, presenting the procedure and the challenges encountered is important to our transparency study process. A decompiler's job is to convert bytecode back into its source code format. This protocol [12] generally unfolds as follows: Firstly, the tool tries to decode the binary files of the contract and converts them into a stream of instructions and other important data. This phase is prone to errors due to various language version compatibility issues with binary file formats. The instruction streams generated in the previous step are transformed into assembly code. This is particularly challenging as it is hard to differentiate between program code and data. Assembly programs are then developed into various Intermediate Representation (IR) forms, from abstract syntax trees to control-flow graphs. Without control structures, this process is complex. Finally, the process is concluded by generating high-level code from the previous IRs. The reconstruction is a challenging procedure in Ethereum, for example, due to the absence of identifiers (variable names and types).

In this survey, we study the decompilation and analysis processes of various tools and evaluate their effectiveness.

2. Related work

Numerous studies have covered this topic but usually target only one Blockchain. For instance, Liu et al. [12] conducted an empirical study of Ethereum's smart contract decompilers, gathering the insights and challenges of the major tools present on the market in one paper. On Algorand, the research mainly focused on formal verification and analysis tools. Notable contributions include the work by Bartoletti et al. [13] and analysis tool Panda [14]. On the other hand, efforts on Dfinity's Internet Computer for smart contract decompilation were relatively limited, with the primary focus on WebAssembly (discussed in Subsection 3.3) decompilation efforts made by Dfinity [15] and Google [16] to debug and analyze the low-level language code. In this paper, we aim to bridge these gaps by providing a comprehensive overview of smart contract transparency across these various ecosystems.

3. Representative Blockchains

The following sections present a detailed overview of each ecosystem focusing on how they manage and implement smart contracts.

3.1. Ethereum

Buterin [17] introduced Ethereum in 2013 as an "alternative protocol for building decentralized applications," it has since fulfilled its promise, developing into one of the major platforms in the Blockchain space. Ethereum's Blockchain technology goes beyond financial transactions; it has several real-life applications such as insurance, saving wallets, or even cloud computing.

Ethereum's differs from the Bitcoin Blockchain by integrating a Turing-complete high-level programming language, Solidity. A Turing-complete [18] language is a programming language capable of creating and computing any wanted program. Additionally, a run-time environment that supports smart contracts [19], called Ethereum Virtual Machine (EVM) was implemented. It executes smart contract's bytecode, also known as EVM Bytecode. These tools allow users to directly interact with the Blockchain by creating their own smart contracts and decentralized applications (dApps) for different purposes beyond just currency exchange. As a result of Solidity's properties, the network members are able to perform any computable function within the Ethereum ecosystem. However, the execution is not free of charge; it is influenced by the "gas" cost, which is merely the price of computational efforts on the Blockchain [17].

3.2. Algorand

First presented in 2017 as a low latency and highly scalable cryptocurrency, Algorand [20] uses a Pure Proof of Stake [21] consensus mechanism. To ensure security, this mechanism applies various techniques based on the Byzantine Agreement protocol (BA*). It creates groups of nodes called "committees" that approve the transactions. BA* ensures that 2/3 of the weighted users in the committee are honest. The same protocol applies cryptographic sortition to privately choose committee members, hence protecting them from targeted attacks. The random sortition process is guaranteed by the so-called "Verifiable Random Function" (VRF), a random number generator that expresses, among others, the probability of the user being part of the committee. Moreover, Algorand's Byzantine Agreement allows committee members to contribute once, and then they are generally replaced. This prevents members from being deliberately targeted by attackers and jeopardizing the consensus. This protocol is designed to reach consensus on transactions securely. One of Algorand's main features is its low-level bytecode-based stack language, the Transaction Execution Approval Language (TEAL) [22]. Introduced as a non-Turing-complete programming language, this property helps reduce the risk of attacks [13]. However, Python provides a high-level language alternative. "Pyteal" is specially tailored to write smart contracts on Algorand. TEAL is executed as a script and returns a boolean that either approves or rejects the transaction. An important additional feature of Algorand is its Virtual Machine (AVM) [14], capable of executing the bytecode resulting from compiled TEAL code.

Smart contracts in Algorand are categorized into single State and Algogeneous contracts [22]. Single State contracts can be used for various purposes such as transactions and creating applications. Such smart contracts can be Stateless or Stateful and they have distinct functions. Stateless smart contracts are primarily used for transaction validation. They approve and deny transactions and can also serve as "signature delegators". On the other hand, stateful smart contracts are mostly used to store and manage data on the Blockchain. Both smart contract types could be combined to produce complex applications.

Algogenous contracts represent a more advanced type of smart contracts. They comprise the functionalities of both Stateless and Stateful contracts. This design allows it to do multiple tasks, combining validation and verification.

3.3. Dfinity's Internet Computer

Dfinity's Internet Computer [23] represents a relatively new member of the Blockchain family. The particularity of this Blockchain is its use of a hybrid model, named DAO-controlled network, which is a consensus mechanism based on subnets that use a permissioned consensus mechanism. These subnets are chosen by the network nervous system (NNS) to manage the network functions. This step is similar to PoS, as the members of the network stake tokens to vote for the entities that create "replicas" and perform other tasks. These replicas are stored on distributed servers to ensure their security. On the Internet Computer, smart contracts are written in a high-level language, such as Rust [24] or Motoko, a Dfinity-tailored language that aligns with IC's semantics. The written high-level code is then compiled down to WebAssembly (Wasm), a binary instruction format that provides a way to run code on the web. After the compilation, the program is then deployed on the Blockchain in a Canister [23]. A Canister is similar to the concept of a "process" in traditional computing, they are coded in Wasm and consist of a program and its state. Canisters run autonomously on the Internet Computer and interact with each other through an interface called Candid.

4. Smart Contract Transparency

The methodologies applied for the transparency analysis differ from one Blockchain platform to another. The following subsections handle the specifics of smart contracts transparency approaches used by major platforms.

4.1. Smart Contracts decompilation on Ethereum

Writing smart contracts directly in EVM bytecode, an assembly-like language [25], is an intricate operation and rather prone to errors. Therefore, Ethereum supports various high-level programming languages besides Solidity to implement smart contracts, such as Vyper [26]. The compilation process from high-level to EVM bytecode occurs before deploying to the network. A smart contract's bytecode comprises three parts [12]: a deployment code is responsible for deploying smart contracts on Ethereum. It is put to execution as soon as it is created. It also checks if the function can receive Ether payments. Runtime Code defines the contract's functionality on the Blockchain. Auxiliary data contains a hash value linked to the metadata of the deployed contract and can be used for verification.

Having established the fundamental challenges of smart contract decompilation in Subsection 1.2, we will now focus on the approaches and solutions adopted by decompilers to tackle these issues.

The EVM uses 256-bit pseudo-registers containing 160-bit addresses called "accounts" to identify them. EVM's pseudo-registers fundamentally operate as a stack, which facilitates passing parameters to perform various operations [27]. This observation is exploited by Porosity [27] to extract the addresses from the bytecode using bitmasks to isolate the 160-bit address from the 256-bit EVM pseudo-registers. Gigahorse [28] addresses the issues of disassembly and intermediate representation by applying "Context-sensitivity" that considers varying states and conditions available at each step of the program. Basically, heuristics are used to determine the program control-flows. Elipmoc [29], a decompiler based on Gigahorse, uses the same principle but creates IRs using only stack locations that might contain jump targets. Using this technique, Elipmoc claims a 99,5% success rate in fully decompiling Smart Contracts. A better ratio than Gigarhorse's 62,8% decompilation rate.

4.2. Smart Contracts Transparency on Algorand

Although significant efforts have been deployed toward smart contract decompilation on Algorand, fully decompiled smart contracts are still not the standard. Other approaches and methodologies are frequently employed.

Stateless Smart Contracts on Algorand or ASC1 [13], are programmed using non-Turing-complete language to reduce vulnerability risks. However, there are still potential threats without a formalized mathematical model that ensures the contract's accuracy and security. Bartoletti et al. [13] decided to create a formal model that defines the behavior of Algorand accounts, transactions, and smart contracts using a state machine that acts to fundamentally understand their functioning and experiment on them. An attacker model was also developed to simulate attacks on their formal smart contract model. This model can potentially be a valuable tool for debugging and identifying security flaws and susceptible points of attack.

Panda [14], a security analysis tool of Algorand smart contract, has an architecture composed of several components. The main components we are interested in are: **(1)** The user interface for user input and settings, **(2)** a Blockchain Explorer that fetches the TEAL bytecode from the contract and disassembles it, **(3)** a control-flow graph (CFG) Builder generates graphs from TEAL programs, and **(4)** a symbolic executor analyzes these CFGs and processes each command symbolically. Moreover, detection rules are defined to address any already known vulnerabilities. Sun et al. [14] have reported several security concerns on Algorand. They were later categorized into five key groups. Three of these categories affect application operations, while the remaining two are vulnerabilities in smart signatures (also known as stateless smart contracts). We examine the vulnerabilities related to stateless smart contracts.

The first refers to an "Unchecked Transaction Fee." This vulnerability is generally caused by transaction fees that are not properly restricted. This allows an attacker to set high transaction fees and deplete the account's funds.

The second weakness was called 'Unchecked Transaction Parameters.' This describes vulnerabilities arising from inadequate verification of transaction parameters. One of the function arguments in the transaction code sets the authorized address for future exchanges. An attacker can gain access to the signature account by modifying this field. Another important parameter directs where the remaining balance of an account should go when the transaction is closed. The attacker can drain all the Algos (Algorand's native currency) from the signature account by setting this parameter to their address.

4.3. Smart Contracts transparency on Dfinity's Internet Computer

To have an overview of the transparency efforts of smart contracts on the Internet Computer, one needs to look at the IC's Wasm Libraries and the key tools provided and developed by Dfinity, especially for Canisters. The ic_wasm library (v0.7.1) [15] includes an experimental feature that instruments Canisters, which can help debug and analyze their code. For instance, the instrumented Canister can provide additional endpoints to access the execution trace log and the current cycle counter. Furthermore, certain flags can also be added to trace logging of a specific function during its execution. Google on the other hand has developed a toolkit to debug and analyze WebAssembly code, namely The WebAssembly Binary Toolkit (WABT) [16]. This repository contains several helpful tools, for instance, **wat2wasm** and **wasm2wat** convert between WebAssembly text and binary formats. Furthermore, **wasm-objdump** generates an objdump providing a general overview of the code structure. Additionally, this toolkit contains a Wasm stack-based interpreter that executes binary files. Another significant tool is a Wasm decompiler that converts a Wasm binary file into an intelligible C-like syntax, providing readable partial decompilation. Although WABT is not directly associated with Dfinity's Internet Computer, it still has potential general application to a wide range of wasm-written code. However, this toolkit's capabilities might eventually be less effective in some IC-specific cases.

5. Discussion

In Section 4, we explore the methodologies and approaches used to fully decompile smart contracts or analyze them through other methods, such as debugging. Given that each discussed platform differs by its fundamental architecture, consensus mechanism, smart contract deployment, publication date, and development phase, a direct comparison might not provide equitable insights. Therefore, we perform a case-by-case analysis to better understand the state of smart contract's transparency on each ecosystem independently.

Ethereum has the most advanced development phase compared to the other platforms, with functional smart contracts decompilers and relatively high success rates. Although each decompiler claims to have one of the best correct decompilation ratios, these percentages generally depend on the used dataset and smart contract types. Consequently, we will refer in our analysis to the work of Liu et al. [12] as the main source of data. This recent empirical study tests with the same dataset on major decompilers and rates their execution based on mathematical formulas.

When applying decompilers on normal datasets with the compiler optimizations turned on, Gigahorse [28], Vandal [30], and Ethervm [31] had success rates all above 99.70%. Whereas Panoramix [32] succeded 98,14% of the time, leaving Erays [33] with the least success rate of 63,92%. Another test on a dataset of buggy contracts yielded a 100% decompilation rate for Gigahorse and EtherVM. Vandal had one failed decompilation, Panoramix encountered 21 failures, and Erays could not decompile 359 buggy contracts.

These results indicate that smart contract decompilers on Ethereum have achieved significant milestones. However, there is still room for improvement, especially with the accuracy and completeness of the decompilation.

Algorand's Panda and the formal model offered important insights to address the transparency problem in smart contracts, but their approaches are limited. Panda, for instance, can be helpful when detecting vulnerabilities, but the symbolic executor cannot process a certain type of opcode. Additionally, there are some challenges when identifying the Validator when the smart signature uses implicit invoking. The formal model, on the other hand, provides a theoretical approach to understanding smart contracts. This might not be suited for practical evaluation since it does not capture every behavior of the Algorand implementation in real life and mainly focuses on stateless smart contracts.

On IC, using the instrumentation feature of 'ic_wasm' could be a tool for understanding smart contracts. Although valuable for bug identification, it lacks the required transparency to fully comprehend the logic of the contract. WABT, however, presents significant means for Wasm analysis and debugging. The tools offered can even partially decompile code. However, this toolkit is more of a general debug and analysis tool and not IC-specific, therefore, its ability to address certain tasks from the Internet Computer might be limited.

Table 1 summarizes the available tools for smart contract decompilation and transparency across the different ecosystems, based on the papers mentioned in this survey:

TABLE 1: Availability of Smart Contract Transparency tools across Blockchains

Platform	Debug	Analysis	Par. Dec[1]	Full Dec[2]
Ethereum	✓	✓	✓	✓
Dfinity	✓	✓	∼[3]	×
Algorand	✓	✓	×	×

1. Partial Decompilation
2. Full Decompilation
3. Not enough data available to make a statement

Table 1, shows a complete set of tools available for Ethereum, ranging from debugging to full decompilation. While Algorand only has debugging and analysis tools, Dfinity offers additionally a potential partial decompiler as well as debugging and analysis transparency instruments. The key factors responsible for these disparities differ from one ecosystem to another. Ethereum for instance, is a more mature and established Blockchain. It has experienced rapid growth in popularity, and its applications, Decentralized Finance (DeFi), for example, have reached 200 billion US dollars in 2021 [34]. Indicating significant transaction traffic and enough data for scientists to work on. Algorand [22] and Dfinity's Internet Computer [23], on the other hand, are still relatively new. But Dfinity has an advantage since its high-level code is compiled into WebAssembly. A widely utilized programming language with various applications [35], in contrast to TEAL which is exclusively used on Algorand.

6. Conclusion and future work

In this paper, we analyze and compare multiple approaches to achieve smart contract transparency. These techniques range from decompiling to analysis and debugging. Their applicability depends on the Blockchain itself; for instance, in Algorand, only debugging and analysis tools were available. Dfinity's Internet Computer introduced, additionally to analysis and debug tools, a potential partial decompiler. On the other hand, smart contracts were successfully fully decompiled on Ethereum. The high transparency level of smart contract decompilation on Ethereum makes the platform more attractive for users and developers to create secure and optimized applications in a trustworthy and secure environment.

Future work could look into integrating machine learning for vulnerability detection and decompilation of smart contracts. Sendner et al. [36], and Gioka et al. [37] have already initiated research efforts towards this topic. Improving the decompilation and analysis tools will help anticipate potential risks and attacks, rendering the Blockchain a more secure platform for developing decentralized applications.

References

[1] M. Javaid, A. Haleem, R. P. Singh, R. Suman, and S. Khan, "A review of blockchain technology applications for financial services," *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, vol. 2, no. 3, p. 100073, 2022.

[2] E. Albert, P. Gordillo, B. Livshits, A. Rubio, and I. Sergey, "Ethir: A framework for high-level analysis of ethereum bytecode," in *International symposium on automated technology for verification and analysis*. Springer, 2018, pp. 513–520.

[3] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (sok)," in *Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings 6*. Springer, 2017, pp. 164–186.

[4] A. S. Rajasekaran, M. Azees, and F. Al-Turjman, "A comprehensive survey on blockchain technology," *Sustainable Energy Technologies and Assessments*, vol. 52, p. 102039, 2022.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009, accessed: Apr. 7, 2024. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[6] P. Paul, P. Aithal, R. Saavedra, and S. Ghosh, "Blockchain technology and its types—a short review," *International Journal of Applied Science and Engineering (IJASE)*, vol. 9, no. 2, pp. 189–200, 2021.

[7] F. Hofmann, S. Wurster, E. Ron, and M. Böhmecke-Schwafert, "The immutability concept of blockchains and benefits of early standardization," in *2017 ITU Kaleidoscope: Challenges for a Data-Driven Society (ITU K)*, 2017, pp. 1–8.

[8] D. Magazzeni, P. McBurney, and W. Nash, "Validation and verification of smart contracts: A research agenda," *Computer*, vol. 50, no. 9, pp. 50–57, 2017.

[9] N. Szabo, "Smart contracts: building blocks for digital markets," *EXTROPY: The Journal of Transhumanist Thought,(16)*, vol. 18, no. 2, p. 28, 1996.

[10] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[11] R. Wilkens, R. Falk, R. Wilkens, and R. Falk, "Rechtliche aspekte," pp. 29–42, 2019.

[12] X. Liu, B. Hua, Y. Wang, and Z. Pan, "An empirical study of smart contract decompilers," in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2023, pp. 1–12.

[13] M. Bartoletti, A. Bracciali, C. Lepore, A. Scalas, and R. Zunino, "A formal model of algorand smart contracts," in *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part I 25*. Springer, 2021, pp. 93–114.

[14] Z. Sun, X. Luo, and Y. Zhang, "Panda: Security analysis of algorand smart contracts," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 1811–1828.

[15] DFINITY Foundation, "ic-wasm: A library for transforming wasm canisters running on the internet computer," 2024, accessed: Apr. 7, 2024. [Online]. Available: https://github.com/dfinity/ic-wasm

[16] G. O. Source, "Wabt: The webassembly binary toolkit, binary," 2023, accessed: Apr. 7, 2024. [Online]. Available: https://chromium.googlesource.com/external/github.com/WebAssembly/wabt/+/refs/tags/binary_0xb/README.md

[17] V. Buterin, "Ethereum white paper: A next generation smart contract & decentralized application platform," 2013, accessed: Apr. 7, 2024. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[18] S. Kepser, "A simple proof for the turing-completeness of xslt and xquery." in *Extreme Markup Languages®*. Citeseer, 2004.

[19] Y. Fu, M. Ren, F. Ma, H. Shi, X. Yang, Y. Jiang, H. Li, and X. Shi, "Evmfuzzer: detect evm vulnerabilities via fuzz testing," in *Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 1110–1114.

[20] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.

[21] C. Lepore, M. Ceria, A. Visconti, U. P. Rao, K. A. Shah, and L. Zanolini, "A survey on blockchain consensus with a performance comparison of pow, pos and pure pos," *Mathematics*, vol. 8, no. 10, 2020. [Online]. Available: https://www.mdpi.com/2227-7390/8/10/1782

[22] A. Chaudhury and B. Haney, "Smart contracts on algorand," 2021.

[23] T. D. Team, "The internet computer for geeks," Cryptology ePrint Archive, Paper 2022/087, 2022, accessed: Apr. 7, 2024. [Online]. Available: https://eprint.iacr.org/2022/087

[24] S. Klabnik and C. Nichols, *The Rust programming language*. No Starch Press, 2023.

[25] E. Hildenbrandt, M. Saxena, N. Rodrigues, X. Zhu, P. Daian, D. Guth, B. Moore, D. Park, Y. Zhang, A. Stefanescu *et al.*, "Kevm: A complete formal semantics of the ethereum virtual machine," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 204–217.

[26] M. Kaleem, A. Mavridou, and A. Laszka, "Vyper: A security comparison with solidity based on common vulnerabilities," in *2020 2nd conference on blockchain research & applications for innovative networks and services (BRAINS)*. IEEE, 2020, pp. 107–111.

[27] M. Suiche, "Porosity: A decompiler for blockchain-based smart contracts bytecode," *DEF con*, vol. 25, no. 11, 2017.

[28] N. Grech, L. Brent, B. Scholz, and Y. Smaragdakis, "Gigahorse: thorough, declarative decompilation of smart contracts," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1176–1186.

[29] N. Grech, S. Lagouvardos, I. Tsatiris, and Y. Smaragdakis, "Elipmoc: advanced decompilation of ethereum smart contracts," *Proc. ACM Program. Lang.*, vol. 6, no. OOPSLA1, apr 2022. [Online]. Available: https://doi.org/10.1145/3527321

[30] L. Brent, A. Jurisevic, M. Kong, E. Liu, F. Gauthier, V. Gramoli, R. Holz, and B. Scholz, “Vandal: A scalable security analysis framework for smart contracts,” *arXiv preprint arXiv:1809.03981*, 2018.

[31] EtherVM, “Online solidity decompiler,” 2024, accessed: Apr. 7, 2024. [Online]. Available: https://ethervm.io/

[32] palkeo, “Panoramix,” 2024, accessed: Apr. 7, 2024. [Online]. Available: https://github.com/palkeo/panoramix

[33] Y. Zhou, D. Kumar, S. Bakshi, J. Mason, A. Miller, and M. Bailey, “Erays: reverse engineering ethereum’s opaque smart contracts,” in *27th USENIX security symposium (USENIX Security 18)*, 2018, pp. 1371–1385.

[34] P. Zheng, B. Su, Z. Jiang, C. Yang, J. Chen, and J. Wu, “Exploring heterogeneous decentralized markets in defi and nft on ethereum blockchain,” in *2023 IEEE 10th International Conference on Cyber Security and Cloud Computing (CSCloud)/2023 IEEE 9th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, 2023, pp. 259–267.

[35] A. Hilbig, D. Lehmann, and M. Pradel, “An empirical study of real-world webassembly binaries: Security, languages, use cases,” in *Proceedings of the web conference 2021*, 2021, pp. 2696–2708.

[36] C. Sendner, H. Chen, H. Fereidooni, L. Petzi, J. König, J. Stang, A. Dmitrienko, A.-R. Sadeghi, and F. Koushanfar, “Smarter contracts: Detecting vulnerabilities in smart contracts with deep transfer learning.” in *NDSS*, 2023.

[37] M. A. Gioka, I. Lagouvardos, and I. Tsatiris, “Machine learning aided tuning of static analysis for evm bytecode decompilation,” 2020, accessed: Apr. 25, 2024. [Online]. Available: https://pergamos.lib.uoa.gr/uoa/dl/object/2923693/file.pdf

Improving MassDNS: Adding CNAME Resolution Output Information

Dimitar Vasilev, Patrick Sattler*, Johannes Zirngibl*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: dimitar.vasilev@tum.de, sattler@net.in.tum.de, zirngibl@net.in.tum.de

Abstract—**MassDNS is an open-source software for resolving domains on a large scale, that is used to capture and study the state of the Domain Name System (DNS). Currently, MassDNS combines in its output all resource records from all DNS responses. As a result, in addition to the address records, this output often contains CNAME records. This complicates the retrieval of IP addresses. We present a solution to this problem, that involves modifying MassDNS to perform the so-called CNAME resolution on each individual DNS response. This allows for more convenient IP address retrieval. CNAME resolution in this context simply means following all CNAME records and retrieving the IP addresses for a particular domain. This can be especially useful for studies, where only the resolved domains and their respective IP addresses are relevant. In addition, we use our new approach to evaluate our previous, post-processing approach for IP address extraction. As a consequence of performing the CNAME resolution on the entire output of MassDNS, our post-processing approach occasionally results in additional, unforeseen domain-to-address mappings. Our new approach prevents this while also performing better. Based on two scans with an input of around 1 Million domains, we find that our new approach takes around 23% less time for a complete scan. We also observed, that the post-processing approach introduced unexpected addresses to around 1% of all domains, which is relatively insignificant.**

Index Terms—**massdns, dns, cname**

1. Introduction

Today, many projects and studies focus on capturing and analyzing the state of the Internet. They usually require huge datasets, containing various information about the participants of the World Wide Web, e.g. IP addresses, Geo IP data, ASN (Autonomous System Numbers). Here, at the Technical University of Munich, there are several ongoing studies on this matter, united under the name GINO [1] - The Global INternet Observatory. For some of these studies, we use MassDNS to perform frequent scans of a relatively large portion of the DNS namespace. In this paper, we focus on implementing the CNAME resolution output functionality directly in MassDNS. This will allow us to easily retrieve the resolved domains and their IP addresses in a separate file. In this process, each DNS response is considered separately from all other responses. In contrast, the post-processing program of our previous approach takes as a basis for performing the CNAME resolution the entire output of MassDNS, which can lead to unforeseen domain-to-address mappings.

There are a few significant advantages of embedding the CNAME resolution directly into MassDNS. Firstly, this process can be performed on each individual DNS response, which prevents the error that our previous approach makes. Secondly, we eliminate the need for executing the post-processing program, thus making the scanning process more straight forward. And thirdly, by offloading the job of following `CNAME` records to MassDNS, we do not affect its performance. As a result, the new scanning workflow is significantly faster than the previous one.

When modifying MassDNS, we also made sure not to interfere with the normal output of the program. This is important, as we want to store these output files in an archive. We have been collecting our scan results for over 5 years now and would therefore like to retain their format.

In the rest of this paper, we will adhere to the following structure: Section 2 gives an overview of some projects, similar to MassDNS. In Section 3, we introduce some core concepts of the Domain Name System, relevant for this paper. Section 4 describes our previous, post-processing approach for extracting IP addresses of domains using MassDNS, along with the one that we propose in this paper. Here, we also provide concrete examples to better illustrate their differences. Alternative possible solutions are considered. Section 5 focuses on the changes we made in MassDNS. In Section 6 we evaluate the differences between the two approaches in terms of performance and final outcome (the set of domain-address pairs they produced). We conclude the paper with Section 7, where we summarize our work and emphasize the most important points of our evaluation.

2. Related Work

Researchers have two ways for acquiring large amounts of DNS related data. They can perform DNS scans themselves, or, alternatively, they can use the datasets, provided by other projects, companies and tools. In the following, we will discuss these options with the main focus being how convenient they are for retrieving the IP addresses of resolved domains.

ZDNS is another fast DNS Lookup Tool, part of a collection of open-source internet measurement tools, called the ZMap [2] project. Just like MassDNS, ZDNS can output all resource records from all DNS responses, meaning this output would need some processing for the aforementioned purposes. According to [3] by Liz Izhikevich et al., the authors of ZDNS have opted for a modular design for implementing the DNS query specific logic. This allows developers to implement custom behavior for performing lookups. If we were using ZDNS, custom module would be a possible way to implement the CNAME resolution output functionality. In this regard, ZDNS is pretty flexible.

OpenINTEL is also a well-known project for DNS measurement. Within this project, huge DNS scans are performed on a daily basis and the data is provided to researchers. Although not open-source, the overall design and implementation are presented in detail by van Rijswijk-Deij et al. in [4]. According to [4], their datasets "store all resource records included in the answer section of the DNS response, including all `DNSSEC` signatures, `CNAME` records and full `CNAME` expansions". Retrieving resolved IP addresses would therefore still require some explicit processing of the datasets OpenINTEL provides.

OpenINTEL and other similar projects have served as a basis for some of our previous studies ([5], [6]). These projects, however, do not offer any kind of control over the scanning process, which is why sometimes we prefer to perform the scanning ourselves and tailor it to our needs.

3. The Domain Name System

DNS is a distributed system for storing various information, assigned to names (domains). While the main goal is to provide a service for translating domains to host IP addresses, there is no restriction for this single application. For example DNS can be used with different internet protocol families, or to store mailbox data [7]. In this paper, we consider DNS only for the translation of domains to IPv4/v6 addresses. As explained in [7], the DNS consists of three major components: the *Resource Records*, the *Name Servers* and the *Resolvers*. **Resource Records(RR)** contain the data, that is associated with the names (domains). The RR-s, relevant for this paper, are described in Table 1. **Name Servers** are responsible for storing part of the Domain Name Space and making it available for others. **Resolvers** are programs, that communicate with Name Servers and extract information in response to client requests.

It is important to understand the purpose of `CNAME` records. Consider the two DNS responses, presented in Listing 1. In this example, we have resolved two domains: `blog.example.com` and `shop.example.com`. We see, that both domains are aliases for `example.com` and therefore have the same IP address. This configuration is convenient, because in case the IP address of `example.com` is changed, the resource records for `blog.example.com` and `shop.example.com` will still be resolved to the correct IP address. In general, `CNAME` records are useful, because usually one domain offers multiple services under different subdomains. At the same time, multiple domains can be hosted on the same machine or subnet and therefore need to be mapped to the same IP address. Here, the main takeaway is that to reach the address records for queried domains, often `CNAME` records must be followed. This applies to both the domain resolution process (which is handled by the resolver) and the output process (which we implement).

Listing 1: Example DNS Responses

```
;; Response for 'blog.example.com':
blog.example.com   CNAME   example.com
example.com        A       1.1.1.1

;; Response for 'shop.example.com':
blog.example.com   CNAME   example.com
example.com        A       1.1.1.1
```

TABLE 1: Record Types, [7], [8]

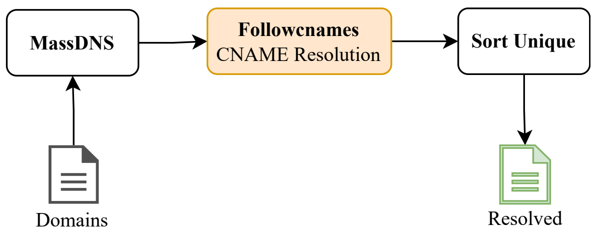
Record Type	Description
A	Address Record
AAAA	IPv6 Address Record
CNAME	Identifies the canonical name of an alias

4. Scan Workflows

In this section, we briefly present our previous scan workflow and the one that we propose. The concrete examples emphasize how exactly the two approaches differ from each other. In addition, we discuss other possible solutions and justify the decision to implement the desired functionality into MassDNS.

4.1. Post-Processing Scan Workflow

Figure 1: Post-processing Workflow Schematic



Until now, our process for resolving domains on a large scale consists of three steps, as visualized in Figure 1:
Step 1. With the right set of command line arguments, we pass a list of domains and resolvers to MassDNS and configure the output format and destination. Then, we execute MassDNS and get an output file, that contains all RR-s received from the resolvers.
Step 2. We pass the output file to our post-processing program, called Followcnames. This program performs the CNAME resolution. For each domain, it simply searches the entire output file of MassDNS for relevant RR-s. In case of `CNAME` records, all possible `CNAME` chains (sequences of `CNAME` records) are followed. In shortly we will show a concrete example to better explain this behavior.
Step 3. We use the Linux `sort` utility to remove all duplicating domain-address pairs. Duplicates can be introduced in the previous step, but in rare cases, DNS responses can also contain duplicating `A` or `AAAA` records. This way we keep our datasets clean and tidy.

Here is a real scenario, that we observed during our tests, in order to illustrate how the post-processing works. Suppose we want to resolve only two domains: `cookbook.openai.com` and app.rifei.com.br. Listing 2 shows a simplified version of the MassDNS output, that we observed.

Listing 2: MassDNS Raw Output

```
cookbook.openai.com   CNAME cname.vercel-dns.com
cname.vercel-dns.com A     76.76.21.22
cname.vercel-dns.com A     76.76.21.164
app.rifei.com.br      CNAME cname.vercel-dns.com
cname.vercel-dns.com A     76.76.21.142
cname.vercel-dns.com A     76.76.21.241
```

We can see, that two different domains are aliases for the same domain, which in turn has four different IPv4 addresses. Two of them were received for the first domain and the other two for the second domain. Now, when Followcnames tries to extract all addresses for the domain `cookbook.openai.com`, it will follow the `CNAME` record to `cname.vercel-dns.com` and then find all 4 different `A` records of `cname.vercel-dns.com`. The same goes for the second resolved domain, `app.rifei.com.br`. As a result all 4 IP addresses will be assigned to both domains. Listing 3 visualizes this result.

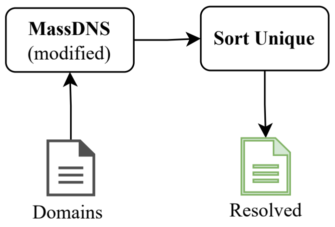
Listing 3: Post-processing Result

```
76.76.21.22, cookbook.openai.com
76.76.21.142,cookbook.openai.com
76.76.21.164,cookbook.openai.com
76.76.21.241,cookbook.openai.com
76.76.21.22, app.rifei.com.br
76.76.21.142,app.rifei.com.br
76.76.21.164,app.rifei.com.br
76.76.21.241,app.rifei.com.br
```

In cases like this, we say that the post-processing program (performing CNAME resolution on the entire MassDNS output) has affected the two resolved domains by assigning them additional IP addresses.

4.2. New Scan Workflow

Figure 2: New Workflow Schematic



Taking advantage of the improvement we made to MassDNS, a single scan with the new workflow, as shown in Figure 2, only includes **Step 1** and **Step 3** from our previous approach. Since we embedded the CNAME resolution into MassDNS, executing the post-processing program Followcnames is unnecessary. In addition, by following the `CNAME` records in each DNS response separately, we expect to get different results. For the same example of resolving `cookbook.openai.com` and app.rifei.com.br, we get the set of domain-address pairs, shown in Listing 4. We see that it is impossible for this approach to mix address records from different DNS responses, which is exactly what we aimed for.

Listing 4: Expected Result

```
76.76.21.22, cookbook.openai.com
76.76.21.164,cookbook.openai.com
76.76.21.142,app.rifei.com.br
76.76.21.241,app.rifei.com.br
```

4.3. Alternative Solutions

MassDNS supports different output formats, e.g. it can store the resource records from each DNS response in `json` format. Again, as a post-processing step, a simple script could parse this `json` output and extract the IP addresses associated with each domain from the respective DNS responses. However, judging by our Followcnames program, we expect similar performance from analogous post-processing scripts, which as we later confirm is not optimal. In addition, we must consider one important constraint when evaluating different possible solutions. As mentioned, we want to preserve the standard output of MassDNS, because over time we have stored a lot of historical data in the same format. With this requirement in mind, we see how modifying MassDNS and tuning it to our needs is the better solution and also aligns with our requirements for higher performance and efficiency.

In the rest of this paper, we call the approach that we propose the direct or the new approach, and the one that we used until now the post-processing or the previous one.

5. Implementation

Implementing the desired functionality into MassDNS does not require any major changes. As we do not want to replace the standard output of MassDNS, we first add a new command line option, called "`--ip-outfile`". It is used for specifying the file for the extracted pairs of domains and IP addresses.

We embed the address extraction process in the `do_read` function, in the `main.c` file. This function processes the responses of all DNS queries. If the query was successful, MassDNS parses the received DNS response and writes to the standard output file in the requested format. Immediately after that, we perform few additional steps. First, we check if `A` or `AAAA` records were requested. If this is the case, the DNS response is parsed once again. We follow the `CNAME` records, if there are any. Eventually, we reach the address records and write them in a separate file, together with the originally queried domain.

We do not expect this tweak to have any performance impact on MassDNS and confirm this in our evaluation. Network communication is rather slow, even when compared to tasks typically considered slow, such as IO operations on a persistent storage device. We suspect that despite the fact that MassDNS is designed with a high concurrency in mind, a significant portion of the total execution time is spent in waiting, whereas just a small fraction in processing the DNS responses. This explains how no slowdown would accumulate even when millions of domains are resolved.

TABLE 2: Evaluation of the Effect of Followcnames

Test	Total Domains	Resolved Domains	Affected Domains	Falsely Resolved Domains
#1	997 382	953 393 (95.6%)	6566 (0.7%)	100
#2	997 382	954 507 (95.7%)	13 002 (1.4%)	104

(a) Domains

Test	Total Pairs	Added Pairs
#1	1 572 013	17598(1.1%)
#2	1 581 746	25799(1.6%)

(b) Domain-Address Pairs

6. Evaluation

In this section we show if the changes we made in MassDNS negatively affect its performance. Afterwards, we compare the two approaches for address extraction with respect to the final result. For this we use real data from two MassDNS scans.

To conduct our tests, we need a set of domains, ideally a large sample, representative for the most frequently used domains in the domain name space. There are several top lists available for this purpose. We opt for the one, provided by CrUX [9] - the Chrome User eXperience Report. This list consists of around one million domains. This size is suitable for our tests, as it is large enough to be considered large-scale, but small enough for it to be feasible to compare the results.

6.1. Speed and Efficiency

For the given input, we did not observe any significant difference between the normal and the modified version of MassDNS. Here are the specific time measurements:

1) **MassDNS** - *1 Min. 40 Sec.* (both workflows)
2) **Followcnames** - *50 Sec.* (previous workflow)
3) **Sort Unique** - *1 Min. 10 Sec.* (both workflows)

However, since we skip the execution of Followcnames in our new workflow, the total compute time is reduced by around 23%. Based on rough calculations, we found that this improvement can save up to 3 hours of compute time within one of our typical scan days, depending on how well our solution scales.

6.2. Output Comparison

In Section 4 we showed that under certain circumstances our post-processing program can artificially introduce unexpected domain-to-address mappings. Given that we have used this scan workflow in the past, and considering it remains the only option for retrieving domain-address pairs from our archived scans, we want to estimate the magnitude of the error, that it produces. For this purpose we take advantage of the fact, that our modification of MassDNS does not affect its standard output. We proceed as follows: we run the improved MassDNS version with the aforementioned CrUX list as an input. Then, we pass the standard output file through the post-processing step. Effectively we just perform the old and the new approach simultaneously, in a single scan. We could also just conduct two scans with the two different workflows separately, but the scans tend not to be exactly reproducible, which could render the comparison invalid or misleading.

The data in Table 2 illustrates the measurable effect of the Followcnames program on the final outcome. In the following, we explain each statistic.

Resolved Domains. This is just a control statistic and shows the ratio of successfully resolved domains over all domains. Lower values should raise suspicion. This can indicate e.g. poor quality of the top list, some kind of error in the configuration or even in the implementation of MassDNS. However, we observe that around 95% of all domains were resolved, which is quite reasonable.

Affected Domains. As previously mentioned, affected are all domains, that received additional IP addresses as a result of the post-processing step Followcnames. We observed, that unforeseen addresses were assigned to around 1% of all queried domains. We have calculated, that each affected domain received on average between 2 and 3 additional addresses, but this value goes up to 56 for some domains.

Falsely Resolved Domains. The Followcnames program was able to find in the MassDNS output IP addresses for around 100 domains that our modification did not report as resolved at all. Upon closer inspection however, we found that all of these domains do exist and can be resolved, so we did not observe any non-existent domains to appear as resolved as a result of the post-processing.

Total and Additional Pairs. Under 2% of the all extracted domain-to-address pairs were artificially introduced by Followcnames.

Even though the amount of affected domains is relatively small, their presence still raises the question of how such differences can occur. Logically if two domains are aliases for the same, third domain, they should always be resolved to the same addresses. However, this is not always true, as we have seen in the example with `cookbook.openai.com` and `app.rifei.com.br` in Section 4. There can be several reasons for this. As an example, although it is unlikely, misconfigured Name Servers or invalid caches could cause such issues. Alternatively, Name Servers often store different information depending on their geolocation, which is a neat way of employing DNS for load balancing. Therefore, by communicating with different Name Servers, a resolver can receive different IP addresses for the same domain.

In the end, whether the differences we observed are significant depends on the context in which the resolved addresses are used. Theoretically, depending on the order in which different Name Servers are queried, a resolver can also receive the additional IP addresses, that our post-processing finds. This is why we believe that our previous use of the post-processing approach should not raise any concerns.

7. Conclusion

For several of our studies here, at the Technical University of Munich, we use MassDNS to perform huge DNS scans on a daily basis. However, the standard output of MassDNS is not convenient for our studies, as it contains the raw DNS responses. This is why our current scanning workflow includes some post-processing on the output of MassDNS. Essentially we perform the CNAME resolution on the entire output file. In this paper we present a new way to perform the CNAME resolution on a single DNS response level, in order to retrieve the resolved domains and their IP addresses. We embedded this functionality directly into MassDNS.

In the evaluation phase, we used our previous approach to assess the new one with respect to speed and efficiency. In addition, we used the new scan workflow to evaluate the impact of the post-processing procedure in our previous workflow. We show that the performance of MassDNS is not affected by our modifications whatsoever. Furthermore, by omitting the execution of the post-processing program Followcnames, we can save up to 3 hours of compute time within one of our typical scan days, which is a major improvement. Based on two large-scale scans with an input of approximately 1 million domains, we found that around 1% of all domains had received additional addresses as a result of the post-processing procedure. Some of them had up to 56 additional addresses. Moreover, during the CNAME resolution, the post-processing procedure was able to find addresses in the MassDNS output for around 100 domains that were not resolved during the scan. However, we believe that with the right combination of Name Servers, a resolver can also reach those IP addresses, that we considered as unexpected (those artificially introduced by Followcnames). For this reason, we do not label the previous approach as strictly invalid, nor do we reject its application to this point.

References

[1] "Global INternet Observatory," https://www.net.in.tum.de/projects/gino/, 2016, [Online; accessed 02-April-2024].

[2] "ZMap Project," https://zmap.io, 2022, [Online; accessed 04-April-2024].

[3] L. Izhikevich, G. Akiwate, B. Berger, S. Drakontaidis, A. Ascheman, P. Pearce, D. Adrian, and Z. Durumeric, "Zdns: a fast dns toolkit for internet measurement," in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 33–43. [Online]. Available: https://doi.org/10.1145/3517745.3561434

[4] R. van Rijswijk-Deij, M. Jonker, A. Sperotto, and A. Pras, "A high-performance, scalable infrastructure for large-scale active dns measurements," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 6, pp. 1877–1888, 2016.

[5] J. Zirngibl, S. Deusch, P. Sattler, J. Aulbach, G. Carle, and M. Jonker, "Domain Parking: Largely Present, Rarely Considered!" in *Proc. Network Traffic Measurement and Analysis Conference (TMA) 2022*, Jun. 2022.

[6] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle, "Scanning the IPv6 Internet: Towards a Comprehensive Hitlist," in *Proc. 8th Int. Workshop on Traffic Monitoring and Analysis*, Louvain-la-Neuve, Belgium, Apr. 2016. [Online]. Available: https://net.in.tum.de/pub/ipv6-hitlist/

[7] "Domain names - concepts and facilities," RFC 1034, Nov. 1987, [Accessed 02-April-2024]. [Online]. Available: https://www.rfc-editor.org/info/rfc1034

[8] V. Ksinant, C. Huitema, D. S. Thomson, and M. Souissi, "DNS Extensions to Support IP Version 6," RFC 3596, Oct. 2003, [Accessed 02-April-2024]. [Online]. Available: https://www.rfc-editor.org/info/rfc3596

[9] "CrUX," https://developer.chrome.com/docs/crux, [Online; accessed 04-April-2024].

Historic Developments in IPv6 Measurements

Florian Briksa, Lion Steger*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: florian.briksa@tum.de, stegerl@net.in.tum.de

Abstract—**Over the last decades, the Internet has become an important and integral part of our daily lives. Handling increasingly large amounts of devices interacting over the Internet, the old address space of Internet Protocol Version 4 (IPv4) is becoming too small. Therefore, in 1998 Internet Protocol Version 6 (IPv6) was developed, which has a significantly larger range of addresses. But IPv6 has also some disadvantages such as readability and, especially important for Internet Providers, it is almost impossible to keep track of all addresses being used on the Internet. This makes lists of currently active addresses, called hitlists, necessary. This paper analyses the growth process of IPv6 along with outliers in the data provided by a selected hitlist and offers a detailed view into composition of aliased addresses and usages of IPv6 around the world.**

Index Terms—**IPv6, measurement, historical analysis, aliasing**

1. Motivation

As IPv6 prevalence continues to grow, analyzing its usage is becoming increasingly important. Network administrators want to know traffic origins, while Internet Service Providers want to reliably allocate addresses and deliver information to their customers. Moreover, observant analysts can detect shifts in IPv6 usage during significant events like wars or disasters. These scenarios underscore the necessity for robust tools to analyze trends and detect anomalies in the IPv6 address space.

This paper focuses on fundamental analyses of IPv6 address space development. Section 5 delves into the composition of prefixes used for addresses and the countries utilizing IPv6 from 2018 to 2024, using a hitlist maintained by the Chair of Network Architectures and Services since 2018 [1] and geolocation tools. It presents a comprehensive view of the address space growth and identifies countries which have the biggest impact on communication over IPv6 according to the referred data.

In Section 6, we further explore outliers in the data, providing a concise before-and-after summary of address space changes and discussing potential origins.

2. Related Work

This paper is based on data obtained from the IPv6 hitlist maintained by the Chair of Network Architectures and Services since 2018 [1]. This hitlist is in the following just referred to as "hitlist". It also utilizes geolocation and technical information from the International Assigned Numbers Authority (IANA) [2], which oversees the assignment and usage of all IPv6 addresses assigned to customers worldwide. The works by Gasser et al. [3], Zirngibl et al. [4], and Steger et al. [5] were particularly helpful in identifying outliers resulting from internal changes in scan execution methodologies.

For comparison between geolocation tools and overall IPv6 usage in different countries, the insights provided by APNIC Labs [6] provided suitable information, particularly in terms of IPv6-capable and IPv6-preferring devices.

3. Methodology

During our research, we developed a tool to process hitlist information in multiple aspects. It was used to filter and create diagrams used in the following, along with a database interface for more efficient processing. To locate the country of IP-addresses, we used the WHOIS [7] database by the Internet Assigned Numbers Authority (IANA) [2].

As in this paper we focus more on countries than on exact addresses, we can reduce the lookups to WHOIS by storing the first 32 bits of each address in a database. This is possible because an ISP typically gets assigned the first 32 or fewer bits of an IPv6 address space for its customers from a Regional Internet Registry (RIR). It can now be assumed that most of the companies or institutions using those addresses operate in their home country, which makes the country identification up to 99% accurate [8].

It is important to mention that the WHOIS database only provides information about the country an AS is assigned to, not the servers on which the AS is running. Therefore, the precision of assigned and operating country may vary.

It also has to be noted that, as an exhaustive scan over all IPv6 addresses is not possible, the results presented in this paper may vary across different hitlist generators, as they possibly have completely different or varying generation methods of finding addresses [5].

4. Background

At first, we provide some background information to offer a clearer view of the research in this paper.

4.1. IPv6 Hitlist

The hitlist used in this paper is maintained by the Chair of Network Architectures and Services and includes lists categorized by aliased and non-aliased addresses, along with lists categorized by used protocols. The entries of the hitlist contain IPv6 addresses of responding servers during a regular scan of the address space. However, this paper primarily focuses on aliased prefixes because they illustrate the structure of IPv6 [9] addresses used in networks and allow a more efficient analysis, although they do not hold as much information as non-aliased addresses, as described in Chapter 4.4.

4.2. IPv6 Notation and Prefix

Each IPv6 address consists of 128 bits available for address location, providing a larger address space than IPv4 which has 32 bits available. It is followed by a number representing the prefix length of this address in bits. For example, an entry could look like this:

```
2401:4900:22dc:fab9::/64            (1)
```

Here we see a common IPv6 address in the usual CIDR (Classless Inter Domain Routing [10]) notation. The number after the slash describes the length of the prefix, in this case, 64 bits. The prefix is used for dividing the address space into sub address spaces of variable size. The longer the prefix, the smaller the resulting sub-address space. In this 64-bit prefix example, we would have 64 bits left to choose addresses for our devices in our network.

4.3. Network Categories and Protocols

The data used contains addresses from diverse network categories, including ISP (Internet Service Provider), NSP (Network Service Provider) and CDN (Content Delivery Network). These categories can be assigned by network operators to their Autonomous System (AS). As described in the paper by Lion Steger et al. [5], more than 42% of hitlist addresses are allocated to ISP networks. Furthermore, this paper considers the distinct behavior of addresses associated with their respective network categories and analyzes possible correlations between protocol and AS/Prefix composition.

Devices communicating over IPv6 use multiple protocols for message transmission. These protocols are the key to measuring the responsiveness of addresses. The hitlist used in our research conducts scans for TCP/80 (HTTP) and TCP/443 (HTTPS), ICMP, UDP/53 (DNS) and UDP/443 (QUIC) on a regular basis [5].

The last protocol to mention here is the ”Internet Control Message Protocol for the Internet Protocol Version 6” (ICMPv6). ICMPv6 is an important part of communicating with IPv6, as it reports errors and provides diagnostics [11], and all parts of this base protocol have to be implemented in all nodes communicating over IPv6.

4.4. Aliasing

IPv6 addresses can be further divided into aliased and non-aliased addresses. Gasser et al. [3] described aliased prefixes as subnets where every address in this subnet is mapped to and responded to by one single host, identified by this aliased prefix. Therefore, the number of aliased prefixes is usually much smaller than non-aliased ones.

However, aliased addresses usually do not hold as much information as non-aliased addresses, as they are used by ASes and not by single devices [3].

5. Basic Analysis

In this chapter we focus on long term trends visible in our processed data. For now, we ignore bigger outliers as much as possible to obtain a better view of the overall development of IPv6 usage in recent years.

5.1. Analysis of AS/Prefix Composition

In this first subchapter we start analysing the prefix composition of responsive addresses from July 2018 to april 2024. In Figure 1 we can observe the development of AS/Prefix composition.

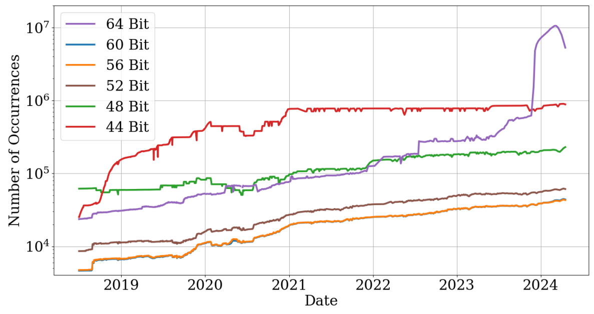


Figure 1: Composition of the six most used prefix lengths from year 2018 to 2024

At first, we can clearly outline that from 35 different prefix lengths from 29 to 120 bits, only three are extensively found: 44 bits, 48 bits and 64 bits. For end users, the use of 64-bit for addresses is recommended, as it is required for Stateless Address Autoconfiguration (SLAAC) to work [12]. SLAAC is used to generate IPv6 addresses for devices in a network without any further control from outside. The usage for end users is evident, as all of our most evaluated prefix lengths are at least 64 bits long. There may be several reasons to further divide the given address space into smaller subnets. One reason could be that an end user wants to connect multiple devices under the same prefix or uses several virtual machines in their network, having an individual internal routing topology. Another reason might be a network plan that is easier to remember. The host gets the original 64-bit prefix address, and then hierarchically structured sub-devices get the next 8 or 16 bits of their corresponding subnet etc [12].

Figure 1 also shows that from the beginning of the measurements in 2018 until July 2022 the 44 bit prefix clearly dominated and rose, while all other prefix lengths remained mostly stable. After January 2021, the number 44 bit prefixes found remained stable. This may be the result of new addresses being added under the already existing prefixes, and therefore not being added to the hitlist. In July 2022, the hitlist added new address candidates from new passive sources and target-generation methods [4]. This led to a general rise in the number of responsive addresses logged as well as in different categories of IPv6 addresses, especially in ISP (Internet Service Provider), NSP (Network Service Provider) and CDN (Content Delivery Network). It follows that, as we only have a rise in 64-bit prefixes at the same time, those categories directly correlate with our change in prefix length composition.

To conclude, it can be noted that not surprisingly most of the devices in the IPv6 address space are likely end users. Furthermore, it is possible that end users divide their address space into smaller subnets to better organize themselves.

5.2. Analysis of Geolocation Data

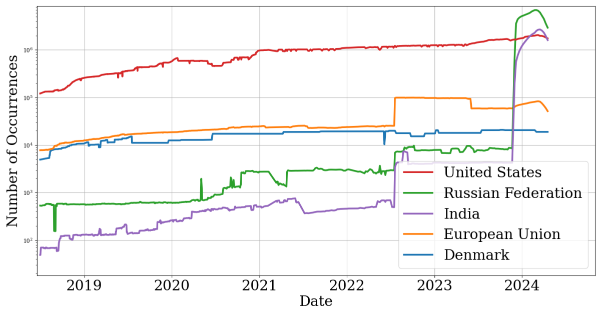


Figure 2: Composition of the six most active Countries from 2018 to 2024

Upon initial examination of Figure 2, it is clear that the United States has the most active and responsive IPv6 addresses, while addresses from outside the US did not play a significant role until July 2022. As mentioned earlier, new methods for finding IP addresses were applied to the hitlist at this date. Following July 2022, we can observe a visible but still relatively small increase in addresses originating from within the European Union.

Up to this point, the found addresses might indicate a lower priority of IPv6 in most countries except for the US. Only in recent months we have noted a slight increase in addresses originating from Vietnam. Upon analyzing the addresses from the EU, Denmark has, according to our data, had the most active addresses found in recent years.

It is worth mentioning that the most active addresses come from the US, India, the EU, and Russia. The only country not in the top five most active countries is China, which is not even noticeable among the other smaller countries in the diagram. This may be a result of the organized censorship of foreign servers under the Great Firewall of China, leading to only a few servers being connected to the rest of the world [5]. As Zirngibl et al. describes, those addresses lead to peaks and inaccuracies in the data. Therefore, most of the Chinese addresses are filtered [4].

We can observe a correlation between AS/Prefix Composition and geolocation data. The increase in 44-bit prefixes, followed by 48- and 64-bit prefixes, corresponds with the growing number of addresses originating from the US.

6. Analysis of Outliers

In this chapter, we deal with the identification of outliers in the examined data and further try to analyze their origins.

6.1. Jumps in Hitlist Data

The way addresses are scanned has a great impact on number and composition. For example, in the following section we describe a jump that occurred in July 2022. During this period of time, the found addresses with 64-bit prefixes increased from 186,000 to 277,000. The possible reason for this may be the paper published by Zirngibl et al. [4] in 2022, which presented new address candidate sources along with target-generation algorithms, the scans of the address space found more aliased addresses especially with 64 bit prefixes. This correlates with a jump in the overall number of responsive addresses found over ICMPv6.

Such jumps are not uncommon, as changes in algorithms are continuously applied and offer a wide research area. On the other hand, sudden breakouts may also happen when networks with greater numbers of addresses block parts of their address spaces from access from outside, making addresses unresponsive and therefore not listed in the data [1].

6.2. Plunge in Responsive Addresses in the US

The first outlier in our data happened in 2020 and was already visible in the previous figures:

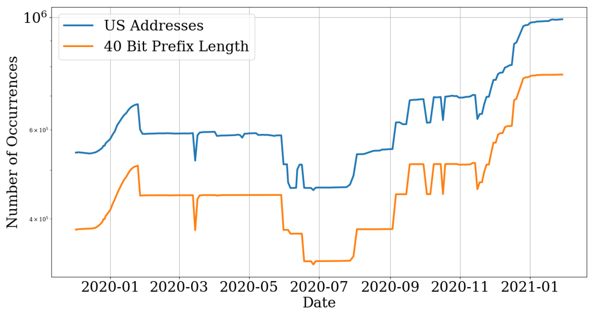


Figure 3: Outlier in Prefix Composition and US Addresses 2020

As prefix composition and number of responsive addresses from the US are clearly connected, we can assume that events inside the US were responsible for that outlier.

Before 2020, we found increasing numbers of responsive addresses originating inside the US. When we have a look at the data, the number of addresses had increased to 670,000 active addresses on January 24th, 2020, and sunk to 460,000 on June 24th.

After November 2020, the number of aliased addresses rapidly increased beyond the number measured before January, reaching a new maximum of one million aliased addresses.

The evaluation of more than 90% of US addresses leads to AWS Cloud Services in Seattle. As [13] states, more than 40 states used at least one of Amazon's election services in the 2020 presidential election campaigns.

As it is quite common for admins to block frequent address scans for hitlists in their firewalls, it may be possible that AWS has blocked addresses to critical infrastructure during the election period, causing this outlier in responding addresses from the US [3]. As AWS uses over 97% of the entire addresses located inside the US, we get a plunge during that election period in 2020.

6.3. Recent Peak in Responsive Addresses

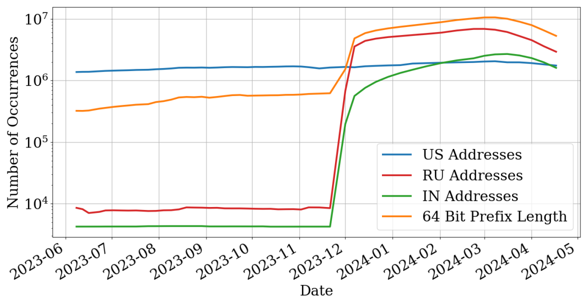


Figure 4: Outlier in Prefix Composition and Russian/Indian Addresses 2024

In the second major outlier to be discussed in this paper we can observe an immensely increasing number of 64-bit addresses especially originating from the Russian Federation and India, surpassing the number of addresses found in the US.

Before of this peak, the scans found fewer than 10,000 aliased addresses originated in Russia and India. During the following three months, responsive addresses from Russia increased to over 6.8 million and India to 2.7 million, while the dominating country in our scans, the United States, continuously increased to 2.1 million.

During our research, this number decreased as abruptly as it increased three months before. As visible in Figure 4, Russia and India still had the majority of scanned addresses, but a much lower level than at its maximum with 2.9 million and 1.6 million responsive addresses, respectively.

This peak may be the result of applying new filters and target-generation algorithms used by scanners to find active IPv6 addresses during this period. As the numbers also decreased at the same rate, we can assume that network administrators have blocked more addresses from being pinged by scans. As multiple Russian servers decreased their responsive addresses at a similar rate, it is possible that those servers have the same administrators, applying filters for their firewalls at the same time [3] [5].

Another reason may be the re-evaluation of addresses after being unresponsive for 30 days, causing a significant increase in protocol responsiveness over ICMPv6 as displayed as event ”I” in Figure 5 [1].

7. Conclusion and Future Work

In this paper, we discussed and analyzed various outliers in IPv6 hitlist data. We provided an overview of

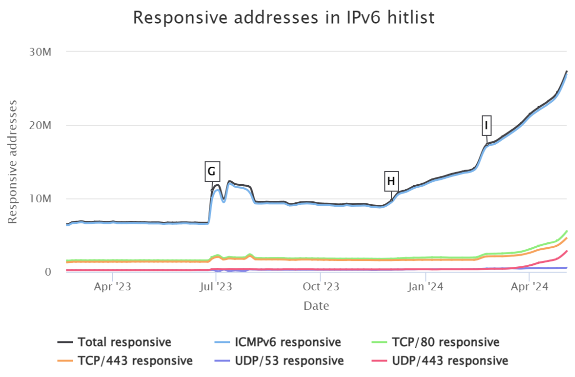


Figure 5: Protocol Responsiveness from January 2023 until now [1]

the overall development of IPv6 usage in recent years and explained the methodology behind collecting and structuring hitlist data.

We can clearly observe the general trends in IPv6 development, although the aliased addresses used for this paper represent only a small portion of globally active and responsive addresses. However, this subset of responsive addresses already provides insights that can help draw conclusions related to specific events in the countries where they occurred. We discovered that Chinese addresses have mostly been blocked, resulting in their underrepresentation in our dataset. On the other hand, India has significantly expanded and modernized its communication infrastructure, leading to a notable increase in responsive addresses.

In future research, it would be beneficial to extend this analysis to non-aliased addresses. Additionally, further analysis of the geographical locations of address origins using the implemented analysis tool could yield valuable insights. By comparing addresses located in different countries with global IPv6 usage statistics, we can draw conclusions about the composition and development of internet service infrastructure in those countries.

References

[1] O. Gasser, J. Zirngibl, and L. Steger:, “IPv6 Hitlist,” https://ipv6hitlist.github.io, 2024, [Online; accessed 5-May-2024].

[2] I. A. N. Authority, “IANA,” https://www.iana.org/whois, 2024, [Online; accessed 12-May-2024].

[3] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczynski, S. D. Strowes, L. Hendriks, and G. Carle, “Clusters in the expanse: Understanding and unbiasing ipv6 hitlists,” 2018.

[4] J. Zirngibl, L. Steger, P. Sattler, O. Gasser, and G. Carle, “Rusty clusters? dusting an ipv6 research foundation,” 2022.

[5] L. Steger, L. Kuang, J. Zirngibl, G. Carle, and O. Gasser, “Target acquired? evaluating target generation algorithms for ipv6,” Jun. 2023.

[6] A. Labs, “IPv6 Preferred Rate by country,” https://stats.labs.apnic.net/ipv6/, 2024, [Online; accessed 14-May-2024].

[7] IANA, “IANA WHOIS,” https://www.iana.org/whois, 2022, [Online; accessed 14-May-2024].

[8] iplocation.net, “How accurate is IP-based Geolocation Lookup?” https://www.iplocation.net/geolocation-accuracy, 2016, [Online; accessed 12-May-2024].

[9] IANA, “IPv6 Specification,” https://www.rfc-editor.org/rfc/rfc8200.html, 2017, [Online; accessed 14-May-2024].

[10] AWS, “Was ist CIDR?” https://aws.amazon.com/de/what-is/cidr/, [Online; accessed 28-May-2024].

[11] T. I. Society, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,” https://datatracker.ietf.org/doc/html/rfc4443, 2006, [Online; accessed 14-May-2024].

[12] serverfault.com, “How does IPv6 Subnetting Work and How does it differ from IPv4 Subnetting?” https://serverfault.com/questions/426183/how-does-ipv6-subnetting-work-and-how-does-it-differ-from-ipv4-subnetting, 2022, [Online; accessed 14-May-2024].

[13] Reuters:, “How Amazon moved into the business of U.S. elections,” https://www.nbcnews.com/tech/tech-news/how-amazon-moved-business-u-s-elections-n1066286, 2020, [Online; accessed 15-May-2024].

Exploring the Impact of Wi-Fi on PTP Synchronization: Leveraging Wi-Fi Features to Enhance Clock Synchronization

Ecem Zehra Büyük, Filip Rezabek[*], Leander Seidlitz[*]
[*]*Chair of Network Architectures and Services*
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: ecem.bueyuek@tum.de, frezabek@net.in.tum.de, seidlitz@net.in.tum.de

***Abstract*—Integrating Precision Time Protocol (PTP) with WiFi technology holds the potential to significantly enhance clock synchronization accuracy in wireless networks. This paper thus explores the feasibility and advantages of this integration, particularly in light of possible implementations of software and hardware timestamping in wireless networks that could lower the margin of error for timestamping in general. Furthermore, incorporating features of Time Sensitive Networking (TSN) into WiFi, we aim to balance the flexibility of wireless connections with the stability and low latency traditionally associated with Ethernet. Our findings suggest that realizing a synergy between PTP and WiFi can provide Ethernet-like latency, revolutionizing real-time applications and offering unprecedented reliability and performance. This synergy could lead to more efficient and better synchronized network systems, meeting the growing demand for precise time synchronization.**

***Index Terms*—PTP, TSN, wireless networks, clock synchronization, timestamping**

1. Introduction

As technology continues to permeate every aspect of human life, the importance of seamless communication between devices has never been greater. Consequently, the quest for precise time synchronization has emerged as a critical and highly discussed topic. Amongst various methods available, this paper focuses on the Precision Time Protocol (PTP) and how it might be impacted by Wifi. Furthermore, this paper also discusses whether features of Wi-Fi can be leveraged to minimize the impact on clock synchronisation.

The Precision Time Protocol is a message-based time transfer protocol that enables synchronization accuracy and precision in the submicrosecond range for packet-based network systems [1]. Because of its low latency, this protocol finds use in various time sensitive areas, such as telecommunications and the energy sector.

Time Sensitive Networking is another service of networking, which values time synchronization, high availability and bounded low latency through an Ethernet connection [2], utilizing PTP. With the cost of latency, WiFi introduces a more flexible, mobile and less complex networking. Unlike TSN, WiFi utilizes wireless connection instead of having a physical Ethernet connection [3]. Trying to minimize this compromise in latency by combining features of TSN within a WiFi implementation can significantly enhance the performance and availability of network systems whilst also remaining relatively uncluttered. Thus this paper focuses on if and how a synergy between Wifi and PTP be realised in order to achieve Ethernet-like latency in a wireless network.

2. Background

This section explains various methods for clock synchronization in PTP, followed by background information about NTP and the Wifi Standard.

2.1. Precision Time Protocol

The Precision Time Protocol (PTP), Figure 1, is designed to provide highly accurate time synchronization for packet-based network systems, achieving precision down to the submicrosecond level. Introduced in the IEEE 1588 standard [4], it operates by exchanging timing messages between network devices, thereby ensuring that all devices maintain a consistent and precise time reference across the network. A PTP packet is composed of a PTP daemon and a lower part, which timestamps the packets [5]. Synchronization is achieved through syncing the slave clock (secondary clock) to the master clock (primary clock) [5]. The accuracy of the synchronization is measured by calculating the difference between the time held on the master node and slave node. PTP offers two delay calculation modes, Peer to Peer and End to End.

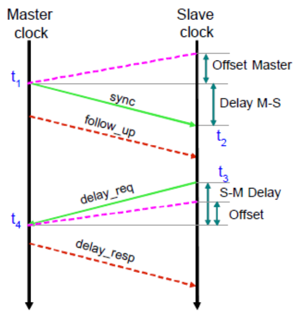


Figure 1: PTP protocol

In Peer to Peer (P2P) delay calculation mode the network operates in a manner where every participant or peer holds capabilities and duties. This means that each peer can both act as a client and server, allowing for direct communication and resource sharing between them without the need of a central server. This is achieved through establishing a distributed system with each node running the same software [6].

End to End (E2E) delay calculation mode refers to a method of communication or data transfer where the information is encrypted at the senders' end and can only be decrypted by the intended recipient at the other end. This ensures that the information remains confidential and secure throughout the transmission, and can only be accesssed by the sender and the recipient.

Both P2P and E2E involve direct communication between participants, P2P refers to the network architecture, while E2E refers to the security and privacy measures applied to data transmissions. Through the P2P mode, a more accurate link delay measurement can be achieved, which then also leads to a better clock synchronization. Although for P2P to be effective, it requires all network devices to be PTP capable, since P2P views each participant as a peer with identical abilities and responsibilities. On the other hand, E2E supports non PTP devices, but in return has a worse clock synchronization performance when it comes to larger network scales [4].

In addition to different delay calculation modes, clocks are also utilized in the PTP standard to guarantee a synchronized and precise transmission of data. PTP offers three clock variations to choose from, the Ordinary Clock, the Boundary Clock and the Transparent Clock.

Ordinary Clock is the simplest version of available clocks and only has one port. That port is either used as a slave or master. In comparison to OC, the Boundary Clock is relatively more complex, possessing two or more ports, which are all in the master state with a single exception being in the slave state, which is then used to synchronize the internal clock within the BC. Thus BC is also considered to be a complete PTP node, allowing its synchronized inner clock to be used by other applications in the PTP topology. This is the main difference between BC and the Transparency Clock. Instead of synchronizing its inner clock, TC forwards the PTP message and adjusts a time correction field in the PTP message according to the residence time in the TC [7]. Transfer Clock feature is used by bridges or routers to assist clocks in measuring and adjusting for packet delay. It computes the variable delay as the PTP packets pass through the switch on the router. Any of these Clocks can be the Grandmaster Clock, which is then used by the network as the main source of time and is used as a reference for other clocks to synchronize their times with.

In the Linux ecosystem, there exists an implementation of the Precision Time Protocol *linuxptp*, a design according to the IEEE 1588 standard. This software can be used to configure PTP service on a system. *Linuxptp* consists of *ptp4l* and *phc2sys*. *Ptp4l* is used for the implementation of PTP, specifically for the OC and BC. Meanwhile *phc2sys* is used for synchronizing two clocks, the PTP hardware Clock (PHC) and system clock, as its name suggests [8]. Depending on the timestamping version, the implementations of this software vary. If hardware timestamping is being used, *ptp4l* is utilized to adjust PHC whilst *phc2sys* adjusts the system clock. If the system opts for software timestamping, then *ptp4l* directly adjusts the system clock and *phc2sys* is not needed [8].

2.2. Network Time Protocol

PTP's predecessor, the Network Time Protocol (NTP), was developed and released in 1985 and is used to organize and maintain a set of time servers and transmission paths as a synchronization subset [9]. With NTP, a precision within the millisecond range is possible [9]. Synchronization of the clocks follow a hierarchical structure, in which clocks near the top are considered more accurate than the ones near the bottom. Clients then take these more accurate clocks as reference to synchronize their time [10]. Due to its simpler build, NTP has become a central protocol for many applications requiring time synchronization over the internet and is still used for applications that do not demand a higher level of precision. However its milisecond precision is not precise enough for modern applications demanding higher accuracy. This limitation paved the way to the development of PTP, achieving a higher precision range. As real time applications continue to evolve, the enhanced precision of PTP becomes even more essential.

2.3. the Wi-Fi Standard

Wireless Fidelity, short for Wi-Fi, is a wireless transmission of radio signals and acts essentially as an alternative for Ethernet for network connectivity in modern systems [11]. WiFi was released in the 1990s with IEEE 802.11 standard [12]. Free of the limitations of a cable, WiFi offers an extended reach to places previously unavailable to a cabled connection. Without such need for a cable infrastructure, WiFi offers a lower cost in comparison to Ethernet while simultaniously enhancing mobility. Nevertheless this flexibility of WiFi comes at the expense of latency and inconsistency [13]. In contrast to stable connection of a wired network, mobility, signal strength and neighboring interference render wireless networks unpredictable [5]. As a synchronization option in wireless networks, IEEE 802.11 introduces the Time Synchronization Function (TSF). This mechanism harmonizes clients with the time broadcasted in the AP's beacons [5]. The problem with this method is that it only works well within the range of one AP, but mobile devices might move across larger areas. To able to synchronize wireless clients streching across large areas, PTP is utilized.

WiFi primarily functions within three frequency bands: 2.4 GHz (802.11), 5 GHz (802.11ac), and the more recent 6 GHz (802.11ax). Newer versions offer more channels and higher data rates with higher speed whilst also lowering latency and solving interference problems the 2.4 GHz band had. However, 5 GHz and 6 GHz implementations have a reduced range [14].

3. Analysis

This section explores the distinctions between PTP and NTP through a comparative analysis and explores the potential integration of precision timing technologies with wireless networks.

3.1. Comparison between NTP and PTP

Although both NTP and PTP provide time synchronization over a packet based network, in analysing the qualities NTP and PTP provide, it becomes evident that both protocols differ in terms of accuracy, topology, hardware and thus may benefit from different applications.

NTP achieves millisecond to sub-millisecond accuracy whereas PTP excels with sub-microsecond precision. Furthermore NTP employs a client-server hierarchical topology, in contrast to PTP, which adopts a peer-to-peer architecture, eliminating traditional hierarchies. While NTP operates efficiently with standard Ethernet hardware, PTP requires specialized equipment to reach its superior accuracy levels.

3.2. PTP with WiFi

Numerous applications rely on precise timing for the exchange of sensor data and control signals. Failure to meet these deadlines can lead to operational issues, instability, and safety risks. Due to PTPs significant improvements in terms of latency compared to its predecessor, a fusion of PTP and Wifi became a very prominent research topic. Wifi's adaptability, when combined with the precision PTP provides, presents potential benefits and improvements in terms of speed and accuracy for wireless local area networks (WLAN). The problem with combining PTP with Wireless Fidelity manifests itself within the uncertainties in PTP timestamps [5]. The uncertainties stem mainly from fluctuating delays, in data packets, signal interference and the intrinsic characteristics of communication all of which can impact the accuracy that PTP strives to deliver. As explained in *Section 2*, PTP is a protocol based on wired connections, mainly Ethernet, whilst WiFi aims to achieve a more flexible and lower-cost connection network sacrificing better synchronization a wired connection brings. Despite these hurdles, continuous research and enhancements in WiFi technology in combination with software and hardware based timestamping in PTP, and pushing TSN towards wireless networks strive to mitigating these issues and achieve highly accurate synchronization.

4. Integrating PTP and WiFi

Even though originally designed for wired LANs, there are several implementations of PTP for wireless networks. In contrast to wired networks, wireless channels introduce uncertainties in PTP timestamps. Recent work to overcome these instabilities involve timestamping and developing a wireless TSN variant, e.g. WTSN.

4.1. Timestamping

One workaround to synchronize wireless devices through PTP is to use timestamping (TS).

Figure 2 assumes Boundary Clock as the default clock method, which means synchronization transpires in several steps. The System Clock of the AP is designated as the master clock, which other clocks synchronize themselves to. The system clocks of the clients are regarded as the slave clock and thus sync themselves to the AP through WiFi. The Master Clock is synced with the help of a PTP clock, connected through LAN. There are two possible approaches, hardware timestamping and software timestamping.

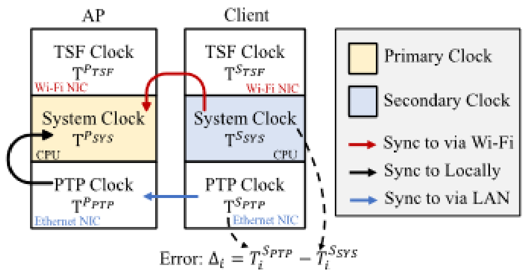
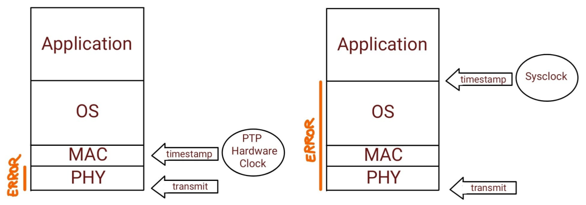


Figure 2: Wireless Clock Synchronization [5]



(a) Hardware Timestamping (b) Software Timestamping

Figure 3: Types of Timestamping

Hardware timestamping involves using dedicated hardware components within network devices, such as network interface cards (NICs), to generate timestamps directly at the physical layer of the network stack. But wireless networks use WNIC (Wireless Network Interface Controller) instead of NIC and thus do not support hardware counters that are needed for hardware timestamping. A solution for this issue presented by [5] is to treat TSF as the hardware clock. Thus, we can emulate the hardware PTP process used for Ethernet NICs on WNICs. The application of hardware timestamping produces sub-microsecond bias error and jitter.

Software timestamping on the other hand, captures precise time information at the software level instead of relying on hardware timestamps. This approach facilitates clock synchronization across networked devices, especially in wireless environments where hardware timestamping may not be feasible due to cost or practical limitations. Software timestamping records the exact time when a PTP event message is processed by the protocol stack. This, however leads to a worse performance and a bigger error margin when it comes to synchronization in comparison to hardware timestamping *(Figure 3)*. Since clock synchronization relies on getting the time from the system clock [8], its synchronization is neither accurate nor stable [5]. To minimize inaccuracy, we can observe the effect Interrupt Mitigation and CPU Power Management has over latency in software timestamping. Interrupt Mitigation aggregates multiple NIC interrupts into one singular interrupt to reduce computational cost and performance impact. When interrupt mitigation is enabled, interrupts of packets that were received at the same time period get delayed and gets a later response from the system, which results in latency [5]. This feature can be disabled to minimize timestamping latency. Moreover, dynamic ticks can also be disabled to boost clock stability [8]. When there are no operations requiring much computational power, modern CPUs turns off several hardware components to be more lightweight and to conserve power [5]. When disabled, software timestamping performs much better with a smaller offset [8].

4.2. Extending TSN towards Wireless Networks

Time Sensitive Networking (TSN) are a set of standards for providing a deterministic and reliable connection in Ethernet networks [15]. But due to the insufficient flexibility of a wired network, newer networks shifted towards wireless connections, sacrificing determinism TSN brings along the way. Wifi 7 (IEEE 802.11be), released in Januay 2024, aims to also integrate TSN extensions for low-latency real time traffic [3]. The central challenge involves adapting TSN mechanisms, initially tailored for Ethernet, to the inherently less predictable wireless environment. This task entails tackling issues like link unreliability, asymmetric path delays, and interference, all while maintaining compatibility with existing WiFi standards. Notably, the wireless network should have less overhead to achieve an accurate clock synchronization [16].

IEEE 802.11be introduces significant enhancements to both the physical (PHY) and medium access control (MAC) layers, specifically tailored to support TSN. On the PHY side, the amendment incorporates the 6 GHz band, allowing for wider channels up to 320 MHz and supporting higher modulation schemes like 4096-QAM [3]. These improvements collectively enhance data rates and reduce latency. Additionally, the expansion to 16 spatial streams optimizes spectrum utilization, benefiting time-sensitive applications by minimizing waiting times in buffers.

At the MAC layer, key advancements include extending multi-user (MU) capabilities such as MU-MIMO and orthogonal frequency-division multiple access (OFDMA) [3]. This extension is included in IEEE 802.11be. These technologies enhance spectral efficiency and decrease channel access latency by enabling simultaneous transmissions between multiple users. According to [17], the introduction of multi-link operation (MLO) is substantial. MLO allows multiple links for a single transmission, improving throughput, reliability, and latency. Opportunistic link selection, link aggregation, and multi-channel full duplex operations further enhance time-sensitive network handling. Their research believes that MLO will allow for a better performance for real-time applications even with the presence of heavy network traffic.

Furthermore, IEEE 802.11be emphasizes multi-AP coordination, bolstering its TSN capabilities [3]. By enabling access points (APs) to coordinate transmissions and share opportunities, the amendment reduces inter-network interference and optimizes overall network performance. This coordination is particularly valuable in operation settings with closely located APs.

The integration of TSN into WiFi 7 via IEEE 802.11be holds promise for various IoT applications, including multimedia, healthcare, industrial automation, and transportation. These applications demand low-latency and high-reliability communication, which WiFi 7 addresses through advanced PHY and MAC enhancements, whilst caving the way for a wireless implementation of the TSN regulations. Future challenges for next WiFi implementations include optimizing the existing PHY layer to reduce computational costs and achieve ultra-low latency, whilst also maintaining efficiency and network management [17]. While challenges persist in adapting TSN to wireless contexts, ongoing research shows a bright future for time-sensitive wireless communications.

5. Conclusion and Future Work

Evident with the advances made in the latest Wifi releases and continuous research refining the accuracy of software and hardware timestamping, merging Precision Time Protocol with WiFi not only seems feasible but also promises significant rewards in terms of precision and clock synchronization. Developing WiFi in the direction of TSN regulations and finding a middle ground between flexibility of wireless connections and the stability TSN brings could revolutionize real-time applications, offering unparalleled reliability and performance. When it comes to optimizing timestamping, disabling Interrupt Mitigation and dynamic ticking would end up reducing power efficiency, which is also an important aspect to consider in mobile devices. Conducting research on how to configure the operating system in a way to allow logical duty cycling between active and idle modes inbetween transmitting can optimize power management and enhance overall network efficiency. Combined with the evolution of TSN regulations, WiFi can offer Ethernet-like latency, setting a new standard for real-time communication in diverse applications.

References

[1] S. T. Watt, S. Achanta, H. Abubakari, E. Sagen, Z. Korkmaz, and H. Ahmed, "Understanding and applying precision time protocol," pp. 1–7, 2015.

[2] N. Finn, "Introduction to Time-Sensitive Networking," vol. 2, no. 2, pp. 22–28, 2018.

[3] T. Adame, M. Carrascosa-Zamacois, and B. Bellalta, "Time-sensitive networking in IEEE 802.11 be: On the way to low-latency WiFi 7," *Sensors*, vol. 21, no. 15, p. 4954, 2021.

[4] M. Aslam, W. Liu, X. Jiao, J. Haxhibeqiri, G. Miranda, J. Hoebeke, J. Marquez-Barja, and I. Moerman, "Hardware Efficient Clock Synchronization Across Wi-Fi and Ethernet-Based Network Using PTP," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 3808–3819, 2022.

[5] P. Chen and Z. Yang, "Understanding PTP Performance in Today's Wi-Fi Networks," *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, pp. 3037–3050, 2023.

[6] S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "VON: a scalable peer-to-peer network for virtual environments," vol. 20, no. 4, pp. 22–31, 2006.

[7] F. Rezabek, M. Helm, T. Leonhardt, and G. Carle, "PTP Security Measures and their Impact on Synchronization Accuracy," pp. 109–117, 2022.

[8] K. Ichikawa, "Precision time protocol on linux introduction to linuxptp," in *Proc. Linux Conf.*, 2014.

[9] D. Mills, "Internet time synchronization: the network time protocol," vol. 39, no. 10, pp. 1482–1493, 1991.

[10] D. L. Mills, "RFC0958: Network Time Protocol (NTP)," 1985.

[11] *An Introduction to Wi-Fi*, Digi International Inc., 2007-2008.

[12] A. A. O. Wafa S. M. Elbasher, Amin B. A. Mustafa, "A Comparison between Li-Fi, Wi-Fi, and Ethernet Standards," vol. 4, no. 12, 2015.

[13] C. Pei, Y. Zhao, G. Chen, R. Tang, Y. Meng, M. Ma, K. Ling, and D. Pei, "WiFi can be the weakest link of round trip network latency in the wild," pp. 1–9, 2016.

[14] S. Fan, Y. Ge, and X. Yu, “Comparison Analysis and Prediction of Modern Wi-Fi Standards,” pp. 581–585, 2022.

[15] M. K. Atiq, R. Muzaffar, Ó. Seijo, I. Val, and H.-P. Bernhard, “When IEEE 802.11 and 5G Meet Time-Sensitive Networking,” *IEEE Open Journal of the Industrial Electronics Society*, vol. 3, pp. 14–36, 2022.

[16] J. Haxhibeqiri, X. Jiao, M. Aslam, I. Moerman, and J. Hoebeke, “Enabling TSN over IEEE 802.11: Low-overhead Time Synchronization for Wi-Fi Clients,” vol. 1, pp. 1068–1073, 2021.

[17] D. Cavalcanti, C. Cordeiro, M. Smith, and A. Regev, “WiFi TSN: Enabling Deterministic Wireless Connectivity over 802.11,” *IEEE Communications Standards Magazine*, vol. 6, no. 4, pp. 22–29, 2022.

Attestation Capabilities of Trusted Execution Environments in the Wild

Eber Christer, Filip Rezabek*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: eber.christer@tum.de, frezabek@net.in.tum.de

Abstract—**Attestation is one of the most crucial mechanisms of confidential computing, allowing trust to be established between software systems. While the underlying concept of software attestation remains consistent in all Trusted Execution Environments (TEEs), each silicon manufacturer has their own unique approach to this process – different architectures, isolation guarantees, measurements, and attestation flows. This paper aims to abstract the intricacies behind TEE technologies by providing a general overview of the underlying concepts behind TEEs, as well as present the attestation capabilities of industry-recognized TEE solutions, such as Intel SGX, Intel TDX, and AMD SEV-SNP.**

Index Terms—**confidential computing, trusted execution environments, attestation**

1. Introduction

In an era of digital transformation, the notions of data integrity and confidentiality are becoming increasingly prevalent. While a traditional computing system is able to protect data at rest and in transit, data in use remains vulnerable [1]. Confidential computing is a technique that aims to mitigate this problem by ensuring that sensitive computations are performed inside a Trusted Execution Environment (TEE). At its core, TEE is a hardware-based mechanism that isolates software execution in a secure region within the memory and CPU, preventing unauthorized access and tampering [2], [3]. Regardless of the TEE implementations, these capabilities of ensuring data integrity and confidentiality are particularly useful when dealing with public cloud platforms, where the underlying system is considered untrusted, opaque, and uncontrollable [4].

Before sharing confidential information and executing any processes inside a TEE, the trustworthiness of said execution environment must be proven in a process called attestation. The inherent mechanisms and guarantees that can be attested by each TEE differ from one vendor to another. Therefore, this paper explores the attestation capabilities of different state-of-the-art TEEs. In Section 2, the theoretical background of TEEs and attestation methods are given, followed by the architecture and attestation methods of different state-of-the-art TEE technologies in Section 3. Finally, Section 4 concludes the paper by highlighting the key findings and outlining directions for future work and improvements.

2. Background

This section presents relevant background information on the concepts relevant to this paper, namely the different TEE models and the general attestation flows.

2.1. TEE Models

Process-Based Model. Process-based TEEs, such as Intel Secure Guard Extensions (SGX) and ARM TrustZone, provision encrypted memory areas called secured enclaves where sensitive workloads can run in isolation [5]. This process involves separating an application into two components: trusted and untrusted. When a workload is to be executed in isolation, an enclave with the necessary resources (mainly memory) is created, where the computations will take place [4]. The untrusted component serves as an interface that communicates with the OS and bridges the secure enclaves with the rest of the system through dedicated channels [6].

VM-Based Model. On the other hand, VM-based TEEs involve dynamically encrypting the memory of a confidential VM (CVM) [5]. This process isolates the whole application running inside the VM from the hypervisor itself. On top of data confidentiality, state-of-the-art VM-based TEEs such as Intel Trusted Domain Extensions (TDX) and AMD Secure Encrypted Virtualization-Secure Nested Paging (SEV-SNP), have additionally introduced integrity-preserving features [2], [3].

2.2. Attestation Types

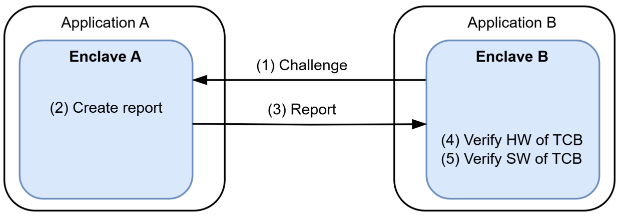


Figure 1: Generic local attestation protocol [6]

Local Attestation. As the name suggests, local attestation refers to the process in which one enclave verifies the identity and integrity of another on the same TEE-enabled CPU via a challenge-response protocol [6]. In general, since the enclaves reside in the same hardware, the authenticity of the local attestation request can be verified using a Message Authentication Code (MAC)–based symmetric-key scheme [4].

Figure 1 visualizes a generic flow for a local attestation. In this case, Enclave B wants to verify that Enclave A is running on genuine TEE-enabled hardware. The process is as follows [4], [6]:

1) Enclave B sends Enclave A its enclave-unique information.
2) Enclave A generates an attestation report containing a cryptographic hash of its initial state, Enclave B's identity, and a symmetric key (i.e., Diffie-Hellman Key).
3) Enclave A sends Enclave B the attestation report encrypted with a platform-specific key
4) Enclave B retrieves the platform-specific key. If the attestation report can be verified by Enclave B using the platform-specific key, then both enclaves are on the same TEE platform.
5) Enclave B verifies the report content to authenticate the software component of the TCB.

Assuming both parties have verified their security measures are as expected, a secure channel can be created using the symmetric key propagated from the attestation reports.

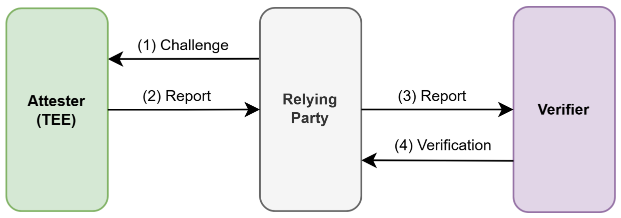


Figure 2: Generic remote attestation protocol [7]

Remote Attestation. On the contrary, remote attestation establishes trust between environments running on different hardware or platforms [4]. Generally, this process involves the following components [7]:

- **Verifier**: A tool used to authenticate the evidence provided by the TEE based on configured policy, ensuring that the certificate is genuine and that the issuer is trusted.
- **Attester**: A TEE looking for its evidence or measurements to be appraised by the verifier, such that it can be trusted to execute workloads.
- **Relying Party**: An entity that uses information about an attester to determine its trustworthiness.

Before workloads can be executed inside a TEE, trust between the *relying party* and the TEE must be established through the *verifier*. The *relying party* first sends a challenge to the *Attester* (TEE), requesting an attestation (1). The TEE submits a set of claims to prove its trustworthiness to the *relying party* through a signed attestation report. Depending on the requirements enforced by the attestation protocol, additional claims – such as roots of trust, trusted computing base (TCB), and metadata – may be requested (2). The *relying party* relays this report to the *verifier*, which will appraise the evidence by applying constraints and enforcing policies. Upon successful verification, an asymmetric key is issued to the *attester* through an attestation report, which it can use to build a safe communication channel between the *relying party* and the TEE (4). Figure 2 shows a generic remote attestation flow, summarizing the overall process [7].

2.3. Challenges and Limitations

While TEEs offer an additional layer of security to applications, a barrier impeding the wide adoption of this technology includes their proprietary nature and lack of standardization. In fact, a majority of successful attacks stemmed from specific TEE design flaws [8]. For example, early iterations of AMD-based TEEs were susceptible to unencrypted register attacks. This reliance on a vendor for updates, security patches, and support can create a single point of failure. Moreover, since there is limited transparency about the internal workings of TEEs, it becomes difficult to assess their security or trustworthiness independently.

This paper aims to identify common design patterns across different state-of-the-art TEEs. For the following analysis, we additionally assume that the silicon manufacturer can be fully trusted to provide secure and reliable TEE solutions.

3. Attestation with State-of-the-Art TEEs

This section explores state-of-the-art TEEs widely adopted by various cloud service providers (CSPs), namely Intel SGX, Intel TDX, and AMD SEV-SNP. While the inherent mechanism of TEEs remains consistent across different solutions, the enabling architecture differs. For each TEEs, we aim to study their security guarantees and attestation capabilities by answering the following questions:

- How are data **confidentiality and integrity** achieved?
- What **measurements** are collected?
- How is the **attestation** conducted?

3.1. Intel SGX

Intel SGX extends the instruction set architecture, allowing it to generate and manage process-based TEEs called enclaves.

Confidentiality and Integrity. Enclave data and code are stored in an encrypted memory region within the DRAM called the Enclave Page Cache (EPC) [6]. The system software (i.e., OS kernel or hypervisor) is responsible for allocating and freeing pages in the EPC for the enclaves, which are created through the application software [9]. The EPC is designed so that an enclave's artifacts are inaccessible to non-enclave software, including the application that created it. This restriction serves as the basis for SGX's confidentiality guarantees. In addition, SGX offers integrity guarantees of its enclaves through its local and remote attestation procedures [4].

Measurements. An enclave is initialized with an SGX Enclave Control Structure (SECS) within a dedicated EPC page to store its identity [9]. SECS comprises two measurement registers called `MRENCLAVE` and `MRSIGNER`

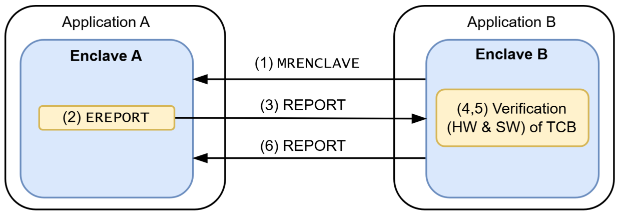


Figure 3: Intel SGX local attestation flow [6]

[10]. The former represents the enclave measurement and contains logs pertaining to the enclave's memory – this includes the contents, positions, and security measurements of the pages used by the enclave. While the latter represents the sealing identity and contains a hash of the enclave author's public key. SECS also includes other fields, such as the attributes of the enclave, product ID, and Security Version Number (SVN) of the modules [9].

Attestation. Local attestation establishes trust between enclaves residing in the same SGX platform. We describe this process as shown in Figure 3 by adapting the generic naming conventions and local attestation process used in Figure 2. The following local attestation flow is adapted from [9], [10], and the SGX developer guide [6]. After establishing a communication channel between Enclave A and B, Enclave B retrieves its `MRENCLAVE` and sends it over to Enclave A (1). Using Enclave B's `MRENCLAVE` as input, the CPU instruction `EREPORT` generates a signed attestation report (REPORT), which binds Enclave A's identity information from its SECS using a MAC tag. This MAC tag is computed using a symmetric key shared exclusively between the target enclave and the same SGX SVN (2). Enclave A sends this signed attestation report over to Enclave B (3), where the report's authenticity can be verified using the MAC tag (4). By invoking the `EGETKEY` command, Enclave B is capable of retrieving the Report Key (symmetric key) needed to recompute the MAC of the REPORT. If the MAC generated by Enclave B matches the MAC of the attestation report, then that would mean that both enclaves reside within the same platform, and the hardware component of the TCB can be verified. Enclave B proceeds to examine the content of REPORT to authenticate the software component of the TCB (5). After all components are verified, Enclave B generates a new REPORT using Enclave A's `MRENCLAVE` and sends it over to Enclave A (6). Enclave A can use this attestation report to verify that Enclave B resides on the same platform as it does.

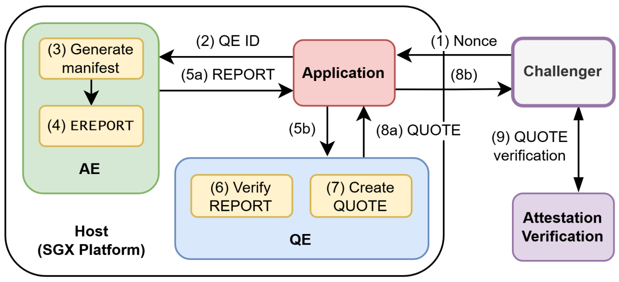


Figure 4: Intel SGX remote attestation flow [10]

Remote attestation, on the other hand, is enabled by a special enclave called the *Quoting Enclave* (QE) which replaces the MAC-bound REPORT (locally verifiable) with a signature generated by a device-specific asymmetric key, creating what is known as a QUOTE (remotely verifiable) [11]. This key is provisioned by the Intel Enhanced Privacy ID (EPID), which protects the identity of the signer [10].

Figure 4 details the process of attestation by an *Application Enclave (AE)* to a remote *Challenger*, adapted from [10] and the SGX developer guide [6]. In this case, the *Challenger* first sends a challenge (nonce) to the *Application* (1). The *Application* relays this challenge along with the *QE* 's identity to the *AE* (2). The *AE* generates a manifest containing the challenge answer and an ephemeral public key. This key is used to facilitate communication between the *Challenger* and the *AE* (3). The *AE* then invokes `EREPORT` to generate a REPORT, containing the hash of the manifest (4) and sends it to the *QE* for signing (5). The *QE* calls `EGETKEY` to obtain the Report Key used to verify the correctness of the REPORT (MAC tag recalculation process as in local attestation) [11] (6). Upon successful verification of the report, the *QE* creates a signed QUOTE from the REPORT using its EPID key (7). The *QE* then sends the QUOTE and the associated manifest to the *Challenger* for verification (8). The *Challenger* may validate the QUOTE 's signature using the EPID public key certificate or an independent attestation verification service. The *Challenger* verifies the manifest by checking its response with the initial challenge (9).

3.2. Intel TDX

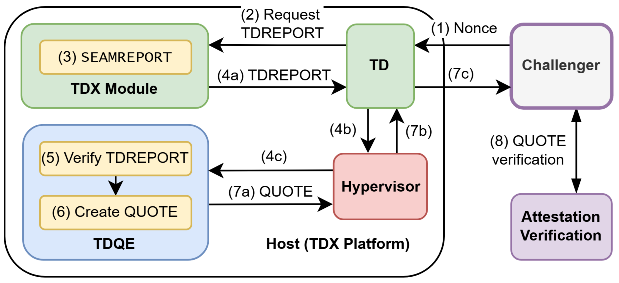


Figure 5: Intel TDX attestation flow [2], [11]

Intel TDX is a VM-based TEE solution responsible for creating and managing confidential VMs called Trusted Domains (TDs). This multi-component system introduces a new CPU mode that extends the functionality of VMX, called Secure-Arbitration Mode (SEAM) [2]. Intel TDX also uses special Intel SGX enclaves called the TD-Quoting Enclave (TDQE) to generate remote attestations for TDs. Behind all this mechanism is the new Intel TDX Module, an Intel-signed software module that interfaces the hypervisor and the TDs [11].

Confidentiality and Integrity. Memory confidentiality is primarily achieved through Intel's Multi-Key Total Memory Encryption (MKTME) technology, which offers encryption at cache line granularity using TD-specific keys [2]. Additionally, in the case of unauthorized attempts to access the cache lines, a fixed bit pattern will be returned to prevent cyphertext analysis. TDX also offers an option for cryptographic integrity protection, which will prematurely terminate a TD in case of any write attempts on secure memory [11]. Furthermore, TDX uses two extended page tables (EPTs) to ensure address-translation and memory-layout integrity, as well as allow TDs to communicate with untrusted entities while maintaining isolating TD's private memory [2], [11].

Measurements. The TDX architecture provides a TD with two types of measurement registers: TD Measurement Register (TDMR) for buildtime and Runtime Measurement Register (RTMR) for runtime of the TD [2]. During the creation of a TD, the TDX Module extends the TDMR with the measurements and metadata of the initially allocated pages for the TD. RTMRs, on the other hand, are used for code and data measurements at runtime [11].

Attestation. TDX supports an explicit remote attestation, enabled through an implicit local attestation [11]. For the following, we go through each TD attestation step as detailed in Figure 5, which is adapted from the TDX white paper [2] and [11]. A remote *Challenger* first sends an attestation request to a TD by transmitting a nonce (1). Upon receiving this request, the TD requests a local report called TDREPORT from the *TDX Module* (2). The *TDX Module* then invokes the `SEAMREPORT` operation to request the CPU to generate an HMAC-protected report containing TD measurements, TD attributes, TD identities, and the TCB SVNs (3). The generated TDREPORT is then handed over to the *TDQE* for further processing (4). The *TDQE* uses the `EVERIFYREPORT2` instruction to check whether the header information, TCB SVNs, and the computed MAC match their expected values (5). If the verification is successful, the *TDQE* replaces the MAC in the TDREPORT with a digital signature generated by an asymmetric-attestation key to form what is known as a QUOTE (6). The generated QUOTE is then sent back to the TD and is relayed back to the remote *Challenger* (7). On receiving the QUOTE, the *Challenger* verifies the signature of the QUOTE, as well as the MRTD and RTMRs of the TD (8). After successful verification, the *Challenger* can trust the *TD* for further communication and computation.

3.3. AMD SEV-SNP

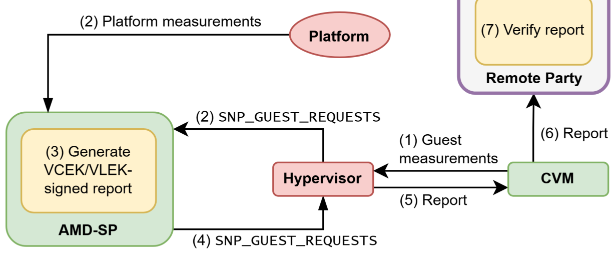


Figure 6: AMD SEV-SNP attestation flow [3], [12]

AMD SEV-SNP is AMD's third generation VM-based TEE solution, which builds upon the previous SEV technologies, specifically [13]:

1) **SEV**: Assigns each VM with a unique encryption key to protect their in-use data, providing guest memory isolation and confidentiality.
2) **SEV-Encrypted State (ES)**: Encrypts the VM register state and obfuscates data being used by the VM from the hypervisor, preventing information leakage to untrusted components.

AMD SEV-SNP introduces new hardware-based security protection with strong memory integrity guarantees through the AMD Secure Processor (AMD-SP) [3].

Confidentiality and Integrity. Similar to TDX, confidentiality is primarily achieved through memory encryption. Each CVM is initialized with a unique ephemeral encryption key, which ensures that specific data is only accessible to the associated SEV-SNP guest [13]. In addition to confidentiality, SEV-SNP architecture enforces data integrity by ensuring that a CVM must be able to read the value it last wrote if it can read a private encrypted page [3]. This guarantee is realized through a structure called the Reverse Map Table (RMP) and an updated nested page table walk, which enforces proper access control to memory pages in the system.

Measurements. The measurements included in the attestation report can be broken down into two parts: platform and guest. Platform measurements ensure that the platform running the SEV-SNP guests is running the latest firmware and microcode by collecting information regarding the TCB SVN [13], SVN threshold, unique chip ID, and specific platform properties [14]. On the other hand, the guest measurements collect various information about a specific guest during its initialization lifecycle [12]. Measurements of the pages and metadata associated with the guest are stored inside a launch digest, which is then compared against the expected guest measurements [14].

Attestation. SEV-SNP only supports remote attestation. In contrast to TDX's multi-module approach, *CVMs* only need to communicate with the *AMD-SP* during the attestation process, as shown in Figure 6. At the guest creation process, a set of private communication keys is created by the *AMD-SP* to facilitate secure direct communication between the *CVM* and the *AMD-SP* [13]. These guests can request an attestation report to the *AMD-SP* at any time through the *hypervisor* by first constructing a `MSG_REPORT_REQ` message, which includes the guest-specific measurements [12] (1). The hypervisor then invokes `SNP_GUEST_REQUESTS`, which wraps the message and sends it to the *AMD-SP*. It is important to note that the *hypervisor* cannot access (read or write) the messages without detection [12] (2). *AMD-SP* then generates the attestation report containing the measurements, optional public keys for communication, and an arbitrary report data field [13]. This report is signed with either the Versioned Chip Endorsement Key (VCEK) or the Versioned Loaded Endorsement Key (VLEK). While both keys are provisioned by the AMD Key Distribution Service (KDS) and the current TCB SVN, VCEK uses chip-unique seed, whereas VLEK uses CSP-unique seed maintained by the KDS [12] (3). The signed report is returned to the *CVM* embedded within the `MSG_REPORT_RSP` message, which is again sent via the `SNP_GUEST_REQUESTS` command [12]

(4,5). This report can be sent to a *remote party* looking to establish a secure communication channel for confidential computing [3] (6). Before trust can be established, the *remote party* verifies the software component of the TCB by checking the report content. Additionally, verifying the VCEK/VLEK-signed report proves the platform's authenticity, thus verifying the hardware component of the TCB (7).

4. Conclusion and Future Work

In this paper, we abstracted intricate concepts revolving around modern TEEs by giving a generic overview of the different TEE models and attestation flows. To observe how these concepts fit into state-of-the-art TEEs, we explored Intel SGX, Intel TDX, and AMD SEV-SNP. Specifically, we investigated how these TEE solutions guarantee confidentiality and integrity, what measurements are being attested, and how their attestations are carried out.

Our findings highlight the complex attestation capabilities of different TEEs. While various studies have conducted benchmarks measuring the impact TEEs have on the performance of a system, there is a lack of literature exploring the performance overhead incurred by generating attestation reports. Moving forward, conducting more practical research to explore such technicalities is beneficial as it also plays a significant role in gauging the scalability of using TEEs.

References

[1] Confidential Computing Consortium, "Confidential Computing: Hardware-Based Trusted Execution for Applications and Data," Nov. 2022, (Accessed: Jun. 16, 2024). [Online]. Available: https://confidentialcomputing.io

[2] Intel, "Intel Trust Domain Extensions (Intel TDX)," *White Paper*, Feb. 2023, (Accessed: Jun. 16, 2024). [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/documentation.html

[3] AMD, "Strengthening VM isolation with integrity protection and more," *AMD White Paper*, Jan. 2020, (Accessed: Apr. 06, 2024). [Online]. Available: https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf

[4] J. Ménétrey, C. Göttel, M. Pasin, P. Felber, and V. Schiavoni, "An Exploratory Study of Attestation Mechanisms for Trusted Execution Environments," Mar. 2022, (Accessed: Jun. 16, 2024). [Online]. Available: https://doi.org/10.48550/arXiv.2204.06790

[5] T. Geppert, S. Deml, D. Sturzenegger, and N. Ebert, "Trusted Execution Environments: Applications and Organizational Challenges," *Frontiers in Computer Science*, vol. 4, 2022, (Accessed: Jun. 16, 2024). [Online]. Available: https://www.frontiersin.org/articles/10.3389/fcomp.2022.930741

[6] Intel, "Intel Software Guard Extensions (Intel SGX) Developer Guide," *Developer Guide*, Mar. 2020, (Accessed: Jun. 16, 2024). [Online]. Available: https://www.intel.com/content/www/us/en/content-details/671581/intel-sgx-developer-guide-for-windows.html

[7] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote ATtestation procedureS (RATS) Architecture," RFC 9334, Jan. 2023, (Accessed: Jun. 16, 2024). [Online]. Available: https://www.rfc-editor.org/info/rfc9334

[8] M. Li, Y. Yang, G. Chen, M. Yan, and Y. Zhang, "SoK: Understanding Design Choices and Pitfalls of Trusted Execution Environments," in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1600–1616. [Online]. Available: https://doi.org/10.1145/3634737.3644993

[9] V. Costan and S. Devadas, "Intel SGX Explained," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 86, 2016, (Accessed: Jun. 16, 2024). [Online]. Available: https://api.semanticscholar.org/CorpusID:28642809

[10] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for cpu based attestation and sealing," 2013, (Accessed: Jun. 16, 2024). [Online]. Available: https://api.semanticscholar.org/CorpusID:14218854

[11] P.-C. Cheng, W. Ozga, E. Valdez, S. Ahmed, Z. Gu, H. Jamjoom, H. Franke, and J. Bottomley, "Intel TDX Demystified: A Top-Down Approach," 2023, (Accessed: Jun. 16, 2024). [Online]. Available: https://doi.org/10.48550/arXiv.2303.15540

[12] AMD. (2023, Sep.) SEV Secure Nested Paging Firmware ABI Specification . (Accessed: Jun. 16, 2024). [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/specifications/56860.pdf

[13] D. Kaplan, "Hardware VM Isolation in the Cloud: Enabling confidential computing with AMD SEV-SNP technology," *Queue*, vol. 21, no. 4, p. 49–67, sep 2023, (Accessed: Jun. 16, 2024). [Online]. Available: https://doi.org/10.1145/3623392

[14] AMD. (2022, Sep.) AMD SEV-SNP Attestation: Establishing Trust in Guests. (Accessed: Jun. 16, 2024). [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/developer/lss-snp-attestation.pdf

Performance Impact of the IOMMU for DPDK

Marcel Gaupp, Sebastian Gallenmüller*, Stefan Lachnit*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: m.gaupp@tum.de, gallenmu@net.in.tum.de, lachnit@net.in.tum.de

***Abstract*—The DPDK framework is used in a range of industries where high-performance packet processing is required. One of the available DPDK drivers is based on the VFIO Linux API, which makes use of the system's IOMMU. The VFIO based drivers are particularly useful for virtual machines.**

Previous research suggested that the IOMMU can have significant performance impacts on I/O operations. This paper measures the performance differences between a VFIO driver and an UIO driver, which does not use the IOMMU. We measure throughput and latency of NICs in a reproducible testing setup.

The results show that the performance difference is not significant in the average case and that a few packets show higher latency. This is explained by the caching behavior of the TLB.

***Index Terms*—computer networks, system buses, measurement, high-speed networks**

1. Introduction

The Data Plane Development Kit (DPDK) enables the development of various high-performance network applications. It is used in data centers, the network edge and infrastructure systems where data processing is performance critical.

Virtual machines (VMs) are often used to improve scalability and security. The secure and efficient use of network interface controllers (NICs) within VMs require special consideration. The use of the VFIO Linux API enables the secure assignment of devices to virtual machines.

In this paper, we compare the performance of two DPDK drivers: An older UIO based driver and the newer VFIO based driver. The VFIO driver mainly differs from the UIO driver in its use of the I/O Memory Management Unit (IOMMU). We show how the IOMMU affects the throughput and latency performance of NICs when used with DPDK.

This paper is structured as follows: Chapter 2 presents related work this paper is based on and differentiates our work from theirs. Chapter 3 introduces background information on PCIe, DPDK, and the IOMMU required to understand the analysis. Chapter 4 analyses differences of the different drivers and how these differences might affect performance. Chapter 5 describes the measurement setup and methodology. Chapter 6 presents and interprets our results and explains how they could have come to be. Chapter 7 concludes with a summary of our findings and their implications.

2. Related work

This paper is based on the findings of Neugebauer et al. [1] which show a significant drop in bandwidth if an IOMMU is used with low packet sizes. Their experiments differ from ours in that they allocated memory buffers sequentially within varying ranges in host memory. In our experiments, we relied on MoonGen's memory allocation strategy, which is not sequential.

Furthermore, Neugebauer et al. did not apply their research on DPDK, but on the use of PCIe in general. That is why they decided not to use the hugepages feature of Linux which increases the page size. Hugepage support is required by DPDK, which is why we did enable hugepages.

3. Background

Several key components contribute to the resulting performance analysis. This section provides an overview of the foundational knowledge about PCIe, DPDK and the IOMMU.

3.1. PCIe

The Peripheral Component Interconnect Express (PCIe) is a communication and interconnect standard interface [2]. It is the de-facto interface for connecting peripheral I/O devices, like NICs, to x86 based computers [1].

It uses a packeted communication protocol to send data between devices. Direct Memory Accesses (DMAs) are data transfers between the device and the host memory. In PCIe, they are implemented by Memory Read and Memory Write packets.

3.2. DPDK

DPDK is a set of libraries which provide a framework for creating network applications [3]. One use case is the creation of poll mode drivers (PMDs). PMDs can achieve higher performance in bandwidth and latency than their interrupt based counterparts, because they disable interrupts and poll write-back descriptors in host memory, thus leaving more bandwidth on the PCI bus for packet data [1].

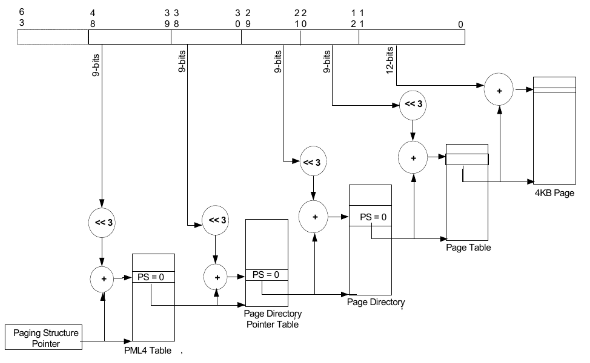


Figure 1: Page Table Walk, taken from [6]

3.3. IOMMU

Newer systems include an IOMMU. If enabled, every DMA passes through the IOMMU. The IOMMU interprets the memory address as an I/O Virtual Address (IOVA) and translates it into a physical address [4]. This is similar to how an MMU translates virtual addresses of a process on the CPU into physical addresses.

Without an IOMMU, giving a virtual machine access to a DMA capable device would give the VM access to all of the host memory, because the DMAs can access all physical addresses [5]. This poses a threat to the security and robustness of the system as memory locations not belonging to the VM can be read and overwritten. Thus hypervisors have to emulate the device with a managed memory space. The hypervisor intercepts the real device's DMA and copies the data to the VM's memory space. This indirection has a detrimental effect on performance.

The IOMMU isolates the IOVA space from the physical address space, therefore restricting the devices to the configured memory pages. Like with MMUs, the mapping is configured in page tables. Multiple memory locations need to be accessed, as shown in Figure 1 to find the page of an IOVA [6]. The results of this page table walk are cached in the Translation Lookaside Buffer (TLB), which is where subsequent address translations of recently looked up pages are found in.

4. Analysis

The Linux kernel currently provides 2 interfaces for accessing IO memory from userspace: Userspace I/O (UIO) and Virtual Function I/O (VFIO).

Both types of drivers provide an interface which allows to map the device's memory ranges to the address space of an userspace process.

UIO requires kernel code, which initializes the device, defines device memory ranges to be mapped and potentially registers an interrupt handler [7]. Once its driver is bound to the device, UIO provides a device file located at `/dev/uioX`. Its file descriptor can be used with a call to `mmap()` to map the defined memory ranges to userspace.

VFIO extends this by allowing the creation of Virtual Functions and by allowing configuration of the IOMMU [8]. Virtual Functions are virtual copies of the device, which can be assigned to multiple virtual machines. This enables the sharing of the physical NIC in multiple VMs with almost no overhead.

TABLE 1: Node Configuration

Component	Description
CPU	Intel(R) Xeon(R) CPU D-1518 @ 2.20GHz
Microarchitecture	Broadwell
Memory	32 GiB
OS	Debian 12
Kernel	Linux 6.1.0-17
NIC	Intel X552 10 GbE SFP+

The configuration granularity of VFIO is that of an IOMMU group [9]. An IOMMU group is a group of devices which can be isolated from the rest of the system. How many and which devices are in such a group depends on the system topology. Once all devices are bound to the VFIO driver, the device file `/dev/vfio/`*`group`* can be used with calls to `ioctl()` and appropriate arguments to access device memory and to configure the IOMMU.

The IOMMU enables virtual addressing for DMAs. The virtual address of each access is translated by the IOMMU to the corresponding physical address. The translation is cached inside the TLB. On a TLB-miss, the page table, which resides in host memory, has to be walked. Because this page walk needs to access multiple memory locations, we theorize that page misses could incur significant performance impacts.

Since the UIO drivers do not use the IOMMU while the VFIO drivers do, one would expect to measure differences in throughput and latency.

5. Implementation

We conducted our experiments on a three node setup depicted in Figure 2. All nodes are identically configured as shown in Table 1. The optical splitters allow the node named `bitcoingold` to passively listen into the communication between `bitcoin` and `bitcoincash`.

For deploying and running our scripts on these hosts, we used the Plain Orchestrating Service (*pos*) [10]. It is a framework, which allows full automation of orchestration, measurement and evaluation with the goal to make the experiments easily reproducible. Our experimentation scripts[1] can be executed on a *pos* testbed controller connected to the three nodes configured as described before.

For packet generation and measurement we used MoonGen [11]. MoonGen is a Lua wrapper for DPDK and will, as such, use the DPDK driver bound to the NIC. We ran the MoonGen tasks with 4 threads where applicable, so that no operation would be CPU bound.

The drivers we compare and their configuration are listed in Table 2. While other options like `uio_pci_generic` and `vfio enable_unsafe_noiommu_mode=1` exist [12], our chosen drivers have been available for the longest time. Therefore, they particularly are of interest for legacy applications.

5.1. Throughput

For the throughput measurement, we used the `bitcoin` and `bitcoincash` nodes and ignored the `bitcoingold`

1. https://gitlab.lrz.de/marcel.gaupp/dpdk-iommu-effects

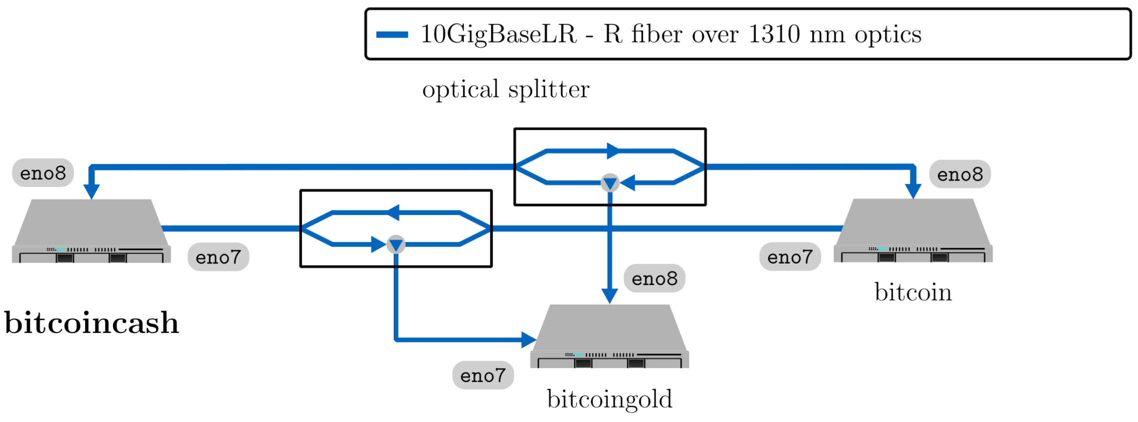


Figure 2: Topology

TABLE 2: Driver Configuration

Kernel Module	Boot Parameters
`igb_uio`	`intel_iommu=off iommu=pt`
`vfio-pci`	`intel_iommu=on`

host. One host was configured as a load generator and the other was configured as a Layer-2-Forwarder. The load generator generated packets as fast as possible and the forwarder returned them to the sender. Both nodes only used one interface port (eno7) to communicate.

The load generator sent packets from sizes 48 Bytes to 88 Bytes in increments of 4 Bytes. The packets were sent for 60 s for each packet size. `igb_uio` was always selected as the driver, assuming as a UIO driver it would be at least as fast as the other option.

This command was run on the load generator:

```
/root/moongen/build/MoonGen
  /root/code/pktgen.lua 0
  -s "$PACKET_SIZE" --threads 4
```

The `pktgen.lua` script sends UDP packets to some nonexistent destination on the given DPDK port id (0). `$PACKET_SIZE` was a *pos* loop variable, which differed for each experiment run.

The measurements were made on the forwarder, which was the device under test (DUT). We measured ingress and egress throughput for each driver configuration. This command was run on the DUT:

```
/root/moongen/build/MoonGen
  /root/code/l2-forward.lua 0 0
```

The `l2-forward.lua` script simply forwards every packet received on the first port to the second port (both 0). It also produces an average throughput statistic every second.

5.2. Latency

For the latency measurement, once again, the `bitcoin` and `bitcoincash` nodes were configured as load generator and forwarder, with the forwarder being the DUT, which is tested with both drivers. But this time the forwarded response was sent back on the other interface port:

```
/root/moongen/build/MoonGen
  /root/code/l2-forward.lua 1 0
```

The `bitcoingold` node was configured to timestamp the packets in both directions. The NIC supports hardware timestamping and achieves an accuracy of below 100 ns [11]. The time difference between the same packet being received by the DUT and the same packed being transmitted by the DUT was recorded as the latency. The packets were identified by a unique identifier in the UDP payload. To generate these packets, this was run on the load generator:

```
/root/moongen/build/MoonGen
  /root/code/traffic-gen.lua 1 0
  -t "$DURATION" -s $PACKET_SIZE -r $RATE
```

The `traffic-gen.lua` script generates UDP packets with an increasing 32-bit integer as the payload. Packets were sent on interface port 1, while port 0 received packets for statistics. The transmission rate was fixed at 1 Gbit/s and the duration was 60 s.

This time we varied the packet size starting at 64 Bytes, doubling until the MTU of 1500 Bytes was reached.

To measure the latency, the capturing host (`bitcoingold`) ran this command:

```
/root/moongen/build/MoonGen
  /root/code/sniffer.lua 1 0
  -t 300 --seq-offset 42
```

This script captures and timestamps the packets on interface ports 1 and 0 and records their timestamps and identifiers in the `latencies-pre.mscap` and `latencies-post.mscap` respectively. The runtime `-t 300` is set to 300 s, much longer than the packet generation time of 60 s. But the script is killed once

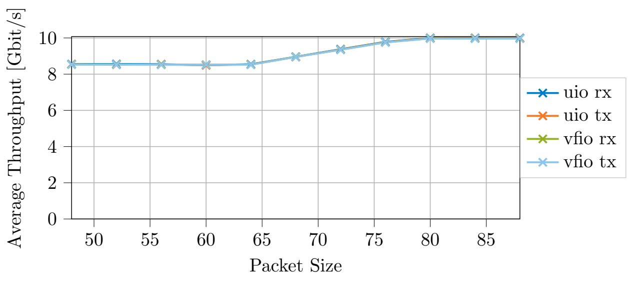


Figure 3: Throughput

the *pos* framework detects that the packet generator has stopped running. The `--seq-offset` option specifies the offset of the identifier from the start of the Ethernet frame. We calculated this with the Ethernet header being 14 Bytes, the IPv4 header being 20 Bytes and the UDP header being 8 Bytes: $14 + 20 + 8 = 42$

Afterwards, we processed these records to generate histogram data:

```
/root/moongen/build/MoonGen
  /root/moongen/examples/
      moonsniff/post-processing.lua
  -i latencies-pre.mscap -s latencies-post.mscap
```

This generates the `hist.csv`, which pairs latencies in nanoseconds with their occurrence count.

6. Evaluation

The results show that the average performance impact is not as significant as we expected.

6.1. Throughput

Figure 3 shows the average ingress and egress throughput of both driver variants. However, all variants are similar enough such that no difference is visible.

For sizes 48 B to 64 B, the throughput is constant but slightly lower than the capacity of the NIC. It increases from 64 B to 80 B where it reaches the full 10 Gbit/s.

The lower throughput for packet sizes below 80 B can be explained by the packetized structure of the PCIe protocol [1]. For smaller network packet sizes the size of the PCIe level packets are dominated by the PCIe packet headers. Therefore, less data is transmitted while the PCIe bus bandwidth is used up.

For packet sizes below 64 B our theory is that the hardware pads the size up to 64 B because this is the minimum Ethernet frame size.

6.2. Latency

The latency's percentiles below 99 show very little variation as show in Figure 4. We notice a slight linear increase of the latency for increasing packet sizes.

If we look at the higher percentiles shown in Figure 5, significant differences between the drivers become visible. This means that something has an effect on a few packets and this effect differs for the drivers.

We theorize that the TLB does not get completely filled up and that only the first packet of the corresponding

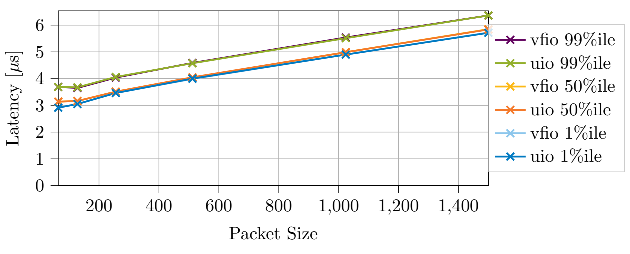


Figure 4: Latency middle percentiles

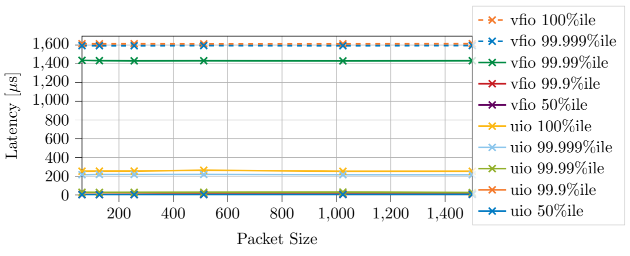


Figure 5: Latency upper percentiles

page causes a page walk to happen. This explains how only a few packets (the packets causing a page walk) do have a measurable difference.

This result differs from those of Neugebauer et al. [1] because they allocated DMA buffers linearly, while we used MoonGen's allocations which reuse buffers. A linear allocation strategy covers a wider range of memory addresses and uses more memory pages. Because fewer memory pages are used this results in the TLB not filling up.

Furthermore, we were forced to use hugepages, which increases the page size from 4 KiB to 2 MiB. Not only does this reduce the page count per memory used, it also increases the size of the page tables. This significantly reduces how often a full page walk has to be executed.

7. Conclusion

This paper answers whether the IOMMU does have a performance impact if used with DPDK. We show that the `vfio-pci` driver for DPDK and its use of the IOMMU do not have a significant performance impact for regular memory access patterns. Only the first few accesses show increased access time. Further accesses of cached pages show no measurable delay if accessed via an IOMMU.

VFIO drivers do not come at a performance cost. That is why we believe that the use of them will make systems more reliable and virtual machines more efficient securely without compromise.

References

[1] R. Neugebauer, G. Antichi, J. F. Zazo, Y. Audzevich, S. López-Buedo, and A. W. Moore, "Understanding pcie performance for end host networking," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 327–341. [Online]. Available: https://doi.org/10.1145/3230543.3230560

[2] H. Zhu, *Data Plane Development Kit (DPDK): A Software Optimization Guide to the User Space-Based Network Applications*. CRC Press, 2020.

[3] "About dpdk," 2024, accessed 15. June 2024. [Online]. Available: https://www.dpdk.org/about/

[4] O. Peleg, A. Morrison, B. Serebrin, and D. Tsafrir, "Utilizing the IOMMU scalably," in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. Santa Clara, CA: USENIX Association, Jul. 2015, pp. 549–562. [Online]. Available: https://www.usenix.org/conference/atc15/technical-session/presentation/peleg

[5] R. Jithin and P. Chandran, "Virtual machine isolation," in *Recent Trends in Computer Networks and Distributed Systems Security*, G. Martínez Pérez, S. M. Thampi, R. Ko, and L. Shu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 91–102.

[6] *Intel Virtualization Technology for Directed I/O Architecture Specification*, 4th ed., Intel Corporation, June 2022.

[7] J. Corbet, "Uio: user-space drivers," *LWN.net*, 2007, accessed 16. June 2024. [Online]. Available: https://lwn.net/Articles/232575/

[8] ——, "Safe device assignment with vfio," *LWN.net*, 2012, accessed 12. June 2024. [Online]. Available: https://lwn.net/Articles/474088/

[9] *The Linux Kernel documentation / VFIO - "Virtual Function I/O"*, 2024, accessed 16. June 2024. [Online]. Available: https://docs.kernel.org/driver-api/vfio.html

[10] S. Gallenmüller, D. Scholz, H. Stubbe, and G. Carle, "The pos framework: a methodology and toolchain for reproducible network experiments," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 259–266. [Online]. Available: https://doi.org/10.1145/3485983.3494841

[11] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," in *Internet Measurement Conference 2015 (IMC'15)*, Tokyo, Japan, Oct. 2015.

[12] *DPDK documentation / Linux Drivers*, 2024, accessed 12. June 2024. [Online]. Available: https://doc.dpdk.org/guides-24.03/linux_gsg/linux_drivers.html

How Precise Is the Clock in the Cloud?

Gee-Il Han, Filip Rezabek, Leander Seidlitz*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: geeil.han@tum.de, frezabek@net.in.tum.de, seidlitz@net.in.tum.de

Abstract—**A precise clock is a necessary element in a system and most devices have their own internal clock. Over time these clocks experience clock drift due to influences such as hardware variations or operating conditions. As a result, the time of internal clocks varies from the time they are supposed to be. Differences between internal clocks of cloud servers and devices can result in data inconsistencies and authentication errors, ultimately impacting the reliability and security of distributed systems.**

To overcome these deviations protocols are used, for instance, network time protocol (NTP), precision time protocol (PTP), or simple precision time protocol (SPTP). Some providers already implemented these protocols and can be used for clock synchronization, e.g., Google Public NTP, Amazon Time Sync, or Alibaba Cloud Time Synchronization Service. They are used to achieve synchronized clocks between the server and client.

Index Terms—**NTP, network time protocol, PTP, precision time protocol, SPTP, Simple precision time protocol, synchronized clock, time sync service**

1. Introduction

Before devices within a system try to synchronize their clocks, they communicate with each other through data packets, containing timestamps of the sender's device and more to ensure its chronological order for time-sensitive data. The time values are read from an individual clock to each device.

However, a problem occurs when the clocks in a system are not synchronized. These deviations are called clock drifts and can occur for the following reasons:

Hardware variations between the components in each system result in different magnitudes of deviations leading to varying aberrations for every system [1].

Operating conditions in the context of clock consistency describe multiple influences on the clock through environmental factors. One of them is the electronic component aging. It decays over time, altering the components' electrical properties and causing changes in their frequency characteristics. This effect causes deviations in the clock. Another operating condition affecting the clock is temperature fluctuation which causes the materials in the clock to expand or contract, therefore affecting the timekeeping mechanism. This physical phenomenon leads to shifts in the oscillation frequency [1].

The desynchronization of clocks, which is an outcome of, e.g., clock drift and network and processing latencies across processes, is followed by anomalous behavior in the system [2]. To illustrate, two computers A and B issue requests a and b with the condition that a is sent earlier than b but the timestamp of a is later than bs. This situation can cause request b to be ordered before request a even though a was sent earlier than b, which is known as anomalous behavior [2].

The problem is solved by using protocols such as the network time protocol (NTP [3]), precision time protocol (PTP [4]), and its abbreviation, the simple precision time protocol (SPTP [5]). These protocols are applied to synchronize the clocks of the devices connected to a system with its grandmaster clock. PTP and SPTP have optional-PTP-enabled hardware that supports the synchronization process which leads to a more accurate offset time in order of nanoseconds. At the same time, the NTP is only a software-deployed protocol. The precision of NTPs is in order of milliseconds (ms) or microseconds (μs) [6].

Instead of implementing a protocol, it is more efficient to use clock synchronization options already available through cloud providers since this method saves time and effort. The services researched in this paper are Google Public NTP from Google Cloud Platform (GCP [7]), Amazon Time Sync provided by Amazon Web Services (AWS [8]), and Alibaba Clouds Alibaba Cloud Time Synchronization Service [9].

This paper presents background information about NTP, PTP and SPTP and their differences. Afterwards, available clock synchronization options are presented and details are provided.

2. Background Information

In order to understand the differences between the protocols, background information about NTP, PTP, and SPTP is provided.

NTP works by continuously exchanging time information between server and client in a hierarchical order to ensure synchronization across the clocks in all nodes with an acceptable deviation of milliseconds. The grandmaster clock is called Stratum 0 and after it has been synchronized with another device, the other device is considered a Stratum 1. Every clock that is synchronized with Stratum 1 becomes Stratum 2, every clock synchronized with Stratum 2 becomes Stratum 3, and so on. [3]. A simplified version of a single cycle is presented in Figure 1.

A simple cycle of the NTP can be described as follows: The client first sends a request to the server with its current timestamp attached. The server receives the message and and sends out a response containing three

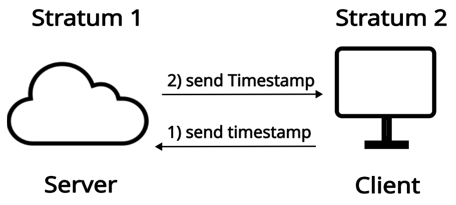


Figure 1: The Network Time Protocol [3]

timestamps: the arrival time of the message, the time when the response was sent, and the server's current time. After the client receives the response, the offset and the delay of the round-trip are calculated. Using the phase-lock loop (PLL) the clock is gradually adjusted to minimize abrupt changes [3]. This process is executed periodically in order to maintain synchronization.

PTP works by synchronizing clocks in a network to a master clock, reducing deviation to nanoseconds. This section references PTP to security extension of PTP, the PTPv2 Standard IEEE-1588-2019 [10]. During the first step of this protocol, PTP chooses a grandmaster clock using the Best Master Clock Algorithm (BMCA) by determining each clock's quality and accuracy. The grandmaster clock has the highest hierarchical order. Once it is established, the rest of the hierarchy is formed, which builds master and slave relationships. The latter has to synchronize with the master clock. In order to adjust the clock, messages have to be exchanged between master and slave. The master sends a Sync request to the slave and follows it up with a follow-up request. This message contains the timestamp the Sync message was sent. It assures the accuracy of the Sync timestamp since it could have diverged due to processing delays. The slave sends a delay request, which is needed to measure the time from slave to master after the follow-up message arrives. Afterward, the master responds to it with a delay response containing the timestamp when the delay request arrives.

Now, all important timestamps are available, and the offset and delay can be calculated, but two different kinds of delay calculation modes have to be considered. First is the End-to-End (E2E [11]) delay, where the round-trip time between a master and a slave is measured. The other method is measuring the Peer-to-Peer (P2P [12]) delay. This method is used in networks with redundant paths. It measures the link delay between each pair of devices and calculates the delay and offset [4]. After succeeding with the measurements comes the synchronization of the clocks. The delay and offset are calculated and the slave clock is adjusted. The Figure 2 shows a simplified cycle of a PTPv2.1.

It is also important to note, that this protocol differentiates clocks. First, there are transparent clocks which are clocks with a switch. The delay of the switch has to be calculated depending on the state of the switch but the transparent clock does not directly synchronize with other clocks. Instead, it works as a forwarding device with a delay that is added to the timestamp calculation. Second, boundary clocks receive the time in one port and distribute

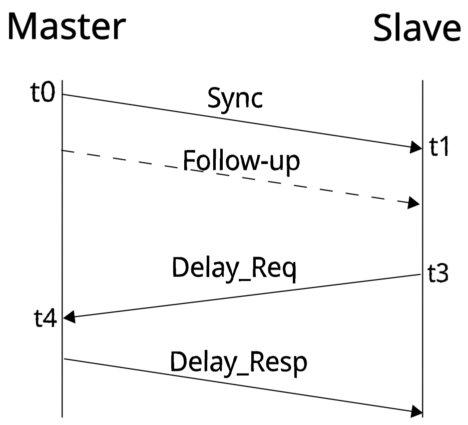


Figure 2: The Precision Time Protocol PTPv2.1 Standard IEEE-1588-2019 [13]

it through another port. They act as a master clock for devices with lower hierarchy therefore reducing the direct connection to the grandmaster. Last but not least are the ordinary clocks. They only have a single PTP Port and act either as a master clock or slave clock [4].

The SPTP is a simplified and advanced version of PTP that maintains compatibility with current equipment that is capable of supporting PTP. It also reduces the number of exchanges between master and slave nodes. As a result, more efficient network communication is possible [5].

3. Assessment of Time Protocols

Even though the concepts of NTP, PTP, and SPTP are similar, they have distinct variations in their performance, accuracy and component utilization. This section compares the previously mentioned protocols and sets their differences side by side.

3.1. NTP vs PTP

Although the network time protocol and the precision time protocol fulfill the same roles, both have significant differences in their implementation. While NTP uses Strata to determine the grandmaster clock and the hierarchical orders of the connected devices, the PTP determines its grandmaster by running the BMCA. NTP's Strata are more easily implemented but PTP's BMCA, while being more complex, finds the most optimal grandmaster clock to send Announce messages to. Figure 3 depicts a network with multiple grandmaster clocks but due to the BMCA only one of them sends out Sync messages while the other is dormant.

Another difference is the introduction of transparent and boundary clocks in the precision time protocol. This differentiation does not exist in the network time protocol resulting in more accuracy for the PTP. This can be explained since NTP is an entirely software-implemented protocol, which is sufficient should ms-level deviations be acceptable. NTP does not require any hardware support and can therefore be used on most networked devices [3] [6]. As a result, the NTP is commonly used for general-purpose time synchronization and is also the most common type of time synchronization protocol available to the

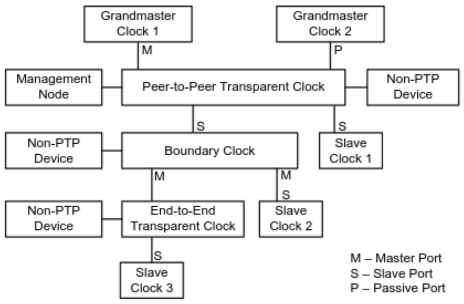


Figure 3: The Precision Time Protocol [4]

public. The network time protocol is also easier to implement since it is less complex. Unlike NTP, two approaches for the implementation of PTP have been accepted [6]. One approach is a precision time protocol with software-implemented time stamping resulting in clock deviations in microseconds μs. The other more accurate approach is hardware-supported PTP. This allows hardware time stamping, resulting in precision within nanoseconds [6]. That is the reason why PTP is the most commonly used where the accurate synchronization of time is crucial, such as in the financial area, telecommunication, and industrial automation. In addition, PTP slaves and masters exchange more messages with each other in order to maintain higher accuracy than NTP while the message size stays similar to NTPs. PTP also has a higher frequency of message exchanges compared to NTP. This can be explained by to the goal of making PTP as accurate as possible. While NTP sends messages in an interval of seconds, PTP exchanges data every few microseconds [3] [6]. As a result of the higher frequency, PTP has a higher processing overhead in addition to a higher network load.

Let t_0, t_1, t_2, t_3 be timestamps for a NTP cycle with t_0 as the sending time of the request, t_1 its arrival time, t_2 the sending time of the response, and t_3 its arrival time. In addition, let $a = t_1 - t_0$ and $b = t_2 - t_3$. The roundtrip delay δ and the clock offset θ of B relative to A at Time T is calculated as followed [3] :

$$\delta = a - b \text{ and } \theta = \frac{a+b}{2}$$

PTP calculates its offset δ and delays θ differently in order to achieve higher accuracy for the clock synchronization. Let t_0, t_1, t_2, t_3 be timestamps for a PTP cycle with t_0 as the sending time of the sync request, t_1 its arrival time, t_2 the sending time of the delay request, and t_3 its arrival time. There are two different kinds of delay in the PTP; therefore, there are two mechanisms to calculate the delay. One is the request-response mechanism for the End-to-End delay and the peer-delay mechanism for the Peer-to-Peer delay. During the following calculations, only the delay through the request-response mechanism without any switches between master and slave is considered. Additionally, CF_M is the sum of switch delays from the master to the slave and CF_S the sum of switch delays from slave to master [14]. The offset and the delay of the slave clock from the master clock are calculated as presented [4]:

$$\delta = (t_1 - t_0) - \theta \tag{1}$$

and

$$\theta = \frac{(t_3 - t_2) + (t_1 - t_0) - CF_M - CF_S}{2} \tag{2}$$

It is apparent that the calculations for the delay and offset are different from each other with PTPs calculation being more accurate by considering different types of delays and using more efficient algorithms.

3.2. PTP vs SPTP

The SPTP is a simplification of PTP and therefore shares many similarities. The typical complete exchange for IEEE 1588-2019 two-step PTPv2 unicast UDP flow is depicted in Figure 4

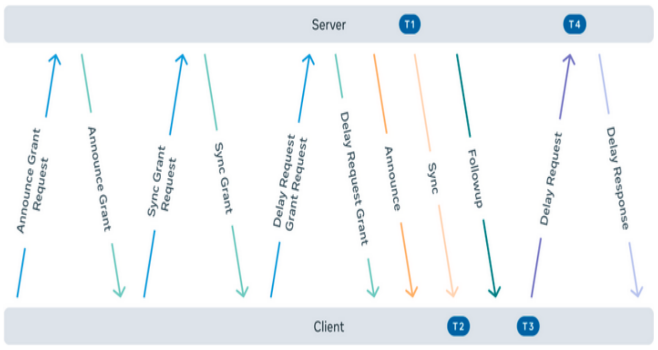


Figure 4: Two-step PTP exchange [5]

This sequence repeats itself and can be extended or reduced depending on the negotiation results between master and slave. Designing the PTP this way allows it to be flexible. The trade-off is that the slave and master have to keep their state in memory, resulting in excessive usage of resources, such as CPU and memory as well as increased code complexity.

SPTP does not need states to be preserved and reduces the number of exchanges needed while still being compatible with a two-step PTPv2 exchange, as seen in Figure 5 [5].

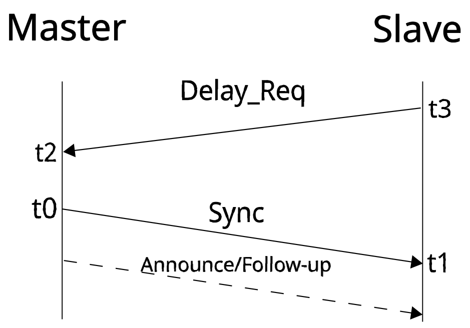


Figure 5: Simple Precision Time Protocol [5]

Unlike a PTP exchange, the SPTP starts with a delay request from the client to the server initializing the variables t_2 and t_3 as well as the correction field CF_S. The Sync message then gets dispatched from the server to the client containing t_0 and CF_M. Afterward, an Announce/Follow-up package is sent with t_2 and other information such as clock class, accuracy, and so on. With the values of this process, the offset and delay are calculated using equation 1 and 2. As a result, the 11 exchanges in Figure 4 are reduced to 3 in Figure 5 while maintaining compatibility with PTPv2.

In general, SPTP is simpler and easier to deploy while maintaining microsecond-level accuracy, while PTP is more precise but also more complex.

4. Different Clock Synchronization Deployments

Implementing the protocols is a complex matter since many different factors have to be considered, e.g. security concerns, precision and accuracy requirements, scalability, and more. Clock synchronization services are supplied by cloud providers, removing the need to implement a time protocol. This section analyzes Google Public NTP from GCP, Amazon Time Sync provided by AWS, and Alibaba Cloud Time Sync Service.

Google Public NTP is a NTP that uses a 24-hour linear smear from noon to noon UTC as a leap smear in order to maintain system stability during the insertion of leap seconds by gradually adjusting the extra second across the hours before and after each leap [15]. The servers are Stratum 1 which means that the Google servers are referred to as more accurate clocks, such as atomic clocks or GPS clocks. This NTP is also publicly available without the need for an account or cloud service usage with the server address being `time.google.com` or from `time1.google.com` to `time4.google.com` [7].

The Amazon Time Sync service is also a Stratum 1 referencing GPS and atomic clocks but unlike Google Public NTP it is only available to EC2 instances within AWS. This ensures that the service is already integrated with AWS infrastructure. In addition, Amazon Time Sync service uses Leap Smearing for the same reason as Google Public NTP. Even though this service is only available for EC2 instances it is still accessible through the instance metadata at `169.254.169.123` for IPv4 address endpoints and `fd00:ec2::123` for IPv6 [8].

Last but not least, the Alibaba Cloud Time Sync service is just like the other two services, a Stratum 1 referencing atomic and GPS clocks. The leap smear used is a 12-hour smear on either side of the leap second [16]. Alibaba Cloud Time Sync is designed to be used within Alibaba Cloud Environments but can be used publicly. In order to access the server, the address `ntp.aliyun.com` can be used [9].

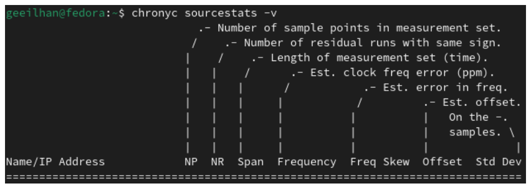
Another service to be noted is Meta's SPTP [5]. It can be used in a PTP environment due to its compatibility with the precision time protocol. SPTP may need to be used with PTP TLVs (type-length-value [5]) should the system, where the PTP/SPTP is used, require subscriptions and authentications [5].

4.1. Testing AWSs and GCPs Clock Sync Services

The device used for testing has the specifications listed in Table 1. This Listing 6 shows a screenshot of synchronization with Amazon Time Sync services and Google Public NTP for a duration of approximately 2 hours. Here, the table for the output is generated, and the Columns are defined using `chronyc`, which is an interface program used to interact with `chronyd` daemon for monitoring and controlling purposes. The commands used to monitor the output is `chronyc sourcestats -v`.

Specification	Details
OS	Fedora
CPU	Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
RAM	SODIMM DDR4 Synchronous 2133 MHz
GPU	GeForce GTX 960M
Motherboard	ASUS GL552VW
Networking	Intel Wireless 7265

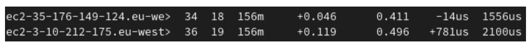
TABLE 1: Hardware Specifications of the Test Device



```
geeilhan@fedora:~$ chronyc sourcestats -v
                             .- Number of sample points in measurement set.
                            /    .- Number of residual runs with same sign.
                           |    /    .- Length of measurement set (time).
                           |   |    /      .- Est. clock freq error (ppm).
                           |   |   |      /           .- Est. error in freq.
                           |   |   |     |           /         .- Est. offset.
                           |   |   |     |          |          |   On the -.
                           |   |   |     |          |          |   samples. \
                           |   |   |     |          |          |             |
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
==============================================================================
```

Figure 6: Screenshot of `chronyc sourcestats -v`

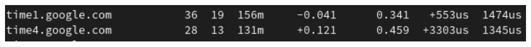
Following Listing 7 shows an offset of ~400 μs. This is within the appropriate time of a network time protocol.



```
ec2-35-176-149-124.eu-we>  34  18  156m     +0.046      0.411    -14us  1556us
ec2-3-10-212-175.eu-west>  36  19  156m     +0.119      0.496   +781us  2100us
```

Figure 7: Screenshot of `chronyc sourcestats -v` output for Amazon Time Sync
clock synchronization options

The next Listing 8 shows for `time1.google.com` an offset of 553 μs which is within the NTPs deviation. `time4.google.com` shows an offset of 3303 μs which is significantly bigger than `time1.google.com`s offset. This can be explained to synchronization issues and network latency.



```
time1.google.com           36  19  156m     -0.041      0.341   +553us  1474us
time4.google.com           28  13  131m     +0.121      0.459  +3303us  1345us
```

Figure 8: Screenshot of `chronyc sourcestats -v` output for Google Public NTP

This shows that many time sync services provided by AWS and GCP can be used as reliable NTPs. Depending on the environment the clock synchronization service applied should fit the instance it runs on if provided, e.g., Amazon Time Sync service should be used on an AWS instance, which reduces the workload of incorporating the service. If the instance does not provide a time sync service a good fallback service is the Google Public NTP.

5. Conclusion and Future Work

In conclusion, this study has demonstrated that the PTP is a more accurate protocol than a NTP, with the trade-off that the NTP is easier to implement than the PTP. Additionally, the time deviation of NTPs are calculated in microseconds, while the deviations of PTPs is calculated in nanoseconds. Due to its relatively easy implementation compared to PTP, NTP is mostly used for public time sync services.

SPTP is a simplification of PTP that reduces the number of exchanges needed during a full cycle. This leads to more efficient network communication therefore, improvements in resource utilization. The SPTP is also compatible with PTP, with an example of this being Meta implementing a SPTP compatible with almost any PTP environment.

A publicly available time sync service that can be used as a fallback service, for any instance, is the Google Public NTP. Otherwise, if an instance provides a time synchronization service, it should be used, such as the Amazon Time Sync service for AWS instances. Future studies should explore SPTP and its implementations since it is easier to implement the simple precision time protocol than the PTP while its offset is calculated in nanoseconds.

References

[1] F. Bizzarri and X. Wei, "Phase noise analysis of a mechanical autonomous impact oscillator with a mems resonator," in *2011 20th European Conference on Circuit Theory and Design (ECCTD)*. IEEE, 2011, pp. 729–732.

[2] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 179–196.

[3] D. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[4] S. T. Watt, S. Achanta, H. Abubakari, E. Sagen, Z. Korkmaz, and H. Ahmed, "Understanding and applying precision time protocol," in *2015 Saudi Arabia Smart Grid (SASG)*. IEEE, 2015, pp. 1–7.

[5] O. Obleukhov and A. Byagowi, "Simple precision time protocol at meta," https://engineering.fb.com/2024/02/07/production-engineering/simple-precision-time-protocol-sptp-meta/, February 2024, [Online; accessed 07-June-2024].

[6] Z. Idrees, J. Granados, Y. Sun, S. Latif, L. Gong, Z. Zou, and L. Zheng, "Ieee 1588 for clock synchronization in industrial iot and related applications: A review on contributing technologies, protocols and enhancement methodologies," *IEEE Access*, vol. 8, pp. 155 660–155 678, 2020.

[7] Google, "Configuring Clients," https://developers.google.com/time, 2024, [Online; accessed 09-June-2024].

[8] Amazon, "Set the time for your Linux instance," https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/set-time.html#configure-time-sync, 2024, [Online; accessed 09-June-2024].

[9] Alibaba, "Manage the time synchronization service," https://www.alibabacloud.com/help/en/ecs/user-guide/alibaba-cloud-ntp-server, 2024, [Online; accessed 09-June-2024].

[10] F. Rezabek, M. Helm, T. Leonhardt, and G. Carle, "PTP Security Measures and their Impact on Synchronization Accuracy," in *18th International Conference on Network and Service Management (CNSM 2022)*, Thessaloniki, Greece, November 2022.

[11] Z. Gao, Y. Hua, X. Jin, S. Liu, and L. Tang, "End-to-end delay testing and research on the internet," in *2023 3rd International Symposium on Artificial Intelligence and Intelligent Manufacturing (AIIM)*, 2023, pp. 13–18.

[12] B. Zhang and S. Wang, "An optimization model of load balancing in peer to peer (p2p) network," in *2011 International Conference on Computer Science and Service System (CSSS)*, 2011, pp. 2064–2067.

[13] "Ieee standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008)*, pp. 1–499, 2020.

[14] O. Obleukhov and A. Byagowi, "How precision time protocol is being deployed at meta," https://engineering.fb.com/2022/11/21/production-engineering/precision-time-protocol-at-meta/, November 2022, [Online; accessed 07-June-2024].

[15] Google, "Leap Smear," 2023, [Online; accessed 09-June-2024].

[16] S. Moss, "A second look," https://www.datacenterdynamics.com/en/analysis/a-second-look/, November 2022, [Online; accessed 09-June-2024].

Pacing User-Space Applications

Luis Kleinheinz, Marcel Kempf*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: luis.kleinheinz@tum.de, marcel.kempf@tum.de

Abstract—**Pacing improves network performance by evenly spacing packet transmissions, reducing delays and packet loss caused by bursty traffic. While Linux kernels support pacing, making it applicable for TCP, user-space protocols like QUIC fall short due to their lack of direct access to kernel-level mechanisms. This paper investigates using Time-Sensitive Networking (TSN) queueing disciplines (qdiscs) to provide effective pacing for user-space protocols. The goal is to bring the benefits of pacing to user-space protocols by discussing the technical details and evaluating which TSN qdiscs is best-suited for this task.**

Index Terms—**pacing, user-space, tsn, qdiscs**

1. Introduction

In today's networking environment, managing traffic flow is important to ensure efficient and reliable data transmission. Congestion control algorithms can help with that by dynamically regulating the rate at which data packages are sent over the network. By doing so, we can maximize our throughput and ensure a fair distribution of network resources, ultimately providing reliable communication by reducing package loss and retransmissions.

However, congestion control algorithms can produce irregular and bursty traffic, where large volumes of data packets are sent in quick succession. This sudden traffic can lead to queueing delays and increased packet loss, weakening the reliability and efficiency that congestion control aims to achieve in the first place [1]. To address this issue, packet pacing has proven to be a suitable solution. This technique smoothens the traffic by evenly spacing packets over time, and with that reducing bursts and their associated negative impacts on network performance.

Modern Linux kernels have integrated packet pacing within TCP [2], using kernel-level controls to ensure a smoother and more predictable traffic flow. This integration has shown massive improvements in managing network congestion and maintaining steady data transmission rates. However, not all network protocols operate within the kernel space. User-space protocols operate outside the kernel, presenting unique challenges for implementing effective pacing.

The Quick UDP Internet Connections protocol, or QUIC for short, is one such user-space protocol [3]. Like TCP, it is a transport layer protocol designed to make internet connections faster and more reliable. However, QUIC offers some benefits over the well-established protocol, like reduced latency or multiplexing. This is accomplished by combining the transport and the cryptographic layers into a single protocol, and supporting multiple lightweight streams that allow for efficient multiplexing. Yet, its user-space operation means it can not benefit from the kernel-level pacing mechanisms available to TCP. Therefore, achieving effective packet pacing for user-space protocols like QUIC is a problem that has to be solved.

One promising solution is to utilize Time-Sensitive Networking (TSN) queueing disciplines, or qdiscs. TSN qdiscs are advanced network scheduling mechanisms that manage packet transmission timing with high precision [4]. As TSN qdiscs can regulate the intervals at which packets are sent, they could be used to implement mechanisms to bring the same pacing benefits to user-space protocols as those used by kernel-space protocols. How the functionalities of the qdiscs can be used for this is explained later on in this paper.

This paper investigates the feasibility of using TSN qdiscs for pacing user-space protocols, focusing on QUIC. We will explore the mechanics of various TSN qdiscs, compare their effectiveness in providing packet pacing, and evaluate their applicability to user-space protocols. Furthermore, we aim to find a solution as to how TSN qdiscs can be used by user-space protocols, ultimately improving the performance and reliability of modern network communications.

2. Introduction to Pacing

Before investigating whether using TSN qdiscs for pacing user-spaced protocols is feasible, this section will give a rough overview over pacing itself. The aim is to understand the concept behind the mechanism.

2.1. Pacing at kernel-level

The concept of pacing was brought up as a solution to the limitations of traditional congestion control algorithms, which often resulted in packet loss and queueing delays due to their bursty nature [1]. Research investigated the benefits of spacing out packet transmissions to achieve a more stable and efficient data flow [5]. This led to the development and implementation of practical pacing mechanisms in mainstream networking protocols.

Pacing operates by controlling the intervals at which packets are transmitted, resulting in smoother traffic bursts and improving overall network performance. The process begins with packet queuing, where incoming packets are initially queued in the network stack or within the application layer, awaiting transmission. Pacing mechanisms then use rate-limiting algorithms to calculate the sending rate based on bandwidth, congestion signals, or application requirements. This mechanism brings many advantages, like the reduction of sudden bursts of traffic and, therefore, lower queueing delays and reduced latency. In general, pacing enables the network to work more stable and consistently, leading to better bandwidth use and higher overall throughput [5].

Pacing support for the Linux kernel was officially introduced with the addition of TCP Small Queues (TSQ) in the kernel version 3.6 in 2012 [6] and the Fair Queue (fq) pacing scheduler in version 3.12 in 2013 [7]. TSQ aims to reduce bufferbloat by limiting the amount of data that could be queued in the network stack. The fq scheduler further enhances the pacing capabilities by implementing fair queuing with per-flow pacing. With these integrations of pacing in the kernel, protocols like TCP could provide for more consistent and predictable network performance. Having these benefits is important for any protocol that operates on the network stack today; therefore, using them for user-spaced protocols like QUIC is desirable. The following section will explore why this can not be done easily and what challenges we are facing.

2.2. Pacing at user-space applications

As pacing offers great advantages and solves various problems of congestion control algorithms, pacing user-space applications would be beneficial to profit from all these aspects as well. For this, we have to adapt to the specific challenges posed by protocols such as QUIC, which operate independently of direct kernel control and cannot use the traditional way of pacing implemented in the Linux kernel. QUIC has gained popularity due to its significant improvements over TCP. While TCP operates over IP and relies on a separate TLS (Transport Layer Security) layer for encryption, QUIC combines both transport and security functionalities into a single protocol. This integration reduces the overhead of multiple handshakes when connections are established and with that enhances security by ensuring encryption is applied at the beginning of communication by default. Moreover, QUIC directly incorporates various additional features, making it a more suitable choice in many scenarios than the traditional TCP. One example is connection migration, which allows sessions to seamlessly switch between network interfaces or IP addresses without interrupting ongoing transmissions, making it especially useful in mobile environments. Another feature is built-in packet pacing in QUIC within its application, which provides a consistent data transmission rate. [3]

However, this pacing, of course, differs from our kernel-level pacing, as it only operates within its own protocol stack instead of directly influencing how packets are scheduled. These technical advancements make QUIC well-suited for modern applications, especially for mobile and multi-homed environments.

Despite all these advantages, QUIC still faces some problems when it comes to implementing packet pacing in its user-space environment. Unlike kernel-level protocols like TCP, which benefit from being integrated with the operating system's network stack and hardware, user-space protocols like QUIC operate at a higher layer, relying on application-specific libraries and interfaces. Therefore, using kernel-level optimizations like the pacing scheduler is not possible, and another solution has to be found to mitigate this problem.

Because of that, it is important to research different methods to mimic kernel-level pacing mechanisms within user-space applications. One interesting method we will talk about is TSN qdiscs, which enable user-space applications to use Time-Sensitive Networking.

3. TSN Qdiscs as solution

3.1. Introduction to TSN qdiscs

Time-Sensitive Networking (TSN) is a set of IEEE standards developed to provide reliable real-time communication over networks by introducing advanced traffic management techniques to support applications like industrial automation and live audio-video transmission that demand precise timing and low-latency data transmission. The Linux kernel has implemented different TSN queueing disciplines (qdiscs) to introduce various mechanisms for managing packet transmission in the kernel space. For pacing our user-space applications, we will focus on these qdiscs provided by the Linux kernel as well, as there are far too many different qdiscs to evaluate all of them.

Linux currently supports three different qdiscs related to TSN. [8]

Credit-Based Shaper (CBS): CBS, which is defined in the IEEE 802.1Qav standard [9], assigns credits to different traffic classes and manages when packets can be transmitted based on the accumulated credits. By dynamically adjusting the transmission rate according to available credits, this qdisc can manage the use of available resources and allocate the packets accordingly.

Enhancements for Scheduled Traffic (TAPRIO): TAPRIO implements features introduced by the IEEE 802.1Qbv standard [10]. It introduces precise packet transmission scheduling, so that network administrators can define transmission schedules for different traffic classes, including priority access for important data. This deterministic approach to packet scheduling requires more manual scheduling by the administrators in contrast to the other two alternatives, but has less operational overhead. With multiple priority levels and strict scheduling policies, TAPRIO provides an efficient transmission of time-sensitive data streams in our network.

Earliest TxTime First (ETF): This qdisc is not part of the IEEE TSN standards. However, it is a specialized queueing discipline in the Linux kernel designed to prioritize packet transmission based on transmission time (TxTime) values. The problem is that only certain network interface cards, like the Intel Ethernet Controller I210, support this feature by letting packets be tagged with specific TxTimes that specify when they should be sent. ETF then proceeds to transmit the packets according to their TxTime, enabling real-time applications to maintain their synchronization and timing. [11]

All in all, time-sensitive networking and the queueing disciplines are great tools in the Linux system that offer various possibilities to manage packet transmission with precision timing and prioritization. If they are applicable in user-space applications, they can be used to implement a pacing-like mechanism for applications like QUIC. If this is successful, we can mitigate bursty traffic and minimize latency, effectively gaining the same benefits as from pacing itself. Therefore, by evaluating their applicability to user-space environments in the following sections, we can potentially find a way to get all the kernel-level benefits we already have to a user-space-level as well, ultimately supporting a similar mechanism to pacing for protocols like QUIC.

3.2. Evaluating methods of using TSN qdiscs in user-space

Usually, TSN (Time-Sensitive Networking) queuing disciplines (qdiscs) are implemented at the kernel level [12] [13] [11] and interact with the network stack to manage packets. Operating at the kernel level provides qdiscs with the advantage of relatively low overhead and the ability to interact closely with the operating system. However, to use TSN qdiscs with user-space protocols like QUIC, modifications are needed to adapt the kernel-level qdiscs for compatibility with these protocols.

3.2.1. Implementing qdiscs in user-space. Implementing qdiscs directly in the user space to be used by our user space applications allows for greater flexibility and customization. Certain measures can be taken to adapt the implementation to fit the user space and our application. As our implementation is independent of the kernel-level implementation, it is isolated from changes and updates of the kernel implementation. This brings advantages like more control and more stability to our implementation. However, new findings and improved implementation have to be maintained by ourselves. Another downside is that user-space implementations generally introduce more overhead than kernel-level operations, potentially affecting the performance. Also, using kernel-level functionality in user-space can be complex and may be time-consuming to develop. As it is our own implementation, extensive testing and performance checks are necessary. However, as the kernel implementation is rather well-tested and optimized, it is probable that our implementation will not quite match the quality of the one at the kernel-level.

3.2.2. Using the kernel-level implementation. In contrast to that, when trying to rely on the kernel-level implementation, communication between the kernel-level and our user-space application is necessary. A common approach involves creating an interface or API to expose the kernel-level qdiscs to our application. With this method, we can take advantage of the efficiency of kernel processes, providing minimal performance overhead. Kernel-level implementations are generally more robust and stable due to extensive testing and integration with the operating system. Therefore, we benefit from these aspects by directly using the well-tested implementation. Also, developing an API to use these qdiscs in user-space applications is naturally less effort and can be done faster than developing the qdiscs on the user-space itself. Therefore, it is also less prone to development errors and offers a more robust method of using qdiscs in our application. Of course, some overhead will exist due to the communication with the kernel level. However, this is usually quite minor when working with an API on the same machine.

3.2.3. Conclusion of using qdiscs in user-space. Due to the improved quality of the implementation, the reduced development effort, and the potential performance advantages, using the kernel-level implementation by introducing an API or other interface for communicating with the kernel implementation from our user-space application seems like the better method for using TSN qdiscs in user-space applications. As the Linux kernel currently offers three different types of implemented qdiscs, the question of which of these qdiscs is best suited for our task arises. This will be evaluated in the following section.

3.3. Evaluating the best suited qdisc for pacing

As we figured that using the already implemented TSN qdiscs of the Linux kernel seems to be the most appropriate solution for our purpose, we will only evaluate the three qdiscs supported by the kernel. It is definitely notable that many more qdiscs are standardized by IEEE. However, they would need to be manually implemented in the user space. This approach has its downsides, and the other qdiscs do not provide massive advantages compared to the existing implementations. Therefore, we will focus on the existing qdiscs in the Linux kernel and evaluate the best option for our task [8].

3.3.1. Credit-Based Shaper (CBS). As previously mentioned, CBS (Credit-Based Shaper) operates by allocating credits to different traffic classes and dynamically adjusting transmission rates based on the availability of these credits. [9] This dynamic adjustment is also one of the biggest advantages of CBS as a qdisc, as it offers a high flexibility for managing packet transmission. The credit-based mechanism regulates the timing of packet transmissions, which results in smoother bursts of traffic which is what we want to achieve with our pacing emulation. By controlling the rate at which packets are sent, CBS can provide a consistent transmission rate and optimize our network performance. This adjustment is especially beneficial for user-space applications like QUIC that require an adaptive rate, depending on the required usecase.

However, using CBS in user-space applications also has some problems. When implementing a pacing-like solution that uses CBS, we will naturally introduce some overhead that could potentially damage our performance. This overhead comes from the additional computing power needed for managing the credit system and the calculations of the transmission rate based on the remaining credits. This amount of overhead may not be that prevalent with other types of qdiscs, as they do not have a credit management mechanism.

In conclusion, CBS has a great flexibility for managing transmission rates and controlling packet timing, making it suitable for implementing pacing-like mechanisms in protocols like QUIC. Its dynamic credit-based adjustment ensures smoother traffic flows, which reduces latency spikes and jitter in high-traffic scenarios when compared to the other TSN qdiscs. There is some overhead that comes with using CBS, but that may be worth it for complex systems that can benefit off the flexibility that the credit system offers.

3.3.2. Enhancements for Scheduled Traffic (TAPRIO). Enhancements for Scheduled Traffic, or TAPRIO for short, enables network administrators to define different traffic classes for packages and create transmission schedules for those [10]. TAPRIO also offers multiple priority levels for critical data and, with that, provides consistent and time-sensitive data transmission, which is really beneficial for user-space applications like QUIC due to a low latency and precise timing. This approach of packet scheduling makes this qdisc a relatively precise and efficient option with little overhead, which is another major advantage of this qdisc. Especially compared to CBS, TAPRIO mitigates the heavy computational overhead needed in CBS by not using a credit-based management system. With those mechanisms, this qdisc can effectively emulated pacing in user-spaced protocols.

However, TAPRIO's deterministic nature also introduces some challenges. Manually configuring the schedules appropriately means a high effort from the network administrator side. Especially in more complex applications, this may be prone to more errors. Especially a dynamic user-space application like QUIC requires frequent adjustments to the transmission schedule based on real-time network conditions. Additionally, the synchronization between the application and schedules introduces overhead and also potential security risks if configured poorly. For example, misconfigured TAPRIO schedules could allow attackers to exploit predictable packet transmission times by using timing attacks.

TAPRIO's biggest downside, however, is that the manual configuration might not offer the same level of flexibility as dynamically adjusted qdiscs like CBS. While TAPRIO is best suited in environments where precise and stable scheduling is important, it is not as adaptable to changing network conditions that require real-time adjustments to transmission rates. For user-spaced applications, flexibility should be a priority to offer a suitable solution for most applications.

In conclusion, even though TAPRIO offers robust and precise packet scheduling, its restricting manual configuration is very limiting for our purpose. For supporting a pacing-like mechanism for user-space applications it is important to provide flexibility and minimize the configuration overhead on the administrator side. As TAPRIO does not offer that flexibility and introduces a large configuration overhead for the transmission scheduling, it does not offer a more suitable solution than CBS does, even though it would mitigate the computational overhead provided by CBS.

3.3.3. Earliest TxTime First (ETF). The Earliest TxTime First (ETF) qdisc is quite different from the other two supported qdiscs. First, it is not part of the IEEE TSN standard, as mentioned, making it less standardized. This qdisc is designed to prioritize packet transmission based on a unique value named transmission time, or TxTime. By allowing packets to be tagged with specific TxTimes, ETF can specify the exact moment they should be sent. This feature is particularly supported by certain network interface cards, such as the Intel Ethernet Controller I210, to enable the accurate timing of packet transmissions. [11]

ETF offers some advantages over CBS and TAPRIO; however, they are really situational. This qdisc can ensure that packets are transmitted at precise intervals, maintaining synchronization and, with that, minimizing latency and jitter. While this sounds like a great advantage, it comes with the cost of severe dependency on hardware support. Not all network interface cards support TxTime tagging, making ETF a specialized and niche tool for managing certain applications like high-frequency trading in finances. Even if the hardware aspect was to be overcome, the fixed TxTime still lacks the flexibility we seek in our qdisc, making it rather similar to TAPRIO. All in all, for an all-purpose tool and a solution for pacing in the user-space, ETF does not provide a suitable solution and is probably not worth investigating further.

4. Conclusion

This paper analysed the feasibility of using Time-Sensitive Networking (TSN) queueing disciplines (qdiscs) as a way to emulate effective pacing for user-space protocols like QUIC. Firstly, we concluded that building upon the already implemented qdiscs in the Linux kernel via an interface or API that is yet to be implemented seems like the best approach. Then, after evaluating the three main TSN qdiscs supported by the Linux kernel—Credit-Based Shaper (CBS), Enhancements for Scheduled Traffic (TAPRIO), and Earliest TxTime First (ETF)—we conclude that CBS offers high flexibility with its dynamic credit-based mechanism, which is a desirable feat for implementing pacing for our applications. ETF falls short due to its hardware constraints, and TAPRIO introduces a high configuration and maintenance overhead that should be avoided in the user-space.

Therefore, CBS offers the most promising option for implementing pacing in user-space protocols. However, the integration of any TSN qdisc in user-space applications like QUIC would still require consideration of overhead and compatibility issues. Also, it is notable that there might be other qdiscs described by IEEE that are better suited than the preexistent ones in the Linux kernel. Those would need to be implemented manually, of course, providing a whole range of different challenges but also advantages.

In further research, the feasibility and implementation of an API for communicating between the application and the kernel-level qdisc should be evaluated to fully realize the benefits of pacing in user-space environments.

References

[1] C. Nandhini and G. P. Gupta, "Exploration and evaluation of congestion control algorithms for data center networks," *SN Computer Science*, vol. 4, no. 5, p. 509, 2023. [Online]. Available: https://doi.org/10.1007/s42979-023-02016-4

[2] M. Ghobadi and Y. Ganjali, "Tcp pacing in data center networks," in *2013 IEEE 21st Annual Symposium on High-Performance Interconnects*, 2013, pp. 25–32.

[3] E. J. Iyengar and E. M. Thomson, "Quic: A udp-based multiplexed and secure transport," Internet Engineering Task Force, Proposed Standard RFC 9000, May 2021, accessed: 2024-06-16. [Online]. Available: https://www.hjp.at/doc/rfc/rfc9000.html

[4] N. Finn, "Introduction to time-sensitive networking," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 22–28, 2018.

[5] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of tcp pacing," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 3, 2000, pp. 1157–1165 vol.3.

[6] E. Dumazet, "Tcp small queues," *LWN.net*, July 2012, accessed: 2024-06-16. [Online]. Available: https://lwn.net/Articles/506237/

[7] ——, "Fq: Fair queue traffic policing," https://www.man7.org/linux/man-pages/man8/tc-fq.8.html, iproute2 project, September 2015, accessed: 2024-06-16. [Online]. Available: https://www.man7.org/linux/man-pages/man8/tc-fq.8.html

[8] "Configuring tsn qdiscs — tsn documentation project for linux* 0.1 documentation," 2018, readthedocs.io. [Online]. Available: https://tsn.readthedocs.io/qdiscs.html

[9] "Ieee standard for local and metropolitan area networks– virtual bridged local area networks amendment 12: Forwarding and queuing enhancements for time-sensitive streams," *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pp. 1–72, 2010.

[10] "Ieee standard for local and metropolitan area networks – bridges and bridged networks - amendment 25: Enhancements for scheduled traffic," *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–57, 2016.

[11] Man7.org, *tc-etf(8) - Linux manual page*, 2018, accessed: 2024-06-16. [Online]. Available: https://www.man7.org/linux/man-pages/man8/tc-etf.8.html

[12] ——, *tc-cbs(8) - Linux manual page*, 2018, accessed: 2024-06-16. [Online]. Available: https://www.man7.org/linux/man-pages/man8/tc-cbs.8.html

[13] ——, *tc-taprio(8) - Linux manual page*, 2017, accessed: 2024-06-16. [Online]. Available: https://www.man7.org/linux/man-pages/man8/tc-taprio.8.html

Evolution of Wireless Security

Nils Lorentzen, Leander Seidlitz*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: nils.lorentzen@tum.de, seidlitz@net.in.tum.de

***Abstract*—Wireless connections have become one of the most commonly used connection types, and securing connections over an open medium has a long history. A lot has changed since the beginning of WEP in 1997 to today's standard WPA3, released in 2018. While WPA2 was the prevailing standard for a significant period, its vulnerabilities grew to a critical point, necessitating the development of a new, more robust standard. This paper focuses on the transition from WPA2 to WPA3 and gives an overview of their design principles and reasons for change. There is also a short look at other additional improvements to wireless security made for open networks.**

***Index Terms*—WPA2, WPA3, Wireless Networks, Security**

1. Introduction

Millions of people use the Internet over a wireless connection at work, home, or while travelling daily. In an industrial nation, almost everyone uses a mobile phone or a laptop in their daily lives. Without proper protection, it is no problem to read, intercept, and change messages sent over a wireless connection because it is not a closed system, and theoretically, anyone can access the used frequencies. The IEEE committee introduced different security protocols to ensure the confidentiality and integrity of those messages, starting with the Wired Equivalent Privacy protocol (WEP) in 1997. Because of serious vulnerabilities, it was replaced by the first Wi-Fi Protected Access protocol (WPA). WPA was introduced in 2003 with the IEEE 802.11i [1] standard as a temporary solution because of the weak Rivest Cipher 4 (RC4) encryption algorithm used in WEP and was soon updated to WPA2 in 2004 [1]. Over the years, there were some amendments to this standard, but until 2018, when WPA3 was announced, no newer version existed. WPA2 is still widely used today, but it definitely has weaknesses, some of which were discovered over the years. This is why WPA3, the newer standard, became increasingly necessary, as extensions developed to counter vulnerabilities were just optional. To understand the main changes from WPA2 to WPA3, we will first examine the basic protocol procedure of WPA2 and then examine what changed with WPA3. Afterwards, we will also look at the weaknesses of WPA3 [1], [2].

2. Wi-Fi Protected Access 2 (WPA2)

The WPA2 protocol introduced in the IEEE 802.11i-2004 standard establishes a secure connection to an access point and is meant to provide confidentiality, integrity, and mutual authentication between a device and the access point. The big problem with WEP and WPA was the weak RC4 encryption algorithm, which was shown to have multiple vulnerabilities. WPA was only a temporary solution to address this critical weakness of WEP and it used a longer key size for the RC4 stream cipher. This changed with WPA2, which implements the AES-CCMP encryption algorithm as defined in the IEEE 802.11i-2004 [1] standard. AES-CCMP is based on the Advanced Encryption Standard and uses the Counter Mode with Cipher Block Chaining Message Authentication Code Protocol, which is computationally more demanding than RC4. So much more demanding that new hardware was needed for access points. This was ultimately why WPA with RC4 and a longer key size exists at all [1].

2.1. Key Management

According to the IEEE standard, the access point is referred to as the authenticator, and the client is referred to as the supplicant. These definitions will also be used here. WPA2 does not use a single key to achieve the desired security goals. Instead, two key hierarchies are defined, one for each of the following scenarios. There are two scenarios for communicating in wireless networks, either directly via unicast between the supplicant and authenticator or as a broadcast/multicast to the network. WPA2 also uses two keys for these two scenarios. For unicast messages, the Pairwise Transient Key (PTK) is used. As the name states, this key is pairwise, unique between the authenticator and a supplicant. At the top of the unicast key hierarchy stands the Pairwise Master Key (PMK). This 256-bit key needs to be known by the authenticator and supplicant before the four-way handshake explained in Section 2.2 can be done. This can be done in different ways, such as using a Password-Based Key Derivation Function, in this case PBKDF2, to derive the PMK from the passphrase. The PTK is not directly used to encrypt or decrypt messages after the handshake. Instead, it is split into different parts for different tasks. The first 128 bits of the PTK will be the Key Confirmation Key (KCK), which provides data origin authenticity for handshakes. The following 128 bits will be the Key Encryption Key (KEK), which encrypts handshake messages and provides confidentiality. The following 128 bits will finally be the Temporal Key (TK), which is then used to encrypt and decrypt messages after the handshake. The other type of key used is the Group Temporal Key (GTK), shared between all supplicants and the authenticator. The

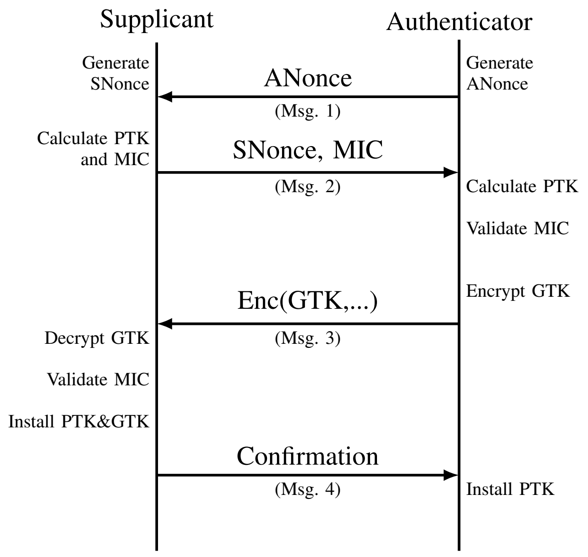


Figure 1: Short diagram of the four-way handshake [1]

GTK is derived from the Group Master Key GMK, a cryptographically secure random number generated by the authenticator. It makes sense to reinitialize the GMK after a specific time interval and redistribute a new GTK to all supplicants. This can be done via a separate simple group key handshake. PTK and GTK are distributed during the four-way handshake, further explained in Section 2.2 [1].

2.2. Four-Way Handshake

Before the four-way handshake starts, the supplicant sends an association request to the authenticator, who then sends an association response. Now, both participants generate a cryptographically secure nonce. The authenticator starts the handshake by sending his Authenticator Nonce (ANonce) to the supplicant. This first message and the following messages can be seen in Figure 1. After receiving the ANonce, the supplicant can compute the PTK using both nonces, the PMK and the MAC addresses of both participants. The nonces are used to protect against replay attacks. To ensure that no one interferes with the first two messages, the supplicant computes a message integrity code (MIC). This is where the KCK is used. The second message's primary information is the SNonce, which is integrity-protected by the MIC. The authenticator can now also compute the PTK and validate the MIC. If a third party changes the ANonce or SNonce in the first or second message, this will be detected at this point. Essentially, both supplicant and authenticator now have a shared key and can communicate encrypted. The only thing missing is that the authenticator sends the GTK to the supplicant and the handshake can be finished with a last confirmation message of the supplicant [1].

2.3. Security Considerations

Especially with wireless communication compared to wired communication, it is easy to intercept or eavesdrop on messages, so it is crucial that the protocols used offer as little attack surface as possible. For example it is possible for anyone to sniff the four-way handshake and this alone is no problem but can get one if the station listening is malicious and also has access to the pre-shared secret (in most cases a passphrase) used for the network. With this extra information, it is also possible for the attacker to compute the exchanged keys. Even if the attacker is too late to eavesdrop on the handshake or has no access to the pre-shared secret, there are still open attack vectors, as shown in the following.

2.3.1. Deauthentication Attack. One uncomplicated attack on a wireless connection secured by WPA2 is the deauthentication attack. Management and control frames are not part of the payload in these connections and are, therefore, not encrypted nor authenticated. Usually, a deauthentication frame is sent by either the supplicant or the authenticator to indicate that the connection should be closed, but it is easy for an attacker to spoof the source MAC address and send this frame repeatedly to either the authenticator or the supplicant [1], [3].

2.3.2. KRACK Attack. The Key-Reinstallation Attack (KRACK) was first demonstrated in 2017 by Vanhoef and Piessens in [4] and raised major concerns about the security of WPA2. This attack focuses on the four-way handshake and it works by tricking the victim into reusing replay counter values with the same key. After installing a key through the regular four-way handshake, the replay counter value starts at zero and increases after sending messages. The attack concept is to trick the victim into installing the same key as before and reinitializing the replay counter to zero. This enables the attacker to read all sent packets even with AES-CCMP in use [4], [5].

2.3.3. Handshake Capture Dictionary Attack. By eavesdropping on a successful handshake, an attacker can use this technique to obtain the passphrase used by the authenticator. The attack is based on an offline dictionary attack. It exploits the fact that both nonces are sent in plain text and are the only random source for calculating the PTK. After both nonces have been intercepted, it is possible to force the passphrase and validate the current attempt with the MIC sent in the second message. Therefore, this can be performed as an offline attack, making the brute force and dictionary attempt possible [6].

3. Wi-Fi Protected Access 3 (WPA3)

Over the years, amendments have been made to the original IEEE 802.11i standard to eliminate vulnerabilities. However, these changes were voluntary and must be used by both communication partners. Therefore, the Wi-Fi Alliance introduced WPA3 in 2018, just one year after the publication of the KRACK attack method. This version aims to eliminate all known vulnerabilities of the old standard, including the previously mentioned KRACK and deauthentication attacks. There are different variants of WPA3, for example, WPA3-Enterprise only mode or WPA3-Personal only mode, which we will focus on here because of simplicity. The Enterprise mode is mainly used for bigger company and institutional networks while the Personal mode is also deployed in many private households [6], [7].

3.1. Simultaneous Authentication of Equals

With several security improvements, WPA3 adds another layer to the initial key exchange handshake. The technique used is called Simultaneous Authentication of Equals (SAE), which Dan Harkins first introduced in 2008 [8]. SAE Public Key (SAE-PK) is the extended version used in WPA3 to counter attacks like the "evil twin AP" attack [7] and also the previously mentioned KRACK attack. With this extension, an asymmetric cryptography key pair also authenticates the access point. SAE is a version of the Dragonfly key exchange that is based on the discrete logarithm problem and, therefore, works with prime modulo groups or elliptic curve groups. In comparison to integer exponentiation modulo a prime problems are elliptic curves still harder to solve. Therefore, the keys for elliptic curves can be smaller and still be considered secure. In contrast to the regular four-way handshake, the passphrase is not directly used to compute the PMK. Instead, the passphrase is converted to a specific elliptic curve similar to a hash function, which is then used to compute a password element (PE). To increase the entropy in this equation, an increment counter, supplicants, and authenticators' MAC addresses are used in an iterative procedure to determine the PE. The increment counter is increased in each iteration, and then a new hash is computed for all factors. This hash is then used as x and if there exists a solution for y in (1), the coordinates (x,y) will be used as PE in the following handshake. a,b and p are factors of the specific elliptic curve that is used [6], [8], [9].

$$y^2 = x^3 + ax + b \mod p \tag{1}$$

Now, this PE will be used in the dragonfly handshake, which outputs an initial PMK that can then be used for the normal four-way handshake [6].

The way this dragonfly handshake is constructed it is not computationally feasible to reconstruct the PMK after learning about the passphrase. So this protocol is perfect forward secret, which was not the case with standard WPA2 [6], [9]. Because of the extra entropy added in this procedure, offline dictionary attacks are no longer possible as well [9].

3.2. Protected Management Frames

Another issue with WPA2 was the relatively easy deauthentication attack 2.3.1, which is one reason why in 2009 the amendment IEEE 802.11w [10] was made where the Protected Management Frames (PMF) protocol was introduced. The usage of these frames became mandatory for WPA3. IEEE 802.11w protects specific frames as Robust Management Frames (RMF). These are disassociation, deauthentication and robust action frames. So the amendment itself is named Protected Management Frames while the frames itself are part of the Robust Management Frames. PMF uses the Broadcast Integrity Protocol (BIP) to guarantee data integrity and replay protection. Essentially, a MIC is computed not only over the data frames but also for management frames [6], [10], [11].

Protected Management Frames protect against deauthentication attacks, if an attacker sends an unprotected deauthentication request the receiving station will no longer directly deauthenticate the device but will temporarily reject this request and also send a Security Association (SA) query back. If the original station that the attacker wanted to deauthenticate is in the network, it will be able to answer this SA query with the correct key. Otherwise, the SA query will timeout, and it can be assumed that the original station is either already disconnected or no longer able to use the key and needs to re-associate [6], [10], [11].

3.3. WPA3 Security Considerations

Even though WPA3 was intended to eliminate all vulnerabilities of WPA2, this is not the case. Several attacks were found on different WPA3 mechanisms, including the previously mentioned Protected Management Frames and Dragonfly key exchange.

3.3.1. Deauthentication Attack on PMF. In a scenario with one supplicant and one access point in a unicast communication channel it is possible to deauthenticate these two peers. To achieve this, "a large number of spoofed unprotected unicast deauthentication frames" [11] are sent to both peers. This means that the supplicant and access point will start sending SA queries to the other peer. As soon as the access point sends the SA query to the supplicant it ignores any SA query coming from the supplicant. This will then lead to a timeout on the supplicant's side and cause a disassociation [11].

3.3.2. Dragonblood Attacks. In April 2019, M. Vanhoef, who also participated in the KRACK attack [4] and E. Ronen published a paper [9] about multiple attack vectors on the dragonfly handshake. The so-called Dragonblood attacks contain, among others, timing side-channel, downgrade and denial-of-service attacks. For example, it is enough to know the SSID of the network and be close enough to the victim to perform a downgrade attack by just advertising a WPA2-only network. During the four-way handshake, the downgrade attack will be detected by WPA2, but with the information gathered through authenticated four-way handshake messages, a dictionary attack becomes possible [9].

3.4. WPA Conclusion

Keeping wireless connections secure is not a Task which is done at some point. Over the time someone will eventually come up with an idea to attack the protocols in place and this is also the case for WPA3 as well as it was the case for WPA2. Protocols have to evolve and adapt constantly to vulnerabilities as well as other factors that may change the way wireless connections work. WPA3 may not be perfectly secure today but for most cases the attacks on it are hard enough to not be really worth it.

4. Opportunistic Wireless Encryption

WPA3 is not the only protocol currently available to secure wireless connections, in a different use case a different approach might be better. For example in todays

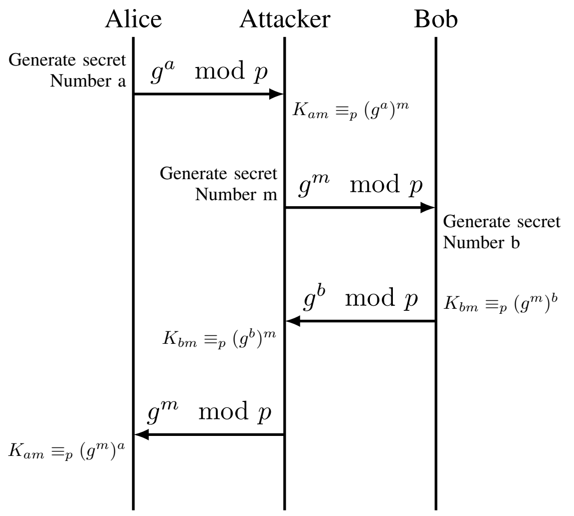


Figure 2: A Diffie-Hellman key agreement man in the middle attack with publicly known integer prime numbers g and p. K_{am} and K_{bm} are the resulting DH-keys between Alice/Bob and the Attacker. They can be used as seed for a proper key derivation function [14].

world it is common to have internet access almost everywhere at public places like restaurants or airports offered by unprotected wireless networks. It is just not practical enough for large public spaces to distribute shared secrets to everyone, and this would also not be convenient enough to attract customers. The only possible way to do this without much effort is to publicly advertise the passphrase to the network. This technique became popular over the years and as explained in 2.3 it is possible to completely bypass this which is even worse because users get a wrong impression of security in their connection to the internet. So a way to protect these kinds of connections is needed and a one way to improve this is called Opportunistic Wireless Encryption. OWE was standardized in 2017 with RFC8110 [12] and is currently not part of the WPA3 standard [7] but was introduced as the Wi-Fi Enhanced Open certification by the Wi-Fi Alliance in 2018 [13]. Essentially, a Diffie-Hellman key exchange is done, and the resulting shared secret is used in the four-way handshake. This way no public passphrase is needed and every participant has an unique shared secret with the access point but it can not be guaranteed that this is the correct access point. Plain Diffie-Hellman does not provide any authentication [12].

OWE is a replacement for unencrypted communication and can contribute to a more secure connection. The end user does not have to actively participate in this protocol, which maintains the convenience of an open connection. However, the problem is that an active man in the middle attack on the Diffie-Hellman key exchange is still possible. This can be seen in Figure 2. This is why it does not provide any type of authentication, as said in Section 4. The only things protected by OWE are packet integrity, confidentiality, and authenticity between two peers. Which peer is really on the other side of the connection is not known. This is where the Opportunistic approach comes from. The protocol hopes that on the first connection the correct access point is chosen and therefore a secure connection can be established. A proper end-to-end connection on a higher layer should still be established to mitigate the security risks in an open connection [12].

5. Conclusion

For many years, WPA2 was the newest standard in terms of wireless security. It was constantly updated and improved, but the use of these amendments was not mandatory. Because of this, a new standard is necessary at some point, and with the KRACK attacks described in Section 2.3.2, this point was reached. WPA3 was introduced and includes new features and techniques already used as amendments to the WPA2 standard, which has now become mandatory to use in WPA3. The goal is to eliminate all WPA2 vulnerabilities. The KRACK attack is countered by a new extended handshake, the Dragonfly handshake. Nevertheless, just one year after its release, the Dragonblood attacks revealed significant weaknesses in this handshake. Another new security mechanism is the Protected Management Frames, which are designed to prevent deauthentication attacks, among other things. It took more time to break these, but eventually, in 2022, Lounis et al. [11] published several ways for deauthentication attacks on Protected Management Frames. Overall, WPA3 is harder to attack than WPA2, which is an improvement, but it is still vulnerable as shown in Section 3.3. Achieving a high safety standard requires much work and constant further development. Solving all these vulnerabilities is a hard task, but as of today WPA3 is the newest standard regarding wireless security and has solved many vulnerabilities of the past versions and therefore should be used. It is uncertain for how long WPA3 will be the newest standard around. Maybe sometime in the future, WPA4 will be necessary because some vulnerabilities can not be solved with an amendment to the WPA3 standard or a completely new standard will be introduced to accomplish the goal of securing wireless connections to the Internet. One example of an independent addition to wireless security in open networks is Opportunistic Wireless Encryption, even though it was discussed to be included in the WPA3 standard. Of course, this protocol has its own problems and is no replacement for a proper authentication and encryption method. This is one of the goals of future research in this area and it is important to keep updating security standards because also the attacks on wireless networks are constantly evolving.

References

[1] "Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements-part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications: Amendment 6: Medium access control (mac) security enhancements," *IEEE Std 802.11i-2004*, pp. 1–190, 2004.

[2] "Ieee standard for wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-1997*, pp. 1–445, 1997.

[3] C. Mitchell and C. He, "Security analysis and improvements for ieee 802.11 i," in *The 12th Annual Network and Distributed System Security Symposium (NDSS'05) Stanford University, Stanford*, 2005, pp. 90–110.

[4] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in wpa2," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1313–1328.

[5] C. Cremers, B. Kiesl, and N. Medinger, "A formal analysis of {IEEE} 802.11's {WPA2}: Countering the kracks caused by cracking the counters," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1–17.

[6] C. P. Kohlios and T. Hayajneh, "A comprehensive attack flow model and security analysis for wi-fi and wpa3," *Electronics*, vol. 7, no. 11, 2018.

[7] W. Alliance, "Wpa3 specification v3.3," 2024.

[8] D. Harkins, "Simultaneous authentication of equals: A secure, password-based key exchange for mesh networks," in *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*. IEEE, 2008, pp. 839–844.

[9] M. Vanhoef and E. Ronen, "Dragonblood: Analyzing the Dragonfly handshake of WPA3 and EAP-pwd," in *IEEE Symposium on Security & Privacy (SP)*. IEEE, 2020.

[10] "Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements. part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 4: Protected management frames," *IEEE Std 802.11w-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, and IEEE Std 802.11y-2008)*, pp. 1–111, 2009.

[11] K. Lounis, S. H. H. Ding, and M. Zulkernine, "Cut it: Deauthentication attacks on protected management frames in wpa2 and wpa3," in *Foundations and Practice of Security*, E. Aïmeur, M. Laurent, R. Yaich, B. Dupont, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2022, pp. 235–252.

[12] D. Harkins and W. Kumari, "Rfc 8110: Opportunistic wireless encryption," USA, 2017.

[13] W. Alliance, "Opportunistic wireless encryption specification," *Specification v1.1*, 2020.

[14] L. Lizama and J. R., "Non-invertible public key certificates," *Entropy*, vol. 23, p. 226, 02 2021.

TSN and ATS: The Influence of Shaping on the System

Bruna Giovana Machado, Florian Wiedner*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: brunagiovana.machado@tum.de, wiedner@net.in.tum.de

***Abstract*—Rather automotive, aerospace or transportation, multiple industries depend on deterministic communication with strict timing requirements. In order to fulfill the necessary requirements, delays caused by traffic interferences must be minimized. The success of managing this traffic is critical to these industries and one possible solution is traffic shaping. This paper analyzes traffic shaping within Time-Sensitive Networking (TSN) and its impact on network performance and costs , focusing on Asynchronous Traffic Shaping (ATS) and comparing it with a synchronous shaper Time-Aware Shaping (TAS). We examine three ATS algorithms: Urgency-Based Scheduler, the Paternoster mechanism, and the ATS standard draft. Through evaluation of different simulations, we compare ATS with unshaped traffic, various scheduling mechanisms for ATS and TAS, highlighting the benefits and costs associated with each approach.**

***Index Terms*—time-sensitive networking, asynchronous traffic shaping, traffic shaping**

1. Introduction

Time-Sensitive Networking (TSN) is a set of IEEE 802 standards that ensure deterministic communication over standard Ethernet. One central mechanism of TSN is the traffic shaping techniques, managing traffic and providing bounded latency and reduced frame loss [1].

Time-Aware Shaping (TAS) is a TSN shaper, providing deterministic transmissions through synchronization among all network participants, however, in dynamic networks this characteristic is limiting [2]. Asynchronous Traffic Shaping (ATS) offers a flexible alternative, adapting to diverse network conditions [3]. We focus on the shaping mechanisms of ATS, their influence and costs.

In Section 2, we explore the theoretical background of TSN, the role of shaping and the functionality of TAS. In Section 3, we introduce three ATS's algorithms. In Section 4, we evaluate ATS, comparing it with unshaped traffic, different scheduling mechanisms, and TAS. In Section 5, we concludes and suggest ideas for future work.

2. Theoretical Background of TSN

From TSN's perspective, there are only two types of devices: bridges and end stations. End stations are further divided into talkers (sources) and listeners (targets) [1].

2.1. Shaping

In the case of data with strict time constraints, which is present in multiple industries, managing the delay caused through interferences by other participants of the network traffic is critical. TSN shapers introduce controlled delay aiming at bounded low latency and zero congestion loss by controlling the traffic flow at every hop, thus avoiding long bursts [1]. We focus on two TSN shapers: TAS and ATS.

2.2. Time-Aware Shaper

TAS requires the scheduling of traffic classes to be synchronized across all bridges from the talker to the listener(s), depicted in Figure 1. TAS schedules traffic streams in two reserved time-triggered windows: (i) for low-priority traffic, such as best effort (BE) and (ii) for scheduled traffic (ST) [4]. TAS employs a gate driver mechanism that opens and closes according to a time schedule for each port in a bridge. The Gate Control List (GCL) contains Gate Control Entries that define the transmission eligibility of a queue. A frame is allowed to be transmitted if (i) the queue has a frame ready to transmit, (ii) higher priority queues with an open gate have no frames to transmit, and (iii) the frame transmission can be completed before the gate of the queue closes [2], [4].

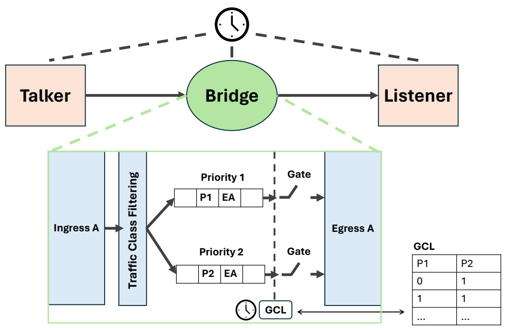


Figure 1: Visual representation of TAS adapted from [5]

3. Asynchronous Traffic Shaping

To avoid the critical failure of a timing misalignment, ATS is introduced as an alternative. It imposes similar traffic determinism without strict timing synchronization by introducing an independent clock at every bridge and end station [4].

The original concept of ATS [6] occurs at every hop and is depicted in Figure 2. First, individual frames are queued at a shaped queue of the desired egress port according to the flow state. The separation process of per flow state queues is called interleaved shaping. These queues follow three Queuing Admission Rate (QAR) schemes:

- **QAR1:** Frames from different sources are stored separately.
- **QAR2:** Frames from the same source with different priorities must be kept apart.
- **QAR3:** Frames from the same source with the same sender's priority but different receiver's priorities are separated [6].

Following these queueing schemes ensures traffic isolation [7]. Additionally, prioritizing high-priority traffic reduces their queuing time by allowing them to bypass lower-priority traffic [8].

Afterwards, the shaper merges the shaped queues conforming to the receiver's priority traffic class. The frames in the shared queue are then regulated by the transmission selection algorithm based on eligibility time [6]. In figure 2, the chosen transmission selection algorithm is strict priority FIFO.

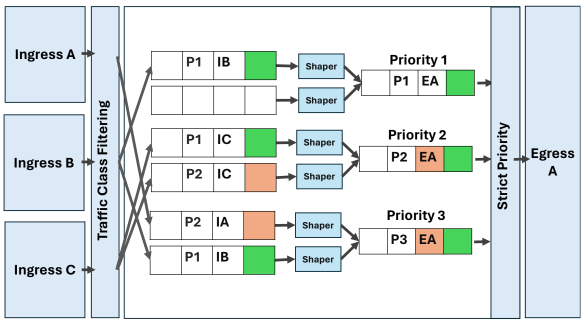


Figure 2: Visual representation of ATS adapted from [6]

Figure 2 depicts a switch with three ingress ports and one egress port with three priorities and therefore three shared queues. All of them follow the queuing schemes. The priorities from the ingress ports are depicted with "P1" or "P2" followed by the ingress port, e.g. "IA".

3.1. UBS algorithms

ATS, formally known as Urgency Based Scheduler (UBS), was created by Specht and Samii [6]. UBS has two different algorithms that utilize a per flow state:

- **Length Rate Quotient (LRQ)**: a frame-by-frame leaky bucket algorithm
- **Token Bucket Emulation (TBE)**: a token-based leaky bucket algorithm

LRQ: aims at a consistent transmission rate even for unpredictable flow patterns. The state of each flow f_i contains a timestamp t_i with an eligibility time for the current frame, based on the size of the previous frame l and the permitted burst rate of a flow $\hat{r}_i$. The frame is delayed at least until the local time of the device t_{now} reaches t_i. The eligibility time for the next frame of f_i is $t_i = t_{now} + l/\hat{r}_i$ [6].

TBE: focuses on achieving a transmission with an average rate. The state of each flow f_i contains a timestamp t_i as well as a bucket level b_i. The frame is delayed until the token count T is greater than or equal the frame size l. The tokens are measured as $T = b_i + (t_{now} - t_i) \cdot \hat{r}_i$. The eligibility time for the next frame is the current device time $t_i = t_{now}$ and the bucket level b_i = $\min\{\hat{b}_i, (t_{now} - t_i) \cdot \hat{r}_i\} - l$. This means that the delay between packets from the same flow is removed if enough "tokens" are available, possibly causing bursts [6].

TABLE 1: Variable definitions for equations (1) and (2)

Parameter	Definition
d	Upper bound on per hop delay
I	Set with all flow indices
b	Burst size
l	Frame size
r	Burst rate
$C(i)$	Flows with the same priority as i
H	Flows with higher priority
L	Flows with lower priority
$\hat{a}$	Maximum
$\check{a}$	Minimum

The mathematical evaluation of the worst-case delay of a single hop is given by [6]:

$$d_{LRQ} \leq \max_{i \in I} \left(\frac{\hat{b}_H + \hat{b}_{C(i)} + \hat{l}_L}{r - \hat{r}_H} + \frac{\hat{l}_i}{r} \right) \qquad (1)$$

$$d_{TBE} \leq \max_{i \in I} \left(\frac{\hat{b}_H + \hat{b}_{C(i)} + \hat{b}_i - \check{l}_i + \hat{l}_L}{r - \hat{r}_H} + \frac{\check{l}_i}{r} \right) \qquad (2)$$

The variables of equations (1) and (2) are defined in table 1.

3.2. Paternoster queuing and scheduling

The Paternoster algorithm [9] is an improvement over the peristaltic shaper (802.1Qh Cyclic Queue and Forwarding (CQF)) [9]. It operates in a four-phase cycle: prior, current, next, and last. Packets are first added to the current queue. If they exceed the reservation's bandwidth for an epoch, they are moved to the next and then to the last queue. Once the three queues are full, the incoming frames are discarded. These phases rotate left after each epoch duration τ, with current becoming prior, prior becoming last, etc. Each queue has a reserved bandwidth allocation for an epoch. Unlike CQF, Paternoster works asynchronously, reduces average delay, and handles multiple epoch reservations within a single epoch.

According to [9], the algorithm's per hop worst-case delay is defined as

$$d_{Paternoster} \leq 3 \cdot \tau, \qquad (3)$$

This delay occurs when both the current and next queues are full, forcing the frame to wait in the last queue for up to three cycles before transmission.

3.3. ATS algorithm

The ATS standard algorithm [3, Sec. 8.6.11.3] is a derivation of TBE.

According to [3], each bridge in a network has a set of tables for different purposes, which include parameters necessary for the traffic regulation. These tables include the ATS Shaper Instance Table [3, Sec. 12.31.5] with parameters and variables for independent instances of ATS shapers, the ATS Shaper Group Instance Table [3, Sec. 12.31.6], catering to group instances of ATS shapers, and the ATS Port Parameter Table [3, Sec. 12.31.7], which contains parameters shared by all ATS shaper instances connected to a reception port.

The final eligibility time is determined by taking the maximum of three values [3, 8.6.11.3]: the frame's arrival time, the group eligibility time (the most recent eligibility time processed by any ATS shaper in the group), and the scheduler eligibility time (the earliest moment when a frame has accumulated enough tokens to be considered for transmission). For a frame to be considered valid, its eligibility time must be less than or equal to the arrival time plus the $MaxResidenceTime$ parameter, which limits how long a frame can reside in the bridge [3, Sec. 8.6.11.3]. This eligibility time is then used by the ATS transmission selection algorithm [3, Sec. 8.6.8.5].

Due to the worst-case delay equation for the ATS standard algorithm [3, Annex V] being an extension of the equation 2, it is not covered in this paper.

4. Evaluations of ATS

To determine the influence and costs of shaping through ATS, it is crucial to evaluate and compare its algorithms from different perspectives. Given the diversity of the simulations compared in this paper, their setups will be explained.

4.1. Comparison of ATS and unshaped traffic

Setup. The evaluation in [6] simulates two different scenarios to evaluate UBS algorithms by delay. The first scenario features four talkers connected to one switch (S0), which is then connected to another switch (S1), leading to the only listener. In the first scenario, switch S0 is equipped with four queues, and S1 with one, meaning one queue per ingress port. All four talkers transmit four flows each, totaling 16 flows. The second scenario involves one talker (T0) and one listener (L0) connected through five switches, dealing with interfering flows and increased link utilization. In the second scenario, only three flows are transmitted from T0 to L0, with eight additional flows introduced along the path to simulate a more realistic multi-hop environment. Each scenario includes two series: (i) with a single priority level and (ii) with dual priority levels. Both scenarios utilize the equations 1 and 2 to predict the expected worst-case delays for LRQ and TBE, which are anticipated to be identical in the first series. Specht and Samii [6] compare LRQ, TBE, per-flow queues shaped with LRQ (LRQ-F), and strict priority FIFO scheduling (SPO) through trajectory analysis. Here, we focus on comparing LRQ and TBE with SPO.

First scenario analysis. In the first series, each of the four flows occupies one queue at S0 and then compete for the single queue at S1. The simulation results indicate equal delays caused by both LRQ and TBE algorithms, with a high discrepancy between expected and simulated results upon entering the second switch, and a low discrepancy at the first switch. With only one priority level, the delay induced by the shapers is higher than that of SPO. This effect is especially clear at the second hop, where the delay is considerably higher due to the single queue scheme [6].

In the second series, flows are assigned different priorities at each hop. Discrepancies between expected and actual delays increase at each hop, similar to the first series. High-priority flows experience lower delays than low-priority ones. However, at S1, the delay for low-priority flows under SPO is notably higher compared to UBS algorithms. This occurs because SPO suppresses low-priority flows (last eight flows at S1) due to the buildup of high-priority flows (first eight flows at S1) [6]. LRQ maintains a consistent transmission rate, while TBE averages rates with minimal bursts, thus avoiding this issue.

Second scenario analysis. At the first hop in the first series, delays for the three flows are identical across all methods. However, at the following hops, SPO exhibits significantly higher delays than the expected worst-case delays for all UBS algorithms. This likely stems from SPO's inability to manage traffic bursts, leading to congestion under heavy traffic loads [6].

In the second series, low-priority flows face higher delays compared to high-priority flows. With priorities changing at each hop, delays from UBS algorithms closely align with the expected worst-case scenarios 1 and 2. Nevertheless, SPO's delay is nearly double that of UBS algorithms for low-priority flows.

Evaluation. Shapers are highly effective for networks with multiple priorities, as they manage traffic efficiently by minimizing bursts. In particular, asynchronous shapers are twice as beneficial in environments with interfering flows. The only downside to a shaper occurs when a network has only one priority and few interfering flows. However, such scenarios are uncommon for many networks, making shapers a valuable solution for such network traffic management issues.

4.2. Comparison of scheduling mechanisms for ATS

Setup. The UBS algorithms and Paternoster are compared regarding frame loss rate, average number of queued frames and average per-hop delay in the simulations done by Zhou et al. [7], [8]. The topology used in both studies is the same, a talker is connected to a switch, which connects to a listener. Paternoster is simulated with three different epoch durations τ: 0.01s, 0.005s and 0.0025s. The results vary with the bandwidth of the input flow ranging from 4,096 to 20,48 MBit/s in [7] and 32 to 192 MBit/s in [8]. The reserved bandwidth being 5.76 MBit/s and 50 MBit/s accordingly. This generates similar outcomes in both papers.

Frame loss rate. With the increase of sent frames, the frame loss rate rises across all algorithms. Lower epoch durations τ result in higher frame loss rates due to reduced queuing time. This is because the reserved bandwidth is calculated as $3 \cdot \tau \cdot datarate$ [7]. Overall, UBS algorithms typically show a lower or equal frame loss rate compared to Paternoster in both simulations.

Average number of queued frames. LRQ stores frames longer than TBE, given the fact that LRQ must wait before transmitting multiple frames from the same flow, unlike TBE, which allows bursts. In comparison to other Paternoster variations, Paternoster A has the least amount of queued frames until the reserved bandwidth is reached. Once it is reached, Paternoster C has the least amount of queued frames [7]. The more frames that are sent, the closer each Paternoster algorithm gets to an equilibrium, which depends on the τ value. Lower τ values result in lower equilibrium levels due to higher frame loss. UBS algorithms follow this pattern, losing more frames as the input flow increases, resulting in less frames in each queue.

Average per-hop delay. The analysis confirms the worst-case delay of equation 3. All Paternoster variations show increased delay with higher input flow, however, the lower the epoch duration, the smaller frames can be forwarded at faster rates, causing Paternoster C to have the lowest delay of all. In the case of LRQ and TBE, both reach their peak delay at input flows 5.78 MBit/s [7] and 80 MBit/s [8], which is the moment when the traffic is almost overloading. Nevertheless, as soon as the overload is reached, the characteristics from LRQ and TBE of keeping the traffic constant and at an average rate create a sharp decrease [7]. In this environment, the UBS algorithms are focusing on smaller frames, which are not being discarded, clearing out the queue much faster.

Evaluation. While the given simulations does not accurately describes a multi-hop network, they successfully show the correlation between frame loss, average number of queued frames and average delay. The average delay and number of queued frames are directly linked to the frame loss rate [7]. Both ATS algorithms exhibit similar frame loss rates, resulting in a similar average amount of queued frames. For networks with many small frames, a low epoch duration τ yields the best result for Paternoster. Due to the leaky-bucket characteristic of the UBS algorithms, they perform well in overloaded networks, but transmit mostly small frames.

Since the simulation works with one queue and one priority, we can deduced out of Section 4.1 that an unshaped system, would have shown a lower delay, especially for very high input flows. If the simulations included more than one priority, the results would be more insightful. We can, however, deduce that in overloaded networks with priorities, only frames with the highest priority and smallest sizes would be transmitted, as seen in Section 4.1.

4.3. Comparison of ATS and TAS

Setups. Nasrallah et al. [4] compare the frame loss rate, mean and maximum frame delay of ATS and TAS with a ring network topology. The comparison includes sporadic and periodic scheduled traffic sources. With the knowledge that ATS does not generate extra overhead in a worst-case delay of a FIFO queue system [10] and the network calculus method introduced by Mohammadpour et al. [11], Zhao et al. [5] compare the performance from ATS and TAS using NC for only one priority. The compared aspects are the worst-case backlog (WCB), delay (WCD) and jitter (WCJ). It works with five different topologies, which are variations of ring and tree topologies. Moreover, both simulations use the standard ATS algorithm.

Frame loss rate. The results for sporadic ST sources show that ATS has a much lower frame loss rate for ST and a higher frame loss rate for best effort than TAS. The reason for this is that ATS prioritizes ST, causing more congestion in the BE queues, while TAS works with time-scheduled windows, transmitting both ST and BE consistently. Once the ST sources are periodic, the scheduled-windows work more in favor for TAS than ATS [4].

Mean frame delay. For high-priority traffic in the sporadic scenario, ATS performs with lower delays than TAS. As the load increases, ATS keeps a consistent delay for ST, whereas TAS with a 20% gate usage time for ST (TAS 1) increases slightly and with a 30% gate usage time for ST (TAS 2) increases significantly. For low-priority traffic, on the other hand, ATS performs more similarly to TAS 1 and better than TAS 2. The cause for this is the same as for the low frame loss rate. For the periodic scenario, ATS shows similar results [4].

Worst-case scenarios. In the case of sporadic ST sources, the WCD for low-priority traffic for ATS is significantly higher than any other traffic from both ATS and TAS. The WCD of ATS for high-priority traffic, however, is the lowest of TAS 1 and TAS 2. On the other hand, for periodic ST sources, the higher the period, the worse the WCD becomes for ATS, whereas TAS stays extremely low [4]. Zhao et al. [5] mention that sporadic flows are not supported by TAS, therefore it shows the same result as previously mentioned. Additionally, TAS has the lowest WCB, WCD and WCJ by far. They compares all three worst-cases for five different topologies. Moreover, they conclude that the increased concentration of transmissions and number of hops increase the traffic transmission determinism [5]. Besides, they hypothesize, that ATS will perform better, once the load increases [5]. The average hop delay discussed in Section 4.2 supports this hypothesis.

Evaluation. In scenarios with sporadic transmissions, ATS has a clear advantage over TAS in regards to frame loss rate, mean and maximum frame delay. If the topology of a network creates a high flow transmission concentration or if the transmissions are periodic, the transmissions become more deterministic, which is beneficial for TAS. The hypothesis from Zhao et al. [5], however, introduces the idea that an increased load would generate a better performance for ATS. Therefore, in networks without determinism ATS would certainly perform superior and in the case of high traffic loads, ATS might achieve more advantages.

5. Conclusion and future work

Overall, ATS algorithms offer a flexible and efficient traffic management solution, aiming for bounded low latency and zero congestion loss, especially for high-priority traffic. They perform exceptionally well in dynamic and high-load environments and operate independently of the incoming flow pattern and network synchronization.

Future work should prioritize integrations of ATS with other shaping mechanisms and evaluate its performance in more complex network topologies. Some of the mentioned simulations do not attempt to imitate real-life scenarios with complex topologies and multiple hops, which would be a significant area for future exploration. Another potential improvement for ATS algorithms would be the development of methods to recognize and optimize for deterministic traffic.

References

[1] IEEE 802, "Time-sensitive networking (tsn)," accessed: 2024-05-02. [Online]. Available: https://1.ieee802.org/tsn/

[2] IEEE, *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic*, Std. IEEE 802.1Qbv-2015, March 2016, released: 18.03.2016.

[3] ——. (2020, November) IEEE 802.1Qcr-2020 IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks - Amendment 34: Asynchronous Traffic Shaping. Accessed: 2024-05-27. [Online]. Available: https://1.ieee802.org/tsn/802-1qcr/

[4] 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.

[5] L. Zhao, P. Pop, and S. Steinhorst, "Quantitative performance comparison of various traffic shapers in time-sensitive networking," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2899–2928, 2022.

[6] 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.

[7] 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.

[8] 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.

[9] M. Seaman, "Paternoster policing and scheduling," March 2017, iEEE 802.1Qcr.

[10] J.-Y. Le Boudec, "A theory of traffic regulators for deterministic networks with application to interleaved regulators," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2721–2733, 2018.

[11] E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. Le Boudec, "Latency and backlog bounds in time-sensitive networking with credit based shapers and asynchronous traffic shaping," in *2018 30th International Teletraffic Congress (ITC 30)*, vol. 2. IEEE, 2018, pp. 1–6.

Hide and Seek: On Privacy and Security In Modern 802.11 Wireless Networks

Florian Schmidt, Leander Seidlitz*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: fs.schmidt@tum.de, seidlitz@net.in.tum.de

***Abstract*—Given the widespread adoption of network-enabled devices, security and privacy considerations are of great importance. Since the introduction of IEEE 802.11 in 1997, the wireless standard's associated security and authentication measures have evolved. Insecure methods like WEP have been replaced by the now widespread frameworks WPA2 and WPA3 which, given a correct configuration, reliably ensure confidentiality and integrity of the exchanged traffic. While randomization schemes exist to avoid sending globally unique MAC addresses, fingerprinting and tracking of users using information obtained from the PHY or MAC layer is still an open issue even in modern networks. Comparable mobile networks like 5G offer robust security with concealed identifiers and mutual authentication by default, whereas the security of Wi-Fi networks depends on the specific configuration of the access point. In this work, we provide a comprehensive analysis of security and privacy aspects in modern IEEE 802.11 wireless networks. Over the years, 802.11 has seen a significant improvement in the security level. Nevertheless, some challenges remain regarding fingerprinting, misconfiguration and circumvention of MAC address randomization schemes.**

***Index Terms*—802.11, security, privacy, wi-fi, tracking, fingerprinting, RCM, fuzzing**

1. Introduction

Since the release of the initial Wi-Fi standard in 1997 [1], the growth and technological progress in mobile devices has led to widespread adoption of the standard around the world. This ubiquity necessitates the careful consideration of security and privacy aspects concerning the standard and its implementation in devices, as vulnerabilities or design flaws in the employed protocols can have far-reaching ramifications. It is critical to ensure the confidentiality and integrity of the traffic exchanged over the Wi-Fi networks to protect users from attacks like eavesdropping, Man-in-the-Middle (MitM) or malicious networks. Even when the payload is encrypted, careful examination of the data that can be read from frame headers or is leaked by other layers is critical to safeguard the privacy of users.

This paper aims to explore the 802.11 standard with a specific focus on security and privacy aspects of the offered services. The rest of this work is structured as follows: Section 2 provides an overview over IEEE 802.11 with a focus on historical development, the MAC layer and protocol security measures. Section 3 analyses the security and privacy aspects of Wi-Fi fingerprinting, tracking, MAC address randomization, and fuzzing. Section 4 compares these aspects to 5G mobile networks. Finally, Section 5 concludes.

2. 802.11 Wireless Networks

The IEEE 802.11 standard for wireless networks was released in its original form in the year 1997. The following section will give a brief introduction to the standard.

2.1. Overview and Historical Development

Since its origin in the 1980s, the IEEE 802 project served to standardize communication in local area networks (LANs), with the other most widely used standards also including 802.3 CSMA/CD Ethernet [2].

The 802 standards generally follow a reference model closely related to the ISO/OSI model, but with a few distinct modifications. The main functionality of 802.11 is incorporated to the Physical (PHY) and Data Link Layer (DLL). The DLL is further subdivided into the two sublayers Logical Link Control (LLC) and Media Access Control (MAC) [3].

Definitions for several key terms closely associated with the standard will be provided here. A station (STA) is any device with a wireless interface capable of communication within 802.11 networks. A Basic Service Set (BSS) consists of a number of STAs communicating with each other. An Access Point (AP) is a type of STA which manages the network communication [1]. Imporant identifiers include the Service Set Identifier (SSID), the natural language label for a single network, as well as the BSS Identifier (BSSID) consisting of the MAC layer address of the AP [4].

The original 802.11-1997 standard [1] has since been amended and superseded a significant number of times with a concrete focus on the performance, reliability and security of 802.11 networks.

2.2. MAC Layer Services

This section will discuss the services specified on the MAC layer of 802.11 networks. For the sake of brevity, the PHY layer will not be discussed in this work in much detail. It shall nevertheless be noted that the PHY layer may still be a source of information disclosure from a security perspective.

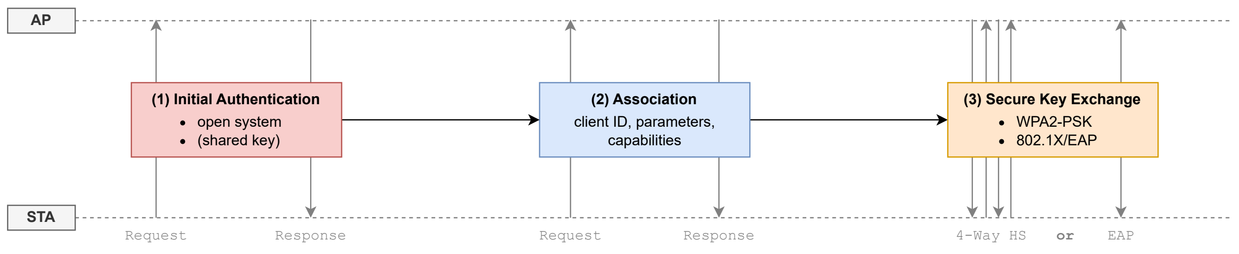


Figure 1: IEEE 802.11 association and authentication procedure, adapted from [4].

Addressing. For addressing communication partners on the MAC layer, the 48-bit LAN MAC addresses standardized in 802-2001 are used. The first 3 octets of this address make up the Organizationally Unique Identifier (OUI) assigned by the IEEE, leaving the remaining 3 octets to be assigned by the organization or device vendor [5]. The second last bit in the first octet of the address constitues the Universally or Locally administered (U/L) flag, indicating that, when set to 0, this address is universally administered and therefore globally unique.

Frame Structure. The primary transmission unit of 802.11 networks is called a frame, comprised of frame control information, sequence numbering, a checksum, the payload, and four MAC addresses. Frames are generally classified into three distinct types by their purpose. Data frames carry payload from the higher reference model layers, which commonly takes the form of Internet Protocol (IP) packets. Management frames are used for network management and can, for example, take the form of beacon frames or probe requests. Lastly, control frames are used to coordinate access to the wireless medium for carrier sensing and collision detection [4].

A critical point to mention is that management frames are typically sent over the medium in plaintext and without integrity protection. This vulnerability has been somewhat rectified by the 802.11w-2009 amendment's introduction of Protected Management Frames (PMF) [4].

Association and Authentication. In order to communicate over a Wi-Fi network, a STA must associate with the corresponding AP. In this process, the STA is assigned an association ID and parameters indicating specific capabilities are exchanged between the STA and the AP [4].

Prior to the association, a device must authenticate itself to the AP. Originally, Wired Equivalent Privacy (WEP) shared key authentication was intended to be used for this purpose in the authentication phase. However, since its deprecation due to critical security flaws, the procedure typically realizes an open system authentication leading to an always positive authentication response by the AP (essentially realizing a dummy authentication), followed by the actual, security-relevant cryptographic authentication handshake *after* the above-mentioned association phase [2], [4]. This procedure is illustrated in figure 1. More details on 802.11 security measures will be discussed in section 2.3.

Network Discovery. There are both active and passive scanning procedures for discovering APs in range of a given STA. With passive scanning, a STA listens for beacon frames broadcast by APs at regular intervals to advertise their networks. When actively scanning, a STA sends out explicit probe requests which may or may not be specific to a single, searched-for SSID. Addressed APs respond with probe responses containing detailed information about its network [2], [4].

2.3. Authentication and Security Measures

Security mechanisms in 802.11 networks are frequently divided into the classes Pre-RSNA and RSNA in literature, with the name stemming from the Robust Security Network Association (RSNA) specified in the 802.11i amendment [2].

Pre-RSNA. Networks using Pre-RSNA security rely on WEP for confidentiality and the procedures *shared key* or *open system* for authentication. The insecurity of the key scheduling algorithm employed in WEP was shown in 2001 by Fluhrer, Maintin and Shamir [10], where the authors outline a well-scaling ciphertext-only attack. Since this original work, even more rapid attacks have been developed, leading to the effect that any WEP key can be recovered within a negligable time period using non-specialized hardware. The mechanisms of shared key authentication and WEP are therefore completely insecure and have consequently been deprecated with 802.11i in 2004. What remains of Pre-RSNA security in modern networks is the open system authentication mentioned in section 2.2 [2].

RSNA. As part of RSNA security, 802.11i specifies the authentication mechanisms pre-shared key (PSK) and 802.1X [2]. The Wi-Fi security frameworks predominantly used today are the above-mentioned open system authentication, WPA2 and WPA3 [4].

First, PSK works by relying on a secret key known to both the STA and AP and performing a 4-way handshake for authentication. In WPA2, the key material is derived from the SSID and network password, where the latter is required to be shared out-of-band prior to the authentication process.

The second option, 802.1X, is a centralized network access control protocol that uses the Extensible Authentication Protocol (EAP). It ensures that unauthenticated devices can only transmit and receive 802.1X traffic. In contrast to PSK, a session key is derived upon positive authentication in 802.1X, from which the encryption key is then generated by the STA.

Statistics based on Wi-Fi network datasets gathered in the context of global open data initiatives indicate that the vast majority of networks still rely on WPA2 (74.5 %), with the more secure WPA3 (1.4 %) alarmingly being less represented than the deprecated WEP (3.0 %) [11].

	iOS/iPadOS 14+	Android 10+	macOS 13	Windows 10+
Randomized for Probe Requests	Always	Always	Never	Optional (default: off)
MAC generated using	BSSID	SSID, security parameters, (FQDN)	–	SSID
Randomized per Network	Always	Always	Never	Always
Randomized per Session	Never	Never	Never	Never
Randomized per Day	Never	Non-persistent in some cases (v12+)	Never	Optional (default: off)
Re-randomized on 'Forget' Network	Always	Never	Never	Always

TABLE 1: Vendor adoption of MAC address randomization schemes (from [6]–[9] and own experiments).

3. Security and Privacy Considerations

Ensuring the security and privacy of user data exchanged over a network is crucial, even more so when the network in question uses a broadcast medium. This section will therefore discuss aspects of security and privacy in modern IEEE 802.11 networks.

3.1. Wi-Fi Fingerprinting and Tracking

The fingerprinting and tracking of users is a sizeable concern in the widespread use of wireless networks, since the messages exchanged by the STAs and APs contain unique identifiers. These identifiers can be used to determine whether a specific STA, and thereby a particular person, is present at a given location.

In the simplest case, the 802.11 frame headers contain the actual universally-administered MAC address of the STA in question, distinctly identifying this specific STA. On older devices, probe requests sent during active scanning also leak the real MAC address of the STA while it is not connected to any network. This fact allows for passive tracking of the STA by eavesdropping on 802.11 frames exchanged over the medium [12].

For mitigating this vulnerability, several vendors have implemented randomization schemes allowing the STA to hide its actual, globally-unique MAC address in favor of a disposable address. The implementation details, adoption and shortfalls of these schemes are further discussed in section 3.2.

Next to the MAC address, other information is also sent over the medium, all of which can potentially be used for fingerprinting. This practice consists of the collection of enough information to either construct a full identifier of the STA, or at least to classify it based on various features.

On the PHY layer, it is conceivable to identify the network interface card used to transmit a frame by analyzing distinctive artifacts contained in the transmission. Moreover, a scrambling process is applied to 802.11 frames prior to transmission to reduce transmission errors. It is possible to correlate the scrambler values used across multiple transmissions to identify the device even if the MAC address has changed [4], [13].

On the MAC layer, it is possible to fingerprint the STA by analyzing the device class, involved operating system, driver software, chipset, and may even include device-specific fingerprinting techniques that are not broadly applicable [12]. Linking multiple frames together is simplified by the sequence number field contained in the frame header. The number, presence, order and contents of Information Elements (IEs) contained in probe requests for advertising device capabilities can also be used for fingerprinting and may even indirectly leak the STA's MAC address [14]. It is further possible to exploit the significant amount of information contained in 802.11 management frames, for example to fingerprint the AP and the environment it is operating in [4].

While fingerprinting is generally possible on all layers of the reference model, considering cost and efficacy yields the MAC layer as the optimal information source: PHY requires dedicated equipment to observe, and transport layer traffic is only sent once a STA is associated and authenticated to the network, thereby limiting the observable information exchange [12].

The procedures and techniques outlined above yield identifiers of the users, either through the actual MAC address of the STA in the simplest case, or through a combination of other information collected during the communication. These identifiers can then be used to detect whether a user is present in a specific location and to track this user through space and over time. In many cases, this involves little effort and requires no specialized hardware, as typical Wi-Fi interfaces in consumer devices support sniffing and recording the frames exchanged over the medium using monitor mode.

3.2. Randomized and Changing MAC Address

The tracking concerns outlined in the section above have led to a wide adoption of Randomized and Changing MAC Address (RCM) schemes by device vendors. The core idea is to ensure that not a single, globally-unique MAC address can be associated with a STA by replacing it with a virtual, randomly generated address.

All modern versions of general purpose operating systems support RCM. However, since there is no standardized specification for this scheme, the concrete implementation varies somewhat across vendors. An excerpt overview of the implementation details can be seen in table 1. While generally the same randomized MAC address is used per network in order to minimize service disruptions resulting from frequently changing addresses, it can be sensible to randomize more frequently in some cases like open networks for better protection. Interestingly, as seen in table 1, macOS 13 does not implement a native MAC address randomization scheme [9].

Even though RCM schemes significantly enhance user privacy, it has been shown that it is still possible, though only with an increased effort, to track and fingerprint RCM-enabled devices [4]. In a sense, it is possible to de-randomize the MAC address, thereby constructing a pseudo-identifier for the tracked STA.

As an example, a passive technique can rely on associating probe requests sent out using randomized addresses with different devices based on their inter-frame arrival times, frequency, sequence numbers, inter-burst time deltas and contained IEs. Some active attacks also include exploiting BSSIDs found in probe requests to create malicous APs or taking advantage of WPS parameters directly linked to the factory MAC address [4], [6].

3.3. Privacy Risks

As described in the sections above, the fingerprinting and tracking possibilities in 802.11 networks come with significant privacy considerations.

Privacy can be defined as the "fair and and authorized processing of Personally Identifiable Information (PII)" [4], a concept that describes information that can be used to either directly or indirectly identify an individual. In this sense, a MAC address of a STA can also be considered PII, given that it allows for determining the presence of the STA's owner in a particular location. It may also be noted that the collection of PII alone does not always pose a direct threat. In some cases, it may even be desirable for an individual to opt-in to the collection of certain data in exchange for easier usage of a system – or it may be necessary from a technical perspective to provide the sought-after service. Nevertheless, the use cases for the extracted PII include broad surveillance and tracking, directed probing, targeted advertisement, profiling of users, or mobility research and statistical analysis [4], [12].

In the context of Wi-Fi networks, the act of observation itself is fairly trivial given the broadcast nature of the medium. Any observer within range of a STA is capable of intercepting the transmitted signals and extracting PII. The possibility for Wi-Fi tracking enabled by this fact therefore carries risks, as the distinctiveness of mobility data is high enough that only few data records in space and time suffice to uniquely identify a large proportion of individuals [4], [15].

In earlier versions of well-known operating systems, STAs using active scanning to discover nearby APs sent out bursts of probe requests targeted to specific SSIDs contained in their Preferred Network List (PNL). The networks contained in a device's PNL can also be considered PII, as it has been shown to be possible to reconstruct the social graph of individuals in a large group by correlating the observed networks. In newer OS releases, this procedure has been replaced with sending wildcard probe requests directed at all APs in the vicinity, with directed probe requests only being used for discovering hidden APs that do not announce their presence by answering wildcard probes or by sending beacon frames [12], [16].

3.4. Wi-Fi Fuzzing

A holistic analysis of the security of 802.11 networks also includes checking for vulnerabilities in the implementations of STAs and APs. A useful technique in this context is fuzzing, which works by supplying the program with large quantities of (semi-)invalid input data with the aim of triggering erroneous behavior. Due to the fact that 802.11 is standardized, no reverse-engineering is required to generate the fuzzing inputs on the protocol level. Various entry points for fuzzing have been used in past works, e.g. kernel hooks or firmware emulation, but the method with the widest applicability to all STAs is over-the-air (OTA) fuzzing.

The latter transmits the fuzzing data as actual 802.11 frames to the target device and can therefore fuzz all types of frames and emulate a STA as well as an AP [17]. Since the 802.11 security frameworks generally only encrypt the payload data carried in the frame, it is possible to forge and send arbitrary management and control frames to STAs as long as PMF is not used to ensure the integrity and confidentiality of these frames. Since PMF is only mandatory on STAs using WPA3, OTA fuzzing frames are in fact processed by a large proportion of devices.

In practice, Wi-Fi fuzzing can reveal a number of implementation issues. For instance, Cao et al. found 23 vulnerabilities in their analysis, including denial of service attacks and memory corruption issues [17].

4. Comparison with 5G Networks

Even though the standards and network architectures differ quite greatly, we aim to compare the security and privacy of 802.11 Wi-Fi with 5G mobile networks.

A first observation is that 5G traffic is encrypted on the network in all cases, whereas Wi-Fi security depends on the configuration of the AP.

Secondly, similar to MAC addresses in Wi-Fi headers, mobile networks also use permanent identifiers: In 4G, the International Mobile Subscriber Identity (IMSI) is used, whereas 5G relies on the Subscription Permanent Identifier (SUPI). Since tracking is possible wherever such permanent identifiers are in use, networks up to 4G suffered from the privacy issues presented by the use of IMSI-catchers (fake base stations that actively request an end user's IMSI) [18], [19]. 5G networks aim to solve this issue by optionally encrypting the SUPI with the public key of the network operator and rotating this identifier for every session, yielding the Subscription Concealed Identifier (SUCI). However, similar to the MAC address de-randomization techniques discussed in section 3.2, it can still be possible to link user identities to SUCIs despite the encryption scheme due to weaknesses in the authentication procedure [18].

Thirdly, contrary to the personal-level Wi-Fi security measures discussed in section 2.3, the 5G authentication procedure performs mutual authentication of both the network and the user's device. In Wi-Fi, proper mutual authentication is only performed in enterprise-level protocols like 802.1X, thereby enabling attacks exploiting rogue APs like the evil-twin attack [4].

5. Conclusion

Through the evolution of the standard over time, the security and privacy of IEEE 802.11 networks improved significantly, especially with the introduction of RSNA networks and WPA3.

In correctly configured, mature networks using WPA2 or WPA3 with good passwords, the security level is very high and barring implementation flaws or elaborate attacks, users are generally not at risk of confidentiality or integrity violations. Fuzzing can help discover implementation flaws and make devices more secure. Nevertheless, privacy risks associated with the practice of Wi-Fi fingerprinting and tracking remain.

Tracking is mainly performed using information obtained from the PHY and MAC layer that, in the worst case, uniquely identifies the device in question. In case the STA uses its factory MAC address for sending probe requests or connecting to the network, this address can be directly used as an identifier. RCM schemes prevent this type of direct tracking by randomizing the exposed address, therefore significantly increasing the level of protection. Still, it is possible in many cases to de-randomize the addresses and track the devices, albeit under an increased level of effort.

Mobile networks like 5G offer a robust security and privacy framework by default, whereas the security of 802.11 networks depends on the configuration of the AP.

The concluding recommendation is twofold: Firstly, given the importance of proper configuration for the security of Wi-Fi networks, up-to-date devices and strong passwords, security-aware users with knowledge of the risks involved are essential. Secondly, ongoing research into improving protective techniques like RCM is vital to further restrict an attacker's ability of address de-randomization and fingerprinting. These measures will further enhance the security and privacy of IEEE 802.11 users in an era of ubiquitous devices.

References

[1] "IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," *IEEE Std 802.11-1997*, 1997.

[2] A. Holt and C.-Y. Huang, *802.11 wireless networks: security and analysis*. Springer Science & Business Media, 2010.

[3] "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture," *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)*, 2014.

[4] D. Ficara, R. G. Garroppo, and J. Henry, "A Tutorial on Privacy, RCM and Its Implications in WLAN," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 2, pp. 1003–1040, 2024.

[5] "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture," *IEEE Std 802-2001 (Revision of IEEE Std 802-1990)*, 2002.

[6] M. Uras, E. Ferrara, R. Cossu, A. Liotta, and L. Atzori, "MAC address de-randomization for WiFi device counting: Combining temporal- and content-based fingerprints," *Computer Networks*, vol. 218, p. 109393, 2022.

[7] J.-C. Zúñiga, C. J. Bernardos, and A. Andersdotter, "Randomized and Changing MAC Address state of affairs," Internet Engineering Task Force, Internet-Draft draft-ietf-madinas-mac-address-randomization-12, Feb. 2024, work in Progress (Last Accessed: June 10, 2024). [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-madinas-mac-address-randomization/12/

[8] Android Open Source Project, "MAC Randomization Behavior," 2024, (Last Accessed: May 27, 2024). [Online]. Available: https://source.android.com/docs/core/connect/wifi-mac-randomization-behavior

[9] Apple Inc., "Apple Platform Security: Wi-Fi privacy," 2024, (Last Accessed: Aug 6, 2024). [Online]. Available: https://support.apple.com/guide/security/wi-fi-privacy-secb9cb3140c/web

[10] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16–17, 2001 Revised Papers 8*. Springer, 2001.

[11] "WiGLE: Wireless Network Mapping," (Last Accessed: July 25, 2024). [Online]. Available: https://wigle.net/index

[12] C. Matte, "Wi-Fi tracking : Fingerprinting attacks and counter-measures," Theses, Université de Lyon, Dec. 2017, (Last Accessed: June 10, 2024). [Online]. Available: https://theses.hal.science/tel-01921596

[13] B. Bloessl, C. Sommer, F. Dressler, and D. Eckhoff, "The scrambler attack: A robust physical layer attack on location privacy in vehicular networks," in *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2015, pp. 395–400.

[14] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 413–424.

[15] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific reports*, vol. 3, no. 1, pp. 1–5, 2013.

[16] M. V. Barbera, A. Epasto, A. Mei, V. C. Perta, and J. Stefa, "Signals from the crowd: uncovering social relationships through smartphone probes," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 265–276.

[17] H. Cao, L. Huang, S. Hu, S. Shi, and Y. Liu, "Owfuzz: Discovering Wi-Fi Flaws in Modern Devices through Over-The-Air Fuzzing," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 263–273.

[18] M. Chlosta, D. Rupprecht, C. Pöpper, and T. Holz, "5G SUCI-catchers: still catching them all?" in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 359–364.

[19] M. Wang, T. Zhu, T. Zhang, J. Zhang, S. Yu, and W. Zhou, "Security and privacy in 6G networks: New areas and new challenges," *Digital Communications and Networks*, vol. 6, no. 3, pp. 281–291, 2020.

ISBN 978-3-937201-80-1
DOI 10.2313/NET-2024-09-1

ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)