Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

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

Winter Semester 2020/2021

Munich, Germany

Editors

Georg Carle, Stephan Günther, Benedikt Jaeger

Publisher

Chair of Network Architectures and Services

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

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

Winter Semester 2020/2021

Munich, August 6, 2020 – March 7, 2021

Editors: Georg Carle, Stephan Günther, Benedikt Jaeger

Network Architectures
and Services
NET 2021-05-1

Proceedings of the Seminar
Innovative Internet Technologies and Mobile Communications (IITM)
Winter Semester 2020/2021

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/`

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

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

Cataloging-in-Publication Data

Seminar IITM WS 20/21
Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)
Munich, Germany, August 6, 2020 – March 7, 2021
ISBN: 978-3-937201-72-6

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2021-05-1
Innovative Internet Technologies and Mobile Communications (IITM) NET 2021-05-1
Series Editor: Georg Carle, Technical University of Munich, Germany

Preface

We are pleased to present you the proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM) during the Winter Semester 2020/2021. 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 afterwards 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 arwards where given to Han My Do with the paper *Collaborative SLAM over Mobile Networks* and Christian Kilb with the paper *EDNS NSID Option* .

Some of the talks were recorded and published on our media portal `https://media.net.in.tum.de`.

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

Munich, May 2021

Georg Carle

Stephan Günther

Benedikt Jaeger

Seminar Organization

Chair Holder

Georg Carle, Technical University of Munich, Germany

Technical Program Committee

Stephan Günther, Technical University of Munich, Germany
Benedikt Jaeger, Technical University of Munich, Germany

Advisors

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

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

Stephan Günther (guenther@tum.de)
Technical University of Munich

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

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

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

Marton Kajo (kajo@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

Dominik Scholz (scholz@net.in.tum.de)
Technical University of Munich

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

Henning Stubbe (stubbe@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/ws2021/seminars/`

Contents

Block Seminar

Survey of Mesh Networking Messengers 1
Simon Blöchinger (Advisor: Richard von Seck)

Collaborative SLAM over Mobile Networks 5
Han Do (Advisor: Marton Kajo)

Time Sensitive Networking - 802.1Qci 9
Abdalla Mahamid (Advisor: Filip Rezabek, Kilian Holzinger)

Current Developments of IEEE 1588 (Precision Time Protocol) 13
Kilian Rösel (Advisor: Max Helm, Johannes Zirngibl, Henning Stubbe)

Seminar

SmartNICs: Current Trends in Research and Industry 19
Tristan Döring (Advisor: Kilian Holzinger, Henning Stubbe)

Debugging QUIC and HTTP/3 with qlog and qvis 25
Dominik Freiherr von Künßberg (Advisor: Benedikt Jaeger)

Recent Developments in Service Function Chaining 31
Patricia Horvath (Advisor: Kilian Holzinger, Henning Stubbe)

EDNS NSID Option 37
Christian Kilb (Advisor: Johannes Zirngibl, Patrick Sattler)

xdpcap: XDP Packet Capture 41
Stefan Lachnit (Advisor: Sebastian Gallenmüller, Dominik Scholz, Henning Stubbe)

TLS Certificate Analysis 45
Jonas Lang (Advisor: Markus Sosnowski, Johannes Zirngibl, Patrick Sattler)

Corona Warn-App – Design, Development and Privacy Considerations 51
Oliver Layer (Advisor: Benedikt Jaeger)

Precision Time Protocol - Security Requirements 57
Tizian Leonhardt (Advisor: Filip Rezabek, Kilian Holzinger)

Network Coding — State of the Art 63
Florian Stamer (Advisor: Jonas Andre, Stephan Günther)

White Rabbit: High Precision PTP 67
Edward Waterman (Advisor: Max Helm, Johannes Zirngibl, Henning Stubbe)

Intra-vehicular Data Sources 73
Paul Wiessner (Advisor: Filip Rezabek, Kilian Holzinger)

Survey of Mesh Networking Messengers

Simon Blöchinger, Richard von Seck*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: simon.bloechinger@tum.de, seck@net.in.tum.de

Abstract—**A centralized architecture utilizing one or more central servers is used by most messenger applications. The messenger will only work if the server is functioning and a connection is possible. Mesh networking messengers use peer-to-peer connections to exchange messages directly, without the need for central servers.**

A decentralized architecture is more resilient against failures. Mesh networking messengers have privacy benefits as well. This paper analyzes different mesh networking messengers and compares their features.

Briar and Technitium Mesh provide secure mesh networking messaging for their respective platforms Android and Windows. Meshenger implements encrypted local audio and video calls.

Index Terms—**mesh networking messenger, mesh messenger, peer-to-peer messenger, mesh network, peer-to-peer network, instant messaging**

1. Introduction

Most messengers use central servers responsible for storing and exchanging messages. These messengers only work when they have access to the Internet and the central servers are available. In situations where the users cannot connect to the internet, for example in remote locations or when the necessary infrastructure fails, the messenger cannot function. The same is true when the central servers are not available. Even if a direct connection between the clients would be possible, messengers relying on central servers do not work without them.

A connection between peers is called *peer-to-peer*. As shown by Akyildiz et al. in [1], when multiple peers are dynamically interconnected peer-to-peer, this is called a mesh network. Mesh networks often allow routing through the client nodes. Messengers that utilize the mesh networking approach are called mesh networking messengers. These messengers do not rely on central servers but connect to each other directly.

The decentralized mesh architecture does not have a single point of failure. As long as there are enough redundant connections between the devices forming the mesh, no device is essential.

Another advantage of mesh networking messengers is that they are privacy friendly. When there is no central server storing the messages, central data mining is not possible. There is also no chance of server data leaks. Vulnerabilities in the messaging application itself can still exist.

Because mesh networking messengers are inherently attractive to people interested in privacy-oriented messaging, most mesh networking messengers are open-source as well. This allows users to inspect the code of the application they are using themselves to make sure that there are no hidden side effects.

In Section 2 three types of mesh networking systems are introduced and use cases explored.

In Section 3 different mesh networking messenger applications are analyzed and compared.

2. Mesh Based Networks

There are three different types of mesh networking systems [1]. The *Infrastructure Mesh Network* is differentiating between infrastructure and clients. The infrastructure is interconnected in a mesh, the clients are connecting to the infrastructure.

The *Client Mesh Network* only has a single type of node, the client. All clients are connected and pose not only as an end-user device but can also be used to route messages. The nodes in this network are communicating using peer-to-peer connections.

The *Hybrid Mesh Network* combines the infrastructure and the client approach. The clients can access the network both through routers, which make up the infrastructure, and through the other clients.

2.1. Use Cases

Infrastructure Mesh Networks can be used to set up routers on a large scale to provide a connection to the Internet in an area that is too big for a single router. The area is set up with multiple routers in such a way that allows every part of the area to be reached by at least one router. Then the routers automatically and dynamically form a mesh network and route messages between them [1].

As presented by Coulouris et al. in [2], Hybrid Mesh Networks are great for hosting big, immutable files such as video files on a large scale. Since the files are immutable, they can be stored in small parts across a distributed network without worrying about keeping them up to date. For downloading purposes, the parts can be supplied by multiple hosts. This makes a bandwidth problem on the hosters side less likely. After the download, the downloader can become a hoster on his own. This can help to balance out supply and demand for a file.

3. Mesh Networking Messengers

The focus of this paper is the comparison of different mesh networking messengers. The features of Briar and Technitium Mesh are evaluated in depth. Also considered are Meshenger, Serval Mesh/Chat, FireChat and Bridgefy.

3.1. Briar

Briar is an open-source messenger with a strong focus on privacy that uses a mesh networking approach which allows users to privately communicate with each other. It was first released in 2018 for Android [3]. The devices can connect anonymously over the Internet via Tor or locally via Wi-Fi or Bluetooth.

All direct communication using Briar can only happen between contacts. There is no possibility to send messages directly to a non-contact.

3.1.1. Adding Contacts. A nearby contact can be added by exchanging QR codes. Using an already existing communication channel, a contact can be added by exchanging a link. If two users share a common contact, they can be introduced to each other via this common contact.

Ways of reaching each contact are stored locally on the user's device. If a connection via multiple transport mediums is available, they will be used in parallel.

When adding a contact by exchanging QR codes, the *Bramble QR Code Protocol* (BQP) is used [4]. When adding a contact at a distance by exchanging a link or by introduction through a common contact, the *Bramble Rendezvous Protocol* (BRP) is used [5].

Both protocols are similar and serve the same purpose: an initial public key exchange is used for authentication and encryption. As a result of the protocol, both users know how to reach their contact and have a shared secret key. This shared secret key is used to derive other keys from it, which then are used to encrypt the communication. Both protocols use the Diffie Hellman key agreement function and are secure, even if an attacker can read, modify, delete and insert traffic on all transports at will, as long as the initial public key exchange is not modified.

During the BQP, a commitment to a public key and information on how to be reached using Bluetooth and Wi-Fi, the short-range transports that are supported by Briar, is shared using a QR code. The participants establish an insecure connection and share ephemeral public keys. Using these public keys, a secure connection gets established. Then the participants agree on a shared master key via the secure connection.

At the beginning of the BRP, the only information known to each participant is the public key of the other. Both parties generate pseudo-random contact details for themselves and the other's endpoint using the shared public keys. A shared secret key is also derived from the public keys. For the next 48 hours both peers listen on their network endpoints. If no connection happens within 48 hours, the rendezvous is considered to have failed. If a connection can be established, the participants exchange long-term contact details.

3.1.2. Methods of Communication. Briar allows for a few different methods of communication between hosts: *private chats*, *private groups*, *forums* and *blogs* [6]. Each method of communication can use any of the transport mediums that are available.

Private chats allow users to chat with one of their contacts.

Private groups are created by one user. This user is the owner of the private group. Only the owner can add his own contacts. If the owner leaves the private group, the private group will be dissolved.

Forums are similar to private groups with the exception that every participant is equal. Everyone can invite their contacts and the forum will not be dissolved if the original creator leaves.

It is possible for two users who are not each other's contacts to be part of the same private group or forum.

The blog behaves similar to a broadcast to all contacts. Anything that gets published in a blog can be read by and commented on from all contacts.

New messages in private groups and forums are shared with all contacts that are in the same private group or forum. This allows private group or forum updates to spread to people without a direct connection to the sender of the update. Note that this sharing only ever happens with contacts. No sharing happens to users who are in the private group or forum, but not a contact.

To receive a message, the sender and the receiver need to be connected with each other. Sending a message that can get received even when the sender is offline is not possible. If the receiver is not online when a message gets sent, the sender periodically tries again until the message was successfully sent and received. Briar uses a background task to send and receive messages.

3.1.3. Bramble Transport Protocol. The transport of data between two parties in the Briar application is done using the *Bramble Transport Protocol* (BTP), which provides a secure channel ensuring confidentiality, integrity, authenticity and forward secrecy across a wide range of underlying transports. The protocol is difficult to distinguish from other protocols. To further hide the use of Briar, techniques like traffic morphing can be used. It is suitable for delay-tolerant networks and can even be used on transports with very high latency, such as sending a physical storage medium through mail. The BTP uses rotating keys to encrypt and decrypt the message stream. [7].

3.1.4. Conclusion. Briar implements secure mesh text messaging for Android. It is able to establish encrypted connections via Wi-Fi, Bluetooth and via the Internet using Tor. Briar is open-source. It is of limited usefulness when it comes to communicating with a group of local strangers in case of infrastructure failure, because communication can only happen between contacts.

3.2. Technitium Mesh

The mesh networking messenger Mesh by Technitium is another messenger that provides peer-to-peer communication. Its alpha version was released in 2019 for Windows. Text messaging and file transfers are possible. It is a direct successor to the Bit Chat project. Most of Bit Chat's design is found in Mesh as well [8], [9].

3.2.1. Technitium Bit Chat. The concept for Bit Chat was invented in 2011. It takes many of BitTorrent's concepts with the goal of instant messaging instead of file sharing. Connections are made using existing BitTorrent trackers, which are centralized servers providing information about the location of files.

Instead of the location of files, the trackers are storing who is part of which channel. This is done by storing the IP addresses of all participants together with a unique infohash that identifies the channel. After receiving the IP addresses of the other participants, a direct authenticated connection can be established via IP. Public key cryptography is used to achieve authentication and confidentiality [10].

Bit Chat requires a central user profile registration based on email. Mesh does not use central user profiles. Instead users are identified with a user ID generated from their RSA key pair.

Mesh also removes the BitTorrent trackers and replaces them with *Distributed Hash Tables* (DHT). Using the BitTorrent trackers can lead to connectivity problems since some ISPs block BitTorrent traffic.

Mesh users can choose to use an *anonymous profile* instead of a *peer-to-peer profile*. Anonymous profiles use Tor onion services to accept inbound requests. For every login a new onion domain name is used to prevent tracking. Communication between anonymous and peer-to-peer profiles is possible. Connections using an anonymous profile are still peer-to-peer connections [9].

3.2.2. Methods of Communication. Mesh provides two different options for communication: private chat and group chat [11].

Using the user ID and an optional password, a private chat can be initiated. These need to be transmitted using a secure channel not provided by Mesh.

To initiate a group chat, a group name and an optional password is needed. The name and password have to be distributed to participants. This can be done through private chat or any other secure external channel.

3.2.3. Protocols. Mesh uses the symmetric-key algorithm AES-256. To share the key with all participants, the Diffie-Hellman key exchange function is used. During the key exchange, the user IDs of the participants are verified using RSA. To provide perfect forward secrecy, a new key exchange is done periodically. Message authenticity is ensured through the use of *HMAC-SHA256*. The local data stored on the user's devices is encrypted using a secure key derived from the user's password by the *Password-Based Key Derivation Function 2* (PBKDF2). In Mesh's implementation, PBKDF2 uses the pseudorandom function HMAC-SHA256.

When a new channel is created, the network ID of the channel is used to uniquely identify the channel. In a group chat, the network ID is the hash of the group name in combination with the group password. In a private chat, it is the hash of the user IDs of the participants in combination with the password. This hash is then stored together with the IP addresses of the channel participants in a *Distributed Hash Table* [11].

3.2.4. Conclusion. Technitium Mesh implements encrypted peer-to-peer text messaging and file transfer in pairs and in groups. Communication is possible locally via LAN/Wi-Fi and globally using IP or using Tor onion services. Mesh is released on Windows and is open-source. Mesh does not implement mesh network routing functionality for messaging.

3.3. Other Messengers

While Briar and Technitium Mesh are analyzed in detail, other messengers are considered as well.

3.3.1. Meshenger. Meshenger is an open-source peer-to-peer messenger for audio and video communication released on Android. The project started in 2018 as part of the Freifunk initiative [12]. Version 1.0 got released in 2018 [13], followed by a repository change [14].

Meshenger supports encrypted audio and video calls in local networks with contacts. Text messaging is not supported. Communication via Bluetooth or via the Internet is also not supported.

To establish a connection with a contact, primarily local unicast IPv6 addresses are used. Other IP addresses or DNS names can be used as well. Meshenger does not use mesh routing for audio and video calls.

3.3.2. Serval Mesh/Chat. The Serval Project has the goal to help the geographically, financially or otherwise unfortunate.

Serval Mesh is an open-source Android app that provides secure mesh networking text messaging, file sharing and audio calls using Wi-Fi. Audio calls only work under good conditions. Group or broadcast messaging is not supported.

Serval Chat is an iOS app providing secure text messaging using Apple's proprietary peer-to-peer wireless network. Group and broadcast messaging are supported. Serval Chat is not open-source [15].

Communication between Serval Mesh and Serval Chat is not possible. While the project in general has interesting and unique features, there has been no development since 2018 for Serval Mesh/Chat. Serval Mesh is not available in Google Play anymore and Serval Chat is also not available in Apple App Store [16].

3.3.3. FireChat. FireChat was a mesh networking messenger that got popular during protests in Iraq and Hong Kong in 2014 [17], [18]. It has since been discontinued and the official website is not available anymore [19].

3.3.4. Bridgefy. Bridgefy is another mesh networking messenger. It is released for Android and iOS [20]. A Bluetooth connection is used to connect the devices. It got used in Hong Kong in 2019 [21].

While there are code samples for developers, Bridgefy is not open source [22]. Bridgefy currently still has security issues and is not able to provide secure messaging [23].

	Briar	**Mesh**	**Meshenger**	**Serval Mesh/Chat**	**Bridgefy**
Communication	Contacts	Chat rooms	Contacts	Private & broadcast (iOS)	Private & broadcast
Bluetooth	Yes	No	No	No	Yes
LAN / Wi-Fi	Yes	Yes	Yes	Yes (Android)	Yes
IP	No	Yes	Yes	No	No
Tor	Yes	Yes	No	No	No
Built secure	Yes	Yes	Yes	Yes	No
Platform	Android	Windows	Android	Android, iOS	Android, iOS
Open-source	Yes	Yes	Yes	Yes (Android)	No
Text	Yes	Yes	No	Yes	Yes
Audio	No	No	Yes	Yes (Android, limited)	No
Video	No	No	Yes	No	No
File sharing	No	Yes	No	Yes (Android)	No

Figure 1: Mesh networking messengers - comparison

4. Conclusion

Briar and Technitium Mesh implement secure mesh networking text messaging for their respective platforms Android and Windows. They implement encrypted peer-to-peer communication both over local and global transport mediums.

Briar only allows communication with contacts, which limits its usefulness in communicating with local strangers, for example in case of a local infrastructure failure.

Mesh allows to connect to an open local LAN chatroom without a password. File sharing is also possible. Mesh does not implement mesh network routing functionality for messaging.

Meshenger allows for secure audio and video communication in local networks. Meshenger also does not implement mesh networking functionality for audio and video calls.

A big weakness of these mesh networking messengers is that they are only able to communicate with other devices using the same application. Since there is a variety of mesh networking messengers that come and go, a messenger is not of great use if there are not enough users.

If a standard would get introduced for mesh networking messaging, the usefulness of these mesh networking messengers might rise. But because these messengers use different protocols and have different design goals, it is unlikely that messengers supporting inter-messenger communication will become common.

References

[1] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," *Computer networks*, vol. 47, no. 4, pp. 445–487, 2005.

[2] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*. pearson education, 2005.

[3] M. Rogers, E. Saitta, T. Grote, J. Dehm, B. Wieder, and N. Alt, "Briar Release," https://briarproject.org/news/2018-1.0-released-new-funding/, 2018, [Online; accessed 03-October-2020].

[4] ——, "Bramble QR Code Protocol," https://code.briarproject.org/briar/briar-spec/blob/master/protocols/BQP.md, 2019, [Online; accessed 21-November-2020].

[5] ——, "Bramble Rendezvous Protocol," https://code.briarproject.org/briar/briar-spec/blob/master/protocols/BRP.md, 2019, [Online; accessed 21-November-2020].

[6] ——, "Briar Manual," https://briarproject.org/manual/, 2016, [Online; accessed 03-October-2020].

[7] ——, "Bramble Transport Protocol," https://code.briarproject.org/briar/briar-spec/blob/master/protocols/BTP.md, 2019, [Online; accessed 21-November-2020].

[8] S. Zare, "Technetium Mesh," https://mesh.im/, 2019, [Online; accessed 21-November-2020].

[9] ——, "Technetium Mesh Release," https://blog.technitium.com/2019/12/technitium-mesh-released.html, 2019, [Online; accessed 03-October-2020].

[10] ——, "Technetium Bit Chat Release," https://blog.technitium.com/2011/07/bitchat-peer-to-peer-instant-messaging.html, 2011, [Online; accessed 03-October-2020].

[11] ——, "Technetium FAQ," https://mesh.im/faq.html#q15, 2019, [Online; accessed 21-November-2020].

[12] D. Dakhno, "Meshenger Original Repository," https://github.com/dakhnod/Meshenger, 2018, [Online; accessed 03-October-2020].

[13] ——, "Meshenger - P2P Local Network Messenger - Final Update," https://blog.freifunk.net/2018/08/14/meshenger-p2p-local-network-messenger-final-update/, 2018, [Online; accessed 03-October-2020].

[14] ——, "Meshenger 2.0 Repository," https://github.com/meshenger-app/meshenger-android, 2018, [Online; accessed 03-October-2020].

[15] P. Gardner-Stephen, "The Serval Project," http://servalproject.org/, 2011, [Online; accessed 03-October-2020].

[16] ——, "Serval Project Blog," https://servalpaul.blogspot.com/, 2011, [Online; accessed 03-October-2020].

[17] A. Hern, "Firechat Updates as 40,000 Iraquis Download 'Mesh' Chat App in Censored Baghdad," https://www.theguardian.com/technology/2014/jun/24/firechat-updates-as-40000-iraqis-download-mesh-chat-app-to-get-online-in-censored-baghdad, 2014, [Online; accessed 03-October-2020].

[18] A. Bland, "FireChat - The Messaging App That's Powering the Hong Kong Protests," https://www.theguardian.com/world/2014/sep/29/firechat-messaging-app-powering-hong-kong-protests, 2014, [Online; accessed 03-October-2020].

[19] O. Garden, "FireChat," https://www.opengarden.com/firechat, 2014, [Online; accessed 03-October-2020; not available].

[20] "Bridgefy," https://bridgefy.me/, 2020, [Online; accessed 03-October-2020].

[21] J. Koetsier, "Hong Kong Protestors Using Mesh Messaging App China Can't Block: Usage Up 3685%," https://www.forbes.com/sites/johnkoetsier/2019/09/02/hong-kong-protestors-using-mesh-messaging-app-china-cant-block-usage-up-3685/, 2019, [Online; accessed 03-October-2020].

[22] "Bridgefy Code Samples," https://github.com/bridgefy, [Online; accessed 03-October-2020].

[23] "Bridgefy's Commitment to Privacy and Security," https://bridgefy.me/bridgefys-commitment-to-privacy-and-security/, 2020, [Online; accessed 03-October-2020].

Collaborative SLAM over Mobile Networks

Han My Do, Marton Kajo[*]
[*]*Chair of Network Architectures and Services, Department of Informatics*
Technical University of Munich, Germany
Email: han-my.do@tum.de, kajo@net.in.tum.de

***Abstract*—Simultaneous Localization and Mapping (SLAM) in general is a problem that is key to the path planning of autonomous robots. The tasks of generating a map of an unknown environment while keeping track of its position are accomplished more accurate in a system with multiple robots. Such collaborative SLAM systems can be found in modern warehouses, where the logistics chain is performed by Automated Guided Vehicles (AGVs). With example industry use cases, this paper gives an overview on the main topics of collaborative SLAM and analyzes the different approaches to its components, architecture and communication. SLAM communication methods over mobile networks are analyzed and provide insights to the synergy potential 5G and SLAM has to offer.**

***Index Terms*—SLAM, Simultaneous Localization and Mapping, mobile networks, collaborative, 5G, logistics, visual, autonomous**

1. Introduction

Due to the need for more automated and flexible logistics systems, Automated Guided Vehicles (AGVs) are gaining foothold in the industry and enable a more efficient way of modernizing the industry. 5G has a lot of industrial focus with its Ultra Reliable and Low Latency Communications (URLLC) and broadband use-cases, and is the perfect fit for the communication solution for these AGVs.

1.1. Visual SLAM

For autonomous robots to function and navigate in a secure and robust way in a highly complex environment such as warehouses, the topic of Simultaneous Localization and Mapping (SLAM) is of high interest. SLAM involves the problem of simultaneously determining the position of a robot and the generation of a map of its environment. The interdependence of those two is made clear when keeping in mind that for path planning of a robot, not only its own position and orientation, but also obstacles such as humans and other robots play a role [1]. Therefore, to generate the map of the robot's environment different kinds of sensors are used.

Visual SLAM describes those systems that use cameras as the only exteroceptive sensor [2]. Cameras are lightweight, inexpensive and offer a lot of visual information. Thanks to the fast development and improvements of visual SLAM, as well as the growing computer performance, cameras have become an increasingly popular sensor for SLAM applications. Especially since inertial measurement units were integrated into visual SLAM entities, the system profits from better robustness and accuracy thanks to the additional inertial information such as acceleration and angular rate [3].

Typically, a visual SLAM system has two task areas. The front-end takes care of processing the image and extracting features to match and track those across different video frames. The back-end computes the camera poses and 3D coordinates. This geometric computation is often done with a filter or a nonlinear least squares optimizer. Further important SLAM issues are loop closure, relocalization, outlier rejection and the architecture [3].

Figure 1 shows the categorization of visual SLAM tasks in front-end and back-end.

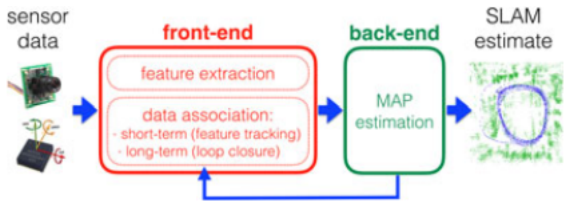


Figure 1: The two task areas of visual SLAM [1]

As SLAM research over the years has mainly developed visual-inertial algorithms, visual SLAM represents the state-of-the-art [1]. Nonetheless, we will take a short look at other systems with a different main sensor than video cameras.

1.2. Nonvisual SLAM

Besides cameras, other exteroceptive sensors in SLAM systems include sonars, range lasers, and global positioning systems (GPS). Even though sonars and range lasers are very precise and offer dense information of the environment's setting, they are limited for automated robots in logistic facilites, since they are heavy and have large pieces of equipment which makes them unsuitable for aerial or smaller robots [2]. Furthermore, they do have difficulties with highly cluttered environments, which makes it difficult for correctly mapping warehouses or comparable facilites. GPS sensors face similar complications when the signal is not available indoors at all times.

To ensure an accurate and robust estimation of the position of the robot, it is of advantage to combine the gained information from multiple exteroceptive sensors and proprioceptive ones. The latter are for example encoders, accelerometers and gyroscopes which measure velocity, position and acceleration [2].

After this short introduction to the differences of visual and nonvisual SLAM, the focus will be on collaborative SLAM which enables the interaction between multiple entities and their environment.

2. Collaborative SLAM

In the first chapter we introduced the hardware specifications of a SLAM system and described the different sensors needed for an autonomous robot. In this part of the paper the focus will be on the software side of the system. Different aspects and methods to solve the collaborative SLAM problem are presented.

What makes multiple-agent SLAM more complex than single systems is that robots must process available data to construct a consistent global map while simultaneously localize themselves within the map [4]. In the following we provide an overview on researched approaches to collaborative SLAM.

2.1. Key Components

SLAM exploration and mapping tasks are fulfilled faster and with more accuracy by multiple robots than by just one. This also allows for a heterogenous team of robots with each one having its own specialization [5].

Another advantage is that the whole system is more robust in a distributed system due to the fact that failure of one robot does not crash the whole system [4].

However, the main difference between the elements of single-agent SLAM and multi-agent SLAM is the processing of data from multiple participants [3]. Otherwise, the building blocks are the same to a single SLAM system. In the following we will shortly present specific elements of a multi-robot collaborative SLAM.

There are two prominent approaches to pose estimation: key-frame based and filter-based methods. Key-frame based methods are more suitable and easier to implement for sharing information among the different robots. Depending on the architecture, the key-frames are sent to the server and can be downloaded by the participants. This means that every agent has access to key-frames produced by the others for their own pose estimation.

After synchronization of the participants' video frames a pose estimation between several robots is possible. This approach considers static points as well as moving points and enables a robust localization even with moving obstacles in the environment [3]. Figure 2 shows the method of camera synchronized pose estimation. Even though the moving object blocks the view of Camera A on the static background, Camera B is able to detect the background as well as the moving object.

The key-frame based approaches proved to be an efficient way for visual SLAM systems as it separates computation of real-time pose estimation and the complex mapping tasks. Pose estimation and mapping are calculated rotationally and can therefore resort to the previously calculated results. A key-frame contains the detected feature points and their corresponding coordinates. Aligning those data from the previous and current key-frame allows

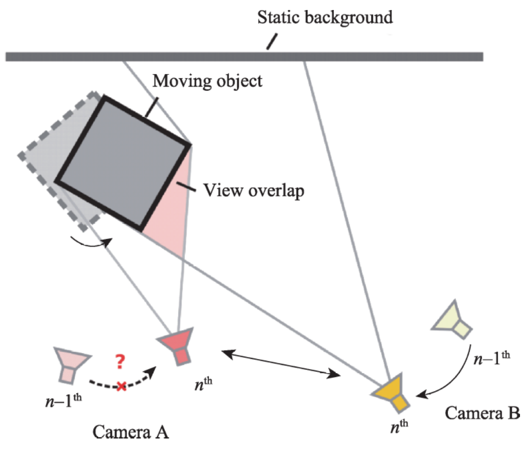


Figure 2: Synchronized pose estimation in collaborative SLAM [3]

for the localization of the agent. The mapping task is solved by triangulation of the matching feature points between different key-frames. On a collaborative level the mapping is also done by using image information captured by the different cameras to generate map points.

The filter-based approaches use the Extended Kalman Filter to estimate the camera pose through iteration. The state vector also includes the 3D coordinates of landmarks in the environment. With every iteration these coordinates and the camera motion are updated and lead to high computation load with an increasing number of landmarks.

Another important task is loop closure, which detects already visited areas to update the map and correct accumulated inaccuracies. Loop closure is done by detecting the overlaps in some specific regions among multiple individual maps for fusion. A globald descriptor is used to check the similarity of two images to detect the overlap. Otherwise, collaborative loop closure follows the same pipeline as in single-agent SLAM algorithms. Such cooperation among the multiple cameras result in more accurate and robust estimations [3].

2.2. Architecture

A major challenge of collaborative SLAM is to distribute the time-consuming computational tasks to different agents with limited onboard computational resources [3]. This involves designing complex distributed algorithms to solve those computational tasks appropriately.

It is also important to consider the communication load to design the distributed algorithms and consider the strict bandwidth constraints when applying the decentralized architecture [3].

Figure 3 illustrates four main issues that arise in the context of data handling in collaborative SLAM.

The first topic is Data Communication. The SLAM system has to provide communication channels that allow for information sharing between the multiple agents. Central factors are bandwidth and communication network coverage.

The Data Sharing can span from exchanging raw sensor data to refined data. Measurements of exteroceptive and proprioceptive sensors are understood as raw information, while the refined data are those that are processed

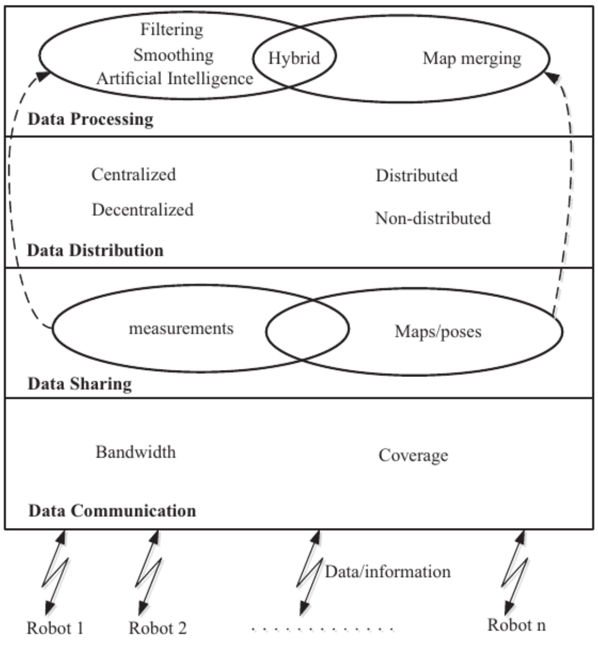


Figure 3: Issues in collaborative SLAM [4]

through filtering or optimization. Examples for processed data are environment maps or robot poses. Even though raw data offers more flexibility, the prerequisites for functioning are high bandwidth and stable communication between the entities. Computational power is essential as well. While processed information does not have the same amount of requirements and even reduces redundancy, the results are determined by the maps' quality.

Data Processing covers a wide range of methods and algorithms with filter-based and key-frame based approaches as their foundation. The choice of the data processing method is again dependent upon several factors such as processing power, type of sensor data and memory space of the entities [4].

For the distribution of data it is possible to deploy a centralized or decentralized architecture. Most collaborative systems use a central powerful server to collect all data and to process the computational-intensive tasks such as map optimization, loop detection and pose graph optimization for each entity [3]. This entity is also responsible for answering requests and providing information. This architecture has the disadvantage that the functioning of the whole system is dependant on the one server to never fail and to always be reachable. It also has to scale to the number of participating robots in computation performance and bandwidth. Decentralized systems do not suffer from such bottlenecks [5], but are much more difficult to deploy as the computational tasks are performed by more than one robot.

2.3. Communication over Wifi and 5G

To fulfil the needs of communication in decentralized SLAM systems, it is advisable to take a look at wireless networks such as Wifi and 5G as they have suitable properties for the wireless and real-time transmission of huge amount of data. Not only is Wifi sufficient for communication, but Wifi sensing can help with the SLAM problem, too. Due to the wide spread of Wifi Access Points in urban environments and the availability of Wifi radios on most robots or mobile devices, [6] and [7] propose to incorporate Wifi sensing into visual SLAM algorithms. A general method for the integration of Wifi into visual SLAM is shown in Figure 4. Similar approaches of using the signal strength of Bluetooth and LTE can be found in [8].

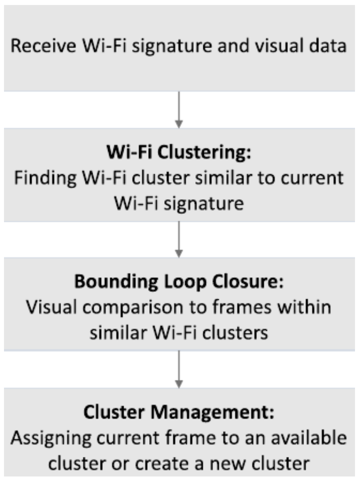

Receive Wi-Fi signature and visual data
Wi-Fi Clustering:
Finding Wi-Fi cluster similar to current Wi-Fi signature
Bounding Loop Closure:
Visual comparison to frames within similar Wi-Fi clusters
Cluster Management:
Assigning current frame to an available cluster or create a new cluster

Figure 4: Pipeline of Wifi integration into visual SLAM, proposed by [6]

The use of 5G for SLAM methods, for example for the estimation of angle and delays of mmWave channels [9] or for the use of available multipath signals from landmarks to accomplish the mapping task [10], is promising as many of the required steps of Wifi integration as shown in Figure 4 can be omitted.

Three usage profiles are defined by the International Telecommunication Union for the International Mobile Telecommunications 2020 requirements for 5G networks. The three key capabilities are enhanced Mobile Broadband (eMBB), massive Machine-Type Communications (mMTC), and Ultra-Reliable and Low-Latency Communications (URLLC) [11].

URLLC lists specifications for seamless interaction between robots in real-time applications. Requirements are robustness, high bandwidth, and low latencies. With such significant advantages, 5G allows for reliable wireless and real-time transportation of high amounts of data, which accelerates the performance and functionalities of mobile robots. In addition, 5G allows to reserve sections of the network with a guaranteed Quality of Service [12].

User localization with 5G offers benefits such as high coverage, high accuracy, low energy consumption and scalability. The improvements in localization of users are possible due to the high concentration of base stations, device-to-device communication and mmWave technology [13]. For users, such as autonomous vehicles in complex settings, one crucial topic is accurate positioning. 5G in combination with collaborative SLAM approaches provide an optimal basis for the positioning task.

3. Use Cases for SLAM

Lastly, we explore applicable use cases of SLAM and show potential future research topics. The approaches to visual SLAM in a collaborative way over wireless networks pave the way to some interesting use cases which are described in the following.

3.1. Logistics

The use of Automated Guided Vehicle (AGV) or autonomous Micro Aerial Vehicles for the automation of modern logistics systems is quite common. Traditionally, SLAM is based on laser reflector and triangulation which is dependant on an established and static structured working environment, which is rarely the case in warehouses. With multi-agent visual SLAM algorithms, entities can localize themselves automatically and trace their path accurately in a dynamic and unstructured environment. Visual SLAM improves working efficiency, system flexibility and reduces constructing cost [14].

3.2. Autonomous Driving

Related to AGV is the use case of self-driving cars. An important aspect is the way data and communication are handled in a centralized or decentralized way. The availability of internet connection in the vehicle also plays a central role. The aspects of real time updates and offloading critical processing aspects onto the cloud spark discussions about safety. A future field of work is the architecture design of software which should be able to handle data flows and to segment updates [15].

3.3. Augmented Reality

Augmented Reality (AR) applications can benefit from SLAM systems, because the gained information enriches the AR experience from a technical aspect. AR systems face important technical challenges which come down to the need of specific information that SLAM can offer. One type of information needed is the current view of the real environment that is supposed to be augmented, while others are the shape of the virtual object and its location within the real world. When combined with other sensors or tracking systems, well-designed user interaction and system design, it is possible to widen the extent of AR to any environment [16]. Such an environment can also be warehouses where AR can be used for information exchange between teams and for prevention of errors and support.

4. Conclusion

In this paper we briefly described the differences between visual and nonvisual SLAM and went on to analyse the characteristics of collaborative SLAM. Focus was also set on the advantages of a decentralized architecture of multi-agent systems and their communication over wireless networks. Based on our findings we referred back to our introductory example use case of collaborative visual SLAM in logistics which was followed by further related use cases of autonomous driving and AR.

Even though visual SLAM in general, as well as in a collaborative way is already discussed thouroughly in existing literature, there is space for further research in the topics of decentralized multi-robot SLAM over wireless networks. Especially 5G in combination with collaborative SLAM is not yet comprehensively researched, but offer many improvements and great potential for synergy as evaluated in chapter 2.3. of this paper.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard, “Simultaneous localization and mapping: Present, future, and the robust-perception age,” *CoRR*, vol. abs/1606.05830, 2016. [Online]. Available: http://arxiv.org/abs/1606.05830

[2] J. Fuentes-Pacheco, J. Ascencio, and J. Rendon-Mancha, “Visual simultaneous localization and mapping: A survey,” *Artificial Intelligence Review*, vol. 43, 11 2015.

[3] D. Zou, P. Tan, and W. Yu, “Collaborative visual slam for multiple agents:a brief survey,” *Virtual Reality & Intelligent Hardware*, vol. 1, no. 5, pp. 461 – 482, 2019, 3D Vision. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2096579619300634

[4] S. Saeedi, L. Paull, M. Trentini, and H. Li, “Multiple robot simultaneous localization and mapping,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 853–858.

[5] T. Cieslewski, S. Choudhary, and D. Scaramuzza, “Data-efficient decentralized visual SLAM,” *CoRR*, vol. abs/1710.05772, 2017. [Online]. Available: http://arxiv.org/abs/1710.05772

[6] Z. S. Hashemifar, C. Adhivarahan, A. Balakrishnan, and K. Dantu, “Augmenting visual SLAM with wi-fi sensing for indoor applications,” *CoRR*, vol. abs/1903.06687, 2019. [Online]. Available: http://arxiv.org/abs/1903.06687

[7] J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal, “Efficient, generalized indoor wifi graphslam,” 06 2011, pp. 1038 – 1043.

[8] P. W. Mirowski, T. K. Ho, S. Yi, and M. MacDonald, “Signalslam: Simultaneous localization and mapping with mixed wifi, bluetooth, LTE and magnetic signals,” in *International Conference on Indoor Positioning and Indoor Navigation, IPIN 2013, Montbeliard, France, October 28-31, 2013*. IEEE, 2013, pp. 1–10. [Online]. Available: https://doi.org/10.1109/IPIN.2013.6817853

[9] H. Wymeersch and G. Seco-Granados, “Adaptive detection probability for mmwave 5g slam,” 03 2020, pp. 1–5.

[10] Y. Ge, H. Kim, F. Wen, L. Svensson, S. Kim, and H. Wymeersch, “Exploiting diffuse multipath in 5g slam,” 2020.

[11] Z. Li, M. A. Uusitalo, H. Shariatmadari, and B. Singh, “5g urllc: Design challenges and system concepts,” in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, 2018, pp. 1–6.

[12] S. Ludwig, M. Karrenbauer, A. Fellan, H. D. Schotten, H. Buhr, S. Seetaraman, N. Niebert, A. Bernardy, V. Seelmann, V. Stich, A. Hoell, C. Stimming, H. Wu, S. Wunderlich, M. Taghouti, F. Fitzek, C. Pallasch, N. Hoffmann, W. Herfs, E. Eberhardt, and T. Schildknecht, “A5g architecture for the factory of the future,” in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, Sep. 2018, pp. 1409–1416.

[13] S. Zügner and M. Kajo, “User localization in 5g mobile networks.”

[14] Y. Chen, Y. Wu, and H. Xing, “A complete solution for agv slam integrated with navigation in modern warehouse environment,” *2017 Chinese Automation Congress (CAC)*, pp. 6418–6423, 2017.

[15] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, “Simultaneous localization and mapping: A survey of current trends in autonomous driving,” *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[16] G. Reitmayr, T. Langlotz, D. Wagner, A. Mulloni, G. Schall, D. Schmalstieg, and Q. Pan, “Simultaneous localization and mapping for augmented reality,” *International Symposium on Ubiquitous Virtual Reality*, vol. 0, pp. 5–8, 07 2010.

Time Sensitive Networking - 802.1Qci

Abdalla Mahamid, Filip Rezabek and Kilian Holzinger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: abedkh.mahamid@tum.de, rezabek@net.in.tum.de, holzinger@net.in.tum.de

Abstract—**Time sensitive Networking (TSN) is a task of the IEEE 802.1 group. It provides a real-time communication and a large bandwidth to transfer big amount of data in time that fulfills the TSN communication requirement of end-devices. This technology plays an important role in several industries, for instance, IIoT (industrial internet of things), automotive and self-driving cars, that have to transfer a big amount of data to the targets that must be collaborating simultaneously [1] [2]. There are different standards which can be used to realize this. The main focus of this paper will be about the IEEE 802.1Qci Standard with its definition Per-stream filtering and policing (PSFP). PSFP consists of three main instance tables, first, Stream filter instance table, second, Stream gate instance table and finally, Flow meter instance table (see Figure 1). These tables have a relationship between each other that realize the functionality of PSFP. Furthermore, there are some other applications that their integration with PSFP give solutions for end-device TSN-requirements, for instance, IEEE 802.1Qbv, IEEE 802.1Qav (Section 3.3) in addition to centralized configuration module (Section 3.2).**

Index Terms—**TSN, Time-sensitive Networking, IEEE 802.1, IEEE 802.1Qci, real-time communication, implementation of TSN.**

1. Introduction

Self-driving cars are a part of the future technology that requires high efficiency, secured systems with real-time communication [1]. This technology must avoid the high latency communication to transport multiple data flows in addition to the sensitivity of packet loss. Time-sensitive Networking with its definitions to the Standards IEEE 802.1Qci, IEEE 802.1Qva and IEEE 802.1Qbv are technologies that can provide the simultaneous, safe and secure transfer of data as well as combining time synchronization and transmission scheduling due to the end-points requirements [3]. It is also an important technology to the industry likewise Automotive, to enable optimization of communication in addition to reducing the cost in general [4].

The Standard IEEE 802.1Qci is supplying the per-stream filtering and policing (PSFP), which is a task of the Time-sensitive Networking group IEEE 802.1. Why is TSN important? TSN gives various of benefits for the industry, for instance, large bandwidth, security, interoperability and low latency and synchronization [2]. Technology like automotive, industrial internet of things or self-driving cars have a large amount of data has to be transferred from one point to another point with low latency and secure transfer because they are very sensitive data [2]. The current ethernet technology that the industry used to use is IEEE 802.3u that limited with 100 Mbit/s of bandwidth and half-duplex communication [2]. The TSN will give a solution that provides a high bandwidth transfer and communications [2]. Furthermore, TSN provides security technology that gives the framework higher level of defense, protection and performance. For the interoperability, it uses existing Standards and integrates them with new applications to satisfy the TSN-requirements, in other words, it is no need to develop everything from zero, TSN application can use existing technologies and improve them to fulfill the TSN-requirements [2].

Finally, TSN has various advantages against the current and common Ethernet Standards (802.3) to reduce the latency as well as enabling the synchronization between End-devices. TSN can transfer data taking into consideration the priority and time-requirements while the current Ethernet does not differentiate between critical data (data with TSN requirements) and normal data (without TSN requirements) . Later in this paper it is explained exactly how it can be secured, and which methods are used in order to fulfill the TSN requirements. To sum up, the essential goals of the TSN are the low latency communication (real-time), security and priority for critical flows.

Self-driving cars technology it is an important future technology that needs very secure, efficient, and low latency communication in addition to protection against Denial-of-Service (DoS) attacks [1]. IEEE Standards could not decide if the received data flows are urgent or not. Therefore, new technologies were developed to solve these problems. IEEE 802.1Qci is a TSN substandard of IEEE 802.1Q which is TSN substandard of IEEE 802.1 that provide the per-stream filtering and policing which is given solutions to particular problems. Per-stream filtering and policing filter and scheduled the ingress flows due to discarding the non-essential streams and scheduled high priority streams first. This paper points the workwise of per-stream filtering and policing and how it guarantees secured and low latency communication.

There are 4 sections in this paper. Section 2 explaines and gives a related work, how the Per-stream filtering and policing works. The 3ed Section shows several implementations of 802.1Qci, in addition to definitions for important concepts that will be used in the rest of the paper. Finally, in the last Section, Section 4, the author's conclusion will be presented.

2. Background and Related Work

IEEE 802.1Q Standard technology of layer 2 that make decisions using Ethernet-Headers and not IP-Headers. The current used IEEE Ethernet Standards do not have layer 2 deterministic capability, thus, the intention of IEEE 802.1Q was to supply deterministic flows on standard Ethernet. The traditional IEEE Ethernet Standards give no attention for the sensitive information and their priorities, in addition to no ensure for safety, security or protection in a network [3]. IEEE 802.1Q managed and delivered flows to minimize the transmission time for real-time applications using scheduling and policing to fulfil deferent requirements of deferent applications. A Sub-Standard of IEEE 802.1Q, IEEE 802.1Qci with their definition for "per-stream filtering and policing" is a Sub-Standard that with other Standards/Sub-Standards meets the critical requirements that can realize transmission of frames in a particular and predictable time due to filtering the sensitive information and queuing them taking into consideration their priorities [4].

2.1. Per-Stream Filtering and Policing (PSFP)

IEEE 802.1 Qci defines Per-stream filtering and policing (PSFP) that consist of three instance tables, Stream Filters, Stream Gates and Flow Meters. The relationships between the tables can be seen in Figure 1.

To start with, stream filter instance table. It consists of several components that help determine which frame should be processed, for example a Stream Filter Instance Identifier is an integer value that works as an ID for this stream and its index in this table [5]. Then, Stream Gate Instance Table. It is an instance that contains parameters for each flow. For instance, a stream gate instance identifier, that nearly the same functionally of above example without the "index as position". Stream gate state has two states, OPEN and CLOSE, that determine which flow is permitted to pass through the gate [5]. Lastly, internal priority value specification, that have also two options, the null value and an internal priority value. Each have different functionality with the same goal to determine frame's traffic class [5].

Finally, Flow Meter Instance Table. It is as the Stream Gate Instance Table contains parameters for each flow, that gives them a specification. An important specification called Bandwidth Profile Parameter and Algorithm. (For more information see [5]).

To sum up how it exactly works, there is an example. The Figure 1 shows, there is a flow ingress (input) that passes to the *Stream Filter Instance Table* where it have several flows. Each flow has multiple attributes, Stream (flow) ID, Priority, Gate ID and Meter ID. The stream filter table with its application can identify each flow from its unique ID, knows its priority level and which Gate it should be passed to. After taking into consideration the priority, a flow will be forwarded to *Stream Gates Table* which have different gates. Each Gate is specified with a unique Gate ID. With a specific application that provides the ability to match each flow with its correct Gate ID the flow is passing to the matching Gate. Each gate in the

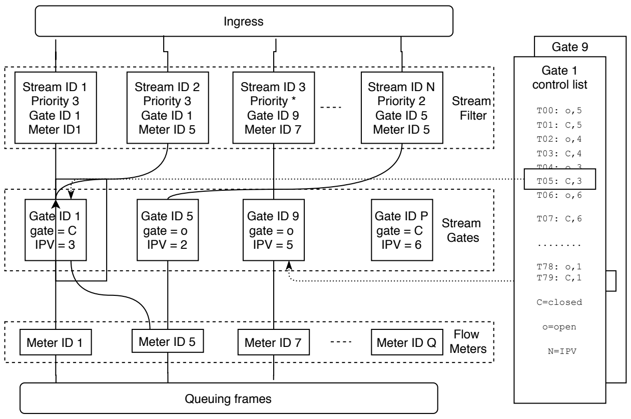


Figure 1: Per-stream filtering and policing [5]

Stream Gate Table has attributes, for instance, Gate ID, gate, internal priority value (IPV) etc. Gate ID is a unique ID that the Stream Filter Table use to match its flow with the correct Gate. The attribute gate is like a status for a gate, with two status, OPEN (o) and CLOSE (c). There is also a *Gate Control List* that is important to control the status of a gate and to update it.

Further, IPV gives the priority of a flow that in this gate. The flow is in the Stream Gate Table and to forward it, the gate controls the status to know if is allowed to let the flow pass or not. If the gate status is OPEN, then the flow allowed to pass to the next table, but if the status is CLOSE then it is not allowed to pass. When the flow is passed, each flow in the Flow Meter Table has also attributes (parameters) that are as specified in *Bandwidth Profile Parameters and Algorithm*. Then the flow will be passed to queue it according to its attributes and priorities. The algorithm that used to schedule the frames is similar to the schedule method of IEEE 802.1Qbv. (more info [5]).

3. Implementations

Before we dive into the details, here some important terms that will make the rest of the explanation more understandable.

3.1. Definitions of Important Components

- **TSN flow**: describes the time-sensitive communication between end devices. Each stream (flow) has a different time requirement that it gives no concessions for its right in strict transmission time [4].
- **End devices**: End devices are the hosts or the source and the destination nodes in our network that the TSN flows weillcbe transmitted between them. Each of these devices has to run an application that requires deterministic communication [4].
- **Bridges**: or "Ethernet Switches" are special switches that capable to transfer or receive frames a TSN flow taking into consideration their schedule and priorities. In other words, the TSN switches should have the ability to forward the

frames on a schedule and receive frames according to a schedule [4].

- **Central Network Controller (CNC)**: It can be defined as a proxy for the Network and the Control Application, where they are needing deterministic communication. The TSN frames are simply transmitted on the schedule defined by the CNC. In other words, it is an application, which provides configuration frames for TSN bridges. These frames are response to TNS stream requirements that received from the Centralized User Configuration (CUC). This application gives it the vendor of the TSN bridges [4].
- **Centralized User Configuration (CUC)**: CUC is an application that communicates with the Central Network Controller (CNC) and End-devices. CUC makes requests to CNC for TNS flows, where each flow has deference requirements. In other words, CUC is an application that receives requests from End-devices and then CUC will transfer the configuration flows to the CNC in the Network for processing [4].

3.2. Security Policies

IEEE 802.1Qci avoids traffic overload condition, that impact the bridges and the end-devices on a network, that is mean is improving the robustness of a network, for instance, daniel-of-Service (DoS) attack, error through streams transmission or likewise if we receive a flow that is not in the schedule time period then it is dropped [6].

After a while had IEEE 802.1Qci a progress. The source states that little progress has been made to connect the standard with existing industrial security systems and architectures [7].
IEEE 802.1Qci Standard is relatively new addition to IEEE 802.1Q Standard, The source states that 802.1Qci does not define how specific policies are created and deployed. Furthermore, there are not a lot of contribution to give explicit ways how the security systems can be deployed and employed, or how the filtering rules are dynamically updated as the Internet and Networks scaling up [7]. TSN networks apply the centralized configuration module, Central User Configuration (CUC) and Central Network Controller (CNC) [7].

How these Components work together? The CNC is a software program works on a costumer own network or premises that communicate with the bridges (a network's component) and controls it. CNC has two principle responsibilities, First, it resolves routes and scheduling TSN flows, second, it configures the bridges for TSN operations. The CNC communicates with the CUC to receive the communications requirements that a network must provide, then the CNC processes all the communications requirements to determine the routes and to schedule the end-to-end transmission for each TNS flow. CNC provides a unique identifier for each flow (include MAC-Address) to help the bridges without doubt identify each flow. Finally, CNC transfers all these processed data flows to bridges the premise [4].

The progress that happens is to integrate the 802.1Qci Standard security system with the Centralized Configurations module (CCM). The idea is to centralize the policies in a policy server to collect the global security policies for the network to dynamically and automatically update them due to the requirements. In the following points it will be explained how the 802.1Qci Standard is integrated with the CCM.

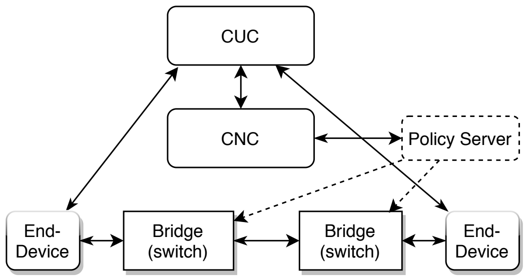


Figure 2: Centralized Configurations module (CCM) [7]

The Policy Server communicates with the CNC.Iin addition, the identifications of TSN endpoints should be imported as objects, besides, the route of the flow should be in a secured mode. Then the Policy Server should be provided on configurated global security policies that are required to be integrable to integrate with the configured IEEE 802.1Qci policies and to enable the end-devices to deal with them. The bridges should be provided with the security system and the policies. For this reason, the Policy Server should enable to update the security system beside to the policies when the requirement for a route change from CNC that received new information and requirements from the CUC. At the same time the control mechanism Quality of Service (QoS) is also should be updated when the bridges receive new flows. The CNC is allowed to deal with per-flow shaper but is not allowed to handle with QoS. Here come the Policy Server to deal with QoS and to adjust the QoS configuration taking into consideration its buffer size and the current number of on-board TSN flows without any meddling from human resource. In general, the Policy Server communicate with the bridges on a network as well as with a CNC on the same premise that configurate the communication requirements for each flow. [7]

3.3. Queuing Frames

After a flow passes all the levels from the ingress till the flow meter, it gets queued taking into consideration all the communication requirements that the End-devices required. To be queued there is an important component that manage this process called Credit Based Meter (CBM). To explain CBM the Credit Based Shaper (CBS) should be referred.

CBS is defined in IEEE 802.1Qav that have two important concepts, $idleslope$ and $sendslope$. $idleslope$ is a definition of the reserved Bandwidth and $sendslope$ is a definition of the Bandwidth subtracted from reserved Bandwidth. CBS has an important role in the TSN network to keep the maintain of a reserved bandwidth [1]. If two End-devices want to communicate, for security reasons the route should be protected and reserved for the sent flow. This reservation takes part from the bandwidth (or the whole, depend on the communication requirements). To control this reservation, it should be controller

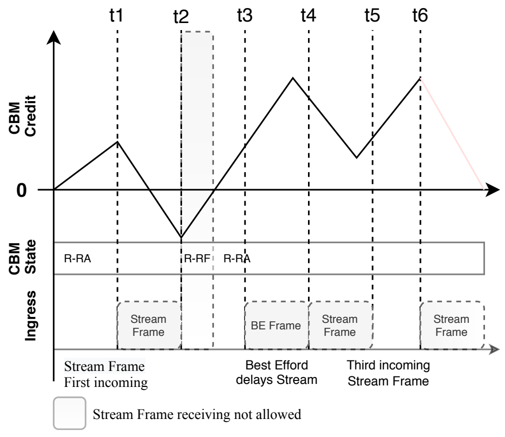

t1
t2
t3
t4
t5
t6
CBM Credit
0
CBM State
R-RA
R-RF
R-RA
Ingress
Stream Frame
Stream Frame
BE Frame
Stream Frame
Stream Frame
Stream Frame First incoming
Best Efford delays Stream
Third incoming Stream Frame
Stream Frame receiving not allowed

Figure 3: Credit Based Metering example [1]

that manages all the traffic flows. CBS is the method that has been defined to solve this problem. It shapes the traffic flows depending on the reservation Bandwidth that gives information about the maximum interval and size of flow frames to that allow to enter the network medium [1].

CBM is based on the credit value of the CBS and others. It takes the two Slopes, $idleslope$ and $sendslope$. It also contains further parameters, maximum burst size parameter ($Burst_{max}$), stream frame sending duration ($T_{duration}$), frame size (FS_{stream}) port bandwidth (B), Ethernet inter frame gap (T_{ifg}) and the maximum credit value ($Credit_{max}$). $Burst_{max}$ is a parameter that gives the number of the allowed streams frame to income burst. $T_{duration}$ defined as following [1]:

$$T_{duration} = \frac{FS_{stream}}{B} + T_{ifg}$$

$T_{duration}$ is important to calculate the $Credit_{max}$ that defined as following [1]:

$$Credit_{max} = sendslop \cdot T_{duration} \cdot (Burst_{max} - 1)$$

In addition to all the definitions, CBM can be in two different statuses R-RA and R-RF.
R-RA = RUNNING RECEIVING ALLOWED
R-RF = RUNNING RECEIVING FORBIDDEN

As expected, the status at the beginning is R-RA with credit 0 that will be changed depending on $idleslope$. The credit still increasing till a flow frame is incoming or the credit hit the maximum then stopped. If it stopped because of $credit_{max}$ reached, then the credit value stays as it is till a frame is incoming . As the credit in R-RA can increase, also it can decrease. $sendslope$ is responsible for decreasing the credit for receiving duration of a flow. Also, if it decreased and still has positive value, still acting normal like before, but if it crosses the zero to the negative value, then immediately the status will be changed to R-RF. In the status R-RF no frames from now allowed to be queued and will be dropped. This will be changed if the credit again increases to positive value to change it to R-RA. This will happen by $idleslope$ if an incoming frame was dropped [1].

4. Future Work

TSN is the future to realize synchronization and simultaneous communication to enable future ideas as self-driving cars the possibility to become true. TSN will be improving and the standards will be implementing in applications. Credit Based Meter (CBM) also will be implementing and testing, where the environment is simulated to analyze and collect more information about the efficiency, performance and the maximum burst configuration [1]. Furthermore, it will be analyzing how this technology integrated with other TSN traffic shaper concepts and how it deals with them in this simulated network [1].

5. Conclusion

Today's industrial requirements are above the current standards Ethernet and there ability to fulfill the requirements. They are not enabling the communication between two end-devices simultaneously, low latency or to determent the critical data and their priority to handle it. Therefore, a new technology was developed as a solution for this problem to satisfy the communication requirements. Time sensitive Networking (TSN) provides low latency transfer, simultaneous communication between two end-devices, security and priority for critical data. TSN has several Standards that provide different functions and applications.

In this paper 802.1Qci and its definition for Per-stream filtering and policing was discussed. Because the technology relatively new, there are some gaps in the implementation and it still developing. A problem that 802.1Qci have it, when the internet daily scaling up and a lot of changes happen, how can be the policy of a network updated to still in full swing with these changes. A solution was to centralize the policies in one server that called Policy Server, to controls all the policies and update them according to the requirements that will take them from the CNC (look Section/Subsection 3.2).

In addition to these implementations, Credit Based Meter (CBM) will tested and analyzed in a simulated environment networks, where also will show how it can deal with other implementations.

References

[1] P. Meyer, "Preventing DoS Attacks inTime Sensitive Networking In-Car Networks through Credit Based ingress Metering."

[2] D. Greenfield, "4 reasons why time sensitive networking matters," 2016, [Online; accessed 28-September-2020].

[3] J. L. Messenger, "Time-Sensitive Networking: An Introduction."

[4] Cisco, "Time-sensitive networking: A technical introduction," 2017.

[5] "Ieee standard for local and metropolitan area networks–bridges and bridged networks–amendment 28: Per-stream filtering and policing," *IEEE Std 802.1Qci-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, and IEEE Std 802.1Qbz-2016)*, pp. 1–65, 2017.

[6] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 88–145, 2019.

[7] J. H. Robert Barton, Maik Seewald, "Management of IEEE 802.1Qci Security Policies for Time Sensitive Networks (TSN)," 2018.

Current Developments of IEEE 1588 (Precision Time Protocol)

Kilian Rösel, Max Helm*, Johannes Zirngibl*, Henning Stubbe*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: roeselk@in.tum.de, helm@net.in.tum.de, zirngibl@net.in.tum.de, stubbe@net.in.tum.de

Abstract—**Precise synchronization of clocks is essential for multiple scientific and industrial applications. Synchronization in networks can be achieved with the IEEE 1588 Precision Time Protocol. This paper gives an overview of this protocol and explores recent developments of this standard. It examines new features for accuracy and security introduced by the 2020 released IEEE 1588-2019 (PTPv2.1) edition of this protocol. Sub-nanosecond accuracy gets supported by the High Accuracy Profile based on the White Rabbit Extension, utilizing Layer 1 signals and a system wide calibration procedure. Several approaches to make the synchronization mechanism more secure are presented. Finally the paper outlines the expected impact of PTPv2.1 functionality on industrial use cases.**

Index Terms—**IEEE 1588, precision time protocol, high accuracy**

1. Introduction

Precise synchronization of clocks in distributed systems is a major requirement in several areas such as telecommunication, finance and power grid. However, many solutions lack in synchronization accuracy, robustness and security to be properly deployed in real industrial scenarios [1].

On 16 June 2020 the IEEE 1588-2019 [2] version of the Precision Time Protocol (PTP) superseded the previous IEEE 1588-2008 (PTPv2) [3] version. This new revision includes the High Accuracy Profile (HA), which allows to achieve sub-nanosecond accuracy as well as several mechanisms to make PTP systems more secure and robust.

This paper gives an overview over the PTP protocol in Section 2. The different PTP devices with their topology and the synchronization mechanism will be discussed. The new High Accuracy Profile allows to achieve sub-nanosecond accuracy. It relies on two key mechanisms: Firstly, calibration and measurement of asymmetries and secondly achieving higher precision in timestamping, presented in Section 3. Furthermore, new features and guidelines for security are presented in Section 4. This paper finally discusses new possibilities and challenges of PTPv2.1 in PTP implementations based on different industries in Section 5.

2. Background

A PTP network consists of multiple PTP devices and non-PTP devices, such as switches and routers.

An Ordinary Clock (OC) is a terminal device which has only one PTP port and maintains the timescale with its local clock. It can either be the Grandmaster Clock, such that it acts as the source of time or a slave receiving time. When it is in the master state, it often uses global navigation satellite systems (GNSS) or terrestrial radio links as time reference.

Boundary Clocks (BC) are network devices with multiple PTP ports. One of them is in the SLAVE state, so they can synchronize their own local clock to the time source. The ports in the MASTER state provide time to other PTP Instances.

End-to-end (E2E) and peer-to-peer (P2P) Transparent Clocks (TC) are network devices as well, but do not synchronize their own internal clock. Instead they measure the residence time of PTP messages and propagate them after adjusting a correction field.

Management Nodes are devices used for configuring and monitoring clocks in a PTP network.

Non-PTP devices such as switches and routers, can cause inaccuracies because they introduce asymmetry in the network through queueing effects. For achieving high accuracy it is therefore essential to only use BCs and/or TCs as network devices.

The logical unit in which the PTP devices synchronize to one timescale is called a domain. Originally multiple domains could exist in the same network, but were strictly separated. The new edition introduces the possibility of inter-domain interactions between PTP devices. This feature can get used to enhance security, presented in Section 4.

2.1. Master-Slave Hierarchy

The PTP domain has to be organized in a treelike master-slave hierarchy, with the best suited clock as grandmaster at the root. To select the grandmaster and to negotiate this topology the Best Master Clock Algorithm (BMCA) may be used. First OCs and BCs exchange the following performance properties via *Announce* messages:

1) priority1: Can be set by administrators to apprise their preferred master clock.
2) clockClass: Describes the traceability, synchronization state and expected performance.
3) clockAccuracy: Describes the accuracy of the Local PTP Clock.
4) offsetScaledLogVariance: Describes the stability of the Local PTP Clock.
5) priority2: Can be set by administrators to arrange equivalent PTP Instances.

6) clockIdentity: Unique identifier for PTP Instances to break ties.

Secondly, each PTP Instance computes the states of its ports according to those properties.

The BMCA also prunes mesh topologies to avoid cyclic network connections. It does so by setting ports on PASSIVE state such that there is no time synchronization on this connection. This way endless circulation of rogue *Announce* messages can be avoided. Figure 1 shows an exemplary PTP network with pruned mesh topolgy [2].

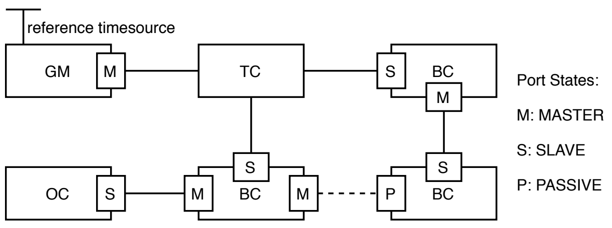


Figure 1: Example PTP network with pruned mesh topology [2].

The BMCA is running continuously, even when the desired topology is already established. This way the network can reconfigure itself automatically, if for example physical connections get lost or the performance properties of a Grandmaster Clock degrade [2].

The new edition of the standard also includes mechanisms for manual configuration of PTP port states. However, setting port states manually may result in "timing islands" where time does not get distributed, illustrated in Figure 2. Additionally it disables automatic reconfiguration [4].

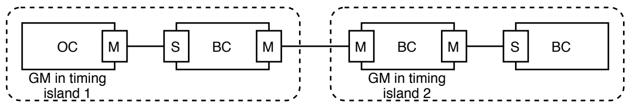


Figure 2: Adjacent ports in master state result in timing islands [4].

2.2. Synchronizing Mechanisms

The time synchronization mechanism takes place between two linked Ordinary and/or Boundary Clocks. One of them is in the master state, the other one in the slave state. They are exchanging a series of event and general messages to calculate the offset of the Slave Clock with respect to the master clock. Event messages are messages that get timestamped when they egress or ingress a port. General messages are not required to be timestamped. Details on timestamp generation are shown in Section 2.3. Eventually, all PTP Instances are synchronized to the grandmaster as time gets distributed through the hierarchy.

To distribute time, the master clock first sends a *Sync* message to the Slave and timestamps the departure time t_1. The slave timestamps the arrival of this message t_2. In a two-step setup the master then sends a *Follow_Up* message containing t_1. In a one-step setup the master clock would already have included timestamp t_1 in the first *Sync* message, rendering the *Follow_Up* message obsolete. In order to calculate the network delay according to the E2E mechansim the Slave clock then sends a *Delay_Req* message, and notes the departure time t_3. The master creates timestamp t_4 at arrival of this message and communicates this timestamp via a *Delay_Resp* to the slave. Figure 4 illustrates this message exchange. When the slave clock possesses all four timestamps, it can compute the mean path delay d and its offset to the master o:

$$d = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}$$
$$o = (t_2 - t_1) - d$$

Knowing the slave-master offset the slave clock can adjust its own clock and is then synchronized to the master.

Calculation of the network delay can also be done with the P2P mechansim. This mechanism does not calculate the network delay between a master and slave port, but between directly neighbouring nodes. The P2P network delay then gets added up along the whole path. Figure 3 illustrates the difference between P2P and E2E.

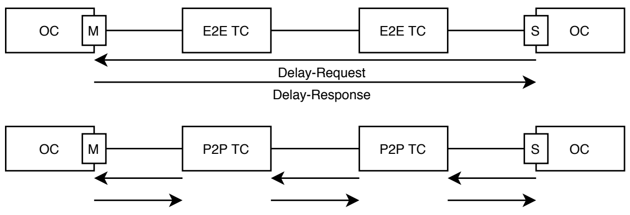


Figure 3: PTP Delay Mechanism [5]

This model assumes the master-slave (t_{ms}) and slave-master (t_{sm}) propagation delay to be symmetric, i.e. messages need the same time to travel in either direction. However, to achieve high accuracy in real scenarios one must take steps to account for asymmetries in the network. The HA therefore defines a system wide calibration procedure, shown in Section 3.1.

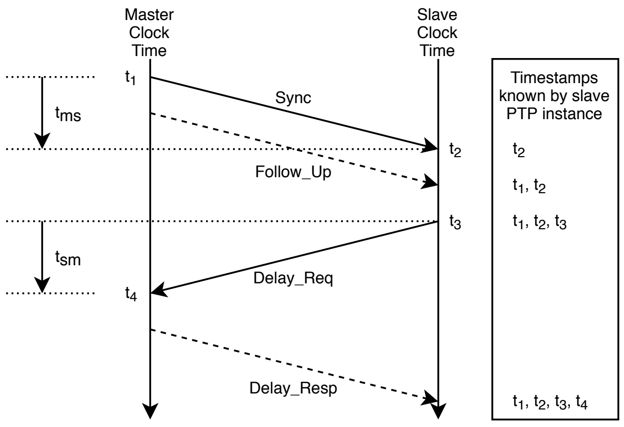


Figure 4: Basic end-to-end PTP Timing Message Exchange [2]

2.3. Timestamp Generation

Precise timestamp generation is crucial for the accuracy of round trip time measurement. A timestamp is defined as the instance the message timestamp point of an event message crosses the reference plane between medium and PTP port. Though in implementations timestamping might take place in the Application Layer (C), in the kernel interrupt service routines (B) or in the physical layer (A), illustrated in Figure 5. Traveling through the protocol stack can introduce latencies, thus it is preferable to choose a point near to the physical layer. In this case, specialized hardware assists in the generation of the timestamp. Nevertheless, any offset from the reference plane has to be compensated for by measurement and calibration [2].

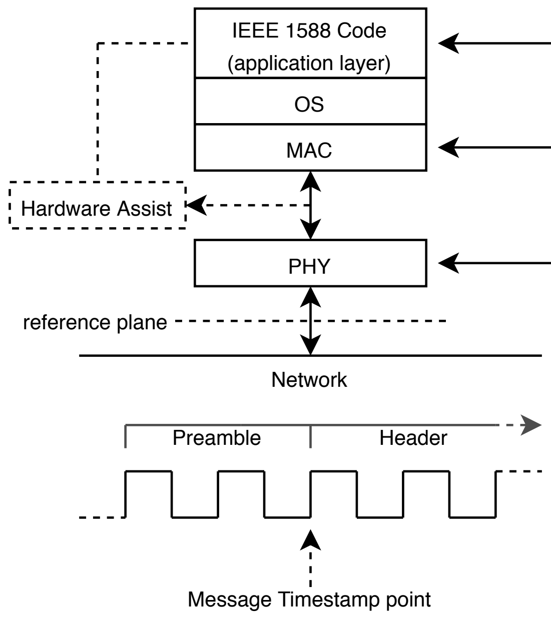


Figure 5: Protocol Stack and Message Timestamp Point [2]

3. High Accuracy

The High-Accuracy Profile is based on the White Rabbit Extension (WR) for PTPv2. WR was developed to renovate the control and timing system at CERN [6] and was later generalized and included in the standard by the P1588 working group [7].

It allows to achieve sub-nanosecond synchronization accuracy by relying on two mechanisms and methodologies: (1) Various sources of asymmetry get recognized, measured and calibrated to compensate for their effects, described in Section 3.1. (2) Utilizing physical transmission and receive signals to increase precision in the hardware assisted timestamping process of PTP event messages, described in Section 3.2 [8].

3.1. Calibration

Asymmetries between two PTP Instances introduce inaccuracy in the synchronization process. There are two sources of asymmetry: timestamp generation latencies and medium asymmetry. Knowing the values allows to compensate for their effects when calculating the offset from the master.

Timestamp generation latencies get introduced on egress and ingress of messages, e.g. because timestamps are captured at a point removed from the reference plane, see Section 2.3. Medium asymmetries originate from the physical communication medium. They can for example be caused by the use of different wavelengths of light in single-strand fibers. The standard defines several procedures, how to calibrate these latencies and asymmetries. Because of different optical phenomena in long distance optical links, these procedures are only intended for Local Area Networks [2]. However, deployment of long distance fiber links has already been investigated [9].

3.2. Precise Timestamping

The accuracy of delay measurements relies on the resolution and precision of timestamping. Timestamps are created by the Local PTP Clock whenever the message timestamp point crosses the implemented point in the protocol stack, see Section 2.3. However, usually the receive and transmit signals on the Physical Layer (L1) are different from the Local PTP Clock signal used for timestamping. This may result in timestamping imprecision. For example, a Local PTP Clock with a frequency of 125 MHz is limited to a resolution of 8 ns [8].

To correct for this imprecision, knowledge about the phase offset between the L1 transmit clock signal (clk_{txL1}), L1 receive clock signal (clk_{rxL1}), and the Local PTP Clock ($\text{clk}_{\text{localPTP}}$) is required. The L1 tx/rx signals are the physical signals used by the medium to transport signals over the wire. The reception phase offset (x_{rx}) and transmission phase offset (x_{tx}) is the offset between the Local PTP Clock signal to the L1 receive signal and L1 transmission signal respectively. This relationship gets demonstrated in Figure 6. Note that the transmit signal of Clock A is the receive signal of Clock B. Knowing the value of x_{rx} and x_{tx} at the instance of the timestamp allows then to compensate the offsets in the calculation process.

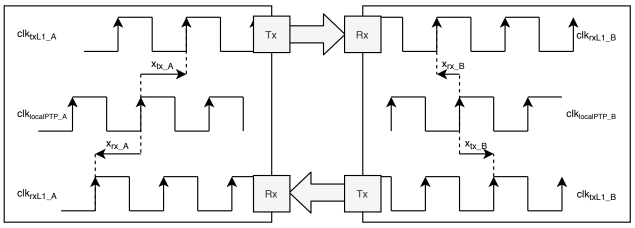


Figure 6: Link Reference Model between two Clocks [8]

Quantifying the phase offsets depends on the variability of the offset. In the simplest case the offsets are constant. That means the L1 tx/rx signals and the Local PTP clock signal are coherent, i.e., they operate on the same frequency. To achieve coherency, ports can base their Local PTP Clock signal on the L1 rx signal recovered from the medium and generate their L1 tx signal from the Local PTP Clock. With this relationship in place, the constant offsets can be measured, for example by using Digital Dual Mixer Time Difference (DDMTD) phase detection [10].

Syntonization in networks can for example be achieved with Synchronous Ethernet (ITU-T Recommendations G.8261 [11] and G.8262 [12]). The PTP Clocks can then take advantage of the Layer 1 syntonization to enhance their timestamping precision [8].

4. Security Mechanisms

Security concerns have long been neglected in the development of PTP. Especially in critical infrastructures such as the power grid this might prove fatal. However, PTPv2.1 describes several mechanisms to make PTP more secure [13].

PTP Integrated Security Mechanism. PTP messages can be extended by a type, length, value (TLV) extension mechanism in order to transfer additional information. There are several different types of TLVs defined.

The AUTHENTICATION TLV, providing a way to authenticate PTP messages, was already introduced in PTPv2. But test implementations of this feature have shown little additional security at the expense of overhead [14]. PTPv2.1 revised the AUTHENTICATION TLV feature.

The included integrity check value (ICV) verifies all fields from the PTP Header up to the AUTHENTICATION TLV without including the ICV itself. Also included in the calculation is a secret key. This key has to be distributed by a key management system. Depending on this system two different verification schemes are possible: (1) Immediate security processing enables verification of the message immediately. To achieve this, the secret key has to be known to the communication partners before processing. This approach also allows mutable fields. For example a transparent clock can adjust the correction field and can then recompute the ICV. (2) Delayed processing enables to share the secret key after message transmission. With this approach the receiver has to store the message until he receives the key to verify it. Those two approaches can also be combined. Figure 7 shows this case. Everything after the first AUTHENTICATION TLV is immediately verified. This method allows to add TLVs that can be modified by intermediate devices [13].

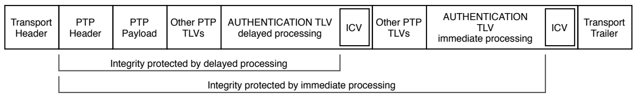

Transport Header
PTP Header
PTP Payload
Other PTP TLVs
AUTHENTICATION TLV delayed processing
ICV
Other PTP TLVs
AUTHENTICATION TLV immediate processing
ICV
Transport Trailer
Integrity protected by delayed processing
Integrity protected by immediate processing

Figure 7: Authentication TLV [2]

The key management system is responsible for the distribution of keys, however the standard does not yet define such a system, but merely gives guidelines [15].

PTP External Transport Security Mechanisms. The standard suggests using MACSec and IPSec as external security mechanisms. Those protocols provide protection against several attacks, as shown by [16].

Architecture Mechanisms. The standard presents various guidelines to enhance security by architectural choices based on redundancy: (1) Redundancy by complementary timing systems means that end-users of time obtain a second reference through a non-PTP way, for example by GPS. This way they can detect malicious behaviour. (2) Multiple domains with separate Grandmaster Clocks work together through inter-domain interactions. End users then can obtain time in a voting process from multiple domains and are therefore able to exclude malfunctioning time information. (3) Lastly redundant network paths between nodes can ensure distribution of timing messages even when some connections get lost [2].

Monitoring and Management Mechanisms. Monitoring and managing the performance of the PTP network can reveal clues about potential security attacks, e.g. delay attacks. These can be identified by detecting unexpected offset jumps or large changes in measured path delays. The new version has also introduced a standardized format in which all PTP devices can share their performance data in an uniform way with Management Nodes [2].

5. Applications of PTPv2.1 Functionality

The White Rabbit Extension has already proven useful in multiple scientific applications, e.g. in particle accelerators. But also other sectors have already made endeavours in adapting this technology [17]. Deutsche Börse, for example, uses WR to synchronize their own timestamping devices. Additionally they provide means for their trading partners to synchronize their own clocks to theirs [18]. As the WR technology matures through the standardization as High Accuracy Profile, it will grow even more attractive for industrial use. So it is to be expected to see an adaption in multiple areas. Especially the operation of power grids can profit from increased timing accuracy. As the grid evolves to being powered by sustainable but unpredictable energy sources, precise monitoring is essential. For example, multiple synchrophasers can detect characteristic voltage spikes caused by malfunctioning equipment. When the measurements are precisely synchronized conclusions on the origin can be drawn [19].

Deployment in such critical infrastructure was previously hampered by security concerns. An implementation of the AUTHENTICATION TLV feature for Linux PTP has already proven to be feasible with a low computational overhead [13]. This result and the other security guidelines may encourage adopters in utilizing PTPv2.1 functionality in their own implementations.

6. Conclusion and Future Work

The new version includes options for achieving high accuracy and mitigating security risks. These two features are essential for PTP to be further adapted as time synchronization technology. This paper has presented these new features and has briefly outlined their impact on industrial scenarios. However, PTPv2.1 includes even more innovations, not presented in this paper, such as profile isolation, special PTP ports and mixed multicast/unicast operation [20].

References

[1] F. Girela-López, J. López-Jiménez, M. Jiménez-López, R. Rodríguez, E. Ros, and J. Díaz, "IEEE 1588 High Accuracy Default Profile: Applications and Challenges," *IEEE Access*, vol. 8, pp. 45 211–45 220, 2020.

[2] "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.

[3] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–300, 2008.

[4] D. Arnold, "What's in IEEE 1588-2019: DIY PTP Port States," https://blog.meinbergglobal.com/2020/07/24/whats-in-ieee-1588-2019-diy-ptp-port-states/, 2020, [Online; accessed 24-September-2020].

[5] 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.

[6] "The White Rabbit Project," https://white-rabbit.web.cern.ch/Default.htm, 2020, [Online; accessed 26-September-2020].

[7] "IEEE P1588 Working Group," https://sagroups.ieee.org/1588/, 2020, [Online; accessed 26-September-2020].

[8] O. Ronen and M. Lipinski, "Enhanced Synchronization Accuracy in IEEE1588," *2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pp. 76–81, 2015.

[9] E. F. Dierikx, A. E. Wallin, T. Fordell, J. Myyry, P. Koponen, M. Merimaa, T. J. Pinkert, J. C. J. Koelemeij, H. Z. Peek, and R. Smets, "White Rabbit Precision Time Protocol on Long Distance Fiber Links," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 63, no. 7, pp. 945–952, 2016.

[10] P. Moreira, P. Alvarez, J. Serrano, I. Darwezeh, and T. Wlostowski, "Digital Dual Mixer Time Difference for Sub-Nanosecond Time Synchronization in Ethernet," pp. 449–453, 2010.

[11] "Timing and Synchronization Aspects in Packet Networks," *ITU-T G.8261/Y.1361*, pp. 1–120, 2019.

[12] "Timing Characteristics of Synchronous Equipment Slave Clock," *ITU-T G.8262/Y.1361*, pp. 1–44, 2018.

[13] E. Shereen, F. Bitard, G. Dán, T. Sel, and S. Fries, "Next Steps in Security for Time Synchronization: Experiences from implementing IEEE 1588 v2.1," *2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pp. 1–6, 2019.

[14] C. Önal and H. Kirrmann, "Security Improvements for IEEE 1588 Annex K: Implementation and Comparison of Authentication Codes," pp. 1–6, 2012.

[15] D. Arnold, "The PTP AUTHENTICATION TLV," https://blog.meinbergglobal.com/2020/06/04/the-ptp-authentication-tlv/, 2020, [Online; accessed 29-September-2020].

[16] T. Mizrahi, "Time Synchronization Security Using IPsec and MACsec," *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pp. 38–43, 2011.

[17] M. Lipiński, E. van der Bij, J. Serrano, T. Włostowski, G. Daniluk, A. Wujek, M. Rizzi, and D. Lampridis, "White rabbit applications and enhancements," *2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pp. 1–7, 2018.

[18] "High Precision Time (White Rabbit) Pilot," https://www.eurexchange.com/ex-en/find/initiatives/technical-changes/high-precision-time-white-rabbit-pilot, 2019, [Online; accessed 3-October-2020].

[19] A. Jarc, "Use Cases for Timing in Power Grids," https://blog.meinbergglobal.com/2019/07/16/use-cases-for-timing-in-power-grids/, 2019, [Online; accessed 3-October-2020].

[20] D. Arnold, "What's In the 2019 Edition of IEEE 1588?" https://blog.meinbergglobal.com/2017/09/24/whats-coming-next-edition-ieee-1588/, 2017, [Online; accessed 3-October-2020].

SmartNICs: Current Trends in Research and Industry

Tristan Döring, Henning Stubbe*, Kilian Holzinger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: tristan.doering@tum.de, stubbe@net.in.tum.de, holzinger@net.in.tum.de

Abstract—**With ever rising demand, modern cloud environments had to evolve fast in the last years. One of these novel problems are the increasing speed requirements in combination with present Software-Defined Networks (SDNs). This paper gives an overview on a new hardware trend resulting from this. We illustrate the demand, development, implementation and use of the network accelerating SmartNICs. SmartNICs tackle existing problems of NIC-hardware such as the lack of flexibility, a requirement for virtualized networks. Furthermore the SmartNIC term will be analyzed to provide an universal definition.**

Index Terms—**SmartNIC, network accelerator, data processing unit, fpga-based smartnic, asic-based smartnic, soc-based smartnic**

1. Introduction

Demand for network performance grows with the expanded use of the internet and the increasing popularity of cloud-based computing. The combination of standard network interface controllers, dedicated networking hardware and software based virtual packet routing can not keep up with the increasing speed requirements. One reason for the slowing development of CPU performance is the decline of Moore's law. E.g. network speeds in the Microsoft Azure platform improved by 50 times between 2009 and 2017, but CPU performance did not improve in the same speed [1]. Therefore cloud service providers, data center operators etc. had to think about new solutions which enable them to provide their services to the growing and performance demanding audience with faster and cost efficient performance. A problem known to exist is the lack of programmability/flexibility in regular NIC hardware which is built for a number of specific predefined use cases and at heart still serves the functionality as an interface from the host CPU to the network. This leaves much of the workload to the CPU. This static functionality does not comply with evolving Software Defined Network (SDN) policies and Virtual Network Functions (VNF) which are key parts of current virtualized network environments. There is an increasing demand for better performance compared to software solutions without compromising too much flexibility. This leads to the development of the SmartNIC. The novel NICs should be able to handle more complex tasks independent of the CPU to a certain degree. This is called an in-Network Interface Card (in-NIC) processing approach.

2. Trends and Technological Demands in Cloud/Hosting Industry

Before diving deeper into the topic of SmartNICs this section will give a short overview on the current state of the industry. The new guiding trend is the virtualization of networks, storage, GPUs etc. These techniques generate network related workloads not only on network devices as virtualization can not independently run on e.g. NICs. The network processing can be divided into two categories, i.e. the data plane and the control plane. The control plane is responsible for the control of the network structure, i.e. communications between network devices and assigning tasks to network devices. In other words it is the implementation of network policies. The data plane handles the actual movement and modification of application data packets. The legacy hardware layout in a server is a combination of standard NICs and CPU cores. Here the CPU handles the control and data plane. The NICs on the other hand at best accelerate certain data plane functions to ease up load on the CPU. That means in current server environments the network traffic can use up a significant amount of CPU resources [1, section 3] (see also section 5.3).

The following paragraphs will explain a few common network technologies present in typical up-to-date server environments that are demanding on the CPU.

Software-Defined Networking (SDN) decouples the network structure from its realization in hardware. That means the control plane is controlled by software control plane policies. These policies are mapped to the data plane and executed in hardware. That imposes additional packet steering and processing requirements and adds additional workload to the CPU. E.g. Open vSwitch (OVS), an open-source software implementation of a multilayer switch, is part of such a virtual network stack.

For example storage virtualization also called Software-Defined Storage (SDS) is part of the virtualization trend. The idea is to share storage between different servers over the network. Virtualization and abstraction is used to make the storage look local to applications. That means all the data traffic has to go through the network. This brings up some challenges to overcome with even more stress on the CPU which will further increase with higher bandwidths. The newest technology is Non-Volatile Memory express over Fabrics (NVMe-oF) which combines the low latency NVMe-protocol with virtualized storage addressing over the network.

For efficient implementations of these protocols, SmartNIC acceleration is vastly beneficial.

3. What is a SmartNIC?

Network environments develop to be more complex and the trend shifts to more and more virtualized network environments. This involves too much networking overhead on the CPU cores. With rising network speeds of currently up to 200Gbps per link the expensive CPU spends too much work for networking. Even though the true value of CPUs lies in their ability for general processing e.g. applications and data analysis. The virtualization trend makes this even worse with increasing local network traffic in servers due to SDS, SDN and big data. An obvious solution is to offload this kind of processing to specialized external devices. There are already some NIC devices that are not considered smart but assist the CPU by performing a huge diversity of network function acceleration including Network virtualization. These functions are hardcoded into these NICs. They are considered the first step that led to the development of SmartNICs. This combination of standard NICs and servers is not able to deliver enough performance to meet the rising demands for network speed because of the required application of SDN policies [1, section 2.3]. SmartNICs in comparison to CPU and NIC combinations target to offer a faster, more efficient and lower cost solution including the important reduction of CPU usage. Another important factor is the required flexibility that current software solutions offer. The programmable SmartNICs also try to meet this demand by offering various development kits or even the possibility to run existing software to engineers. We will see later how different manufacturers handle this problem.

3.1. The Term SmartNIC

"Also called a 'network interface card' (NIC), a network adapter is a plug-in card that enables a computer to transmit and receive data on a local network. [...]" [2]

On the basis of this definition of a classic NIC and also the knowledge of some SmartNIC functions and use cases a proper definition is worked out. A definition that is applicable to the new type of network accelerators called SmartNICs. The term SmartNIC implicates that it is an extension of standard NIC devices, but it is not as simple as just an increased functionality. The keyword "Smart" implies some kind of intelligence or the ability to act somewhat smart to solve complex tasks. The smartness in this new device class lies in the ability to perform numerous tasks independently and to be flexible enough to tackle future and current network tasks [1, section 3]. This was achieved by adding some kind of general processing unit to a NIC.

3.2. Other SmartNIC Definitions

There are numerous attempts to define what the term SmartNIC means. A widely spread definition is written by Alan Freedman, the author of a tech encyclopedia: "A network interface card (network adapter) that offloads processing tasks that the system CPU would normally handle. Using its own on-board processor, the smartNIC may be able to perform any combination of encryption/decryption, firewall, TCP/IP and HTTP processing. SmartNICs are ideally suited for high-traffic Web servers" [3]. But this definition lacks one of the most important properties of a SmartNIC, the programmability, the biggest difference to a regular NIC. In today's rapidly changing software-defined networks updatability is key to have a future proof network environment. Microsoft also supports this opinion in their paper "Azure Accelerated Networking: SmartNICs in the Public Cloud" where they list "Maintain host SDN programmability of VFP" [1] as one of their design goals for their in-development Microsoft Azure SmartNIC. Also contrary to the definition by Alan Freedman use cases or application environments should not be mentioned in the definition as these are broad and ever changing.

Mellanox's Vice President of marketing Kevin Deierling goes even further and defines three different kinds of NIC devices: Foundational NICs, Intelligent NICs and DPU based SmartNICs [4]. This variety exists because Mellanox has customers who ask for a SmartNIC because they need functionality they refer to as smart and which is superior to that of a Foundational NIC. But these functionalities were also supported by their base line of NIC products, the so-called Intelligent NICs which Mellanox does not refer to as smart. Kevin Deierling's approach is to differentiate NICs by their offered functionality. The further differentiation does make sense for Mellanox, because they want customers to perceive their products superior to standard NICs. Mellanox defines an Intelligent NIC as a NIC with extended functionality that is not programmable. The further differentiation may be confusing because the border between NICs and Intelligent NICs can not be properly defined. In the future the comprehension of what extended NIC functionality is will probably change and with it the definition for Intelligent NICs.

3.3. Definiton of SmartNIC

Going off of multiple press and industry definitions we decided to work out a definition of our own that is universally applicable:

A SmartNIC is a device which connects to a network and a computer that specializes in hardware-acceleration of various standard and software-based network related tasks. The innovation drivers for its development are freeing up CPU resources, the requirement for formerly unreachable speeds and more efficiency. It is able to perform control and data plane functions by combining NICs with flexible processing units. That also allows SmartNICs to offer programmable and updatable functionalities. It can also be called programmable network function accelerator.

4. SmartNIC Hardware Architectures

This section explains different hardware approaches to SmartNICs. There are three different processing units to build a SmartNIC upon. Figure 1 is a presentation slide of Microsofts paper "Azure accelerated networking: Smartnics in the public cloud" [1] and shows 5 different options to accelerate SDN speeds. Also the reciprocity of the flexibility versus the efficiency of the hardware solutions is illustrated. This paper will not explain the G/NPU hardware approach because there are no such devices released yet.

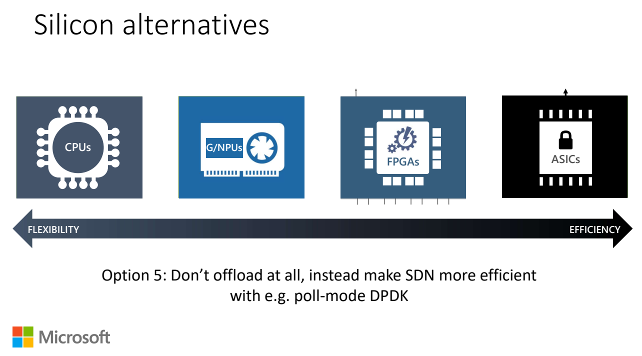

Silicon alternatives
CPUs
G/NPUs
FPGAs
ASICs
FLEXIBILITY
EFFICIENCY
Option 5: Don't offload at all, instead make SDN more efficient with e.g. poll-mode DPDK
Microsoft

Figure 1: Overview on the different implementation options and their flexibility versus efficiency [1]

4.1. FPGA-based SmartNICs

The field-programmable gate array is an integrated circuit which is reconfigurable. This software-like programmability paired with faster performance and lower latency close to that of an Application-Specific Integrated Circuit (ASIC) makes it a perfect fit for SmartNIC hardware. The FPGA achieves faster performance with parallel processing of data flows in comparison to instruction by instruction temporal computing in CPUs. The downside is the comparatively difficult implementation of new functions as it requires expertise in Hardware Description Language (HDL) to make efficient use of the chip. Hiring dedicated development teams with sufficient skills is not a big problem for hyperscalers such as Microsoft, Amazon and Google FPGAs. Thus FPGAs are often used in these hyperscale environments. Though manufacturers try to facilitate development by offering compatibility with common developer environments such as the Data Plane Development Kit (DPDK). Microsoft estimates that the size of an FPGA compared to an ASIC with the same functionality is "around 2-3x larger" because of the flexibility they offer [1, section 4.1.3]. This generally makes them more expensive than ASICs. There are not many example products in the current market that only rely on FPGAs. One rare instance is the Intel FPGA Programmable Acceleration Card (PAC) N3000 that we will discuss later.

4.2. SoC-based SmartNICs

SmartNICs with a System-on-a-Chip work like embedded Linux servers [4, section 4.1.2]. They handle the data and control plane in a traditional manner as they basically run software-based VNFs on their SoC. Due to their compact size format and thus shorter distances between memory, processing unit and network interface efficiency and speed are usually better. Often some kind of other more specialized processing unit like FPGAs or ASICs are paired with SoCs, e.g. in the Silicom N5110A SmartNIC [5]. Many SoC-based SmartNICs offer easy C-programmability. Compared to FPGAs SoCs are easier to program, but slower/less efficient alternatives. As no special programming skills are required SoC based SmartNICs are well-suited for smaller companies e.g. the Broadcomm Stingray SmartNIC architecture which is specifically marketed as easy to deploy [6].

4.3. ASIC-based SmartNICs

Application-Specific Integrated Circuits (ASICs) achieve higher efficiency/performance by sacrificing the flexibility present in an FPGA. ASIC development traditionally means designing a specification and test methodology for everything that a system could possibly want to do over its lifetime upfront. As basic NICs usually rely on ASICs, it results in their development cycles being similar. FPGAs on the other hand allow hardware developers to be far more agile in their approach. The development phase of an ASIC chip can take up to 2 years from requirements engineering to the ready silicon parts [1, section 4.1.1]. But in that time the functional requirements of a NIC probably already changed because new network technologies are being developed all the time. Additionally ASICs are only able to perform data plane functions. To counter these issues manufacturers often add embedded CPU cores to handle new functionality and the control plane. These cores are comparatively slow and can consequently cause a bottleneck when more and more functionality is added to them. In every SmartNIC there is a foundational level of functionalities that is known during the development phases that benefits from the power and cost efficiency of ASICs. As a result ASICs in SmartNICs still play the role of a NIC but only in combination with other flexible compute units.

5. Two SmartNIC Products in Detail

At first sight the market of SmartNICs seems very confusing because every product is marketed for almost every application domain. Performance specifications are rare and differences must be found in the details. Thus the direct comparison of different SmartNIC products is hard. To illustrate the diversity of products two vastly different approaches were chosen to be displayed.

5.1. Intel FPGA PAC N3000

The FPGA-based Intel FPGA PAC N3000 is part of Intels newest line of FPGA-based accelerator cards. This Intel device combines the worlds of programmable accelerator cards (PACs) and SmartNICs into one product. It allows for optimization of data plane performance to achieve lower costs while maintaining a high degree of flexibility. The built-in Intel Arria 10 GT FPGA delivers up to 1.5 TFLOPS [7, section 2.2.1]. As a board management controller the Intel Max 10 FPGA is responsible for controlling, monitoring and giving low-level access to board features [7, section 2.4]. The Intel XL710-BM2 NIC is directly connected to the FPGA to provide basic ethernet connectivity, some virtualization and I/O Features [8]. Consequently the Intel Arria 10 GT can accelerate network traffic at up to 100 Gbps [7]. The Root-of-Trust technology prevents the loading or executing of unauthorized workloads and the unauthorized access to key interfaces and on-board flash storage. The scopes of application are Network Function Virtualizaton (NFV), Multi-Access Edge Computing (MEC), Video Transcoding, Cyber Security, High Performance Computing (HPC) and Finance [8]. Intel tries to make development as comfortable as possible with the implemented compatibility with various developing environments. The Intel Acceleration Stack for Intel Xeon CPU with FPGAs provides an allround development platform. The supported Open Programmable Acceleration Engine (OPAE) technology provides a consistent API across FPGA product generations and platforms. Also the popular Data Plane Development Kit (DPDK) is supported and provides control over both the FPGA and the Ethernet Controller. Due to its lack of a SoC the Intel FPGA PAC N3000 can not offload everything from the CPU as some functions like Internet Protocol Security (IPSec) still need additional CPU resources [9]. The Intel FPGA PAC N3000 compromises on flexibility to offer a network acceleration speed of up to 100 Gbps. It is mostly advertised to telecommunications service providers.

5.2. NVIDIA Bluefield-2 DPU

With the Bluefield-2 DPU, Nvidia fuses specialized network processing and general processing into one SoC. The SoC consist of a Nvidia Mellanox ConnectX-6 Dx intelligent NIC, two Very Long Instruction Word (VLIW) Acceleration Engines and a 64bit Arm processing unit [10]. It offers full programmability to the user. The Arm NEON SIMD execution unit is optimized for vector processing with an extended Instruction Set Architecture (ISA). Nvidia advertises the Bluefield-2 as a DPU made for data movement and security processing. In this case DPUs are defined as programmable Data Center Infrastructure-on-a-Chip. With the built-in Nvidia Mellanox ConnectX-6 Dx Nvidia integrated the fastest NIC in the ConnectX product line [11]. The general idea is to combine easy programmability of an Arm chip with the accelerator especially built for parallel data processing. Even though Nvidia rarely calls it a SmartNIC, it falls in this category when looking at the specifications and features. The DPUs features include some which are commonly seen in SmartNICs such as NFV, en-/ decryption and many other network, security or storage virtualizations such as IPSec, NVMeoF, OVS etc. GPUDirect, a family of technologies that enhances data movement and access for NVIDIA data center GPUs is also supported. Using GPUDirect, network adapters and storage drives can directly read and write to/from GPU memory, eliminating unnecessary memory copies, decreasing CPU overheads and reducing latency, resulting in significant performance improvements. This is an important functionality for AI computing. The Nvidia DOCA SDK integrates industry-standard open APIs for software-defined networking and storage, security services, and programmable P4 functionality. The DOCA SDK was just announced with the Bluefield-2 and is meant to become the standard SDK with intercompatibility to all other Bluefield devices [12]. Another product announcement based on the Bluefield-2 worth mentioning is the so-called AI-Powered DPU Bluefield-2X. With the addition of a Nvidia Ampere GPU it is the first instance of a GPU-based SmartNIC. The GPU enhances AI-based functions like real-time security analytics, identifying of abnormal traffic and dynamic security orchestration [13]. In conclusion the Nvidia Bluefield-2 DPU presents a novel approach to SmartNICs. It combines the hardware acceleration of the data plane with the addition of an easy-to-programm control plane which both run on the custom SoC.

5.3. Nvidia Bluefield-2 performance benchmark

To illustrate the independence of the Bluefield-2 its performance in a DPDK 20.08 benchmark is compared to that of a Nvidia Mellanox ConnectX-6Dx so-called intelligent NIC. Nvidia benchmarked the throughput at zero packet loss at a linerate of 2x25 Gbps. The ConnectX-6Dx uses 4 additional CPU cores, while the BlueField-2 SmartNIC is on its own. The ConnectX-6Dx CPU combination achieves to throughput the maximum amount of packages per second possible at the specified line rate. The BlueField-2 is only slightly slower (maximum 2.06% slower) but uses none of the CPU cores [14].

6. DDoS Mitigation Use Case

The reference paper [15] for this section was published by IEEE, it explores the use of SmartNICs for DDoS attack mitigation. The idea is to offload a portion of the DDoS Mitigation rules from the server to the SmartNIC. For comparison they try three different approaches to the problem.

6.1. Host-based Mitigation

All traffic is processed by the host CPU. The packets matching a given blacklist are dropped. The Linux-Kernel-based solution with iptables and its derivatives is too slow for modern DDoS attacks. The opposite solution using specialized NIC and network drivers in combination with an userspace applications proves to be faster, but requires allocating a fixed number of CPU cores. A mix of the previous solutions is XDP, in essence a kernel framework. The early kernelspace program is injected by userspace before the netfilter framework, hence performing an order of magnitude faster. Also its event-driven execution allows it to use CPU resources only when necessary.

6.2. SmartNIC-based Mitigation

The idea is to spare the CPU cores by prefiltering all packets in the SmartNIC. Depending on its hardware features there are different options to do so. If available the built-in hardware filters should be used, which can only hold a certain number of mitigation policies. That means the SmartNICs CPU has to apply the remaining ones by using custom programs. The surviving traffic is then directed to the server applications in the host system. This option works well until the number of mitigation policies is too big to store for the hardware tables. When additional use of the SmartNICs CPU becomes unavoidable, it presents a bottleneck resulting in declining performance.

6.3. Hybrid (SmartNIC + XDP Host)

A mix of the above combines the SmartNICs hardware tables which run at line rate and the enormous processing power of the host CPU. If the hardware tables prove to be too small to hold all mitigation rules the remaining ones will be implemented by XDP on the CPU.

6.4. Conclusion

In the experiments performed by IEEE, the hardware offloading approaches prove to be the most effective [15]. Precious CPU resources are protected as long as possible and even when additional CPU cores are required due to an increasing number of attack sources the hardware tables provide a proper prefiltering for CPU-based mitigation applications. SmartNICs can help mitigate the network load on congested servers, but only to a certain extent. In a real server environment a DDoS-aware load balancer would come in handy to distribute the load among multiple hosts and thus limit the number of mitigation policies.

7. SmartNICs: Conclusion

In this paper we discussed why the development of SmartNICs is inevitable, the different hardware approaches, two product instances and finally an interesting real world application of them. Beside all good about them it is clearly that their performance is not yet strong enough to solely handle high bandwidth throughput of a server. Often there is still no other choice but to use additional CPU cores as they still exceed the SmartNICs performance as seen in the DDoS Mitigation use case.

The future seems bright for SmartNICs. In the last years most companies in the sector announced their first line of products, e.g. Nvidia, Intel, Broadcom, Silicom, Inventec etc. Also AMD will probably try to enter the market with its acquisition of Xilinx, one of the major FPGA developers. In the future the development will be driven by even faster network speeds. E.g. Nvidia plans to increase the throughput of their devices to 400 Gbps including a 100-times performance improvement until 2023 [13]. The market is expected to grow substantially, with it the hardware-acceleration trend and the utilization of SmartNICs.

References

[1] D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh, M. Andrewartha, H. Angepat, V. Bhanu, A. Caulfield, E. Chung, H. K. Chandrappa, S. Chaturmohta, M. Humphrey, J. Lavier, N. Lam, F. Liu, K. Ovtcharov, J. Padhye, G. Popuri, S. Raindel, T. Sapre, M. Shaw, G. Silva, M. Sivakumar, N. Srivastava, A. Verma, Q. Zuhair, D. Bansal, D. Burger, K. Vaid, D. A. Maltz, and A. Greenberg, "Azure accelerated networking: Smartnics in the public cloud," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 51–66. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/firestone

[2] A. Freedman, "Definition: network adapter," 2020, accessed: 2021-01-05. [Online]. Available: https://www.computerlanguage.com/results.php?definition=network%20adapter

[3] A. Freedman, "Definition: SmartNIC," 2020, accessed: 2020-12-29. [Online]. Available: https://www.computerlanguage.com/results.php?definition=Smart+NIC

[4] K. Deierling, "Defining the DPU (Data Processing Unit) based SmartNIC : What is a SmartNIC and How to Choose the Best One," 2018, accessed: 2020-12-29. [Online]. Available: https://blog.mellanox.com/2018/08/defining-smartnic/

[5] Silicom Limited, "Silicom N5110A SmartNIC Intel® based - Server SmartNIC with Advanced Computing and 100Gbps Network Switching," 2021, accessed: 2021-02-28. [Online]. Available: https://www.silicom-usa.com/pr/fpga-based-cards/100-gigabit-fpga-cards/n5110a-pcie-smartnic-intel-based/

[6] Fazil Osman, Broadcom Limited, "Using SmartNICs as New Platform for Storage Services," 2019, accessed: 2020-12-30. [Online]. Available: https://www.snia.org/educational-library/using-smartnics-new-platform-storage-services-2019

[7] Intel Corporation, *Intel FPGA Programmable Acceleration Card N3000 Data Sheet*, Intel Corporation, 2020, accessed: 2021-01-01. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/documentation/dfy1538000574521.html

[8] Intel Corporation, "Product Brief Intel® Ethernet 700 Series Network Adapters Intel® Ethernet Converged Network Adapter XL710," 2020, accessed: 2021-01-03. [Online]. Available: https://www.intel.de/content/www/de/de/products/docs/network-io/ethernet/network-adapters/ethernet-xl710-brief.html

[9] Intel Corporation, "Accelerating IPSec with Arrive Technologies on the Intel® FPGA Programmable Acceleration Card N3000," Intel Corporation, Tech. Rep., 2019, accessed: 2021-01-05. [Online]. Available: https://www.intel.de/content/dam/www/programmable/us/en/pdfs/literature/solution-sheets/sb-accelerating-ipsec-arrive-technology-intel-fgpa-pac3000.pdf

[10] Nvidia Corporation, "NVIDIA BLUEFIELD-2 DPU Data Center Infrastructure on a chip," 2020, accessed: 2021-01-05. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-2-dpu.pdf

[11] Tolly Group, "NVIDIA Mellanox ConnectX-5 25GbE Ethernet Adapter Adapter Performance vs Broadcom NetXtreme E," no. 219130, Oct. 2019.

[12] Nvidia Corporation, "Nvidia doca sdk data center infrastructure on a chip architecture," 2021, accessed: 2021-01-08. [Online]. Available: https://developer.nvidia.com/networking/doca

[13] Nvidia Corporation, "Nvidia Bluefield Data Processing Units Software-Defined, Hardware-Accelerated Data Center Infrastructure on a Chip," 2021, accessed: 2021-01-05. [Online]. Available: https://www.nvidia.com/de-de/networking/products/data-processing-unit/

[14] NVIDIA Corporation, "Mellanox NIC's Performance Report with DPDK 20.08 Rev 1.1," 2020, accessed: 2020-12-30. [Online]. Available: http://fast.dpdk.org/doc/perf/DPDK_20_08_Mellanox_NIC_performance_report.pdf

[15] S. Miano, R. Doriguzzi-Corin, F. Risso, D. Siracusa, and R. Sommese, "Introducing smartnics in server-based data plane processing: The ddos mitigation use case," *IEEE Access*, vol. 7, pp. 107 161–107 170, 2019.

Debugging QUIC and HTTP/3 with qlog and qvis

Dominik von Künßberg, Benedikt Jaeger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: dominikvon.kuenssberg@tum.de, jaeger@net.in.tum.de

***Abstract*—The powerful properties of the QUIC and HTTP/3 protocols make debugging and inspecting them a challenging task. The qlog format and the qvis toolsuite have been introduced to facilitate this problem. We give an overview of both the format and the visualization tool, introducing and assessing their respective capabilities.**

***Index Terms*—software-defined networks, measurement, high-speed networks**

1. Introduction

The QUIC protocol is a transport protocol designed to offer lower latency for HTTP traffic while also meeting security requirements by encrypting its packets, as described by Langley et al. [1]. Built on UDP, it forms the basis for HTTP/3. Lower latency is attributed to several things: firstly, it streamlines the amount of handshakes needed to establish a secure connection by exchanging cryptographic keys and certificates directly in the initial handshake [1]. It also identifies connections by a connection ID instead of the IP / port 5-tuple, allowing for immediate reconnection to a server after changing IP addresses [1]. To avoid head-of-line-problems like present in the TCP+TLS+HTTP/2 stack, QUIC allows multiple bidirectional streams within a QUIC connection which are independent of each other [1]. Lastly, QUIC packets are entirely encrypted except for fields necessary for routing, forwarding and decrypting the packet [1].

While this is an effective way to protect the packet's data, it makes the protocol difficult to analyze and debug. As there are various implementations of QUIC and its standardization is still ongoing [2], analysis and debugging are indispensable tools to verify the protocol's behavior and find bugs.

Capturing the available metadata from packets in transit alone is not sufficient because fields containing states necessary for analysis are encrypted [3]. Temporarily de- and encrypting the packets while in transit to extract the necessary log information is out of the question as this exposes the full payload and requires the respective session keys [3]. The only places where the later encrypted states are available are the endpoints which send and receive the packets [3]. Logging mechanisms have been implemented, however each is specific to their own implementation of the protocol which makes them difficult to parse [3] for further use.

To alleviate this problem, Marx et al. propose a logging format called qlog [3]. Qlog is based on JSON [3] which allows it to be used across implementations, independent of language-specific characteristics. Each qlog event is characterized by a timestamp, a category, the event type and type-specific data [3]. This format makes it easily extensible: to log a specific type of event which is not yet present, it can simply be added. Qlog files from different connection endpoints can also be aggregated into one single qlog file [3].

Logging events to analyze the performance and behaviour of the QUIC protocol is certainly helpful, however it might be hard to extract the needed information from textual logs only. Because of this, Marx et al. also created the tool qvis to visualize qlogs [3]. This is especially helpful combined with qlog's ability to combine logs from different endpoints; qvis is then able to visualize relations between the endpoints accurately such as packet loss, packet order etc [3].

This paper aims to outline the most important aspects of the qlog format and the qvis visualization tool. We present the qlog format in Section 2 and address how data is collected and its scalability in Section 3. In Section 4, we introduce the qvis toolsuite and its scalability. Section 5 assesses how the qlog format compares to the pcap format commonly used in the TCP+TLS+HTTP/2 stack. In Section 6, we conclude that both qlog and qvis are powerful tools for debugging the QUIC protocol and summarize future plans for qvis.

2. Qlog

The qlog format has so far been defined in two IETF drafts, one describing the general high-level format of qlog [4] and the other defining events specific to QUIC and HTTP/3 [5]. As qlog is a flexible and general format, it can also be used for protocols other than QUIC such as DNS or the TCP+TLS stack by defining the events in the implementation accordingly [3].

Fields inside a qlog file follow a JSON-like format. The basic format is an `object:type` pair. Available standard types are signed and unsigned integers with lengths varying from 8 to 64 bits, floats and doubles, strings, bytes (raw 8 bit long values), booleans, enums and any, which can represent any data type. Additional notations are listed in the Internet Draft for qlog [4].

Every qlog file consists of one top-level file which must contain a `qlog_version` field and an array containing traces [4, Section 3]. Further optional fields can be given such as `title`, `summary`, `description` and `qlog_format` [4, Section 3]. The summary can be useful to get a quick overview of aggregated information about all traces that have been logged, being able to list customizable features such as total lost packets, total number of events and whatever information may be needed in a specific use-case [4, Section 3.1].

2.1. Traces

A trace is a structure which contains the recorded events and additional metadata, however, it usually represents the data flow at a single endpoint [4, Section 3.3]. It must contain a `vantage_point` field to identify which type of endpoint it logged, and an array of events representing all logged events at this endpoint [4, Section 3.3]. Other optional fields allowing for more context are `title`, `description`, and, most importantly, the `common_fields` list [4, Section 3.3], which will be discussed in Section 2.2.

2.2. Events

Each event must at least contain the fields `timestamp`, `name`, and `data` [4, Section 3.4]. Usually it is useful to organize events by assigning them a `group_id`, a `protocol_type` and perhaps a `category` [4, Section 3.4]. Consequentially, fields such as these typically tend to stay the same for the majority of events from the same trace, and thus would need to be constantly logged anew [4, Section 3.4.8]. To avoid unnecessary duplicate data, a trace can contain the `common_fields` list, containing information which is shared by all events of that trace [4, Section 3.4.8]. The mentioned fields can then be omitted in the event itself.

Events can also contain so-called "triggers" in the `data` field [4, Section 3.4.6]. Triggers are a set of possible string values which indicate why an event has occurred [4, Section 3.4.6]. If the event occurs, the applicable trigger string is then included in the log. This gives a direct context to the occurrence of the event and eliminates the need of analyzing logs within roughly the same timeframe to find the reason [4, Section 3.4.6].

The QUIC specific events described in [5] have been divided into three categories: Core, Base, and Extra [5, Section 2.1].

Core events should be present in all qlog files and are used to log very basic information [5, Section 2.1]. Examples of Core events are `packet_sent`, `packet_received`, `version_information` and `packet_lost` [5, Sections 5.3, 5.4.5]. The `version_information` event logs the QUIC versions available for both client and server, as well as the version which has been selected [5, Section 5.3.1].

Base events can depend on Core events but are logged separately for the sake of clarity [5, Section 2.1]. They provide more detailed information which is relevant for debugging. Such events are for example `connection_started`, `packet_dropped`, `packet_buffered`, and `congestion_state_updated` [5, Sections 5.1.2, 5.3, 5.4.3].

Extra events are usually employed to observe the internal behavior of the protocol's implementation, rather than the protocol itself [5, Section 2.1]. Examples for Extra events include `server_listening`, `packets_acked`, `datagrams_sent` and `datagrams_received` [5, Sections 5.1.1, 5.3]. "Datagrams" in this case refers to UDP-datagrams [5, Section 5.3.10].

3. Qlog data collection

How and at which points qlog logs its data is entirely up to the implementation. Any necessary data structures need to be created as well as functions for forwarding and writing information to a qlog file. Coupled with the flexible format of qlogs, it allows for precise logs exactly where it is needed. As an example, the logging of a qlog event in the Go implementation is structured as follows. The file event.go contains and defines all possible events that can be logged [6]. Each event contains the needed and optional attributes which can be set [6]. A struct called `connectionTracer` acts as the trace explained in Section 2.1 [6]. It makes use of Go Channels to record events concurrent to program execution [6]. To avoid race conditions, a mutex is used on the `events` channel so that only one event can be recorded at a time [6]. For instance, when the server sends a version negotiation packet to the client, the `sentPacket` function of the `connectionTracer` is invoked, which in turn records the event and adds it to the `events` channel [6]. This behavior is essentially the same across all functions; when a function is called which necessitates logging, the respective function in the `connectionTracer` is called and adds the event to the log [6]. Upon stopping the server, the aggregated events are written to the qlog [6].

As this means that qlogs are held in memory and only written to the disk when the connection is terminated, this approach might cause unwanted occupation of memory when logging a large volume of events. As an example, the large demonstration file on the qvis website [7] representing a 100 MB download is 31 MB in size, while the qlog file for a 500 MB download mentioned in [3] is 276 MB. Assuming this can be scaled roughly linearly, logging a 10 GB download will then result in a qlog which is somewhere between 3,1 and 5,5 GB in memory before the connection is terminated. It is therefore important to keep this memory occupation in mind and evaluate which events actually need to be logged to minimize the resulting log size, especially when downloading and logging large quantities of data.

3.1. Scalability

In [3], Facebook employed qlog at internet scale and concluded that it "is two to three times as large" and "takes 50 % longer to serialize than their previous in-house binary format." In Facebook's case, this processing surplus was acceptable given the flexibility provided by qlog [3].

To compress qlog's size requirements while preserving the format's desirable properties, an optimized mode [3] was introduced. It relies on two aspects: reducing the initial size of qlogs and encoding the smaller qlogs more efficiently [3]. The former is accomplished by collecting repeated values in a dynamic dictionary [3]. The latter is achieved by using the CBOR (Concise Binary Object Representation) format to encode the qlogs and the generated dictionary [3]. CBOR is a binary format which preserves

JSON's key-value pairs in a concise manner and allows for faster processing than JSON [3].

The combination of these two methods results in significantly smaller file sizes. The qlog for a 500 MB download is usually 276 MB; utilizing the optimized mode results in a file about a third as large as the original one, ending up at 91 MB [3].

4. Qvis

Qvis encompasses a set of tools which visualizes qlog files and their data in an understandable and descriptive manner. It can handle qlog files which contain traces from several endpoints to deduce and display information from the provided data, such as round trip time, congestion control and more [7]. Qvis offers four visualization methods: sequence, congestion, multiplexing and packetization [7]. It also lists general statistics about the provided qlogs such as the number and types of events and frames [7]. Qvis is implemented mainly in TypeScript and Vue and intended to be used in a browser [7]. Scalability issues arising from this are discussed in Section 4.5.

4.1. Sequence Tool

The sequence tool generates a sequence diagram as shown in Figure 1a. The green squares on both sides represent events. If the event is neither `packet_received` nor `packet_sent`, the event name is added next to it [7]. Besides displaying the information contained in transmitted packets and their respective timestamps, all the green boxes, event names and packet information can be clicked which brings up the corresponding qlog file in plaintext, allowing for further, more detailed packet inspection [7].

4.2. Congestion Tool

The congestion tool shows two diagrams: one which shows the amount of data sent over time in bytes, and one displaying the round trip time [7]. What first appears as a slightly jagged line in the first diagram becomes clearer when zooming in; it shows the bursts of data being sent in blue and the acknowledgement of that data in green [7], as shown in Figure 1b. The gap between the blue and green blocks on the same height on the y-axis constitutes the round trip time [7]. The congestion tool, therefore, makes it easier to identify when data is being sent at a different rate indicated by a change of the slope of the graph [7]. It can also display crucial information such as the congestion window size and lost data [7].

4.3. Multiplexing Tool

The multiplexing tool shows how the data sent was divided among the existing QUIC streams (shown in Figure 1d) [7]. It assigns a color to each stream and displays the sent data as colored blocks strung along the timeline, each colored block indicating that the corresponding stream has been used to transmit data [7]. It also indicates which frames had to be resent underneath the corresponding parts of the diagram [7]. This makes it simple to identify unwanted behavior in the applied multiplexing strategy [7]. It is also possible to zoom into the string of blocks and hover over them to display the exact timestamp, utilized stream, number and packet size of the block that is being pointed at [7]. This is especially helpful when inspecting large qlogs.

Additionally, it is possible to enable two more supplementary diagrams: the waterfall and byterange diagram (waterfall diagram shown in Figure 1c) [7]. The waterfall diagram displays a colored bar for each stream between the first and last time it received a frame [7]. This makes it easier to determine roughly when a stream was active, especially when a large number of frames was transmitted [7]. The byterange diagram displays the range of bytes transmitted by the frames which are shown at the current zoom level [7].

4.4. Packetization Tool

The packetization tool visualizes how QUIC packets are composed of QUIC frames and HTTP/3 frames (shown in Figure 1e) [7]. Each layer represents one structure: in ascending order, those are QUIC packets, QUIC frames, HTTP/3 frames and the stream IDs present in the corresponding packet [7]. Headers within packets and frames are represented by taking up half the height of the line compared to the payload. Packet / Frame boundaries can be discerned by the alternating colors within each layer [7]. As with the other tools, it is possible to zoom in on a specific spot and hover over it to view packet or frame information [7].

4.5. Scalability

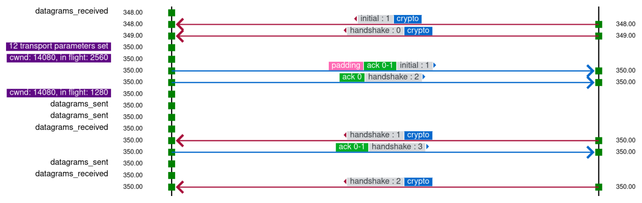
While it is possible to load large files in qvis and the authors of qvis describe in [3] that qvis "scales to loading hundreds of MB in JSON", it significantly impacts the performance of the tool. It is recommended to use a Chromium-based browser, as using another might affect performance even more [8].

Loading the demonstration file of 31 MB representing a 100 MB download [7] is certainly possible but switching between the different tools, using the zoom function to view packet details and other actions noticeably slow down the web browser. Tools especially affected are the sequence, multiplexing and packetization tools.

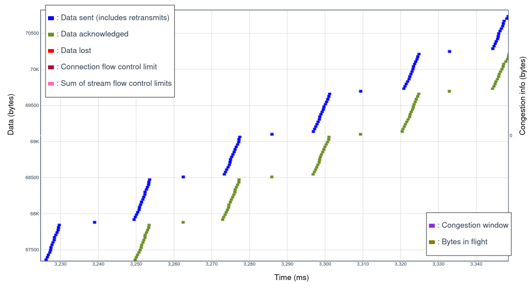
We observed that the sequence tool initially takes between 10 and 15 seconds to load the entries. However, once everything is loaded, the tools work perfectly fine.

The packetization tool also takes about the same time to initially load as the sequence tool. The congestion tool is the quickest to respond of all tools, the zoom works without delay. This is due to the fact that it uses canvas-based rendering [8].

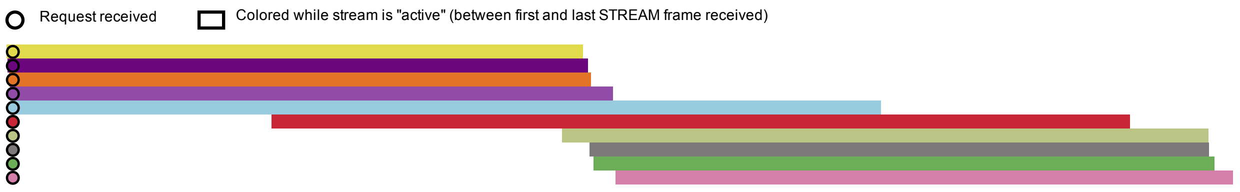
Both the multiplexing and packetization tools share a performance issue with large files concerning the zoom function. Zooming in becomes more important as qlogs get bigger to analyze sections of the graph more closely. To dissect this issue, it is helpful to analyze how the depiction of the diagrams is implemented. Both tools use rendering of scalable vector graphics (SVG) to display the diagrams [8]. Each packet / frame is a separate SVG entity [8]. When zooming in or out, the dimensions of every entity has to be recalculated, which is slow with



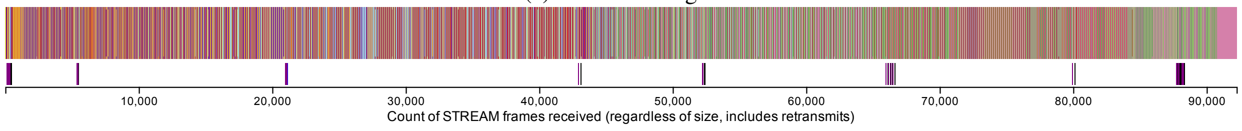
(a) Sequence diagram.



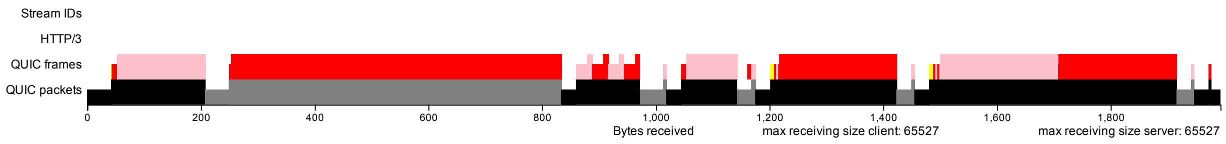
(b) Congestion diagram.



(c) Waterfall diagram.



(d) Multiplexing diagram.



(e) Packetization diagram.

Figure 1: Qvis diagrams.

such big qlogs as this results in tens of millions of SVG entities being resized [8]. Therefore, this large workload is especially noticeable when using the packetization tool, as it contains far more SVG entities than the multiplexing diagram [7].

The SVG rendering approach was chosen because hover effects to display packet / frame information is easier to implement this way compared to the canvas-based rendering used by the congestion tool [8]. However, there are plans to port the remaining tools to the same rendering method for performance reasons [8].

5. Comparison with TCP+TLS+HTTP/2

The TCP+TLS+HTTP/2 (TTH) stack is most commonly debugged with tools such as Wireshark [9] or tcptrace [10]. For this purpose, the log consists of timestamps and the captured packets exactly as they were represented during transmission [3]. As the TTH stack shows most information necessary for debugging in the (unencrypted) headers of the packets, this is sufficient. For QUIC, this would not work as important metadata for debugging purposes such as frame numbers, frame type and stream IDs are in the encrypted section of the packet [3]. This makes a direct analysis of packets similar to that of the TTH stack regarding these properties infeasible.

In terms of log size, pcap files created with e.g. Wireshark can be of varying size depending on the applied options. With default settings, the pcap file for a 500 MB download will exceed 500 MB, as all packets are directly ingested into the log file. However, there are options to limit the capture size of each incoming packet [11], dropping most of the payload. This can dramatically decrease the file size. Measurements showed that when downloading a 500 MB file using Wireshark with default settings results in a pcap file of 550 MB. Restricting the size of each logged packet to 100 B to account for headers, the pcap file size drops to 65 MB.

The qlog file for a 500 MB download is 276 MB or 91 MB [3] when using the optimized mode explained in Section 3.1. This is evidently a noticeable difference in size.

Despite this, qlog has the advantage of offering a more detailed analysis of internal variables such as congestion window, lost packets and bytes in flight [3] in comparison to TCP traces and being able to visualize them accordingly with qvis via the congestion tool [3].

6. Conclusion

Qlog is a powerful logging tool which has tremendous potential for debugging internet protocols, QUIC in particular. Its ability to define custom events and to combine multiple traces into one qlog, paired with the terrific visualization capabilities of qvis, makes it a solid basis for anyone debugging QUIC.

Porting qvis to native code for better performance is currently not planned by the original developers of the tool [8]. While one of the goals is to write qlog importers for existing native tools such as Windows Performance Analyzer, this is not planned for the immediate future [8]. However, the performance issues due to SVG rendering are being worked on as it is planned to convert the respective tools to canvas-based rendering [8].

References

[1] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. B. Krasic, C. Shi, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. C. Dorfman, J. Roskind, J. Kulik, P. G. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, and W.-T. Chang, “The QUIC transport protocol: Design and internet-scale deployment,” 2017.

[2] D. Madariaga, L. Torrealba, J. Madariaga, J. Bermúdez, and J. Bustos-Jiménez, “Analyzing the adoption of QUIC from a mobile development perspective,” in *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, ser. EPIQ ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 35–41. [Online]. Available: https://doi-org/10.1145/3405796.3405830

[3] R. Marx, M. Piraux, P. Quax, and W. Lamotte, “Debugging QUIC and HTTP/3 with qlog and qvis,” in *Proceedings of the Applied Networking Research Workshop*, ser. ANRW ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 58–66. [Online]. Available: https://doi.org/10.1145/3404868.3406663

[4] R. Marx, “Main logging schema for qlog,” Internet Engineering Task Force, Internet-Draft, 2020, work in Progress. [Online]. Available: https://quiclog.github.io/internet-drafts/draft-marx-qlog-main-schema.html

[5] ——, “Quic and http/3 event definitions for qlog,” Internet Engineering Task Force, Internet-Draft, 2020, work in Progress. [Online]. Available: https://quiclog.github.io/internet-drafts/draft-marx-qlog-event-definitions-quic-h3.html

[6] L. Clemente, “A QUIC implementation in pure go,” 2020. [Online]. Available: https://github.com/lucas-clemente/quic-go

[7] R. Marx, “qvis: tools and visualizations for QUIC and HTTP/3,” 2020, https://qvis.edm.uhasselt.be/.

[8] ——, “Qvis performance,” 2020. [Online]. Available: https://github.com/quiclog/qvis/issues/38

[9] “Wireshark,” 2020. [Online]. Available: https://www.wireshark.org/

[10] “Tcptrace,” 2020. [Online]. Available: https://linux.die.net/man/1/tcptrace

[11] “Wireshark documentation,” 2020. [Online]. Available: https://www.wireshark.org/docs/wsug_html/

Recent Developments in Service Function Chaining

Patricia Horvath, Kilian Holzinger, Henning Stubbe*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: patricia.horvath@tum.de, holzinger@net.in.tum.de, stubbe@net.in.tum.de

***Abstract*—A network's infrastructure consists of multiple network functions some of which may include distinct intermediate steps that need to be applied in a particular order.**

In order to ensure the right order of application while still striving for goals such as improving the network's flexibility, improving its independence from physical structure and reducing infrastructure complexity, Service Function Chaining (SFC) is applied.

The resulting service function chains are comprised of an ordered set of network functions that are applied to data packets handled in the network. This technique is beneficial for use in fixed broadband networks, where it is deployed behind the broadband network gateway, as well as in mobile networks for optimization of services such as TCP, and in data centers. This paper gives an in-depth overview over the basic functionality of network function chains and compares the advantages and drawbacks of various implementations for different use cases.

***Index Terms*—network architecture, service function chaining, monitoring, load balancing, service function**

1. Introduction

The infrastructure of a network heavily relies on network services which are applied to the data packets that are being transmitted within the network. These network services are comprised of multiple network functions which may be hardware components or implemented as virtual components. Examples for such network services include firewalls, load balancing, parental control, network address translation and deep packet inspection to name a few.

Grouping these service functions in service function chains (SFC) ensures that the order in which the functions are applied remains unchanged. More specifically, using SFC also allows for an optimization of the network's configuration flexibility, for more independence from its hardware implementation and for less complexity, as the particular order of a set of network functions can be adjusted dynamically to adapt to any necessary changes or external influences, for example in the case of a malicious attack on the network.

In this paper, use cases of the SFC architecture as well as present available implementations, especially in the field of the Internet of Things (IoT), are discussed, while also giving an overview over possible challenges such as monitoring and load balancing and highlighting current trends. The remaining content of this paper is organized as follows: Section 2 provides the background on SFC architectures and describes their basic functionalities. Section 3 discusses use cases and available implementations of SFC in fixed broadband networks, in mobile networks and in data centers as well, while Section 4 examines possible challenges for SFC that are encountered by different implementations and their solutions. Finally, Section 5 summarizes the above content and concludes this paper.

2. Architectural Theory of SFC

SFCs are comprised of an ordered set of multiple service functions (SF) which are applied to packets and specify if packets should be, for example, directed to a firewall or a caching engine as described in RFC 7665 [1]. Additionally, the mechanism to express results of applying a more granular policy and constraints to the abstract constraints is called Service Function Path (SFP), some of which may be vague. It is noteworthy that an SF can be part of multiple SFCs and SFPs.

The logical core components of an SFC are classifiers, Service Function Forwarders (SFFs), the SFs themselves, and SFC proxies which are interconnected by SFC encapsulation which, although is not a transportation encapsulation itself, is the mechanism that allows for a SFP selection while sharing metadata or context information if required and carrying explicit information to identify the SFP.

The service functions a SFP contains may also be altered and result in selecting a new SFP or an update of the related metadata, which is the result of a process called "reclassification". If both of the above occur, this process is specified as "branching". Reclassification and branching are especially needed, if an attack on the traffic within the network has been detected. In this case, traffic can be rerouted to a new SF, e.g. a firewall, to be able to enforce security policies. Keep also in mind that any network transport may be used to carry SFC encapsulated traffic [1].

Beyond that, a tool called Service Function Forwarder is needed to control the flow of packets within the network. A SFF's responsibilities include forwarding packets and frames to one or more SFs associated with a given SFF by utilizing the information transmitted in the SFC encapsulation, terminating SFPs when all required SFs of a SFC have been passed and maintaining the flow state of an SFF, as they may be stateful. A SFC can be abstracted as a graph, where each node signifies a SF.

As such, SFCs may contain cycles and may be unidirectional, in which case the SFC has an ordered path,

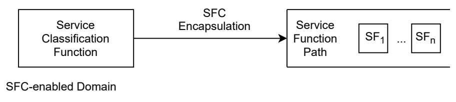


Figure 1: A diagram of a SFP as described in RFC 7665 [1].

or bidirectional, in which case the traffic is symmetric. Furthermore it is necessary to differentiate between SFC-aware and SFC-unaware SFs as SFC-unaware SFs need a SFC Proxy to function as a gateway to the SFC encapsulation by using a local attachment circuit to deliver packets to SFC-unaware SFs. In addition to enabling SFC-unaware SFs to be able to function in a SFC architecture, SFC proxies are also handling the removal of SFC encapsulation.

The SFC-enabled domain contains all SFC Proxies and their corresponding SFC-unaware SFs. Furthermore, the SFC control plane is responsible for managing the resources of the SFC architecture. The responsibilities entail constructing SFPs, translating SFCs to forwarding paths and propagating path information to participating nodes, e.g. SFs.

Further utensils for SFC Operations, Administration and Maintenance (OAM) are subject to ensure fault detection and isolation, as well as performance management and are either in-band, meaning OAM packets are handled the same way user packets are handled, or out-of-band, meaning the tools function in a layer beyond the actual data plane [1].

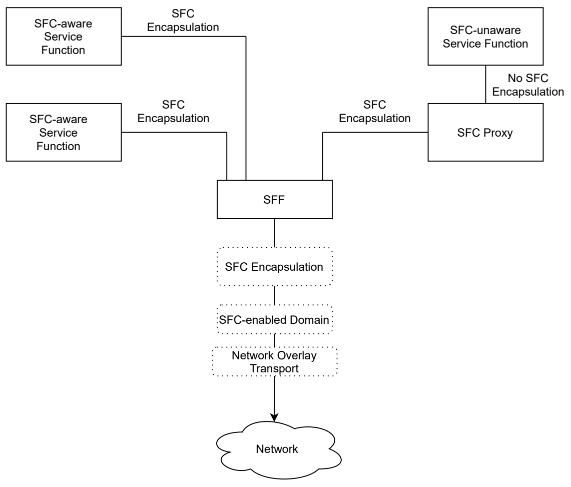


Figure 2: A diagram of the components of the SFC architecture after the initial classification as described in RFC 7665 [1].

3. Analysis of SFC

The following section introduces the reader to use cases of SFC and gives examples for available implementations.

3.1. Use Cases of SFC

The SFC architecture is used in multiple environments which include fixed broadband networks, data centers [2] and cloud customer premises equipment [3].

Fixed broadband network are accessed by their users commonly using technologies such as DSL, Ethernet or Passive Optical Networks, whereas in mobile networks, the Internet is accessed by using SFs (sometimes referred to as "enablers") [2].

In fixed broadband networks, SFC is responsible for services such as Deep Packet Inspection (DPI), NAT44, which is an extension to Network Address Translation, DS-Lite, a tool allowing applications which use IPv4 to access the internet via IPv6, NPTv6, a technology allowing IPv6 to IPv6 Network Prefix Translation, parental control, firewall, load balancer and cache. In mobile networks many SFs are implemented in the Gi interface, which is a reference point between the Gateway GPRS Support Node and an external Public Data Network [4]. Examples for SFs encompass functions such as DPI, billing and charging, TCP optimization, web optimization and video optimization. The answer as to why SFC are used is being able to facilitate resource optimization and a seamless service switchover from one network to the other. Additionally, SFCs are also used to facilitate addressing convergence needs [2]. These network services are comprised of multiple network functions which may be hardware components or implemented as virtual components. Using SFCs is also beneficial in data centers. Traffic flow in data centers can be categorized as either north-south traffic, where traffic originates from outside the data center, or as east-west traffic, where all traffic originates from within the data center [2].

A simple example for north-south traffic is given when a remote worker accesses a specific data center server resulting in incoming traffic for the data center. As you can see, north-south traffic generates the need for traffic analysis, identification of application and its users, authorization of transactions and mitigation and elimination of security threats since communication partners are outside of the data center and as such unknown and potentially dangerous. To be able to fulfill these needs, SFCs are implemented in permutations of service nodes through which the traffic has to flow. Permutations of the service nodes are necessary because not every present service function is suitable to be applied to a certain type of traffic and vice versa. For example, certain SFs are not able to be applied to virtual private network traffic. Furthermore within the network of a data center, SFCs can be either classified as Access SFCs or as Application SFCs depending on the destination of the data packets to whom the SFCs are applied, as highlighted in [2]. Access SFCs assist traffic which enters and leaves the data center, thus making such SFCs suitable for north-south traffic, whereas Application SFCs service traffing destined to applications, which is suitable for east-west traffic.

The following example helps illustrate east-west traffic: In a three-tiered architecture, requests come to the webserver which trigger interaction with the application servers. In turn, interaction with the database servers is triggered and SFs are then applied to enforce security policies between the tiers, while monitoring SFs enable visibility into the application traffic [2]. Along with the above mentioned use cases, there is also the scenario of a SFC architecture consisting of centralized value-added SFs, which are configured by subscribers and enabled in the network side, while the subscriber side box is limited to only Layer 2 and Layer 3 functionalities, which is the case when using Cloud Customer Premises Equipment (CPE). In this case, Cloud CPE translates the subscribers' service requests into SFCs [3]. The ability to reconfigure SFPs as needed allows for pay-per-use and on-demand server-side services.

3.2. Available Implementations

Various frameworks and policies exist where SFC is used. The following section highlights some of these to give some insight into networks with SFC architecture.

One growing field where SFC can be applied to is IoT. In [5], the authors propose using the SFC orchestration system PRSFC-IoT to meet IoT providers' demand of being able to process both the increasing traffic amount and the increasing amount of varied IoT traffic requirements. SFC orchestration can be beneficial to optimize performance and resource consumption. To fulfill the above goals, the authors suggest the use of PRSFC-IoT considering that it encompasses functionalities that enable meeting deadlines while guaranteeing a stable packet rate including efficient resource management.

SFCs are also used in edge computing where high flexibility and low latency are important in regards to the quick execution of various functionalities. In [6] an online orchestration framework for cross-edge SFCs, which strives to improve cost efficiency by improving both resource provisioning and traffic routing, is introduced. In this approach, SFs are split into edge clouds to mitigate the cost of dynamically launching new SFs.

To summarize, SFC is highly beneficial to environments where networks need to be able to adapt to changes quickly and dynamically.

4. Challenges in SFC Architectures

This section discusses some of the prevalent challenges in various SFC architectures and possible solutions.

4.1. Flexibility

Scalability, which results in more flexibility, is an import requirement for SFC architectures as it is needed to be able to accommodate changing demands of the user side, for example in data centers where the number of clients accessing server-side services may change drastically. When using static SFCs, there may be a need to readjust the service nodes by adding or removing some to be able to accommodate new obligations. In other words, static SFCs cannot scale well. Furthermore, SFCs cannot pass metadata which is needed to enforce policies consistently across all of the network. Beyond the above problem, physical and static SFC mechanisms cannot be mixed with virtual and dynamic SFC mechanisms which is problematic as one cannot use the benefits of both implementations at the same time [2].

Another issue regarding flexibility occurs when managing large scale, multi-region data centers with multiple operational teams. In [7], the authors propose an implementation of hierarchical SFC using OpenDaylight platform in a SDN environment to fulfill this demand. The

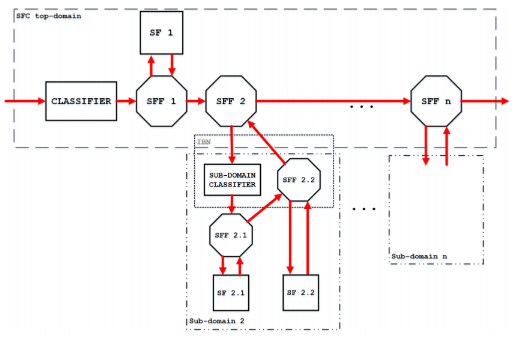


Figure 3: A diagram of a hierarchical SFC architecture [7].

OpenDaylight platform is an open-source project which enables modularity, flexibility, scalability in a multiprotocol SDN controller infrastructure.

4.2. Monitoring

Another functionality which plays a big role in SFC deployments is the monitoring of traffic as it is needed for quality and congestion control as well as anomaly detection and capacity planning among other things.

As there is no standard method of monitoring and detecting failure, the authors in [8] have implemented an alarm-based monitoring with a focus on high availability in SFC for cloud environments using the OpenStack project API and the ODL driver in Tacker. In general, a Virtual Network Function (VNF) Manager's tasks include VNF instantiation, updating and upgrading software, modification, collecting performance measurement results as well as information regarding events and faults, terminating instances when needed and lastly managing the integrity of the VNF instance. Tacker, which is a Generic VNF Manager and NFV Orchestrator and OpenStack project based on ETSI MANO Architectural Framework, deploys and operates Network Services and VNFs and can add functions such as monitoring and auto healing [9]. As Tacker needs a monitoring tool to be able to perform the above introduced functionalities, an alarm-based monitoring tool is needed. Due to alarms being related to hardware resources such as CPU and memory usage, Tacker is used in combination with Openstack Ceilometer, which is responsible for collecting measurements within OpenStack [8].

VNFs are built in such a way that each VNF can be identified with an unique VNF ID which aids in VNF failure detection. Additionally, SFC reliability can be achieved by means of using Ceilometer as a monitoring driver which sends a notification message to the SFC driver after analyzing the alarm message [8].

Another approach to improve network monitoring with the goal for this implementation to be adjustable to recent and future technologies is proposed in [10]. The birth of trends such as network function virtualization (NFV), software-defined networking (SDN) and cloud computing, led to the demand to monitor different planes inside a network, such as the (virtual) infrastructure plane, the user plane and the service plane, having grown considerably. As proposed in [10], using a monitoring framework, that is able to control monitoring in a network as well as in a cloud infrastructure, is favorable to combine the operators' need to monitor end user subscribers' traffic as well as tenant customers' traffic.

As a result, monitoring is not hindered by cloud, physical and virtual boundaries anymore. Moreover, the authors in [10] also recommend classifying traffic flows based on Layer 4-7 information and controlling these flows with a different monitoring approach with the assistance of the new network service header (NSH) and the metadata it carries. The monitoring process of the above mentioned framework is logically centralized. Furthermore, the framework enables for probes to be consolidated based on monitoring requests and for the monitoring of rule consolidations which possibly facilitates the use of monitoring resources in a more efficient manner. Moreover, the authors suggest using Layer 4-7 information to use classifiers to be able to mark packets that pass along the SFPs. The classifier then has the ability to modify the packet's NSH.

In addition, the authors in [10] introduce three possible implementations for the markers: Using coloring markers, which requires a new metadata header that is capable of carrying the color of the packet and the point of time when the marking was done, or timestamp markers, which entails the NHS to carry a new metadata type-length-value that is used to save the timestamps from each probe, and finally, using interception markers, where the intercept metadata is used to gather parts of the packet headers while on their packet data paths. Also, the framework is able to use SFs as well to re-classify or re-mark packets and to modify the metadata field of the NSH. Further, the authors in [10] discuss the classifier function to be adjustable in regards to the deep packet inspection (DPI) function being applied at all. If there is no DPI function, a separate classifier is responsible for a shallow packet inspection, meaning a Layer 2-4 inspection. Otherwise, there is the possibility of using indirect DPI-classifier communication, direct DPI-classifier communication or using an integrated DPI-classifier, where the classifier is part of the DPI engine and the Layer 4-7 classification and marking of the packets is done by the same entity.

Yet another approach to improve monitoring is introduced in [11] which proposes employing in-band network telemetry optimization for NFV service chain monitoring, specifically using a scalable telemetry system called IntOpt that uses active probing, thus making this technique especially effective in dynamic service chaining architectures. Furthermore, this telemetry system also enables specifying monitoring requirements for individual service chains, which are mapped to telemetry item collection jobs. A SDN controller creates the minimal number of monitoring flows needed to monitor the deployed service chains as per the telemetry demands. Moreover, the simulated annealing based random greedy metaheuristic (SARG) is utilized to minimize the overhead caused by active probing and collecting of telemetry items. The IntOpt controller then determines the set of optimal monitoring flows which minimize the total overhead of the network monitoring through use of the SARG approach. In this way, the optimal probing frequency as well as the total number of telemetry items to be monitored for each link in order to cover all service flows with minimal overhead at the data plane as well at the controller is calculated. Reducing the overhead caused by the monitoring allows for dynamic service chaining architectures to function more efficiently as less resources are pointlessly used which can the be rather used by other important tasks so the network retains its fast response time. Next, the the proper telemetry sources, forwarders as well as sinks are calculated by the controller, thus filling in the flow tables. The authors have ascertained that using the above heuristic considerably reduces the total monitoring overhead, including the delays induced by the telemetry operations. Moreover, [11] explains that such a systematic technique can be incorporated with the existing monitoring frameworks to achieve high scalability without losing the generality and expressiveness of the systems.

4.3. Load Balancing

Another important issue that has to be accounted for is load balancing for optimizing response times and making the network work more efficiently. As shown in [12], a joint network and server load balancing algorithm for chaining VNFs is proposed for this purpose. As network load balancing and server load balancing, e.g. in data center environments, are both two separate issues that need to be addressed, the authors propose an algorithm called Nearest First and Local-Global Transformation (NF-LGT) which can address both issues. The above algorithm consists of two phases that are executed consecutively. The first phase involves constructing service chains by a greedy strategy which considers both network latency and server latency. The strategy encompasses choosing the VNF whose latency from the current location is the smallest as the next destination, repeating this process iteratively, until the service chain includes all required NFs. Afterwards, the second phase commences which involves applying a searching technique to improve the result of phase 1. This is accomplished by attempting to find a better service chain, in regards to possessing smaller latency, by replacing a selected VNF with another candidate and swapping the order of VNFs in the SFC. The authors propose using a SDN/OpenFlow concept to implement the above explained algorithm, specifically separating the control plane and data plane from each other. To provide evidence for the superiority of this approach, [12] shows the results of benchmarking tests demonstrating that NF-LGT improves the system bandwidth utilization by up to 45%.

In conclusion, to be able to fully be advantageous, especially in a dynamic context where high and fast adaptability is needed, the main issues in SFC architectures that need to be tackled are monitoring and load balancing as described above. Please keep in mind that the above highlighted implementations are merely an overview over possible solutions and that research in regards to these matters is still being conducted while also touching on adjacent topics that surpass the limits of this paper.

5. Conclusion

In this paper, I presented the fundamentals of service function chains and their composition. SFCs consist of ordered service functions that are consecutively applied to the frames that are being passed through the network. These functions include vital services such as firewalls and caching. SFCs are hugely beneficial to (mobile) networks as well as data centers in particular because they improve latency and promote efficient resource management, modularity and scalability, which are indispensable attributes for managing high volumes of traffic.

Indeed, service function chaining including network function virtualization is because of its dynamic properties of high value for many different application implementations but using service function chains also comes with challenges that need to be addressed to achieve the full potential of the above-mentioned benefits in dynamic environments. In particular, optimizing the load balancing of traffic is necessary, even more so to be able to process the aforementioned high volumes of traffic, to be able to retain response times that are as short as possible, as well as being able to efficiently monitor the network traffic, which is an imperative requisite to be able to properly analyze traffic in order to adapt accordingly to any changes.

References

[1] J. Halpern and C. Pignataro, "Service function chaining (sfc) architecture," Internet Requests for Comments, RFC Editor, RFC 7665, October 2015. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7665.txt

[2] S. Kumar, M. Tufail, S. Majee, C. Captari, and S. Homma, "Service function chaining use cases in data centers," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-sfc-dc-use-cases-06, February 2017, http://www.ietf.org/internet-drafts/draft-ietf-sfc-dc-use-cases-06.txt. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-sfc-dc-use-cases-06.txt

[3] W. Liu, H. Li, O. Huang, M. Boucadair, N. Leymann, Q. Fu, Q. Sun, C. Pham, C. Huang, J. Zhu, and P. He, "Service function chaining (sfc) general use cases," Working Draft, IETF Secretariat, Internet-Draft draft-liu-sfc-use-cases-08, September 2014, http://www.ietf.org/internet-drafts/draft-liu-sfc-use-cases-08.txt. [Online]. Available: http://www.ietf.org/internet-drafts/draft-liu-sfc-use-cases-08.txt

[4] J. Korhonen, J. Soininen, B. Patil, T. Savolainen, G. Bajko, and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)," Internet Requests for Comments, RFC Editor, RFC 6459, January 2012.

[5] J. Wang, H. Qi, K. Li, and X. Zhou, "PRSFC-IoT: A Performance and Resource Aware Orchestration System of Service Function Chaining for Internet of Things," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1400–1410, 2018.

[6] Z. Zhou, Q. Wu, and X. Chen, "Online Orchestration of Cross-Edge Service Function Chaining for Cost-Efficient Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1866–1880, 2019.

[7] A.-V. Vu and Y. Kim, "An Implementation of Hierarchical Service Function Chaining using OpenDaylight Platform," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. IEEE, 2016, pp. 411–416.

[8] L. Yang, D. Van Tung, M. Kim, and Y. Kim, "Alarm-based Monitoring for High Availability in Service Function Chain," in *2016 International Conference on Cloud Computing Research and Innovations (ICCCRI)*. IEEE, 2016, pp. 86–91.

[9] S. Ramaswamy and Brocade. (2015) Tacker: VNF Lifecycle Management and Beyond. IETF. [Online]. Available: https://www.ietf.org/proceedings/93/slides/slides-93-nfvrg-25.pdf

[10] M. Shirazipour, H. Mahkonen, M. Xia, R. Manghirmalani, A. Takacs, and V. S. Vega, "A Monitoring Framework at Layer4–7 Granularity Using Network Service Headers," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. IEEE, 2015, pp. 54–60.

[11] D. Bhamare, A. Kassler, J. Vestin, M. A. Khoshkholghi, and J. Taheri, "IntOpt: In-band Network Telemetry Optimization for NFV Service Chain Monitoring," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[12] M.-T. Thai, Y.-D. Lin, and Y.-C. Lai, "A Joint Network and Server Load Balancing Algorithm for Chaining Virtualized Network Functions," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

EDNS NSID Option

Christian Kilb, Johannes Zirngibl*, Patrick Sattler*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: christian.kilb@tum.de, zirngibl@net.in.tum.de, sattler@net.in.tum.de

Abstract—**The DNS Name Server Identifier (NSID) Option is a DNS extension that helps disambiguate name servers which share IP addresses in anycast setups. NSID is useful for DNS analysis by name server administrators and researchers. This paper evaluates the usage of NSID by 89k name servers authoritative for 533k Alexa Top Sites. It analyzes in particular how widespread its usage is and how identifiers are chosen. The evaluation shows that about a third of name server IP addresses provide NSID support and two-thirds of Alexa domains have at least one name server with support. 81% of observed NSID values are valid UTF-8 strings. Every third NSID value is a domain name. In two out of three cases, these domains resolve back to the name servers' IP addresses.**

Index Terms—**Domain Name System, EDNS, NSID, network measurement**

1. Introduction

In DNS anycast setups, multiple name servers share a single IP address. DNS queries to that IP address are then answered by one of the name servers, for low latency ideally by one in close physical proximity to the client. The DNS response however does not make it apparent to the client which concrete anycast name server handled the query. When researching or debugging anycast, it might be useful or necessary to learn the identity of the responding name server. The DNS Name Server Identifier (NSID) Option, specified in RFC 5001 [1], is a DNS extension that standardizes such a mechanism. It allows name servers to include a name server identifier in their DNS responses, if requested by a client. This NSID value can then be used to disambiguate the anycast name servers.

This paper contributes a usage evaluation of the NSID DNS extension. The analysis addresses the question of how widely it is supported by the most popular name servers. It also evaluates how the NSID values are chosen and whether they follow certain patterns, which is of interest because NSID values are specified as arbitrary byte strings. An NSID dataset obtained from a DNS scan of the Alexa ranked domains is the basis for the usage evaluation.

In Section 2 of this paper, background information about NSID and its requirement EDNS (Extension Mechanisms for DNS) is given. Afterwards, related concepts and work are outlined in Section 3. In Section 4, a DNS scan dataset is evaluated and the usage of EDNS and NSID is analyzed, with a focus on NSID. Finally, a conclusion is drawn and future work is suggested in Section 5.

2. Background

Should name server administrators choose to support the NSID Option, they first have to provide EDNS support, on which NSID is built.

2.1. EDNS

The "Extension Mechanisms for DNS" (EDNS) [2] extends DNS [3] in multiple ways. It increases the maximum DNS message payload size over UDP from 512 B to 65 535 B. It also extends the number of possible return codes and flags.

Classical DNS messages over UDP have a fixed maximum payload size of 512 B. DNS over TCP could be used to circumvent this size limit, which however would be inefficient for single DNS request-response exchanges, as a TCP handshake would have to be performed. EDNS was therefore created to allow for efficient DNS messages over UDP with extended limits in a backward-compatible manner.

EDNS in its version 0 defines a new (pseudo) resource record called "OPT". It contains meta information, but no actual DNS data. DNS clients that support EDNS can include an OPT resource record in the "additional data" section of their request. A DNS server with EDNS support would then process it accordingly and add a corresponding OPT record to its response.

The format of the OPT resource record is shown in Figure 1. Some resource record fields have a different meaning compared to regular DNS records. The `NAME` field always has value 0 to indicate the root domain. A type value of 41 has been assigned to the OPT resource record, to which the `TYPE` field is set. In the reinterpreted `CLASS` field, the requestor specifies the maximum UDP response payload size it is able to receive. The extended return code, EDNS version number and extended flags are embedded in the reinterpreted `TTL` field. The last resource record field `RDATA` contains a list of "options" in the form of attribute-value pairs. The size of the `RDATA` field is found in the preceding field `RDLEN`.

Figure 2 shows the format of an option. They consist of the fields `CODE`, `LENGTH` and `DATA`. The length field specifies the size of the option data. If a DNS client includes an option in its OPT resource record and the DNS server understands it, a corresponding option will be included in the response. Unsupported option codes would be ignored instead.

One example for an extension to DNS that builds on EDNS is the DNS security extension DNSSEC [4]. It

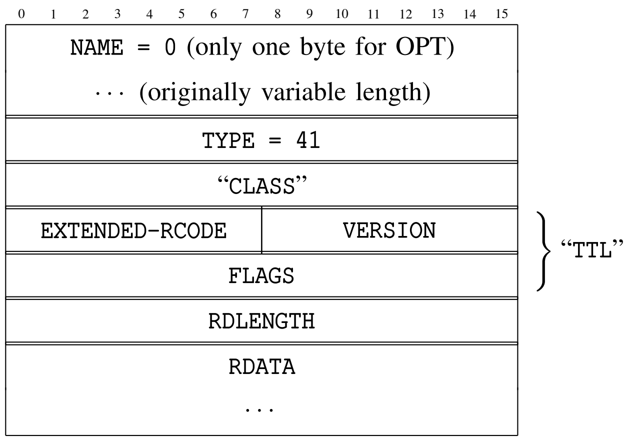


Figure 1: OPT Resource Record Format

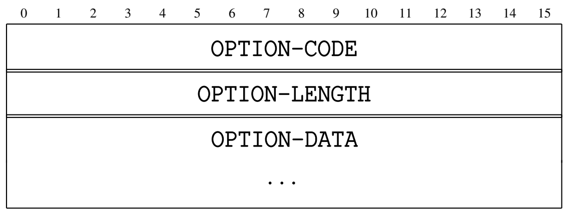


Figure 2: OPT Option Format

introduces new resource records that hold security-related information and also allocates a new EDNS flag in the `FLAGS` section of the OPT resource record.

2.2. NSID

With DNS anycast, multiple name servers can share a single IP address. In such a scenario, the IP address is not enough to tell which of the name servers responded to a query. In order to reliably learn the identity of the responding name server, it would have to include a server identifier directly in the response.

The DNS Name Server Identifier (NSID) Option [1] is a DNS extension that realizes this. With an NSID in the DNS response, clients can disambiguate the name servers with shared IP addresses. This can be useful for debugging DNS problems as well as for internet research.

NSID is realized as an EDNS Option. It is assigned the option code 3. DNS clients can request an NSID by including an EDNS Option with code 3 and empty option data in their request. Should the DNS server support NSID, it includes its identifier in the response, embedded in the `OPTION-DATA` field.

The meaning of NSID values in this option data field is however undefined. NSIDs are specified as raw byte strings or sequences of hexadecimal digits. It is up to the name server administrators to decide on the meaning of NSID values for their servers. Several suggestions are given in the RFC for the NSID meaning. Possible choices are a "real" host name or IP address, a static identifier derived from the name or IP address, a dynamically generated identifier or an encrypted identifier. The administrators can therefore decide whether the NSID should be meaningful for everyone or only for a specific group of people, such as themselves. Should the value be meaningful for everyone, an appropriate encoding must also be chosen, for example UTF-8.

NSID is a hop-by-hop DNS extension, i.e. requests for NSID values are not recursively forwarded by resolvers. Instead, clients learn the NSID value of the DNS server or recursive resolver that they directly addressed with their NSID request.

3. Related work

Work related to EDNS and NSID is rather sparse. In 2020, Stipovic [5] analyzed the RFC compliance of EDNS implementations in popular DNS server software such as BIND.

Before the introduction of NSID in 2007, there was another, non-standard mechanism to query for a name server identifier [6]. A BIND name server could be configured to return the server host name when queried for a TXT resource record of the special domain "HOSTNAME.BIND.". Usually the `CLASS` of DNS requests is "IN" for "Internet". For such an identifier query however, the "Chaosnet" (CH) class was used instead. Similar "CHAOS" queries also allowed clients to request the BIND server version.

Fan et al. [7] made use of such CHAOS queries in combination with traceroutes in 2013 to evaluate DNS anycast. They enumerated as many DNS anycast nodes as they could find. They did however intentionally not use the NSID extension in their scans, due to the lack of standardized NSID values, the lack of recursive queries and too little NSID deployment at that time.

Li et al. [8] also used CHAOS queries in 2018 instead of NSID in their analysis of internet anycast. No specific reason was given this time, but they mentioned that such queries were commonly used to analyze anycast.

4. Evaluation

The Alexa Top Sites [9] domain list provided the basis for a DNS scan with the goal of creating an NSID dataset. The list from 23rd Nov 2020 contained 532 839 entries. Based on these domains, an exhaustive DNS scan was performed one day later. All name servers authoritative for these domains and higher-level domains such as top-level domains were scanned, resulting in 14 782 146 executed queries in total. The query parameters and the server responses have been recorded.

Not all rows of the DNS scan dataset are considered in the NSID evaluation. Queries that resulted in an error response are for example filtered out. The analysis is further being limited to queries for domains from the Alexa Top Sites list. All queries with a `NAME` that is not on the Alexa list are therefore filtered out. The same applies to queries with a `TYPE` other than "A" or "AAAA". The remaining 2 885 428 rows of the dataset (19.5 %) have been included in the following analysis.

4.1. EDNS and NSID usage

89 003 unique name server IP addresses have been found to be directly responsible for the Alexa domains. 87 739 (98.6 %) of these NS IPs supported EDNS, as

TABLE 1: Analysis of EDNS and NSID usage

Description	#IPs	Rel. %
All NS IPs	89 003	100.0
Consistent EDNS support	87 285	98.1
No EDNS support	1264	1.4
Inconsistent EDNS support	454	0.5
NS IPs with EDNS support	87 739	100.0
Consistent NSID support	28 100	32.0
No NSID support	59 203	67.5
Inconsistent NSID support	436	0.5
NS IPs with NSID support	28 536	100.0
Consistent NSID value	25 123	88.0
Varying NSID value	3413	12.0
Description	**#Domains**	**Rel. %**
Alexa Top Sites	474 254	100.0
Any NSID support	312 223	65.8
No NSID support	162 031	34.2

TABLE 2: Analysis of NSID support of top TLDs of the Alexa Top Sites

TLD	#Domains	%Supp.	TLD	#Domains	%Supp.
com	255 759	65.8	ua	7050	65.4
ru	32 946	60.0	au	4814	99.9
net	19 113	67.9	tr	4705	63.8
org	16 421	69.4	co	4512	76.7
ir	15 441	31.7	uk	4012	85.0
in	7577	80.4	gr	3913	50.5

their inclusion of OPT records in DNS responses show. A small amount of servers (454 IPs) however only provided inconsistent EDNS support, i.e. for some queries they did support it and for other queries they did not. Out of all unique name server IPs, 28 536 (32.1 %) supported NSID in addition to EDNS and included name server identifiers in their responses. Similarly, there was a small portion (436 NS IPs) with only inconsistent NSID support.

Most name server IPs that did return an identifier only identified themselves with that single NSID. This applies to 25 123 (88.0 %) of the NS IPs that sometimes or always supported NSID. The other 3413 (12.0 %) IPs have replied with different NSIDs for multiple queries. Such varying NSIDs are an indicator for the presence of anycast, in which case multiple queries to the same IP would have been answered by different name servers. It is however no proof for anycast, as the RFC specification of NSID also allows for dynamic identifiers. In that case, the same name server would respond to possibly every query with a different NSID, whose meaning might only be apparent to the server's administrators.

While only about a third of name server IPs supported NSID, two-thirds of Alexa domains actually did provide at least some NSID support. 312 223 (65.8 %) of the Alexa Top Sites name servers responded with an NSID at least once.

Table 1 summarizes the EDNS and NSID usage findings. In Table 2, the NSID support percentages of the Alexa domains are shown, grouped by the top-level domains with most occurrences. Similar to above, a domain is counted as one that supports NSID if and only if one of its name servers responded with an NSID at least once.

TABLE 3: Analysis of NSID values

Description	#NSIDs	Rel. %
Decodable NSIDs	33 864	80.9
Valid domain name	13 792	40.7
IPv4-like	350	1.0
IPv6-like	0	0.0
Contains IATA airport code	10 253	30.3
Contains “ns” or “dns”	7482	22.1
Hyphenated alphanumeric	14 089	41.6
Specific 32-hex-char pattern	4828	14.3
Non-decodable NSIDs	7973	19.1

TABLE 4: Analysis of NSID domains

Description	#NSIDs	Rel. %
All NSID domains	8090	100.0
Resolved to an IP	6687	82.7
Mapped to original NS IP	5214	78.0
Mapped to multiple IPs	999	14.9
Mapped to IPv4 only	5752	86.0
Mapped to IPv4 and IPv6	932	13.9
Mapped to IPv6 only	3	0.0

4.2. NSID values

In the DNS scan dataset, 41 837 unique (NSID, NS IP)-pairs could be observed (with 10 473 unique NSIDs). The NSID values and their possible meaning are evaluated in the following. NSIDs are counted multiple times if and only if different IPs announce the same NSID. Table 3 gives a summary of the findings.

4.2.1. Decodable NSIDs. 33 864 (80.9 %) of the NSIDs could be decoded to valid UTF-8 strings. These NSIDs were between 1 B and 74 B long. They have been subjected to further automated analysis that tries to match them to certain regular expressions. Subsequent percentage numbers are given relative to the number of UTF-8-decoded NSIDs.

Domain names. In a check of each NSID against a regular expression for valid domain names, 13 792 (40.7 %) of NSIDs fully matched the domain name syntax and 14 148 (41.8 %) NSIDs partially matched, i.e. had a domain string embedded in the NSID. In order to find out whether the fully matching domain strings actually resolve to an IP address, a follow-up DNS scan has been performed on 22nd Dec 2020 on the 8090 unique NSID domain strings. 6687 (82.7 %) of NSID domains indeed resolved to an IP address. A majority of these (5214 or 78.0 %) also mapped back to their original name server IP. Some of the resolved NSID domains pointed to multiple IP addresses (999 or 14.9 %). In most cases, the resolved IPs were IPv4 addresses (5752). In 932 cases, the NSID domain resolved to both an IPv4 and IPv6 address. In three cases, it resolved to an IPv6 address only. A summary of the domain string statistics is given in Table 4.

IP addresses. Next to domain names, name server administrators could also choose to set an IP address as identifier. A check of each NSID against a regular expression for valid IP addresses however revealed that almost none were doing so. Only 15 NSIDs fully matched the syntax of IPv4 addresses and zero that of IPv6 addresses. 34 (0.1 %) of NSIDs partially matched the IPv4 syntax, often in form of the IP address being embedded in a domain name, with still zero IPv6 matches. In the presence of domain names, it might be beneficent to format IP addresses without dots within them. After modifying the IP regular expressions by replacing dots and colons with hyphens, more matching NSIDs have been found. 316 (0.9 %) of NSIDs contained an IPv4 address with hyphens, often as part of a domain name. IPv6 addresses did not seem to appear in any NSID, independent of colon or hyphen as separator.

Airport codes. Sometimes server administrators choose to include a regional identifier in domain names. NSID values could also contain such regional identifiers. Airport codes [10] are one type of location identifier. A check against the three-lettered IATA airport codes revealed that 10 253 (30.3 %) of NSIDs seemed to contain such an airport code. To reduce the amount of false positives, this check has been conducted with some additional conditions applied to the regular expression. The airport code had to appear as whole word and before any dot, limiting its appearance to any first domain label. Additionally, a number of codes have been blacklisted due to them also being technical abbreviations, for example “cdn”, “srv” and “vps”.

Miscellaneous. Some more arbitrary regular expression checks have also been performed. With 7482 (22.1 %) NSIDs, quite a few identifiers contained the string “ns” or “dns” as whole word in lower- or uppercase. 144 (0.4 %) identifiers were just integer numbers. 715 (2.1 %) NSIDs purely consisted of alphabet letters, i.e. a to z in lower- or uppercase. Many NSIDs (14 089 or 41.6 %) consisted of only alphabet letters, digits and hyphens. A small number of IDs were found to be so called “globally unique identifiers” (87 IDs or 0.3 %). Some NSIDs followed a very specific pattern, consisting of 32 hexadecimal digits, followed by two spaces and a dash. This was the case for 4828 NSIDs (14.3 %).

4.2.2. Non-decodable NSIDs. 7973 of the NSID values (19.1 %) could not be decoded to valid UTF-8 strings. These NSIDs were between 2 B and 48 B long. Almost all of the non-decodable NSIDs were duplicates (99.5 %). Only 42 values were unique.

The 7973 NSID values have then be subjected to another decoding attempt. This time however, invalid bytes have been ignored in the decoding process. 84.1 % of the bytes could be decoded to partial NSID UTF-8 strings this way.

A manual review of these partial strings led to more insights. In all except three cases, the string started with the sequence of non-printable ASCII characters `NUL SOH CAN`. In most cases (6458 or 81.0 %), this sequence was only extended by another `NUL`. Sometimes, the sequence continued with `NUL ETX NUL` instead.

Only 1494 partially decoded NSIDs contained printable ASCII characters. There was one repeating ASCII pattern in some of the NSIDs, containing the word “proxy”, a number and presumably a location abbreviation. One example for this is “proxy-121-defra.hivecast-121-defra”, where “defra” seems to mean Frankfurt, Germany. In a similar find, “nlams” apparently means Amsterdam, Netherlands, which solidifies the interpretation as location code. Such proxy patterns were always preceded by a sequence of non-printable characters. The similarities of the proxy text patterns suggest that these NSIDs belong to the same service.

It did not become apparent what the meaning of the other, non-UTF-8 bytes was. As the RFC defines NSID values as arbitrary byte strings, the name server administrators could have chosen a more obscure meaning here.

5. Conclusion and future work

This paper analyzed the usage of the NSID DNS extension by the name servers of the Alexa domains. About a third of the name server IP addresses did support NSID, while about two-thirds of domains supported it at least sometimes. Most NSID values could successfully be decoded to UTF-8 strings. Many of these name server identifiers have been found to be domain names, most of which even resolved back to their original name server IP address.

In future work, a closer look could be taken at the IP addresses and domain names of the name servers that are using NSID in order to learn more about *who* is using it, in addition to *if* and *how*. Next to name servers responsible for the Alexa domains, the use of NSID by other name servers could also be investigated. These could be root servers, TLD servers or other, less popular name servers. Another idea for future research is to evaluate the usage of anycast with the help of multiple NSID scans performed from geographically distinct locations.

References

[1] R. Austein, “DNS Name Server Identifier (NSID) Option,” RFC 5001, Aug. 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc5001.html

[2] J. L. S. Damas, M. Graff, and P. A. Vixie, “Extension Mechanisms for DNS (EDNS(0)),” RFC 6891, Apr. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc6891.html

[3] P. Mockapetris, “Domain Names - Implementation and Specification,” RFC 1035, Nov. 1987. [Online]. Available: https://rfc-editor.org/rfc/rfc1035.html

[4] S. Rose, M. Larson, D. Massey, R. Austein, and R. Arends, “DNS Security Introduction and Requirements,” RFC 4033, Mar. 2005. [Online]. Available: https://rfc-editor.org/rfc/rfc4033.html

[5] I. Stipovic, “Analysis of an Extension Dynamic Name Service – A Discussion on DNS Compliance with RFC 6891,” 2020.

[6] D. R. Conrad and S. Woolf, “Requirements for a Mechanism Identifying a Name Server Instance,” RFC 4892, Jun. 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4892.html

[7] X. Fan, J. Heidemann, and R. Govindan, “Evaluating Anycast in the Domain Name System,” in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 1681–1689.

[8] Z. Li, D. Levin, N. Spring, and B. Bhattacharjee, “Internet Anycast: Performance, Problems, & Potential,” 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. 59–73. [Online]. Available: https://doi.org/10.1145/3230543.3230547

[9] Alexa Internet, Inc., “Alexa Top Sites,” Nov. 2020. [Online]. Available: https://www.alexa.com/topsites

[10] Fubra Limited, “World Airport Codes,” Dec. 2020. [Online]. Available: https://www.world-airport-codes.com/

xdpcap: XDP Packet Capture

Stefan Lachnit, Sebastian Gallenmüller*, Dominik Scholz*, Henning Stubbe*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: stefan.lachnit@tum.de, {gallenmu | scholz | stubbe}@net.in.tum.de

Abstract—**Xpress Data Path (XDP) is a Linux kernel feature that allows high performance packet processing using eBPF programs which are executed before the normal network stack. This however prevents tools like tcpdump from capturing all traffic. Xdpcap is a recently released network capturing program, which uses filters compiled to eBPF and a hook in the tested XDP program to capture packets even if they are dropped by the XDP program.**

This paper explains how xdpcap is implemented and presents benchmarks, which compare xdpcap to tcpdump. We show that xdpcap is not able to achieve the same capturing bandwidth as tcpdump and should thus be used for debugging and capturing only when packets dropped or forwarded by an XDP program are of interest.

Index Terms—**XDP, eBPF, packet capture**

1. Introduction

The ability to capture and filter network packets is an important aspect of debugging network applications. To assess and benchmark the performance of these applications, the recording of large amounts of traffic is needed. Xpress Data Path (XDP) is a Linux kernel feature, which allows users to run small programs to modify, pass, drop or redirect incoming network packets before they are processed by the rest of the networking stack. Using XDP yields performance benefits in applications like Firewalls [1] and DDoS mitigation [2] compared to other solutions like iptables. Additionally, when using XDP, traditional network capturing tools like tcpdump are not able to record all packets, because they could be dropped or modified before they reach the regular network stack. To solve this issue, Cloudflare developed xdpcap [3], a tool which can capture packets (filtered by a user specified expression) directly from an instrumented XDP program.

We performed benchmarks to test the performance of xdpcap and tcpdump by capturing generated test traffic and analyzing how many packets could be recorded. This paper describes these tests and evaluates their results.

The paper is structured in the following way. Section 2 describes the features used by xdpcap and gives an overview of how it is implemented. In Section 3 related work is presented, which discusses XDP and network capturing. After the software and hardware, which was used to benchmark xdpcap and tcpdump, is described in Section 4, the measured data is presented and evaluated in Section 5. In the last section the results are summarized and a conclusion is presented.

2. Background

This section describes the basic concepts of the Linux kernel features used by xdpcap and explains how xdpcap works.

2.1. cBPF and eBPF

Berkeley Packet Filter (BPF; later renamed to classic BPF (cBPF)) is a feature in the Linux kernel which is designed to allow high performance network packet filtering in kernel space. It introduced a virtual machine (VM) that allows users to attach small programs to a network interface, which can parse incoming packets and decide if they should be copied to userspace. One of its primary use cases is the tool tcpdump, which is used for filtering and capturing network traffic for measurements and debugging. It allows the user to specify filtering expressions which are then compiled to a BPF program. Matching packets are copied to userspace where they are stored to a file or parsed and printed [4].

In kernel version 3.15, extended BPF (eBPF) was introduced, which improved the original concept by modifying the VM to allow more complex programs and adding new features. eBPF programs are no longer limited to packet filtering and can now process events in other parts of the kernel. Additionally, the possibility to store persistent data by using maps, which can also be accessed in userspace, and the ability to call kernel helper functions were added. A special type of map allows eBPF programs to dynamically call other eBPF programs, with the limitation that the control flow cannot return to the original code (tail call). To ensure high performance, eBPF programs are compiled to machine code using a just-in-time (JIT) compiler. Since the eBPF program runs in kernel space, it is important to ensure the security of the executed program. This is done by a verifier, which statically analyzes a program before it is attached (e.g., prevents infinite loops, checks if memory accesses are in a valid range) [5].

2.2. XDP

Xpress Data Path (XDP) adds a hook to the Linux network stack, which can be used to run an eBPF program (XDP program) for every packet at the earliest possible moment after it is processed by the driver of the network card. On supported drivers, it is run in the context of the driver or can even be offloaded to specialized hardware on the NIC [6]. The eBPF program has access to the raw packet data and can parse and possibly modify it. It is also possible to add additional metadata to a packet. The action which is applied to the packet can be specified by the return code of the program. A packet can be dropped, passed on to the normal network stack, re-transmitted from the same interface or redirected (e.g., to a different network interface or to a userspace socket) [7].

2.3. xdpcap

Because XDP programs can modify or drop packets before they reach the Linux network stack, traditional packet capture tools like tcpdump are not able to record all traffic. Cloudflare recently released the tool xdpcap with the goal to recreate tcpdump for applications using XDP.

To achieve this, they wrote a compiler that transforms a cBPF program into equivalent eBPF bytecode. The cBPF code is generated from a user-specified filter expression by libpcap, which is also used by tcpdump.

Packets are captured and filtered by this additional XDP program (filter program), which is executed after all other processing steps of the original XDP program are completed. To be able to dynamically start capturing and change filters without removing the original XDP program, the filtering code is executed as a tail call from the original XDP program. This requires manual modification of the original program by adding a hook (a map of filtering XDP programs generated from the filter expression) and replacing returns by tail calls. To allow the execution of the originally specified action (without returning to the original program), xdpcap generates multiple filter programs with every possible return code hard-coded and chooses the matching program when executing the tail call.

When a filter program matches a packet, it has to be transferred to userspace. This is done by generating a perf event using the eBPF helper function *perf_event_output*, which can contain the packet data and additional metadata. Perf events are part of the Linux kernel, which are normally used for profiling and tracing. In userspace, the xdpcap tool creates a ring buffer where the data created by this perf event is put into. When this buffer is filled to a specified number of bytes (by the parameter watermark), it is read by this tool and printed or output to a file [3].

3. Related Work

XDP has been used in many applications, which require high performance packet processing. Firewall rules, which are faster than existing solutions using iptables [1] [8], efficient DDoS mitigation [2] and an XDP based L4 load balancer [9], are examples for such applications.

For capturing network traffic, different approaches exist. The most common tool for debugging and capturing is the software-based tool tcpdump [4]. Capturing tools, which bypass the Linux network stack, are capable of achieving a capturing rate of up to 120 Gbit/s [10] on commodity hardware. Additionally, hardware-assisted capturing based on FPGAs [11] or commercially available capturing hardware (e.g. Endace DAG cards [12]) allows recording of traffic at high data rates and precision.

TABLE 1: hardware setup

	loadgen	**DuT**
OS	Debian Buster	Debian Buster
Kernel	4.19.0-12-amd64	4.19.0-12-amd64
CPU	Intel Xeon E5-2620 v3	Intel Xeon E5-2630 v4
RAM	32 GB	128 GB
NIC	Intel 82599ES 10G	Intel 82599ES 10G

4. Experiment Setup

To measure and compare the performance of xdpcap in a reproducible way, benchmarks were performed on a hardware testbed managed by pos (plain orchestrating service) [13]. This tool manages all test servers using an orchestrating server, which handles the allocation of servers and the execution of benchmarks. Using a script, the required servers can be rebooted and set up automatically. Additionally, the execution of benchmarks and the collection of results are handled by this script. Since all the test servers run live systems in RAM, the required configuration is done automatically every time the benchmarks are executed [13].

The hardware setup of the servers, which were used for the benchmarks in this paper is described in Table 1. The layout of the benchmark servers is presented in Figure 1. Both servers are connected by a 10 Gbit/s connection and managed by an orchestrating server running pos. One of the servers acts as a load generator sending traffic to the other server, which is running xdpcap or tcpdump to capture this traffic.

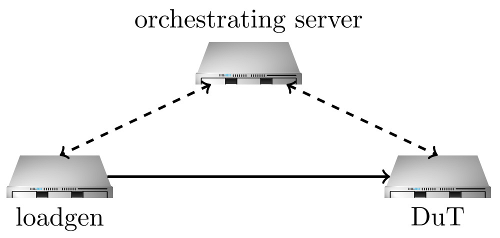


Figure 1: testbed

To generate test traffic on the loadgen server, MoonGen [14] was used. It uses DPDK [15] to bypass the Linux network stack and generate up to 10 Gbit/s traffic on a single CPU core with high precision. The packets can be dynamically created and modified by a Lua-script, which controls the data of each generated packet [14]. The benchmarks in this paper use a modified version of the layer2 example script [16] to create Ethernet traffic with adjustable packet size at a specified bandwidth. For testing how filtering of traffic influences the performance of the tested capturing tools, the script was modified to change the ethertype of every given number of packets to a different value (0x1111 instead of 0x1234). The device under test (DuT) can then filter based on this field.

The other host was set up to capture the packets sent by the loadgen server. Multiple tests with both xdpcap and tcpdump were performed. For the benchmarks of xdpcap an XDP program with two functions (drop all traffic, pass all traffic) was added to the network interface, which is connected to the load generator. This program was modified to contain an xdpcap hook and tail calls, which are required for capturing using xdpcap. Additionally, the network interface had to be set to promiscuous mode to be able to capture packets, which do not match the MAC address of the network interface. When using tcpdump with default settings this happens automatically. Both tools write the incoming data into a pcap file, which is later analyzed using the command *capinfos* to collect performance metrics (number of packets, packet rate, captured bandwidth). All benchmarks performed the capturing for 40 seconds.

The results of both servers (analyzed packet capture, output of Moongen) are then uploaded to the orchestrating server where they were evaluated using an interactive Jupyter notebook.

5. Evaluation

In this section, three benchmarks which were performed using the setup described in Section 4 are presented.

5.1. Maximum Bandwidth

To analyze the maximum achievable capturing bandwidth using both tested tools, benchmarks without filtering expressions were performed. These tests were run with a packet size of 64 B. The bandwidth of the generated Ethernet traffic was scaled from 100 Mbit/s to 3 Gbit/s. For the benchmarks using xdpcap, both an XDP program, which drops all packets and a program which passes all received packets were tested.

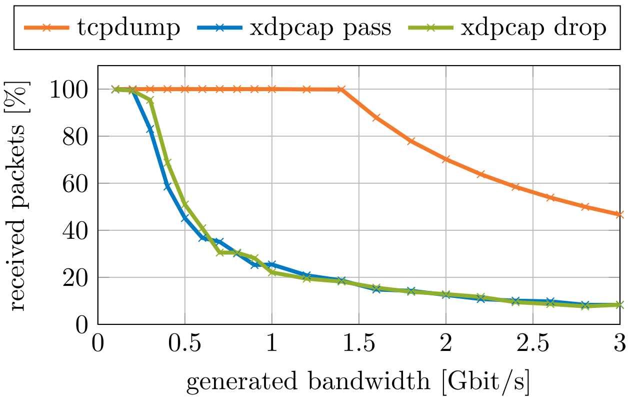


Figure 2: captured packet percentage

Figure 2 shows how many percent of the generated packets could be recorded on the DuT. Tcpdump was able to capture nearly all traffic up to 1.4 Gbit/s. Xdpcap only captured all packets for speeds lower than 200 Mbit/s. The results for xdpcap with different XDP programs (drop all packets, pass all packets) were nearly identical.

Figure 3 shows results of testing the impact of different packet sizes by additionally using 128 B packets. It plots the captured packet rate for transmitted packet rates from 0.1 million packets per second (Mpps) to 5.8 Mpps. Tcpdump was able to capture nearly all traffic up to 2.5 Mpps for a packet size of 64 B. When more traffic was generated, it was only able to record the same 2.5 Mpps and dropped the rest. When using packets with a size of 128 B the maximum number of packets which could be

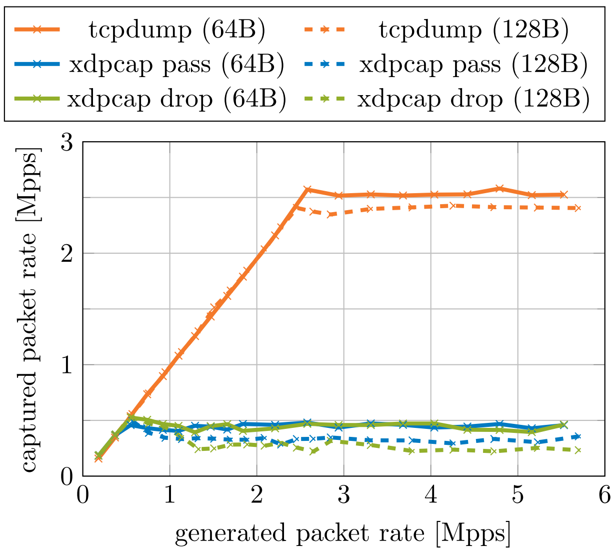


Figure 3: captured packet rate

captured only decreased by approximately 5 %, indicating that the capturing performance of tcpdump is mostly dependent on the number of captured packets and not on recorded bandwidth. Xdpcap showed similar results, but was only able to capture all packets at rates less than 0.5 Mpps with 64 B packets. The achievable packet rates of dropping or passing all packets in the XDP program were nearly identical for 64 B packets. Increasing the packet size to 128 B decreased the possible capturing speed by about 30 % when using xdpcap.

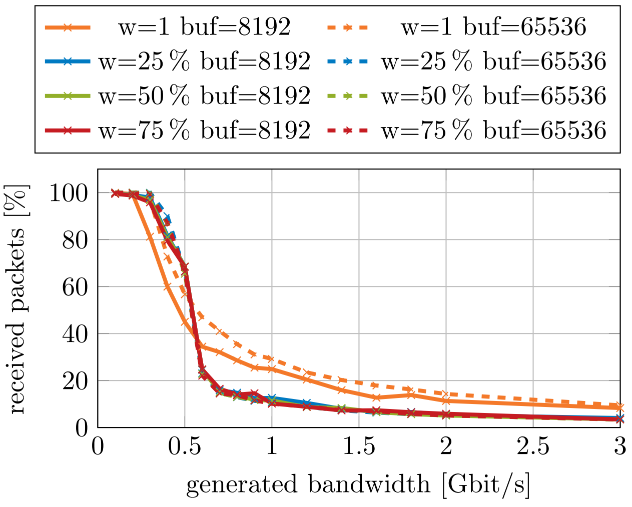


Figure 4: xdpcap parameters: capturing percentage

5.2. Xdpcap Parameters

The userspace xdpcap program offers parameters to tune the buffer size and the "perf watermark", which is used to specify when packets are read from the perf ring buffer. To test the impact of these settings, the percentage of captured packets for different combinations of parameters was tested. The generated bandwidth was scaled from 100 Mbit/s to 3 Gbit/s. The buffer size was set to (the default of) 8192 B and 65 536 B. The watermark was set to 1 (default: transfer immediately), 25 %, 50 % and 75 % of the buffer size.

Figure 4 shows the results of these tests. When the generated bandwidth is low (less than 200 Mbit/s), increasing the watermark value allowed xpdcap to capture more packets than with the default configuration. For bandwidths above 500 Mbit/s of test traffic, the best result was achieved when transferring packets immediately (watermark 1). Increasing the buffer size to 65 536 B with a watermark of 1 increased the number of captured packets by approximately 10 %. All other values of the watermark and other buffer sizes result in nearly identical or slower capturing speeds.

5.3. Filtered Traffic

To evaluate a more realistic measurement and benchmark scenario, where only a small part of the received traffic is of interest, we tested how filtering the incoming traffic would impact the ability to capture all (matching) packets at high data rates. For this, the modified Moongen script described in Section 4 was used to generate Ethernet traffic and set the ethertype of every 1000th packet to a different value. Only this traffic was recorded by specifying the filtering expression *"ether proto 0x1111"*. The packet size was set to 64 B and the generated bandwidth was scaled from 500 Mbit/s to 6 Gbit/s.

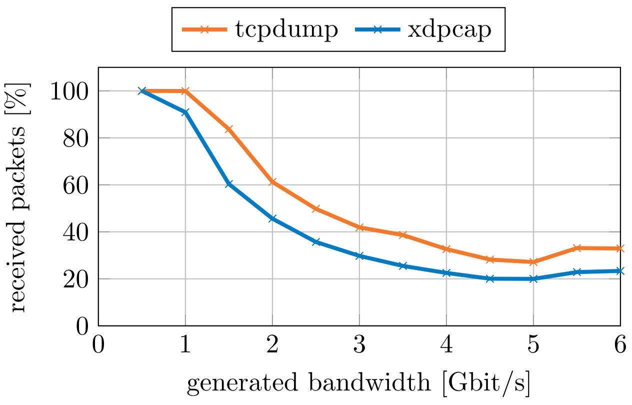


Figure 5: filtered packet capture rate

Figure 5 shows the results of this benchmark. When only capturing a small percentage of the traffic, xdpcap was able to capture all matching packets at rates less than 500 Mbit/s. This is a significant increase from the 200 Mbit/s which could be recorded when no filter was applied. The maximum recording bandwidth of tcpdump decreased compared to the test in Section 5.1. All packets could only be recorded for generated traffic of 1 Gbit/s or less. For every amount of generated traffic, tcpdump captured about 15 % more traffic than xdpcap.

6. Conclusion

The results of the presented benchmarks show that in the measured test scenario, xdpcap performs significantly worse when capturing all packets. Small improvements can be achieved by increasing the ring buffer size and keeping the watermark at the default value. When applying filters, tcpdump was affected more than xdpcap, but still yielded better performance.

Because of this, we conclude that xdpcap should not be used as a tcpdump replacement. However, when used for its intended purpose of debugging or monitoring existing XDP programs, it can be applied where packets are processed before tools like tcpdump can capture them.

References

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

[2] "L4Drop: XDP DDoS Mitigations," https://blog.cloudflare.com/l4drop-xdp-ebpf-based-ddos-mitigations/, accessed: 2020-12-29.

[3] "xdpcap: XDP packet capture," https://blog.cloudflare.com/xdpcap/, accessed: 2020-12-01.

[4] S. McCanne and V. Jacobson, "The bsd packet filter: A new architecture for user-level packet capture," in *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings*, ser. USENIX'93. USA: USENIX Association, 1993, p. 2.

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

[6] "Netronome Agilio SmartNICs," https://www.netronome.com/products/agilio-software/agilio-ebpf-software/, accessed: 2021-01-09.

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

[8] A. Deepak, R. Huang, and P. Mehra, "ebpf/xdp based firewall and packet filtering," in *Linux Plumbers Conference*, 2018.

[9] "katran," https://github.com/facebookincubator/katran, accessed: 2021-01-03.

[10] P. Emmerich, M. Pudelko, S. Gallenmüller, and G. Carle, "Flowscope: Efficient packet capture and storage in 100 gbit/s networks," in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, 2017, pp. 1–9.

[11] Y. E. Kwasi and R. Rojas-Cessa, "High-resolution hardware-based packet capture with higher-layer pass-through on netfpga card," in *2014 23rd Wireless and Optical Communication Conference (WOCC)*, 2014, pp. 1–6.

[12] "endace," https://www.endace.com/, accessed: 2021-01-03.

[13] S. Gallenmüller, D. Scholz, F. Wohlfart, Q. Scheitle, P. Emmerich, and G. Carle, "High-performance packet processing and measurements," in *2018 10th International Conference on Communication Systems Networks (COMSNETS)*, 2018, pp. 1–8.

[14] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "Moongen: A scriptable high-speed packet generator," in *Proceedings of the 2015 Internet Measurement Conference*, ser. IMC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 275–287. [Online]. Available: https://doi.org/10.1145/2815675.2815692

[15] "DPDK," https://www.dpdk.org/, accessed: 2020-12-27.

[16] "Moongen l2-load-latency example," https://github.com/emmericp/MoonGen/blob/525d99917c98a4760db72bb733cf6ad30550d6669/examples/l2-load-latency.lua.

TLS Certificate Analysis

Jonas Lang, Markus Sosnowski*, Johannes Zirngibl*, Patrick Sattler*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: langj@cs.tum.de, sosnowski@net.in.tum.de, zirngibler@net.in.tum.de, sattler@net.in.tum.de

***Abstract*—TLS Certificates contain a variety of different information. In this paper, we give an overview on what information we could extract from a large dataset of TLS certificates. We gather information about who issues these certificates and what they do with select fields. Also, we use zLint, a linter, to check certificates for issues. Finally, we look at the trust chains of leading to certificates.**

***Index Terms*—tls certificate analysis, internet-wide, zlint, trustchain, PKI**

1. Introduction

In recent years, using HTTP over TLS (HTTPS) has become more and more widespread for securely communicating over the web. Major Web-Browsers are marking websites using the plain Hypertext Transfer Protocol (HTTP) as "insecure" [1], [2], so websites are incentivized to provide access by HTTPS [3]. While establishing a Transport Layer Security (TLS) connection, the server has to present a X.509 Certificate. This is also called a TLS certificate. The client has to determine if the certificate is valid, and if the certificate can be trusted. If the certificate is considered invalid or not trusted, web-browsers may block access and display a warning [4], [5]. In 2019, the management of TLS certificates has been standardized by the Automatic Certificate Management Environment (ACME) protocol. This protocol simplifies automated issuance, renewal and revocation of a certificate [6]. In this paper, we focus on the analysis of TLS certificates. We first introduce the concept of TLS certificate in section 2". Then we present a short overview of the design of our analysis in section 3", followed up by the details of the implementation in section 4". The core part of our paper is the evaluation of the results in section 5". Finally, we draw a conclusion and present opportunities for future work in section 6".

1.1. Related Work

There have been other papers that surveyed the certificates used for TLS. The authors of "Analysis of the HTTPS Certificate Ecosystem" [7] presented in 2013 a large-scale study that gave insight into the HTTPS certificate ecosystem. They analysed over 42M certificates in total, and investigated the trust relationships between users, intermediate authorities and root authorities. Another Study in 2017 looked at the misissuance of certificates. "Tracking Certificate Misissuance in the Wild" [8] introduces zLint, a certificate linter. They have been able to check 61M certificates for misissuance and uncovered that mainly smaller organizations misissue certificates.

2. Background

TLS certificates are mainly used by servers to authenticate themselves. Most certificates are leaf certificates, that are signed by a 3rd Party, called a certificate authority (CA). The CAs control root certificates, which are the trust anchor in this system - most clients trust a set of root certificates, transitively trusting each certificate signed by one of the roots. However, most leaf certificates are not directly signed by root certificates, instead they are signed by intermediate certificates. Those intermediate certificates are signed by root certificates. TLS certificates can also be self-signed, therefore they are not signed using the private key from a third party. This means that the client needs to trust the certificate. Most browsers offer the option to trust certain self signed certificates, but self-signed certificates are generally not stored in the trust anchor.

2.1. Chain of Trust

If a server presents a leaf certificate to a client, the client has to determine if it trusts this certificate. As the client implicitly trusts all root certificates, the client has to build a chain of trust to a root certificate, as depicted in Figure 1. In other words, the client has to find a root certificate that has signed the presented leaf certificate directly, or a chain of potentially multiple intermediate certificates that lead to the leaf certificate.

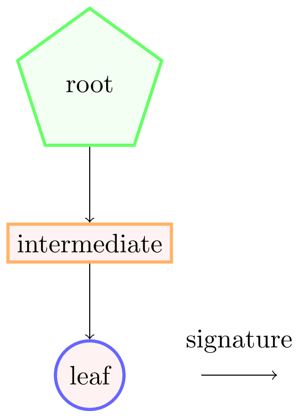


Figure 1: chain of trust

2.2. Certificates with more than one Parent

It is possible that there are two or more valid chains of trust associated with the same leaf certificate. E.g. this occurs when two intermediate certificates share the same private key so both their signatures are identical, as in Figure 2.

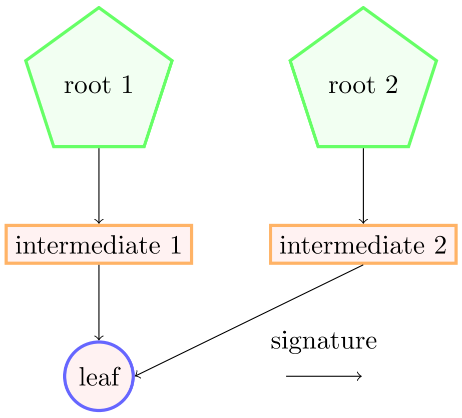


Figure 2: example certificate with two valid chains of trust

2.3. Misissued Certificates

Misissued certificates do not adhere to specifications in RFC5280 [9], or fail to adhere to the CA/Browser Forum Baseline Requirements [10]. If certificate fails to meet those requirements, a client typically does not trust the server.

3. Design

We use a large-scale HTTPS scan on port 443 created in December 2020 using goscanner [11] containing over 126M TLS certificates. The certificates are parsed, processed and then results are written back to disk to be analyzed. We examine the presence of select fields and analyse their content. Each certificate is checked by a certificate linter. Furthermore, we use a library to build trust chains for each certificate, that end with a root certificate in our trust anchor.

4. Implementation

Each certificate is parsed using the zCrypto library, which is based on the standard Go library [12]. If the Basic Constraints extension is present and the cA boolean [sic] is set, a a certificate is considered a CA certificate, otherwise, it is considered a leaf certificate [9].

4.1. Linting with zLint

Every certificate was linted by running zLint on it, which checks for "consistency with rfc standards and other relevant pki requirements" [13]. zlint distinguishes three categories of Lints: "Notice", "Warn" and "Error". Lints in category "Notice" can be non-deterministic and indicate there may be a problem. As over 126M certificates were processed in this paper, it was not possible to examine for every notice if there truly was a problem, so lints of this category were ignored. Lints in category "Warn" check e.g. if a SHOULD or SHOULD NOT Requirement from an RFC has been violated, while lints in category "Error" e.g. check if a MUST or MUST NOT Requirement has been violated [13].

4.2. Building trust chains

We use the mozilla root store [14] as of December 2020 as the trust anchor. The set of intermediate certificates is built by collecting all certificates that are CA certificates. Then we use the Verify function from zCrypto [15] to find all trust chains that lead to a root in the trust anchor.

4.3. Differences to typical clients in trust chain validation

Some certificates that appear valid in our testing may be rejected by certain browsers, and some certificates that appear valid in certain browsers may appear valid in our testing.

4.3.1. Available Intermediates. We try to build trust chains using the set of all intermediate certificates that have been seen in the scan. Typically however, a client should rely on the server to present all intermediate certificates leading to the root. All clients we are aware of use some form of caching for intermediate certificates. If the necessary intermediates are already in this cache, the client may succeed in building a trust chain even if the server does not present every necessary intermediate. Some clients try to use an Uniform Resource Identifier (URI) specified in the Authority Information Access Extension (AIA Extension) [15] to fetch missing intermediate certificates. The platform verifiers on Windows, ChromeOS and MacOS implement this. [16] However, a major client that does not support fetching intermediate certificates using the AIA Extension is Mozilla Firefox [17].

4.3.2. Revoked certificates. In theory, CAs should be able to revoke issued certificates via the Online Certificate Status Protocol (OCSP) and Certificate Revocation Lists (CRL). As Liu et. al found in 2015 however, these mechanisms are often not used by clients [18]. They also revealed that Mobile Browsers on Android did not check for revocation at all. Some Desktop-Browsers use preselected CRLs to check for revoked certificates, which only contain a fraction of all revoked certficates [19], [20]. Because the revocation of certificates is handled so differently across clients, we do not check any certificates for revocation.

5. Evaluation

We ran our analysis on a Dataset that was collected from 24/12/2020 to 27/12/2020. This Dataset contains 126.200.987 certificates that could be successfully parsed, while only 175 certificates were too broken to be parsed.

TABLE 1: Classes of certificates in our dataset

class	total %	absolute
all certificates	100%	126.200.987
roots	<0,01%	78
intermediates	0,18%	236.475
valid intermediates	<0,01%	927
self-signed	1,75%	2.207.833
leaves	98,24%	123.983.769
valid leaves	95,61%	120.658.574

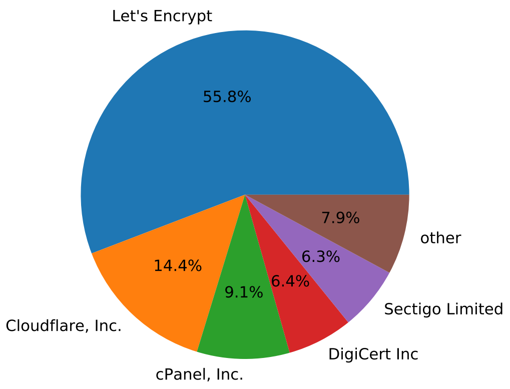


Figure 3: Percentage of valid leaves signed by each organization

5.1. General Landscape

We define a valid leaf as a leaf certificate, to which we could build a trust chain that has been valid anywhere in between 21/12/2020 and 28/12/2020. This is to mitigate that the certificates were collected over multiple days. Table 1 reveals that over 95% of certificates are valid leaf certificates.

5.2. Issuers of HTTPS Certificates

We grouped valid leaf certificate by issuer organisation name in Figure 3 . Let's Encrypt is the dominant issuer of TLS certificates with a share of over 55%.

5.3. Signature Algorithms

Table 2 shows that certificate signatures among valid leaves were almost exclusively made using SHA256-RSA. An exception is Cloudflare, which is responsible for 99,55% of all ECDSA-SHA256 signatures, which amounts to 17.405.658 certificates. This is 99,84% of all Cloudflare issued certificates.

TABLE 2: Signature Algorithms used for valid leaves

algorithm	valid leaves %	absolute
SHA256-RSA	85,23%	102.834.585
ECDSA-SHA256	14,49%	17.483.284
SHA384-RSA	0,27%	316.893
ECDSA-SHA384	0,01%	17.483.284
SHA512-RSA	<0,01%	6.278

TABLE 3: Certificate lifespan by interval

lifetime in days	valid leaves %	absolute
51-100	70.80%	85.427.923
200-398	26.80%	32.341.146
399-800	2,00%	2.413.525
801-1200	0,23%	282.170
101-200	0,11%	132.037
0-50	0,05%	61.773

TABLE 4: Usage of the subject country field

country	valid leaves %	absolute
not used	82,76%	104.443.306
US	14,79%	18.336.360
PL	0,12%	156.190
DE	0,11%	134.287
...	...	...

5.4. Validity Period

Each certificate has a not before and not after field. These indicate the lifespan of a certificate. In Table 3, the lifetime of valid leaf certificates is divided into classes. Starting on September 1st 2020, every major root program decided to reduce the maximum lifespan of each certificate to 398 days [21]. The majority of valid leaves has a considerably shorter validity period of <100 days. This is desirable, as shorter lifespans of certificates are generally better for security, as certificates have to be reissued more often. Certificates that have a longer lifespan than 398 days, but have been issued before September 2020 can still be valid certificates.

5.5. Subject Country Field

As X.509 certificates used for TLS are general purpose, they also feature fields that are not useful for typical browsers. One example is the subject country field, which is left unused by the vast majority of certificates, as Table 4 shows. Surprisingly, some CAs like Cloudflare seem to set the subject country field for all of their issued certificates to "US".

5.6. Extended Key Usage Field

Over 98% of certificates are issued with ServerAuth and ClientAuth as their extended key usage, as Table 5 shows. The purpose of the certificates during the scan was to authenticate the server, so ClientAuth should not be needed as specified by RFC5280, Section 4.2.1.3. [9]. We hypothesize that the vast number of certificates with ClientAuth set is due to the added flexibility for the users.

TABLE 5: Extended key usage

Extended key usage	valid leaves %	absolute
ServerAuth, ClientAuth	98,15%	118.407.285
ServerAuth	1 ,84%	2.243.899
...	...	...

TABLE 6: certificates with warnings per organization sorted by absolute amount

Organisation Name	warnings	issued	% warnings
GoDaddy.com, Inc.	179.679	2.481.067	7,24%
Amazon	40.853	1.392.205	2,93%
SECOM Trust Systems [...]	35.842	35.872	99,91%
Actalis S.p.A.	20.984	408.090	5,14%
DigiCert Inc	20.389	7.773.648	0,26%
Starfield Technologies, Inc.	17.305	322.869	5,35%
Microsoft Corporation	13.572	123.668	10,97%
Sectigo Limited	9.399	7.588.983	0,12%
...	...	...	...

TABLE 7: valid leaf certificates with errors per organization sorted by absolute amount

Organisation Name	errors	issued	% errors
nazwa.pl sp. z o.o.	1.138	271.919	0,41
AC Camerfirma S.A.	479	1.630	29,38
Unizeto Technologies S.A.	53	57.203	0,09
home.pl S.A.	19	54.282	0,03
Dreamcommerce S.A.	5	13.222	0,03
...	...	...	..

5.7. zLint Results

Of all valid leaf certficates, only 2.283 triggered an error, and 353.292 triggered a warning. In Figure 6, GoDaddy and Sectigo triggered warnings, even though they are known to use zLint in some fashion for pre-issuance linting. This may either be due to old certificates that are still valid, or that they choose to ignore these specific lints. In Table 7, most organisations triggered only few errors relative to their total issued certificates. With almost a rate of 30% of their issued valid leaves triggering errors, AC Camerfirma S.A. is clearly an exception in our dataset. The mozilla wiki details 26 potential issues with certificates from this CA [22], and the Chromium open-source project to plans to block this CA in a future release entirely [23].

5.8. Certificate trust chains

We examined the trust chains leading from certificates to a trusted root certificate. As mentioned earlier, we didn't do any revocation checking. Therefore, we also found trust chains containing revoked intermediates. For example, the Let's Encrypt's R3 intermediate certificate signed by DST CA X3 has a twin with different content. This twin intermediate, however, according to crt.sh as of 08/01/2021 has been revoked via OCSP, and CRL by the CA. Those circumstances make it hard to interpret the gathered data, but would provide an interesting starting point for further analysis. If we describe a set of trust chains that lead to a certificate as a trust chain configuration, we can still examine the most used trust chain configurations. The top 20 distinct trust chain configurations combined lead to 98,22% of valid leaf certificates. None of the top 20 trust chain configurations contained a root certificate that directly signed a valid leaf certificate.

6. Conclusion and future work

6.1. Conclusion

We have examined several aspects of TLS certificates, and summarized them. It was possible to build a trust chain leading to a trusted root for a majority of certificates in the dataset. Let's Encrypt currently dominates the TLS certificate ecosystem. Certain attributes of a certificate are characteristic for a CA - e.g. 99.55% of all valid leaf certificates with a ECDSA-SHA256 signature are issued by Cloudflare. With zLint, almost no misissued certificates could be found in our dataset. A CA that is known to issue problematic certificates could be clearly identified.

6.2. Future Work

In this Paper we do not cover every field that may be present in a TLS Certificate. The analysis could easily be extended to cover additional fields. Also, zLint is an established certificate linter that is known to be used by some CAs [13]. More misissued certificates may be unconvered by creating additional lints that have not been published before.

References

[1] Google, "Google Blog," https://blog.google/products/chrome/milestone-chrome-security-marking-http-not-secure/, July 2018, [Online; accessed 7-January-2021].

[2] Mozilla, "Mixed Content Blocking in Firefox," https://support.mozilla.org/en-US/kb/mixed-content-blocking-firefox#w_what-is-mixed-content-and-what-are-the-risks, [Online; accessed 7-January-2021].

[3] P. R. Donahue, "Https or bust: Chrome's plan to label sites as "not secure"," https://blog.cloudflare.com/https-or-bust-chromes-plan-to-label-sites-as-not-secure/, [Online; accessed 7-January-2021].

[4] Mozilla, "How to troubleshoot security error codes on secure websites," https://support.mozilla.org/en-US/kb/error-codes-secure-websites, [Online; accessed 7-January-2021].

[5] Google, "Fix connection errors," https://support.google.com/chrome/answer/6098869?hl=en.

[6] IETF, "RFC8555," https://tools.ietf.org/html/rfc8555.

[7] M. B. Zakir Durumeric, James Kasten, "Analysis of the HTTPS Certificate Ecosystem." IMC '13: Proceedings of the 2013 conference on Internet measurement conference, October 2013.

[8] D. Kumar, Z. Wang, M. Hyder, J. Dickinson, G. Beck, D. Adrian, J. Mason, Z. Durumeric, J. Halderman, and M. Bailey, "Tracking certificate misissuance in the wild," 05 2018, pp. 785–798.

[9] IETF, "RFC 5280," https://tools.ietf.org/html/rfc5280.

[10] CAB Forum, "Baseline Requirements Documents (SSL/TLS Server Certificates)," https://cabforum.org/baseline-requirements-documents/.

[11] tumi8, "GoScanner," https://github.com/tumi8/goscanner.

[12] zMap, "zMap Project," https://zmap.io/.

[13] ——, "zLint," https://github.com/zmap/zlint.

[14] Mozilla, https://wiki.mozilla.org/CA/Included_Certificates.

[15] zMap, "zCrypto," https://github.com/zmap/zcrypto.

[16] Mustafa Emre Acer, Emily Stark, Adrienne Porter Felt, Sascha Fahl, Radhika Bhargava, Bhanu Dev, Matt Braithwaite, Ryan Sleevi, Parisa Tabriz, "Where the Wild Warnings Are: Root Causes of Chrome HTTPS Certificate Errors," *CCS'17*, 2017.

[17] Mozilla, "SecurityEngineering/Certificate Verification," https://wiki.mozilla.org/SecurityEngineering/Certificate_Verification, November 2019, [Online; accessed 30-December-2020].

[18] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson, "An end-to-end measurement of certificate revocation in the web's pki," 10 2015, pp. 183–196.

[19] M. Goodwin, "Revoking Intermediate Certificates: Introducing OneCRL," https://blog.mozilla.org/security/2015/03/03/revoking-intermediate-certificates-introducing-onecrl/, 2015.

[20] The Chromium Projects, "Revoking Intermediate Certificates: Introducing OneCRL," https://dev.chromium.org/Home/chromium-security/crlsets, [Online; accessed 7-January-2021].

[21] P. Nohe, "Maximum TLS certificate validity now one year," https://www.globalsign.com/en/blog/maximum-ssltls-certificate-validity-now-one-year, [Online; accessed 7-January-2021].

[22] Mozilla, "Common CA Database," https://www.ccadb.org/.

[23] Ryan Sleevi, https://groups.google.com/g/mozilla.dev.security.policy/c/dSeD3dgnpzk/m/iAUwcFioAQAJ.

Corona Warn-App – Design, Development and Privacy Considerations

Oliver Layer, Benedikt Jaeger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: oliver.layer@tum.de, jaeger@net.in.tum.de

***Abstract*—Contact tracing applications can help to reduce the spread of the coronavirus disease 2019 by identifying infection chains. The *Corona-Warn-App* is the official German application. While contact tracing apps require a certain amount of users in the population to be effective, there are privacy, effectiveness and security concerns that may diminish the app's acceptance. In this paper functionality and possible privacy and security attack vectors as well as mitigations for the app are reviewed. Furthermore, the app's architecture is compared with other approaches. The results show that privacy and security measures are in place, limiting possible attacks to be infeasible on a large scale. In contrary, there have been several bugs during the introduction phase of the app which could have put off users.**

***Index Terms*—contact tracing apps, privacy, exposure notification, corona warn app**

1. Introduction

In the beginning of 2020 the world has been struck by a pandemic regarding the *coronavirus disease 2019* (COVID-19). The disease is believed to spread especially in situations where people are in proximity to each other. Using their smartphones and their Bluetooth signals, *contact tracing apps* (CTAs) provide information to users indicating if there was a situation in the past where they were exposed to someone who has already been infected. CTAs can help to identify the chain of infections and can therefore slow down the pandemic by breaking them. In contrast, only relying on manual contact tracing is not suitable on a larger scale and for most situations, such as in public transport.

Throughout the pandemic, several CTAs have been developed using different architectures. The *Corona-Warn-App* (CWA) is the open-source contact tracing app of the German government. It is based on the *Exposure Notification API* (ENA) which has been developed jointly by Google and Apple. The approach builds upon a decentralized architecture with the goal of preserving privacy.

Initially, the German government pursued to follow a centralized approach, which may be more prone to privacy breaches than a decentralized one. After being criticized, the German government instead chose to use the ENA. [1]

Broad usage across the population is important for CTAs to have an impact on the development of the pandemic. The CWA has approximately been downloaded 23 million times as of December 2020. [2] The amount of users having the app currently installed may be less. A study shows that approximately one third of the surveyed did not want to install the CWA for several different reasons. [3, Sec. Results] Therefore, the motivation of this paper is to review the architecture of the CWA regarding privacy, security and some other technical considerations which could prevent users from installing the app, such as bugs in the app or the general transparency of the app. Furthermore, this paper compares the ENA approach with different approaches.

In Section 2, it is explained how the CWA and other related ENA-based CTAs work. Privacy, security and other technical considerations are dealt with in Section 3. The ENA is compared with other approaches in Section 4. Afterwards, related work is summarized in Section 5. A conclusion is given in Section 6.

2. Functionality

There are three parts of the CWA that are essential for contract tracing. The proximity detection is responsible for keeping track of nearby users, while sharing the infection information uploads data of the infected user to the central CWA server. The infection risk calculation consists of getting a list of keys which represent infected users from the CWA server and comparing them with the local data captured by the proximity detection.

2.1. Proximity detection

The proximity detection is part of the ENA and uses *Bluetooth Low Energy* (BLE). Smartphones running ENA-based CTAs broadcast and scan for BLE messages with the ENA service identifier around every 3.5 to 5 minutes. The exact interval is determined by a randomized component to prevent tracking. [4, p. 4] [5, `scanIntervalRandomRangeSeconds()` comment]

The payload of a BLE broadcast consists of the *Rolling Proximity Identifier* (RPI) and the *Associated Encrypted Metadata* (AEM). The RPI serves as a temporary identifier for the sending device and is newly derived every 15 minutes. This happens at the same time the randomly generated Bluetooth MAC address changes. The RPI contains a bucketized version of the Unix timestamp with a bucket size of ten minutes and is AES-128 encrypted using the RPI key. The RPI key itself is derived from the *Temporary Exposure Key* (TEK) using the HKDF function described in RFC5869. [7] The TEK in turn is an identifier that is freshly generated every day using a cryptographic random number generator. [4, p. 3, 4] [6, p. 6]

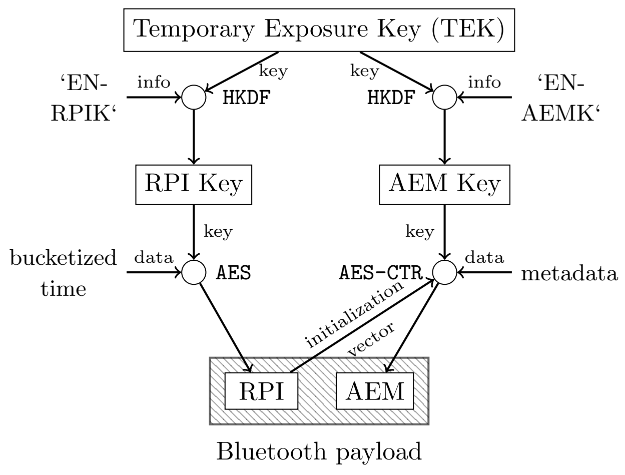


Figure 1: Derivation process of the payload sent in BLE broadcasts. [6, p. 5]

The AEM contains metadata, such as the versioning information about the ENA and the transmit power which was used to send the BLE broadcast. [4, p. 4] This information is later used when determining the infection risk. Similar to the mechanism regarding the RPI, the AEM is encrypted by the AEM key that is derived from the TEK using HKDF. Analogous to the RPI, it also changes every 15 minutes to prevent tracking. As initialization vector for the AES-128-CTR encryption, the current RPI is used. [6, p. 7] The derivation process is visualized in Figure 1.

When scanning and finally receiving a BLE broadcast of another user in proximity, the RPI and the encrypted AEM are stored locally on the smartphone. Decryption of the AEM is only possible with the TEK of the user that initially sent the broadcast. [6, p. 7]

2.2. Sharing infection information

Sharing a positive test result using the CWA allows for other users to check their chance of being infected later in the process of the infection risk calculation.

There are multiple ways on how to mark oneself as infected. Some laboratories print QR codes on the letter that the user receives after conducting the test. The CWA supports scanning this code and will notify users as soon as there is a test result. [8]

Not all laboratories may support this. In this case, the German health authorities may share a code with the infected user that can be used to share their infection status. This code is distributed when the authorities call the users to inform them about the measures they have to take regarding their infection. [8] The user can also take action and phone the CWA call center to receive a code.

As soon as an user is tested positive for COVID-19, there is the possibility of sharing the test result with the CWA server. What is being transmitted to the server in this case are all the TEKs of the last 14 days. The list of TEKs is referred to as diagnosis key. [6, p. 8]

2.3. Infection risk calculation

When installed, the CWA automatically pulls recent diagnosis keys from the CWA server. In previous versions this happened on a daily basis. Recent versions (since v1.7) allow multiple downloads per day, which also means that the infection risk can be invalidated multiple times per day. [9]

Using the downloaded diagnosis keys and the contained TEKs, the CWA can recalculate the RPI keys as well as the AEM keys. Matching locally stored RPIs and AEMs can then be decrypted. For each match, the total encounter time on that particular day is calculated. Additionally, the distance between the smartphones is determined using the signal strength.

If an encounter belonging to the match lasted less than 10 minutes or the distance was larger than 8 meters on average, it is automatically classified as low risk. [10]

For each of the remaining encounters, the total risk score is calculated by multiplying four scaled metrics, ranging from 0 to 8 [11, Sec. Risk Score Calculation], namely:

- days since the exposure has happened
- exposure duration
- signal attenuation
- transmission risk level

The transmission risk level is calculated using an epidemiological model and contained in the uploaded diagnosis key. For example, it can possibly take symptoms entered by the user into account. [12] This particular model is not part of the ENA, but it uses the customizable transmission risk level offered by the ENA.

With taking all exposures into account, a combined risk score is calculated. First, the attenuation levels are grouped into three buckets using predefined thresholds. Each bucket's weight is then multiplied with the sum of the corresponding exposure durations for which the attenuation falls into one of the buckets. The result is called the exposure score. It is multiplied with the normalization of the largest total risk score to finally get the combined risk score. [11, Sec. Risk Score Calculation]

If the combined risk score exceeds a certain threshold, the user is shown a high risk exposure warning.

3. Considerations

CTAs are reliant on a broad acceptance of the population to be effective. The acceptance increases if the app does not interfere with the privacy of the users. For example, this could mean that data that could reveal the users identity is not shared with others.

Moreover security issues, for example attacks that generate fake risks which are shown to the users, can cause uncertainty.

Furthermore different technical considerations could affect the acceptance, such as bugs in the app preventing it from working correctly.

3.1. Privacy

In general, the CWA has been designed with the goal of ensuring as much privacy as possible. Consequently, there have been multiple measures to guarantee privacy, such as frequently changing identifiers, the usage of cryptographic methods for identifier generation / derivation and using a decentralized concept. In contrast to a centralized approach, no information leaves the smartphone, except when sharing a positive test result. Still, the most frequent concerns to not use the CWA are privacy concerns. [3]

Nevertheless, there have been successful attempts demonstrated in literature to circumvent these privacy measures. The resulting privacy threats are mainly deanonymization and movement tracking of users. If exploited, these threats could lead to loss of acceptance in the population.

Movement tracking of users sharing their positive test result is practically possible for all apps using the ENA, such as the CWA. As described in Section 2.2, sharing the result requires the users to upload their TEKs of the last 14 days. The TEKs can then be queried from the CWA backend by anyone. Using BLE sniffers deployed at central locations, such as train stations or supermarkets, one can trace the movement of infected users for at least one day by deriving RPIs from a particular TEK and comparing them with the RPIs picked up by the sniffers. Tracing the movement for longer than one day is also possible, if one manages to match multiple uploaded TEKs using the users movement behavior. Using the movement information, one can possibly also deanonymize users. [13, Sec. III]

Limitations of this approach are certainly not being able to trace users who did not share their test result. In addition it requires the deployment of BLE sniffers. For tracing people in a city with a population of around 160 000 people, approximately 430 strategically placed sensing stations would be necessary. [13, Sec. III] This amount of sniffers needed makes this approach infeasible on a Germany-wide scale. Especially the government, as publisher of this CTA, has access to more suitable methods for tracking users, such as using the data from the mobile networks.

On a smaller scale, deanonymization is also possible using another attack with a BLE device capturing signals at multiple locations. One can then store the RPIs and the signal strength at each location. Observing the locations when capturing the signals establishes a connection between the captured signals and the observed person. Another similar attack is to approach a person and track the RPIs sent by the persons smartphone. When there are not many other signals around, one can likely identify the RPIs belonging to the approached person. If the person is now tested positive for COVID-19 and shares the infection status, one has gained the information that the person is infected. This can be done by deriving the RPIs from the uploaded TEKs and comparing them with the previously picked up RPIs. [14, Sec. 4.2]

These attacks require personal proximity to the victims or camera surveillance and are therefore only possible on a small scale. There are mitigation proposals for both attacks, such as varying signal strength when sending BLE broadcasts. [14, Sec. 4.2]

As seen in the limitations, all presented attacks require a significantly large effort to be able to track users on a large scale. Nevertheless, they are feasible when tracking users on a smaller scale.

3.2. Security

Besides privacy issues, security issues can lead to attacks that stop the app to work in the desired way. For example this could be generating fake risks that lead to warnings for users.

Literature has shown that wormhole attacks (also *relay and replay attacks*) are possible for ENA-based apps. Such an attack picks up a BLE signal at some location, preferably a crowded one. Then the attacker uses a second device at a different location. The second device receives BLE messages that the first device picked up. This is done with the help of a tunnel built by the attacker between the two devices. Now the second device broadcasts these messages and all devices will receive BLE broadcasts originated from the first location, while actually being at the second location. [13, Sec. IV]

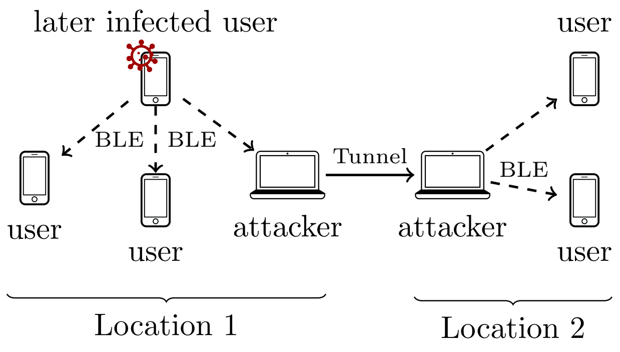


Figure 2: Example setup of a wormhole attack. [13, Figure 7]

Figure 2 shows an example setup of this attack. All users present at location 2 will receive broadcasts from the later infected user at location 1, although this user is not necessarily in proximity in reality.

This attack can be used to generate fake risk contacts, which may tempt users to conduct a test or go into quarantine without a real risk being present. Possible mitigations require either a handshake mechanism or additional verification using the GPS location or the cellular network. [14, Sec. 3.2] A limitation of this attack is that physical presence of some kind (e.g. a smartphone or a microcontroller) is required at the locations where the attack should be performed.

Another possible attack is called power and storage drain attack. It is a denial of service attack, in which the attacker broadcasts a large amount of BLE messages. Devices in the proximity will pick up and process these messages. A large amount of messages to process will result in a higher power consumption. If the attacker manages to generate valid messages, they will also use space on the smartphones storage, as the RPIs and the AEMs are persisted. [14, Sec. 3.1] While this attack may be less severe than the relay and replay attacks, the users' acceptance will decrease if such an attack occurs at her smartphone. A mitigation for this attack is also proposed in literature. [14]

Both presented attacks are hard to apply at a large scale, because they require physical devices at the attacked locations. Nevertheless, anyone exploiting these attacks will certainly lead to the app not working as intended.

3.3. Technical

Since the CWA launched in June 2020, several bugs were discovered. Some of these bugs were preventing users from utilizing the app.

One problem that occurred in early versions was that the CWA was not refreshing the infection risk value on some Android devices without the user manually opening the app. Broadcasting and receiving RPIs would work, but in case an user had an encounter with a later positive diagnosed person, the user would not get a notification without opening the app. A fix deployed later added a setting which, when enabled, lets the CWA run in the background even on Android systems which stop apps running in the background for battery saving reasons. [15]

Another bug appeared for iOS users with the update to version 1.2.0 released in early August 2020. Some users were not able to start the app any more after the update. This was quickly fixed in a follow-up update released five days later. [16]

In September 2020, an additional bug was found which affected smartphones running iOS 13.7. The bug caused the computation of the risk value to be faulty, and would ultimately result in displaying a too high risk for some users. [17]

An additional reason for not using the CWA may be the power consumption of the app. Using the app could lead to a decreased runtime of the smartphone. There is no relevant literature that investigates battery consumption of the ENA or CWA. Nevertheless, BLE was chosen as it is explicitly designed for usage in environments with battery constraints, such as smart home applications for example.

Another technical aspect is transparency. Transparency certainly leads to higher confidence of users that the app contains what is being promised. The CWA is completely open-source and reproducible builds are currently worked on, which then gives certainty that the code in the GitHub repository belongs to the deployed binaries in the app stores. [18]

4. Comparison

Knowing the functionality and issues of the CWA makes it interesting to compare it to other approaches. An overview of selected other approaches is shown in Table 1.

Architecture	Concept	Country
Decentralized	ENA	Germany, Denmark [19], Brazil [20], Italy [21], Spain [22], United Kingdom [23], United States *(partly)* [24], Canada *(partly)* [25]
Partially-centralized	BlueTrace	Singapore [26], Australia [27]
	ROBERT	France [28]
	other	Iceland, India

TABLE 1: Architectures, theoretical concepts and corresponding deployment location of selected CTAs

The underlying concepts of most CTAs can be grouped into two categories regarding their architecture. There are *decentralized* and *partially-centralized* architectures. *Partially-centralized* architectures generally require more interaction of the users with a central server. Examples of an interaction may be an initial registration with personal contact information or an upload of encounters to the server, depending on the implementation. In contrast, for *decentralized* architectures the only transmission of user data to the server may possibly happen when sharing a positive test result, which is not mandatory for using the app.

The most prominent concept using a *decentralized* architecture is the already discussed ENA, which is used by many western countries as seen in Table 1. On the other hand. there are different concepts using a *partially-centralized* architecture, such as *BlueTrace* or *ROBERT*.

In the following subsections selected concepts using a *partially-centralized* architecture are compared with the ENA.

4.1. BlueTrace

BlueTrace has been developed by the government of Singapore. [29, Sec. Abstract] As seen in Table 1, it is currently used in Australia and Singapore.

To use the app, users have to register using their phone number. An account is then created on the backend side, containing the phone number and a randomly generated user identifier. [29, Sec. 4]

Similar to the ENA, the proximity detection uses BLE broadcasts with frequently changing temporary identifiers. In contrast to the ENA, the temporary identifiers are not generated by the user but by the central server. After receiving them from the server, they are broadcasted by the user's smartphone. A temporary identifier includes the user identifier and time information and is encrypted on the server using symmetrical encryption, which enables only the health authority to decrypt it. Analogously to the ENA, received broadcasts are stored on the local smartphone storage. [29, Sec. 4]

If users are tested positive, they upload their locally saved encounters to the central server. The health authority can then decrypt the temporary identifiers and contact the encounters using their phone number saved in the server's database. [29, Sec. 4]

In contrary to the ENA, BlueTrace is only affected by wormhole attacks (see Section 3) to a limited extent. Firstly, this is the case because the broadcasts contain an expiry timestamp, which the server verifies upon uploading the encounter history. Therefore, the broadcast of a user can only be rebroadcasted for a maximum of 15 minutes. Secondly, human operators also verify the locations of both, the infected user and potentially infected users, via phone. [29, Sec. 8] This does not completely rule out wormhole attacks, but may limit their effectiveness.

Bluetooth sniffer attacks by third parties as in Section 3 are not applicable to BlueTrace, assuming a third party cannot decrypt the broadcasts of users and is therefore not able to track them beyond the 15 minutes refresh interval of identifiers. However, such an attack concerning all users could be performed by the health authority, as they are able to decrypt all temporary identifiers.

BlueTrace may also be more effective when it comes to risk classification, as employees of the health authority can decide to contact encounters based on some additional context given by the infected person during the phone call.

Nevertheless, the main weakness of BlueTrace is that if somebody manages to obtain the secret key of the health authority, every temporary identifier could be decrypted. With additional access to the database of the server, every temporary identifier could be connected with the users' phone numbers. This is not possible using the ENA, because the data that is sent in the broadcasts is encrypted with keys generated by the users themselves. Additionally, no personal data of users is stored in context of the ENA.

4.2. ROBERT

The French CTA uses a concept called *ROBust and privacy-presERving proximity Tracing* (ROBERT), which is in turn built upon the *Pan-European Privacy-Preserving Proximity Tracing* (PEPP-PT). The German government initially pursued to implement a CTA based on PEPP-PT. [1]

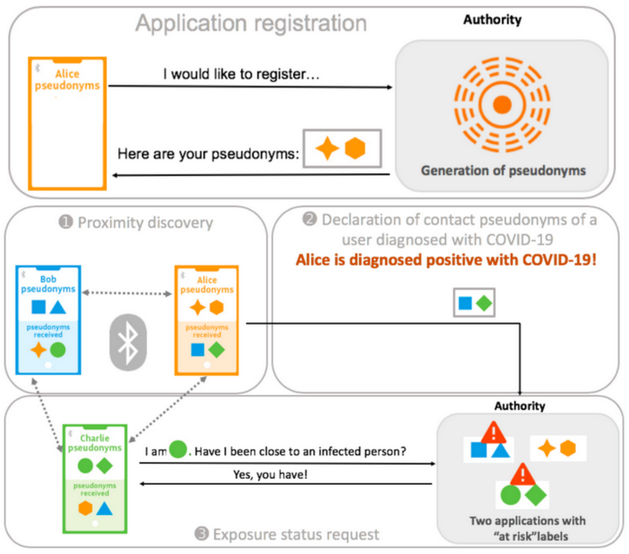

Application registration
Authority
Alice pseudonyms
I would like to register...
Here are your pseudonyms:
Generation of pseudonyms
❶ Proximity discovery
❷ Declaration of contact pseudonyms of a user diagnosed with COVID-19
Alice is diagnosed positive with COVID-19!
Bob pseudonyms
pseudonyms received
Alice pseudonyms
pseudonyms received
Authority
Charlie pseudonyms
pseudonyms received
I am ●. Have I been close to an infected person?
Yes, you have!
Two applications with "at risk" labels
❸ Exposure status request

Figure 3: Tracing flow of ROBERT. [30, p. 3]

Similar to BlueTrace, ROBERT relies on temporary identifiers that are broadcasted via BLE and generated by a central server. [31, Sec. 4, 5.1] The user regularly gets these identifiers from the server and uses them for broadcasts. Received broadcasts are saved to the local smartphone storage, analogously to the ENA and BlueTrace. [31, Sec. 5.2]

If a user is tested positive and wants to share the infection status, the encounter history is uploaded to the central server. The server then calculates the encounter times and adds them to the database entry belonging to the encountered user. [31, Sec. 6]

Every user regularly sends a request to the server with recently used identifiers. The server checks if there has been an encounter and returns the result to the client. [31, Sec. 7]

The tracing flow of ROBERT is depicted in Figure 3.

The main difference to BlueTrace is that the user does not have to send personal information, e.g. the phone number, to the server.

Compared to the ENA, firstly ROBERT uses identifiers generated at the server and not at the local smartphone and secondly relocates the logic of risk calculation to the server. While wormhole attacks may still be viable, sniffer attacks as described in Section 3 become impossible for third parties, as there is no publicly accessible list of infected identifiers. On the other hand, ROBERT makes it possible to perform sniffing attacks for all users when having access to the key used by the server.

4.3. Other approaches

There are other approaches which do not use BLE for proximity detection. The Icelandic CTA uses the locations services (e.g. GPS) of the smartphone's operating system. Only when sharing the infection status or upon request of the authority, the location history can be uploaded to a server. [32] The authority can then take measures, for example by warning people regularly being present at one of the locations.

The CTA of India uses yet another approach. It combines both, Bluetooth and GPS data for proximity detection. Additionally, it requires users to register themselves by providing personal information, such as their name, their age or their phone number. If an infection happens, the Bluetooth encounter history is uploaded to the server together with the location information. [33]

5. Related Work

Most of the literature focuses on one particular aspect of the CWA. The analysis in [14] gives a theoretical, detailed overview of security and privacy issues in the ENA, while [13] contains two case studies demonstrating security and privacy issues of the CWA. Similar to this paper, both papers give a short overview about the functionality.

While being technical, the influence of issues on the apps acceptance is not discussed. Most literature discussing reasons why not to use the CWA are not technical, but rather only conduct representative surveys of the population, like it is the case for [3].

There is no literature that explicitly looks at the population's concerns about the app and compares them with the technical background. This paper tries to fill this gap.

6. Conclusion

There are privacy issues in the CWA that could lead to deanonymization and tracking of users in the worst-case. In addition, wormhole attacks can decrease the usefulness of the app by generating fake risk warnings. Also the user experience was cumbersome especially in the beginning, as some users were not able to correctly use the app due to bugs. Nevertheless, the most critical bugs were fixed in the meantime.

While there exist these privacy and security issues and real world attacks may be possible for single cases, they are not feasible for a large scale. Even on a small scale, a large amount of effort is required. In fact, the CWA provides a decentralized architecture which ensures that no sensitive data leaves the smartphone. Information like the location or identity are in no means transmitted to the server, instead, a design of locally generated and frequently changing identifiers is used. For other approaches, such as the partially-centralized ones, this mostly is not the case. In their case, this could possibly lead to more drastic worst-case privacy breaches than it is the case for the decentralized approach.

Especially in regards to previous software projects developed by the government, the CWA seems to be an good example in terms of privacy and transparency.

To conclude, the privacy and security measures of the CWA are good enough for attacks only to have a limited impact on a large scale. Users with privacy concerns may not know about the effort of the measures taken to ensure this level of privacy and security.

References

[1] K. Becker and C. Feld, "Corona-Tracing: Bundesregierung denkt bei App um," *Tagesschau*, April 2020, https://www.tagesschau.de/inland/coronavirus-app-107.html, [Online; accessed 10-December-2020].

[2] Robert Koch-Institut, "Kennzahlen zur Corona Warn App," December 2020, https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/WarnApp/Archiv_Kennzahlen/Kennzahlen_04122020.pdf, [Online; accessed 10-December-2020].

[3] K. T. Horstmann, S. Buecker, J. Krasko, S. Kritzler, and S. Terwiel, "Short report: Who does or does not use the "Corona-Warn-App" and why?" *European Journal of Public Health*, 12 2020, ckaa239. [Online]. Available: https://doi.org/10.1093/eurpub/ckaa239

[4] Apple/Google, "Exposure Notification Bluetooth® Specification," April 2020, https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-BluetoothSpecificationv1.2.pdf, [Online; accessed 12-December-2020].

[5] Google, "Exposure Notifications Internals - ContactTracingFeature.java," August 2020, https://github.com/google/exposure-notifications-internals/blob/main/exposurenotification/src/main/java/com/google/samples/exposurenotification/features/ContactTracingFeature.java#L367, [Online; accessed 07-January-2021].

[6] Apple/Google, "Exposure Notification Cryptography Specification," April 2020, https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-CryptographySpecificationv1.2.pdf, [Online; accessed 12-December-2020].

[7] H. Krawczyk and P. Eronen, "Hmac-based extract-and-expand key derivation function (hkdf)," Internet Requests for Comments, RFC Editor, RFC 5869, May 2010, http://www.rfc-editor.org/rfc/rfc5869.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5869.txt

[8] Corona-Warn-App Team, "Software Design Verification Server," July 2020, https://github.com/corona-warn-app/cwa-verification-server/blob/master/docs/architecture-overview.md, [Online; accessed 14-December-2020].

[9] ——, "Android App Releases," December 2020, https://github.com/corona-warn-app/cwa-app-android/releases, [Online; accessed 24-December-2020].

[10] ——, "Risk Assessment," December 2020, https://github.com/corona-warn-app/cwa-documentation/blob/master/cwa-risk-assessment.md, [Online; accessed 24-December-2020].

[11] ——, "Solution Architecture," December 2020, https://github.com/corona-warn-app/cwa-documentation/blob/master/solution_architecture.md, [Online; accessed 24-December-2020].

[12] ——, "Epidemiological Motivation of the Transmission Risk Level," October 2020, https://raw.githubusercontent.com/corona-warn-app/cwa-documentation/master/transmission_risk.pdf, [Online; accessed 24-December-2020].

[13] L. Baumgärtner, A. Dmitrienko, B. Freisleben, A. Gruler, J. Höchst, J. Kühlberg, M. Mezini, R. Mitev, M. Miettinen, A. Muhamedagic, T. D. Nguyen, A. Penning, D. F. Pustelnik, F. Roos, A.-R. Sadeghi, M. Schwarz, and C. Uhl, "Mind the gap: Security & privacy risks of contact tracing apps," 2020, https://arxiv.org/abs/2006.05914.

[14] Y. Gvili, "Security analysis of the covid-19 contact tracing specifications by apple inc. and google inc." Cryptology ePrint Archive, Report 2020/428, 2020, https://eprint.iacr.org/2020/428.

[15] A. Wilkens, "Corona-Warn-App: SAP erläutert Problem mit der Hintergrundaktualisierung," heise online, July 2020, https://www.heise.de/news/Corona-Warn-App-SAP-erlaeutert-Problem-mit-der-Hintergrundaktualisierung-4851648.html, [Online; accessed 29-December-2020].

[16] L. Becker, "Corona-Warn-App öffnet sich nicht mehr auf iPhones: Update soll helfen," heise online, August 2020, https://www.heise.de/news/Corona-Warn-App-oeffnet-sich-nicht-mehr-auf-iPhones-Update-soll-helfen-4869676.html, [Online; accessed 29-December-2020].

[17] J. Hoerdt, "Problems with iOS 13.7," September 2020, https://www.coronawarn.app/en/blog/2020-09-10-ios-13-bug, [Online; accessed 29-December-2020].

[18] Corona-Warn-App Team, "F-Droid release and reproducible builds," December 2020, https://github.com/corona-warn-app/cwa-app-android/issues/1483, [Online; accessed 29-December-2020].

[19] Smittestop, "About the app," https://smittestop.dk/about-the-app/, [Online; accessed 11-February-2021].

[20] Governo do Brasil, "Coronavírus-SUS: aplicativo alerta contatos próximos de pacientes com Covid-19," August 2020, https://www.gov.br/casacivil/pt-br/assuntos/noticias/2020/agosto/coronavirus-sus-aplicativo-alerta-contatos-proximos-de-pacientes-com-covid-19, [Online; accessed 11-February-2021].

[21] Presidenza del Consiglio dei Ministri, "Immuni - Domande Frequenti," https://www.immuni.italia.it/faq.html, [Online; accessed 15-February-2021].

[22] RadarCOVID Team, "RadarCOVID iOS App Repository," https://github.com/RadarCOVID/radar-covid-ios, [Online; accessed 15-February-2021].

[23] NHS, "I got an "Exposure Check Complete" notification from the app. What does this mean?" https://faq.covid19.nhs.uk/article/KA-01319, [Online; accessed 15-February-2021].

[24] The Stanford Daily, "CA Notify app offers COVID-19 exposure alerts for Stanford community," December 2020, https://www.stanforddaily.com/2020/12/28/ca-notify-app-offers-covid-19-exposure-alerts-for-stanford-community/, [Online; accessed 15-February-2021].

[25] Canadian Digital Service, "COVID Alert Mobile App Repository," https://github.com/cds-snc/covid-alert-app, [Online; accessed 15-February-2021].

[26] Bluetrace, "BlueTrace Protocol," https://bluetrace.io/, [Online; accessed 15-February-2021].

[27] Australian Government, "Technology behind COVIDSafe," https://www.covidsafe.gov.au/technology.html, [Online; accessed 15-February-2021].

[28] INRIA, "TousAntiCovid Repository," https://gitlab.inria.fr/stopcovid19/accueil, [Online; accessed 15-February-2021].

[29] J. Bay, J. Kek, A. Tan, C. Sheng Hau, L. Yongquan, J. Tan, and T. Anh Quy, "BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders," 2020, https://bluetrace.io/static/bluetrace_whitepaper-938063656596c104632def383eb33b3c.pdf, [Online; accessed 27-February-2021].

[30] Inria and Fraunhofer AISEC, "ROBERT: ROBust and privacy-presERvingproximity Tracing Summary," April 2020, https://raw.githubusercontent.com/ROBERT-proximity-tracing/documents/master/ROBERT-summary-EN.pdf, [Online; accessed 28-February-2021].

[31] ——, "ROBERT: ROBust and privacy-presERvingproximity Tracing," May 2020, https://raw.githubusercontent.com/ROBERT-proximity-tracing/documents/master/ROBERT-specification-EN-v1_1.pdf, [accessed 28-February-2021].

[32] The Directorate of Health of Iceland and The Department of Civil Protection and Emergency Management of Iceland, "Rakning C-19," 2020, https://www.covid.is/app/en, [Online; accessed 28-February-2021].

[33] FTI Consulting Asia Pacific, "A Review of India's Contact-tracing App, Aarogya Setu ," September 2020, https://www.lexology.com/library/detail.aspx?g=f54419a1-4823-404c-92f3-c5e4f193b733, [Online; accessed 28-February-2021].

Precision Time Protocol - Security Requirements

Tizian Leonhardt, Filip Rezabek*, Kilian Holzinger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: tizian.leonhardt@tum.de, rezabek@net.in.tum.de, holzinger@net.in.tum.de

Abstract—**The ever-growing amount of timing-sensitive applications necessitates clock synchronization that can offer guarantees of high precision. The Precision Time Protocol offers sub-microsecond accuracies for clock-based networks. However, there is sufficient evidence that attacks are a substantial threat that can have devastating consequences. In this paper, we examine security requirements in the context of the Precision Time Protocol and evaluate how they may be met by different security solutions, as well as one of its open-source implementations,** `linuxptp`**.**

Index Terms—**time protocols, network security, ptp**

1. Introduction

The increasing need for clock synchronization with high precision requirements demands protocols that can achieve accuracies in the micro and nanosecond ranges. The Precision Time Protocol (PTP), standardized in the `IEEE1588` standard, can cater to these requirements. It largely surpasses the Network Time Protocol (NTP); compared to NTP delivering accuracies in the millisecond range, PTP allows for sub-microsecond accuracies [1]. This is achieved with timestamps at the hardware level, effectively bypassing any noise that would be introduced by the network stack [2].

We now want to motivate why the topic of security is worth discussing in the context of timing protocols. An obvious result of an attack is the falsification of one or more clocks in the network. The implications of this seemingly harmless effect are not to be underestimated. Smart grids, as an example, rely on accurate timestamps and often have the obligation to deliver accuracies in the microsecond range [3]. This enables them to effectively deliver electricity, a crucial resource. Attacks on power delivery can have devastating consequences [4]. Systems that rely on high accuracies are also more sensitive to attacks, as deviations have a higher influence, making PTP an attractive goal for attackers. This paper aims to analyze the requirements of a secure PTP environment, as well as to evaluate different security solutions concerning these demands.

In Section 2, we first lay the technical foundation needed for understanding PTP, as well as its different versions. This also entails a discussion of the aforementioned security requirements. Section 3 contains the analysis of a handful of security solutions in the context of different attack scenarios, the results of which are summarized in a table. In Section 4, we compare the previous results with `linuxptp`, an open-source implementation of PTP. Section 5 concludes the paper and gives an outlook on future work.

2. Background

In this section, we discuss the technical intricacies surrounding PTP, as well as its different versions. We also review the security requirements defined in RFC7384 [5].

2.1. Precision Time Protocol

The following findings are, unless otherwise noted, based on [1]. As already stated, PTP allows the synchronization of multiple clocks in a network with high precision. The protocol is made up of a multitude of clocks serving different purposes, laid out in a master-slave hierarchy. Figure 1 seeks to give an overview of this. Ordinary clocks (OC) have one external port and act as either master or slave. A boundary clock (BC) on the other hand features n ports, where $n > 1$. It is responsible for synchronizing the network segment it governs and on one port listens as a slave for the synchronization of its own clock. The BC propagates this clock to the remaining $n-1$ ports associated with clocks in the segment. While more variants of clocks exist, we only presented the ones we deem necessary for a general understanding of a PTP network.

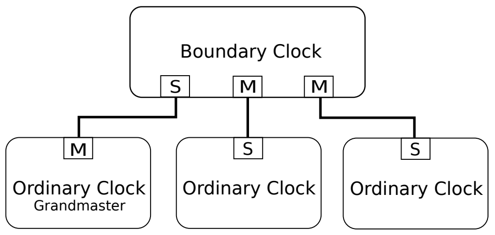


Figure 1: A master-slave clock diagram [1, 6.6.2.4]. 'M' marks a port as master, 'S' as slave. The grandmaster clock is highlighted.

In the case of Figure 1, the OC marked grandmaster presents a special case. This clock is responsible for propagating the time reference and therefore establishes the idea of time in the system. This reference can be fetched from a reliable external source, such as a GPS signal. The grandmaster clock is dynamically chosen by the Best Master Clock (BMC) algorithm, which picks the best candidate according to various criteria, such as the quality of the time source; the candidates have to announce their parameters in order to register for this election.

The IEEE1588 standard also defines a way to deal with cyclic paths in mesh topologies. To avoid synchronizing a BC from multiple sources, superfluous paths are removed by setting the corresponding slave port to passive; this prevents any timing information from being exchanged. Figure 2 illustrates this with an example where one path is pruned, resulting in a tree structure. Note that only the ports necessary for this example have a connection. Besides electing the most suitable master clock to be the grandmaster, the BMC algorithm is also responsible for selecting the path to be excluded.

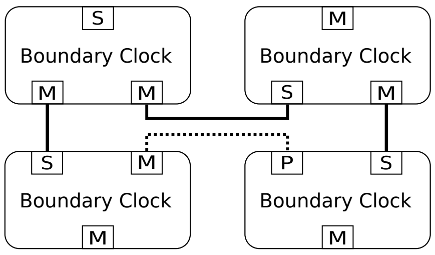


Figure 2: An example of mesh topology pruning [1, 6.6.2.5]. The pruned path is dashed. 'P' marks a port as passive.

The second PTP component we consider is the mechanism responsible for synchronizing the clocks. Propagating the grandmaster clock value itself is trivial, but the delay between the master and its slaves also has to be accounted for. This is determined with a sequence of protocol messages that compute the delay. When a master initiates the synchronization, the slave denotes the time when the message arrived, which is followed up by the master transmitting the time of his initial request. This is subsequently done in the direction of the slave to his master as well. With this information, the correct offset is computed. It should be noted that this process relies on the central assumption that the delay between master and slave is equal in both directions, i.e., the paths are symmetric [1, 6.2].

2.2. PTP Versions

Currently, three versions of the IEEE1588 standard exist. The 2002 version is not of interest for this paper as it is outdated and incompatible with the newest revision [6]. In contrast, the 2008 revision remains largely compatible with the newest standard [6]. During the last twelve years, many issues with this version have been identified [2], [7]. This raises the need for an improved standard, which is now released as revision 2019 and aims to fix many of the aforementioned issues. Besides that, there is also the IEEE standard 802.1AS, an adoption of the IEEE1588 standard to better accommodate to time-sensitive audio and video traffic [8]. The remaining sections of the paper revolve around the two latest IEEE1588 versions.

2.3. Security Requirements

Based on the previous insights, it is discernible that we need to protect against attacks to warrant the security of time-critical systems. This is a relevant topic to PTP, not just as a result of its high accuracy demands, but also because security was not a main concern during the design of the first two revisions [9]. For the 2008 version, an experimental annex ('Annex K') to the standard exists, providing "group source authentication, message integrity, and replay attack protection for PTP messages." [1, K1] On top of its experimental nature, multiple sources state the obsoleteness of this annex [2], [10], which is why we pay little attention to it going forward.

In order to better understand the demands of a secure PTP environment, RFC7384 [5] offers a guideline by listing security requirements in various contexts. Our evaluations going forward are largely based on this RFC. We focus on the so-called MUST-types (see also [11]), i.e., requirements that *have* to be implemented to create a secure PTP environment. Unless otherwise stated, all requirements in the following sections are of this type. The relevant requirements for later parts of the paper include [5]:

- Authentication and Authorization
- Integrity protection
- Spoofing prevention
- Replay protection
- Protection against delay and interception
- Availability

'Authentication and Authorization' is concerned with uniquely identifying clocks in the network and ensuring that their respective behavior does not violate permission boundaries. The 'Integrity protection' requirement necessitates techniques to verify that messages have not been corrupted or tampered with. 'Availability' describes the protection against Denial of Service (DoS) attacks.

3. Threat Mitigation

We now highlight a selection of attacks and their respective security solutions. We also evaluate them in regard to the aforementioned requirements. In the context of the 2008 revision of PTP, we focus on security solutions that are novel to the standard. For the 2019 version, we focus on the new security features that are integrated into the standard; this features some general security considerations instead of solely focusing on a attack scenario. This section concludes with a table that presents the results in a compact form.

3.1. IEEE1588-2008

We begin the analysis with the 2008 revision. Each attack addressed is placed in a new subsection. Even though this revision is already superseded, it is still of interest for this paper due to the newness of the IEEE1588-2019 standard at the time of writing.

3.1.1. Delay Attacks. Reference [3] revolves around exploiting the assumption of symmetric paths discussed in Section 2.1. The threat model is based on an attacker with access to the internal network infrastructure. The attack is executed by delaying the messages used for computing the delay between two PTP nodes in one direction, effectively creating an asymmetry that "introduces an error in the computed value of the clock offset.” [1, 6.6.3] This ultimately leads to a skewed clock, violating the requirement of ’Protection against delay and interception’ [5]. Figure 3 seeks to explain this asymmetry caused by a rogue node.

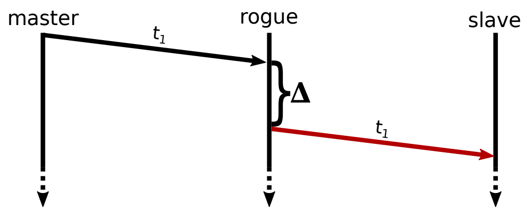


Figure 3: A rogue node intercepts and delays a synchronization message by Δ. Note that this only happens unidirectional. In [3], Δ is chosen randomly.

In order to combat this vulnerability, [3] proposes a solution based on detection and mitigation. Detecting a delay attack is made possible by installing a second, redundant clock that is retrieving its timing information from the same source as the grandmaster (e.g. a GPS signal). This node also responds to delay computation requests and calculates the new clock value; if the difference of this value is not equal to the external time reference, an attack is likely in progress. To reduce the impact of the attack, a cumulative average based on previous offsets for each node is used for calculating the clock value. The fact that this does not completely annul the effects of the delayed messages is discussed by [3], with the conclusion that this method leaves enough time for authorities to respond to the attack, as the rogue node has to be inside the network. Whether or not this is a realistic assumption is not further evaluated. With enough time, an attacker can still skew the clocks, leaving the system open to attacks if the response is not timely enough.

3.1.2. Denial of Service. Even though RFC7384 defines the protection against DoS attacks as a SHOULD-requirement (’Availability’) [5], the low amount of effort needed for the DoS-attack demonstrated by [9] is sufficient evidence for treating it as an important requirement that should not be overlooked. The technique demonstrated relies on forging spurious synchronization packets that are sent to slaves at a rate of around 292 packets per second. The forged packets contain correct identification details (e.g. the clock ID) for the corresponding master node, which can be obtained by sniffing the traffic; the semantics of the synchronization itself are non-existent, as they are not needed for the attack to succeed. After gathering this, the packets can be sent without any knowledge about the slaves through the fixated multicast address. The setup for the attack is therefore comparatively simple, and indeed, [9] reports delays of multiple hours in the test environment by overburdening the nodes with the forged traffic. Even though further tests in real-world PTP networks would be necessary to assess the actual impact of the attack, it still is a significant result.

Just as discussed in Section 3.1.1, [9] proposes solutions based on mitigation, as well as detection. For the former, multiple approaches are suggested. We only discuss the introduction of a digital entity, as the other methods are not further evaluated. This digital entity is introduced for master nodes, opening up the possibility of identifying themselves through cryptographic means. Utilizing this, nodes are able to filter packets based on whether or not they “originate from masters with a valid identity.” [9] The desired effect is furthermore confirmed, substantiating complete protection from the demonstrated attack. We note that the specific implementation of the digital entity is out of the scope of this paper, but it is not guaranteed that other approaches would yield the same efficacy.

3.1.3. Best Master Clock Spoofing. The Best Master Clock algorithm [6], as stated in Section 2, chooses the best clock out of a set of potential masters to act as the grandmaster. This procedure, at its core, compares software-defined quality parameters that clocks announce and acts accordingly. The parameters include, but are not limited to [6, 6.6.2.3]:

- `priority1`
- `clockClass`
- `clockAccuracy`

The value `priority1` is an integer chosen by the administrator to allow for own priority suggestions, whereas `clockClass` categorizes clocks into further subcategories. Especially interesting is `clockAccuracy`, which provides an upper bound for the accuracy offered by the individual clocks. Accuracy bounds range from more than ten seconds down to one picosecond [6], several orders of magnitude smaller than the performance advertised by the standards surrounding PTP. It is apparent that there is a significant potential for abuse by spoofing values that no clock in a real-world scenario would offer. The need for protection against this kind of attack is described by the ’Spoofing Prevention’ requirement [5].

This attack is successfully demonstrated in [2]. Two types of attackers are considered: an external attacker that can only see the public multicast traffic, as well as an internal attacker, which is also a node of the PTP network. Both approaches make use of setting a selection of the previously discussed quality parameters to the best possible values. Announcing these parameters guarantees a win in the election and therefore control over the time propagation. Reference [2] suggests the use of symmetric cryptography in order to mitigate the attack from an external standpoint; this is also the elected method in Annex K, which is applicable here [2], despite its flaws. In contrast, the internal attacker, as part of the PTP network itself, would know the secrets of a symmetric encryption. To counter this, the employment of asymmetric cryptography is suggested, which is confirmed as an effective measure. Reference [9] also discusses this technique, but extends it by closely mimicking the behavior of other master clocks in the network; the details are gathered through sniffing.

3.2. IEEE1588-2019 - Annex P

As a replacement for the obsolete Annex K, the 2019 revision of PTP includes a new security model on which this section is based, defined in Annex P [6]. This is based on four prongs [6], each serving a different purpose in terms of security. The standard even acknowledges RCF7384, stating that the approach presented is tied to the requirements mentioned there. The prongs are made up of the following concepts:

A Integrated Security Mechanism
B External Transport Security Mechanisms
C Architecture Mechanisms
D Monitoring and Management Mechanisms

While all of these prongs play an important role, we only focus on prongs A and D in this paper, as they are the closest to the protocol itself. Prong B is concerned with more general networking techniques that could increase PTP security (for example MACsec), whereas prong C discusses topology enhancements, such as redundancies for master clocks.

The integrated security mechanism of prong A employs symmetric cryptography by adding 'type-length-value' (TLV) attributes to the protocol messages. They allow the direct extension of messages with attributes of arbitrary length. For `IEEE1588-2019`, the so-called 'Authentication TLV' provides "source authentication, message integrity, and replay attack protection [...]." [6, 16.14] To do so, the TLV carries all the necessary data to enable the secure processing of messages, such as the 'Integrity Check Value' (ICV) that is used to verify the integrity of a message. The concept of authentication by adding a TLV is also present in Annex K of the `2008` revision with the same goals in mind. Prong A therefore already addresses 7 out of the 12 MUST-requirements found in RFC7384 in a cryptographically sound way. The feasibility of this approach is asserted by [7], confirming that the accuracy of PTP is not negatively impacted. Although prong A addresses many of the requirements, PTP is still possibly open to delay attacks [7] (besides potential others). This is one part of the issues that prong D should address, though the general responsibilities are much broader and can be tailored to fit the needs of the underlying system. One possible way to combat delay attacks has already been presented in Section 3.1.1, which could be implemented for the newest revision as well. A similar approach that is based on prong D is discussed in [12].

3.3. Results

We conclude this section with Table 1, allowing for a convenient point of reference. It contains an overview of the requirements addressed by the references that were cited in Section 3. Note that this table might include additional details about contributions that were not discussed earlier. An x means addressed, a dash means not addressed. References [1] and [6] refer to Annex K and P, respectively.

4. Case Study: linuxptp

Having reviewed a multitude of security solutions, we now shift our perspective towards the available security features of `linuxptp` [13]. This is based on a comparison of the insights already garnered, as well as security features not previously mentioned. We first discuss the supported security TLV types, defined in `tlv.h`. Although the `AUTHENTICATION` TLV introduced in Annex P [6] is present, it is simply ignored during the processing of protocol messages [13, tlv.c]. The same applies to the authentication TLVs that are used in Annex K [1], leaving authentication through those means impossible without additions to the code. Further research reveals that no other options for authentication currently exist.

TABLE 1: Comprehensive overview of requirements [5] addressed by various contributions

Addressed Requirements						
	[3]	[9]	[2]	[12]	[1]	[6]
Authentication and Authorization	-	x	x	-	x	x
Integrity protection	-	-	x	-	x	x
Spoofing prevention	-	x	x	-	x	x
Replay protection	-	-	x	x	x	x
Protection against delay and interception	x	-	x	x	-	x
Availability	-	x	-	x	-	x

A comparatively simple way of checking for attempts at skewing clocks is to compare the value used for offsetting the clock to a maximum and minimum value; either being exceeded could hint at a possible attack. `linuxptp` reacts to unexpected jumps by issuing a warning and returning from the corresponding function with an error value [13, clockcheck.c]. While this does catch obvious attacks and is mentioned as a mitigation mechanism for prong D in Annex P [6], continuously introducing delays, as demonstrated earlier in Section 3.1.1, would still go unnoticed if Δ is chosen within an appropriate range.

Another attractive attack vector are master clocks. They play the central role of synchronizing their slaves. It is therefore unwanted that rogue masters can influence clocks. This is captured by the 'Spoofing Prevention' requirement in RFC7384, which mentions authentication as a possible solution [5]. Even though no authentication mechanism currently exists in `linuxptp`, there still is a check in place to mitigate this attack. Whenever a slave processes synchronization messages (for example in `process_delay_resp` [13, port.c]), the identity of the source port is checked; should the sender of the synchronization messages not align with the currently associated master clock, the message is discarded [13, port.c]. However, as this is based on values that could be obtained by sniffing the traffic (see also [13, ddt.h]), an attacker could simply determine the correct identification for each slave node. The feasibility of sniffing traffic for this type of information is illustrated in [9].

In summary, there is a lot of work that could be done regarding security features in `linuxptp`, especially in light of the integration of the new security features found in Annex P [6]; only the integration of the authentication TLV and the surrounding techniques would address a great number of critical requirements.

5. Conclusion

We have shown that PTP is an interesting target for potential attackers that could have far-reaching consequences. Only slight deviations could influence the accuracy needed for systems that are reliant on it. We then analyzed security solutions with respect to the requirements defined in RFC7384. The results, which are also showcased in Table 1, are promising; the solutions presented are theoretically able to mitigate many critical attacks. The case study on `linuxptp` illustrated the need for better security options, such as mechanisms for authentication. This paper could serve as a solid foundation for security considerations regarding the latest two PTP versions. It also paves the way for further evaluations regarding the state of security solutions. Future work could analyze new experiences with annex P (`IEEE1588-2019`), as the standard was comparatively novel at the time of writing. There are also many opportunities concerning the design of improved security features for `linuxptp`.

References

[1] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std., Jul. 2008.

[2] E. Itkin and A. Wool, "A security analysis and revised security extension for the precision time protocol," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 1, pp. 22–34, Jan. 2020.

[3] B. Moussa, M. Debbabi, and C. Assi, "A detection and mitigation model for PTP delay attack in a smart grid substation," in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, Nov. 2015.

[4] N. Kshetri and J. Voas, "Hacking power grids: A current problem," *Computer*, vol. 50, no. 12, pp. 91–95, Dec. 2017.

[5] T. Mizrahi, "Security requirements of time protocols in packet switched networks," Internet Requests for Comments, RFC Editor, RFC 7384, Oct. 2014, last accessed on 2021/01/08. [Online]. Available: https://tools.ietf.org/html/rfc7384

[6] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std., Jun. 2020.

[7] E. Shereen, F. Bitard, G. Dan, T. Sel, and S. Fries, "Next steps in security for time synchronization: Experiences from implementing IEEE 1588 v2.1," in *2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. IEEE, Sep. 2019.

[8] M. D. J. Teener and G. M. Garner, "Overview and timing performance of IEEE 802.1as," in *2008 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, sep 2008.

[9] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. A. Wojciak, and S. Guendert, "Impact of cyberattacks on precision time protocol," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 5, pp. 2172–2181, May 2020.

[10] D. Maftei, R. Bartos, B. Noseworthy, and T. Carlin, "Implementing proposed IEEE 1588 integrated security mechanism," in *2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. IEEE, Sep. 2018.

[11] S. Bradner, "Key words for use in rfcs to indicate requirement levels," Internet Requests for Comments, RFC Editor, BCP 14, Mar. 1997, last accessed on 2021/01/08. [Online]. Available: https://tools.ietf.org/html/rfc2119

[12] W. Alghamd and M. Schukat, "A detection model against precision time protocol attacks," in *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, Mar. 2020.

[13] R. Cochran, "linuxptp," version 3.1; last accessed on 2021/01/07. [Online]. Available: https://sourceforge.net/projects/linuxptp/

Network Coding — State of the Art

Florian Stamer, Jonas Andre*, Stephan Guenther*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: florian.stamer@tum.de, andre@in.tum.de, guenther@tum.de

***Abstract*—Network Coding (NC) [1] confers to intermediate nodes of a network the ability to combine packets via code. Instead of the traditional store-and-forward mechanisms of routing this new paradigm of store-code-forward mechanism has the potential to increase throughput, robustness against network loss, and security.**

In this paper we give an overview of recent advancements in network coding. We present an implementation of a Random Linear Network Coding (RLNC) data plane in P4 as introduced in [2]. Furthermore we focus on two optimization approaches for RLNC, with one being a novel Online Directed Acyclic Graph (DAG) algorithm [3] that tries to improve the decoding process and the other being an optimization to the encoding process of RLNC through the use of processor specific SIMD vector extensions [4]. By comparing the benefits of using the online DAG algorithm or SIMD vector extensions, we conclude the latter to be more practical since the online DAG algorithm is quite complex and only provides slightly better performance compared to already existing offline DAG algorithms.

***Index Terms*—Network Coding, Random Linear Network Coding, Directed Acyclic Graph, AVX, SIMD**

1. Introduction

In traditional networks packets are forwarded through store-and-forward mechanisms, but some networks can profit from combining packets to improve throughput. Ahlswede et al. [1] proposed the idea of combining packets via code and creating a store-code-forward mechanism called Network Coding (NC). This way inner nodes of a network can freely combine packets and provide the benefit of improved throughput [5], robustness against network loss [6], and security [7].

In Figure 1 we give an example of a butterfly network to illustrate how network coding can outperform traditional routing. The two source nodes (*server A/B*) transmit information *A* and *B*, respectively. Both must be received by the destination nodes (*PC 1/2*). With traditional routing only information *A* or *B* can be sent between the two switches at a time, thus both destination nodes do not receive all information at the same time. With network coding the information can be combined by a simple operation (*XOR*) and reconstructed at the destination nodes, in this case with antoher *XOR* operation.

The purpose of this paper is to give an overview of the current state of network coding and advances that have been made. We structure the paper as follows:

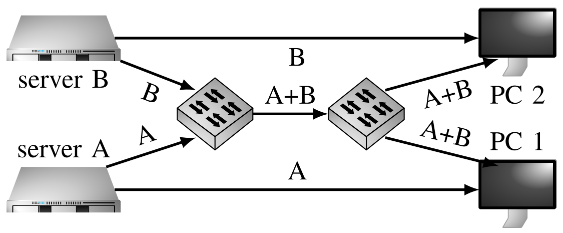


Figure 1: Butterfly Network

We first give a short summary of the network coding basics. We proceed to take a look at a P4 implementation of a Random Linear Network Coding (RLNC) data plane in Section 2. In Section 3 the focus lies on optimizing the RLNC decoding process through the usage of an Online Directed Acyclic Graph (DAG) algorithm. Another potential optimization to the performance of RLNC are Single Instruction Multiple Data (SIMD) vector extensions of certain processors. We take a deeper look at this idea in Section 4. We provide a comparison of the online DAG and SIMD vector extension approaches in Section 5 and give our conclusion as well as thoughts on future work in Section 6.

Basics of Network Coding. Network Coding introduces the ability to combine packets via code. In the case of Linear Network Coding (LNC) an encoded packet is created by linear combinations of N packets. In general the group of packets is referred to as a *generation* and the number of packets N is called the *generation size*. These linear combinations can be expressed as a matrix-vector multiplication over a given finite extension field. A finite field is a Galois field of the form $GF(2^n)$. For (random) linear network coding the finite extension fields $GF(2^1)$, $GF(2^2)$, $GF(2^4)$ and $GF(2^8)$ are of particular interest as mentioned in [4]. For convenience of notation we use F_q with $q = 2^n$ instead of $GF(2^n)$.

The following mathematical description of the encoding process is based on [4]. A packet with M symbols, each of size n, can be written as vector $\boldsymbol{a} = [a_1, a_2, \ldots, a_M]^T$ with the symbols $a_i \in F_q$. This gives for a generation of N packets the matrix

$$\boldsymbol{A} = [\boldsymbol{a}_1 \ldots \boldsymbol{a}_N] = \begin{bmatrix} a_{11} & \ldots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \ldots & a_{MN} \end{bmatrix} \in F_q^{M \times N}. \quad (1)$$

An encoded packet $\boldsymbol{b}$ is generated by multiplication with an encoding vector $\boldsymbol{c} \in F_q^N$.

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{c} = \sum_{i=1}^{N} c_i \boldsymbol{a}_i. \tag{2}$$

The components of $\boldsymbol{c}$ are chosen from the finite extension field F_q, either independently and identically distributed for RLNC or otherwise by some deterministic algorithm. In the given extension fields the addition operation is always a bit-wise XOR operation while the multiplication is a more complex modulo operation given a reduction polynomial specific to the field.

The decoding process is more complex and uses algorithms such as *Gauss-Jordan elemination* or *LU decomposition* on the recieved encoded packets.

2. An RLNC Implementation in P4

A recent implementation of a random linear network coding data plane in P4 has been proposed by Gonçalves et al. in [2]. By using RLNC, the network needs to be able to transmit additional information such as the encoding vectors. Thus a new packet format is designed to cope with this new way of packet processing. Since they use RLNC for their approach, the coefficients are chosen independently and uniformly at random from the finite field F_q. This provides the bonus of decentralizing code generation computation, but has the drawback that enough linearly independent *coded packets* are needed to decode the original message. We elaborate more on the packet format, the P4 program, and different methods of finite field multiplication in the following subsections.

2.1. Packet Format

The packet format of the generation-based RLNC protocol uses an inner and an outer header. Both are carried over Ethernet frames. The inner header contains the symbols and the coding vectors, if present, and carries information about the packet length as well as the type of packet. The types of packets are either *coded* or *uncoded*. The outer header holds information about meta parameters, such as the generation id, generation, finite field and symbol size.

2.2. RLNC P4 Program

The P4 PISA-like switch architecture buffers packets of different generations, which are limited in number per buffer. A generation stays buffered until enough packets of this generation are collected and the coding process starts. New linear combinations of the packets are transmitted until receiving an acknowledgment. Afterwards the buffer is flushed and starts accumulating packets of a new generation. The outer header of the newly coded packets stays the same while the inner header, i.e., the symbols and coding vectors, are readjusted.

2.3. Finite Field Multiplication

For the finite field arithmetic module the authors have featured two multiplication techniques. One is a compute-intensive method, based on simple shift and add operations, that results in an iterative algorithm which operates bit by bit. The other algorithm is based on pre-computed lookup tables, containing values for the log and $antilog$ of the elements in the finite extension field F_q.

3. Parallelization of the Decoding Process

While random linear network coding improves throughput, robustness against network loss, and security, it suffers from decoding delay since enough linearly independent packets have to arrive at the node before the decoding process can commence. This can be improved by using a *progressive* RLNC decoder that can partially decode a generation before all packets arrive. Based on progressive RLNC decoding Wunderlich et al. [3] proposes a novel strategy using directed acyclic graph scheduling. By arranging matrix block operations in a DAG manner, multiple operations are worked on in parallel by different threads. The novelty of this approach lies in it being an *Online* DAG algorithm, thus constructing the graph on the fly, instead of pre-computing, and making optimal use of progressive RLNC decoding. In the following subsections we give an explanation about the difference between non-progressive and progressive RLNC decoders, how the matrix block operations are defined and how the online DAG scheduling works, based on the information provided in [3].

3.1. Non-Progressive vs Progressive RLNC Decoder

There exist two categories of RLNC decoders, the non-progressive and progressive decoders. The classic non-progressive decoder expects all information to be present before starting the decoding process. An example is the generation-based RLNC approach as presented in [2], which we elaborate on in Section 2. Such a decoder can make use of matrix inversion algorithms other than Gauss-Jordan elimination, such as LU inversion.

A progressive RLNC decoder on the other hand can partially decode the data that has already been gathered and does not need to wait for the arrival of all data. While new encoded packets and coding vectors can be fed into the decoder as they are received. When considering conventional full-vector RLNC code, the decoded packets can only be released after the last packet of the generation is decoded. An optimization for such a progressive RLNC decoder is the use of low-delay codes, like sliding window codes [8, 9] or systematic generation based codes [10]. This way the decoder can already release any fully decoded information to upper levels without receiving all coded packets of a generation.

A hybrid scheme aims to combine the strengths of both non-progressive and progressive RLNC decoders. By performing sub-generation, more than one encoded packet, but less than the normal generation size, based progressive RLNC decoding [11].

3.2. Matrix Block Operations

A given matrix, of coding coefficients or encoded packets, gets split into blocks of size $b \times b$, where $b \leq N \wedge N/b \in \mathbb{N}$ and $16 \leq N \leq 1024$ is the generation size, e.g. a matrix of dimension 16×16 is split into four blocks of size 4×4. Each block gets separately processed by the three phases of the Gauss Jordan elimination with the use of helper matrices, hence the name *Matrix Block Operations*.

Given the number of symbols per packet M, the generation size N and the finite extension field F_q, let $\boldsymbol{C} \in F_q^{N \times N}$ be the coding coefficient matrix and $\boldsymbol{D} \in F_q^{N \times M}$ be the data matrix. Both are initially padded with zeros. $\boldsymbol{C}' \in F_q^{b \times N}$ and $\boldsymbol{D}' \in F_q^{b \times M}$ describe the encoded coding coefficient matrix and data matrix respectively, for the sub-generation of size b. In the following the 'row of a block *X*' is used to reference every block left and right of *X* spanning over the same rows and respectively 'column' for the blocks above and below of *X*. For a more in depth explanation and helpful figures we refer the reader to [3].

Forward Elimination. When a new sub-generation of encoded packets arrives, i.e. $\boldsymbol{C}'$ and $\boldsymbol{D}'$, the blocks in $\boldsymbol{C}$ containing the pivots on the diagonal are used to fill the corresponding blocks in $\boldsymbol{C}'$ with zeros. The row operations for a block are recorded in a helper matrix and applied to the other blocks in $\boldsymbol{C}'$ and to the blocks of $\boldsymbol{D}'$. Gaussian elimination is used on the first block in $\boldsymbol{C}'$ containing non zero values. The row operations are once more recorded in a helper matrix and applied to the rest of $\boldsymbol{C}'$ and $\boldsymbol{D}'$.

Backward Substitution. Continuing in this phase $\boldsymbol{C}'$ contains blocks of zeros followed by a block with the pivots on the diagonal. This block is used to fill the corresponding block *X* in $\boldsymbol{C}$ with zeros. The row operations are once again recorded in a helper matrix and applied to the row of *X* in $\boldsymbol{C}$ and $\boldsymbol{D}$. This process is repeated for every non zero block in the column of X.

Row Swapping. After the backward substitution concludes, both $\boldsymbol{C}'$ and $\boldsymbol{D}'$ are moved to the corresponding row in $\boldsymbol{C}$ and $\boldsymbol{D}$. If $\boldsymbol{C}$ is still missing pivots on the diagonal the algorithm starts once more after collecting enough new packets for a sub-generation. This is repeated until $\boldsymbol{C}$ is an identity matrix.

3.3. Online DAG Algorithm

New block operations are added on the fly to the Directed Acyclic Graph, instead of collecting all operations and constructing the entire DAG a priori (offline). This way the algorithm can take advantage of the properties of a progressive RLNC decoder. The iterative RLNC program is executed by a main thread. Block operations are added to the online DAG as new *task descriptions*, each summarizing the read and write dependencies regarding the other task descriptions. The main thread can then delegate tasks to a later time or another worker thread. The worker threads check independently for task descriptions in the DAG which have all of their dependencies resolved, pick and execute them.

Task Descriptions. These are objects in the DAG that hold information about the type of operation they represent, pointers to memory where matrices are stored, parameters describing said matrices (e.g. size) and more. Every object also has an access queue that keeps track of the sub tasks, that need to be concluded beforehand.

4. Encoding Process Optimization Using SIMD Vector Extensions

Contrary to optimizing the decoding process of (random) linear network coding, Günther et al. [4] try to improve the encoding process. In particular they implement and evaluate algorithms for finite field multiplication using processor specific vector instructions. For this study they implement two algorithms using the new family of instruction sets AVX512 in their finite field library *libmoepgf* [12]. AVX512 is a family of extensions and the two subsets AVX512-F (foundation) and AVX512-BW (byte and word) are the focus for the presented algorithms. While the byte-wise operations of AVX512-BW set of instructions are only supported by Intels Skylake-X and Ice Lake processors as of the time [4] was written, the AVX512-F extension will be supported by any processor that supports AVX512.

As we present in Section 1, the encoding process (2) expects the vectors $\boldsymbol{a}_i$ to be multiplied by the constant values c_i and finally accumulated into $\boldsymbol{b}$, this is commonly know as *multiply and add (madd)*. One such algorithm using vector instructions has been proposed by Plank et al. [13] and is called *shuffle* algorithm. This algorithm requires a shuffle instruction to swap words in vector registers. Hence the byte-wise operations of the AVX512-BW instruction set are necessary. Another algorithm introduced by Günther et al. in [12] is called *imul*. Contrary to the shuffle algorithm, it does not need any special instructions, but its complexity linearly depends on the word size. For the implementation this algorithm relies on the AVX512-F instructions set. Both algorithms are implemented for the finite expansion fields F_2, F_4, F_{16} and F_{256}.

Shuffle Algorithm . This algorithm expects the accumulator array $\boldsymbol{b}$, the source packet vector $\boldsymbol{a}_i$ and the coefficient c_i to calculate $\boldsymbol{b} := \boldsymbol{b} + \boldsymbol{a}_i \cdot c_i$. The shuffle algorithm needs certain constants, such as lookup tables and bit masks, different for each finite field F_q. Those are preloaded into the register variables. After handling the trivial cases for $c_i \in \{0, 1\}$, either no operation or a simple XOR, additional temporary registers are preloaded and the necessary madd operations using the shuffle instruction are performed.

Imul Algorithm. Similar to the previous algorithm, the imul algorithm expects the accumulator array $\boldsymbol{b}$, the source packet vector $\boldsymbol{a}_i$ and coefficient c_i. This algorithm also preloads lookup tables and after caching the trivial cases, sets up the temporary registers. One holds bit masks to isolate the coefficients of $\boldsymbol{a}_i$ and the other the powers of the constant c_i. These again are different depending on the finite field F_q. Afterward polynomial multiplication is performed inside a loop.

5. Evaluation and Comparison

In this Section we give a compact overview of the results and evaluations of both optimization approaches and put the results into perspective. It is important to mention, that the presented algorithms (online DAG vs shuffle/imul) try to optimize different processes (decoding vs encoding) and have been tested on different hardware.

Online DAG. The algorithm has been tested on both the ODROID-XU-3 and ODROID-XU+E, each equipped with four Cortex-A15 (big) cores and four Cortex-A7 (LITTLE) cores. The Cortex-A15 are clocked at 2.0GHz in XU-3 and 1.6GHz in XU+E, while the Cortex-A7 are clocked at 1.4GHz and 1.2GHz respectively [3].

While keeping the finite field (F_{256}) the same throughout the tests, a multitude of combinations for different values of the other parameters such as generation size, symbol size, number of threads and more are examined. This way optimal parameterization for high throughput and low delay are collected. For the benchmark the online DAG algorithm is compared to its offline DAG counter part and a state-of-the-art progressive *coefficient matrix duplication* (CD) approach [14]. The online DAG approach performs in general similar to the conventional offline method, while resulting in slightly better performance for smaller generations sizes. This is expected to be the result of the computational complexity, that increases with growing generation size. This approach performs also better then the CD approach for small symbol size, while falling short when the symbol size is especially large.

AVX512 Instruction Set Extensions. The shuffle and imul algorithms are tested on a multitude of processors, but in this overview we only mention the ones that support AVX512. These are the Intel Xeon Gold 6130 (clocked at 3.7GHz), Silver 4116 (clocked at 3.0GHz) and D-2166NT (clocked at 3.0GHz) [4].

For the tests only a single core is used and the generation size is kept at 16, while different finite fields ($F_2, F_4, F_{16}, F_{256}$) get analyzed. The AVX512 implementations perform in general better then the AVX2 implementations, but when the packet size reaches the size of the L2 cache the throughput drops regardless of extension used. Even for the commonly used finite field F_{256} an average of roughly $30Gbit/s$ of throughput can be achieved.

Encoding vs Decoding Optimization. We compare and evaluate both optimizations based on their performance gain compared to existing methods, as well as their implementation complexity. The online DAG algorithm is a complex approach, needing support for matrix block operations and the DAG scheduling with custom task descriptions. It provides only slight performance improvements compared to the common offline DAG algorithms, with throughput measured in MiB/s. Contrary the AVX512 based shuffle and imul algorithm already have library implementations and perform better compared to the older AVX2 extension, with throughput measured in $Gbit/s$. The difference in the units of measurement are most likely linked to the higher computational complexity of the decoding process or the different hardware used. To improve the performance of an RLNC implementation, the optimization of the encoding process through the use of vector extensions should be prioritized.

6. Conclusion and future work

We provide an overview of current advances in network coding, with the focus on two approaches to improve and optimize random linear network coding. One uses a progressive RLNC online directed acyclic graph based algorithm to parallelize the decoding process. The other provides the shuffle and imul algorithm, which make use of AVX512 vector extensions to speed up the finite field multiplication of the encoding process. In Section 5 we provide a summary of the evaluations of both approaches and came to the conclusion, that the use of processor specific vector extensions yield better results with less complex algorithms. Thus the optimization of the encoding process is more appealing.

Different implementations of network coding protocols or other areas of network coding, such as network security, can be of interest and be the focus of future works.

References

[1] R. Ahlswede, N. Cai, S. . R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[2] D. Goncalves, S. Signorello, F. M. V. Ramos, and M. Medard, "Random linear network coding on programmable switches," in *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2019*, 2019.

[3] S. Wunderlich, F. H. P. Fitzek, and M. Reisslein, "Progressive Multicore RLNC Decoding with Online DAG Scheduling," *IEEE Access*, vol. 7, pp. 161 184–161 200, 2019.

[4] S. M. Günther, N. Appel, and G. Carle, "Galois Field Arithmetics for Linear Network Coding using AVX512 instruction set extensions," 2019.

[5] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497–510, 2008.

[6] D. S. Lun, M. Médard, R. Koetter, and M. Effros, "On coding for reliable communication over packet networks," *Physical Communication*, vol. 1, no. 1, pp. 3–20, 2008.

[7] L. Lima, M. Médard, and J. Barros, "Random linear network coding: A free cipher?" in *IEEE International Symposium on Information Theory - Proceedings*, 2007, pp. 546–550.

[8] S. Wunderlich, F. Gabriel, S. Pandi, F. H. P. Fitzek, and M. Reisslein, "Caterpillar RLNC (CRLNC): A Practical Finite Sliding Window RLNC Approach," *IEEE Access*, vol. 5, pp. 20 183–20 197, 2017.

[9] F. Gabriel, S. Wunderlich, S. Pandi, F. H. P. Fitzek, and M. Reisslein, "Caterpillar RLNC With Feedback (CRLNC-FB): Reducing Delay in Selective Repeat ARQ Through Coding," *IEEE Access*, vol. 6, pp. 44 787–44 802, 2018.

[10] D. E. Lucani, M. Médard, and M. Stojanovic, "Systematic network coding for time-division duplexing," pp. 2403–2407, 2010.

[11] M. Kim, K. Park, and W. W. Ro, "Benefits of using parallelized non-progressive network coding," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 293–305, 2013.

[12] S. M. Günther, M. Riemensberger, and W. Utschick, "Efficient GF arithmetic for linear network coding using hardware SIMD extensions," in *2014 International Symposium on Network Coding (NetCod)*, 2014.

[13] J. S. Plank, K. M. Greenan, and E. L. Miller, "Screaming fast galois field arithmetic using intel simd extensions," *USENIX Conference on File and Storage Technologies*, vol. 11, 2013.

[14] H. Shin and J.-S. Park, "Optimizing random network coding for multimedia content distribution over smartphones," *Multimedia Tools and Applications*, vol. 76, 10 2017.

White Rabbit: High Precision PTP

Edward Waterman, Max Helm*, Johannes Zirngibl*, Henning Stubbe*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: edward.waterman@tum.de, helm@net.in.tum.de, zirngibl@net.in.tum.de, stubbe@net.in.tum.de

***Abstract*—White Rabbit is a time synchronization technology based on the Precision Time Protocol. It is used to synchronize clocks between different entities on an Ethernet network. Promising sub-nanosecond accuracy it is well suited for time and latency sensitive distributed applications. This paper gives an overview of the functionality, performance and application domains of White Rabbit.**

***Index Terms*—White Rabbit, Precise Time Protocol, Synchronous Ethernet, Time Synchronization**

1. Introduction

In 2008 development begun at CERN to replace its old timing infrastructure. The result of this work was White Rabbit (*WR*) – inspired by the habitually late rabbit of Alice in Wonderland. A requirement for the new implementation was compatibility with existing infrastructure. As a result an Ethernet-based application was chosen, adding enhancements to the Precision Time Protocol (*PTP*). PTP is a sub-microsecond accuracy time synchronization protocol for network devices, using a master-slave architecture [1]. As an improvement White Rabbit synchronizes the master and slave's clock frequencies using Synchronous Ethernet (*Sync-E*). This reduces the problem of determining latencies in the synchronization procedure to one of detecting phase offsets, enabling sub-nanosecond accuracy [2]. In 2020, after 12 years of development White Rabbit was included into the latest PTP release as High-Accuracy profile [3].

White Rabbit is used at CERN and other scientific institutions, helping to synchronize telescope arrays and distributed measurement units. It has also gained traction in the financial sector where time synchronization is important to manage stock transactions.

In this paper we will introduce the building blocks of White Rabbit: PTP (Section 2.1) and Sync-E (Section 2.2). We will continue with an overview of the components and topology of WR (Section 3.1), its time synchronization procedure (Section 3.2), applications (Section 3.4) and performance (Section 3.5). We provide further reading material in Section 4 and end with a short summary in Section 5.

2. Background

White Rabbit is mainly based on on two technologies: The (1) Precision Time Protocol which, as the name suggests, attends to precise time synchronization. And (2) Synchronous Ethernet which enables synchronization on the physical layer.

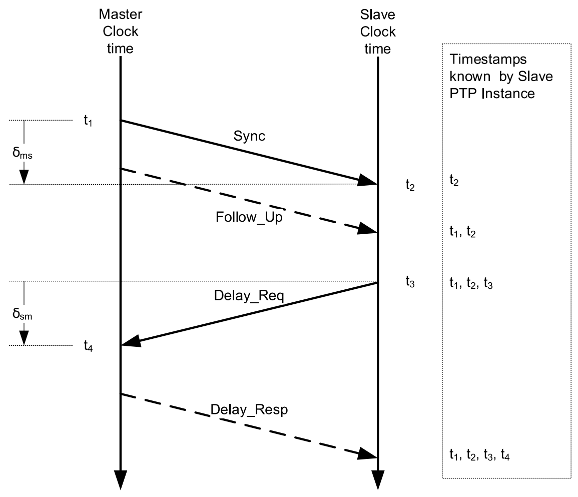


Figure 1: PTP synchronization sequence diagram [4]

2.1. Precision Time Protocol

The Precision Time Protocol is defined in its latest version v2.1 by IEEE Standard 1588-2019 [4]. It is used to synchronize clocks in networks with sub-microsecond accuracy. PTP makes use of a master-slave architecture. Its benefits are that the protocol supports heterogeneous clocks, has low latency and minimal resource usage. The protocol can be enhanced with profiles to meet use case specific requirements. One of these profiles is White Rabbit [4, Annex M] which enables PTP to synchronize clocks with sub-nanosecond precision.

The PTP procedure is shown in Figure 1. A PTP master node sends a `Sync` message to one of its slave node, signaling it to listen for a `Follow_Up` message. In the `Follow_Up` the master includes its egress timestamp t_1 of the `Sync` message. To account for communication latency δ_{ms} the slave sends a `Delay_Req` request to the master which returns a `Delay_Resp`, containing the timestamp t_4 of the message reception. With timestamps t_1, t_2, t_3, t_4 known by the slave, it can estimate the roundtrip time $\delta_{mm} := \delta_{ms} + \delta_{sm}$, the sum of message delays from master to slave and vice versa, see Equation (1). Under the assumption that communication delay is symmetrical, then one-way delay $\delta = \delta_{ms} = \delta_{sm}$ is half of the roundtrip time, which we can estimate using Equation (2).

$$\hat{\delta}_{mm} = (t_2 - t_1) + (t_4 - t_3) \tag{1}$$

$$\hat{\delta} = \frac{\hat{\delta}_{mm}}{2} \tag{2}$$

The slave updates its local time t with the estimated clock offset $\hat{o}_{ms}$, see Equations (3) and (4).

$$\hat{o}_{ms} := \left(t_2 - t_1 + \hat{\delta}\right) \tag{3}$$

$$t := t - \hat{o}_{ms} \tag{4}$$

PTP provides accuracy in the sub-microsecond range [1]. Problems in accuracy stem from the assumption of symmetrical delay, which when violated invalidates the one-way delay computation seen Equation (2). Another source for low synchronization accuracy are imprecise timestamps, where errors propagate into the roundtrip delay, given in Equation (1).

2.2. Synchronous Ethernet

Synchronous Ethernet is a standard defined by the ITU-T (International Telecommunication Union - Telecommunication Standardization Sector) [5]. Sync-E enables clock frequency synchronization – also called syntonization – between a master and a slave node. In standard Ethernet, clock oscillators are free running, introducing clock drift and diminishing the ability to synchronize clocks accurately. Synchronous Ethernet operates on the physical layer with little overhead.

Sync-E syntonization functions as following [5]: the grandmaster node is connected to a precise reference clock, similar to Figure 3. The reference clock input is passed to a central timing card on the network interface which calibrates and handles the input accordingly. With the reference clock signal the master synchronizes its physical layer line code frequencies. A slave can recover the reference clock by extracting the frequency via a clock data recovery unit from the line codes. This is possible because the medium in Ethernet is never idle [2], thus line codes are sent with a constant frequency.

3. White Rabbit

Building upon PTP and Synchronous Ethernet, a White Rabbit network enables sub-nanosecond accuracy time synchronization.

WR uses special hardware to provide its high accuracy. Fortunately, it is an open hardware initiative, firmware and hardware designs are open sourced and freely available [6]. Several commercial implementations for WR switches and nodes are available.

In the following sections we will focus on WR as presented in its specification [7]. The specification focuses on Gigabit Ethernet over fiber, thus we will too.

3.1. Topology

Like PTP and Sync-E, White Rabbit networks use a master-slave architecture. The main component of White Rabbit is the WR switch which functions as time synchronization source and sink. A White Rabbit node is only a synchronization sink [8].

One or more WR switches can be connected to a reference clock or GPS, with one grandmaster and possibly several backup grandmasters [9]. WR nodes and switches connected to a downlink port are slaves to the

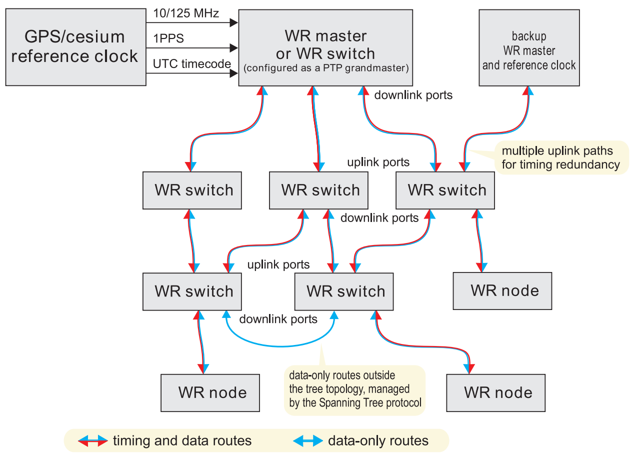

GPS/cesium reference clock
10/125 MHz
1PPS
UTC timecode
WR master or WR switch (configured as a PTP grandmaster)
backup WR master and reference clock
downlink ports
multiple uplink paths for timing redundancy
uplink ports
WR switch
WR switch
WR switch
downlink ports
uplink ports
WR switch
WR switch
WR node
downlink ports
WR node
data-only routes outside the tree topology, managed by the Spanning Tree protocol
WR node
timing and data routes
data-only routes

Figure 2: Exemplary White Rabbit network topology [8]

switch. White Rabbit networks must have a tree topology in order for time synchronization to function properly. However, additional connections can be established to ensure redundancy. This layout is exemplarily displayed in Figure 2.

In general, White Rabbit functions transparently, working alongside non-compatible hardware. During link detection, a White Rabbit master determines if a node is compatible by sending a `ANNOUNCE` message. If a node responds with a `SLAVE_PRESENT` message, White Rabbit synchronization is enabled.

3.2. Time Synchronization

The main source of errors in PTP are the inaccuracies in time-stamping and delay measurement. Using syntonization both time-stamping and delay measurement can be reduced to a problem of phase detection. In Figure 3 a connection between a master and a slave node is shown. The master and slave clocks are running with the same frequency. Additionally, the slave node adds an estimate $phase_s$ offset to its network clock to compensate for the phase introduced between master and slave. This corrected frequency is used to transmit data to the master and for clock correction. From the loopback from master to slave and back, the master can measure a roundtrip phase offset $phase_{mm}$ [7, Sec. B.1].

White Rabbit's initial time synchronization procedure is as follows [7, Sec. B.1]:

1) Syntonization.
2) Calibration.
3) Roundtrip delay.
4) Phase measurement.
5) Fine delay.
6) Determine link asymmetry.
7) One-way delay computation.

Syntonization. Using Sync-E, clock frequencies of the master and the slave are synchronized and locked [7, Sec. 5.1].

Calibration. Constant delays, shown in Figure 3 as $\Delta_{\{txm,txs,rxm,rxs\}}$, are measured. Depending on requirements, different calibration techniques can be employed:

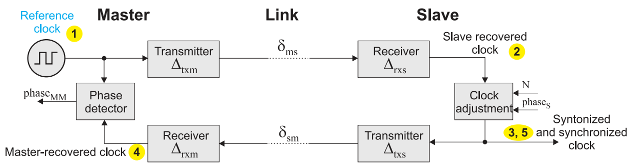


Figure 3: WR link connection [7, Fig. 11]

First, factory calibration and measurements can be used to compensate constant delays by physical latencies. Secondly, compensate active interference sources, e.g. temperature with a model of the interference's influence on the delay. Lastly, ensure similar operation conditions and setup between master and slave to minimize the delay asymmetry between nodes [7, Sec. B.6.1].

Roundtrip delay. The slave obtains $t_1, t_2, t_3, t_4, \hat{\delta}_{mm}, \hat{\delta}$ using standard PTP. To minimize timestamp errors due to clock jitter, timestamps are obtained via specialized hardware. This enables timestamp accuracy of one clock cycle [7, Sec. B.5]. Additionally, rising and falling edges of the timestamp trigger event are captured, for further precision.

Phase measurement. The roundtrip phase offset from master to slave and back is extracted on the master node [7, Sec. B.1]. Shown as $phase_{mm}$ in Figure 3.

Fine delay. PTP roundtrip time $\hat{\delta}_{mm}$ is refined using the established phase difference $phase_{mm}$.

The timestamps t_2 and t_4 are further calibrated, due to the fact that these incoming timestamps come from a different frequency domain. Either the rising or falling edge timestamp of the timestamp trigger event is used as new timestamp basis. This rising or falling edge timestamp is then adjusted using a simple algorithm to take phase offset into account [7, Sec. B.5]. For t_2 and respectively t_4, $phase_s$ / $phase_{mm}$ is incorporated to obtain t_{2p} / t_{4p}. Using these new precise timestamps t_{2p} and t_{4p} the new roundtrip delay is computed with the standard PTP roundtrip delay formula [7, Eq. (20)]:

$$\hat{\delta}_{mm} := (t_{2p} - t_1) - (t_{4p} - t_3) \tag{5}$$

Determine link asymmetry. White Rabbit loosens the assumptions of standard PTP that $\delta_{ms} = \delta_{sm}$. To transmit and receive data different wavelengths, with different refractive indexes, are used. This results in different propagation delays between the master-slave connection and the slave-master connection in the link medium. White Rabbit uses a delay asymmetry coefficient α defined as [7, Eq. (23)]:

$$\alpha := \frac{\delta_{ms}}{\delta_{sm}} - 1 = \frac{n_{1550}}{n_{1310}} - 1 \tag{6}$$

Here, n_{1550} and n_{1310} would be the refractive indexes of the transmitting wavelength of 1550 nm and receiving wavelength of 1310 nm over fiber. The delay asymmetry coefficient can be expressed in relation to δ_{mm} and the master and slave clock offset o_{ms} [7, Eqs. (8, 24, 25)]:

$$\Delta := \Delta_{txm} + \Delta_{txs} + \Delta_{rxm} + \Delta_{rxs} \tag{7}$$

$$\delta_{mm} := \Delta + \delta_{ms} + \delta_{sm} \tag{8}$$

$$o_{ms} := \frac{\delta_{ms} - \delta_{sm}}{2} \tag{9}$$

$$\alpha = \frac{\delta_{mm} - \Delta + 2 * o_{ms}}{\delta_{mm} - \Delta - 2 * o_{ms}} \tag{10}$$

One-way delay computation. With the dependency of α on δ_{mm}, as seen in Equation (10), one-way delay and offset can be computed [7, Eqs. (30,31)]:

$$\hat{\delta}_{ms} := \frac{1+\alpha}{2+\alpha}(\hat{\delta}_{mm} - \Delta) + \Delta_{txm} + \Delta_{rxs} \tag{11}$$

$$\hat{o}_{ms} := t_1 - t_{2p} - \hat{\delta}_{ms} \tag{12}$$

Multiple offset correction terms are computed [7, Eqs. (32–34)]:

$$corr_{UTC} = \left\lfloor \frac{\hat{o}_{ms}}{1\,\mathrm{s}} \right\rfloor \tag{13}$$

$$corr_{CNT} = \left\lfloor \frac{\hat{o}_{ms} - corr_{UTC}}{T_{ref}} \right\rfloor \tag{14}$$

$$corr_{PHASE} = \hat{o}_{ms} - \lfloor \hat{o}_{ms} \rceil \tag{15}$$

$corr_{UTC}$ for the UTC time, $corr_{CNT}$ for the clock counter, $corr_{PHASE}$ for the estimated offset $phase_s$, T_{ref} is the duration of one reference clock cycle (8 ns for Gigabit Ethernet over fiber).

The corrective terms are used for the following:

1) The slave updates its UTC time counter to: $t_{UTC} := t_{UTC} + corr_{UTC}$.
2) Updates its reference clock counter: $t_{CLOCK} := t_{CLOCK} + corr_{CNT}$.
3) And its phase offset: $phase_s := phase_s + corr_{PHASE}$.

In subsequent synchronizations not all synchronization steps have to be performed. It is sufficient to recalculate phase differences.

3.3. PTP Extension

After twelve years of development [10] White Rabbit was added as High-Accuracy profile to the latest PTP version, IEEE Standard 1588-2019, in 2020 [4, Annex M]. For the integration the WR protocol was split into multiple features which are described separately in the standard [11]:

- How layer 1 syntonization shall be used within PTP [4, Annex L].
- Asymmetric delay estimation and correction for PTP [4, Clause 16.7f].
- Hardware calibration and delay asymmetry coefficient estimation [4, Annex N].
- Master-Slave assignment [4, Clause 8.2.15.5.2, 9.2.2.2 & 17.6].
- High-Accuracy delay request-response default PTP profile [4, Annex I.5].
- High-Accuracy profile for sub-nanosecond accuracy [4, Annex M].

Overall the PTP High-Accuracy profile is a generalization of White Rabbit offering more configuration options [4, Annex L].

3.4. Applications

Initially developed to replace CERN's old timing infrastructure, White Rabbit has gained traction in other scientific projects. While CERN is still one of the main users of White Rabbit [12], many other research organizations adapted the technology. Some examples are [12]: KM3NET, a deep-sea neutrino telescope, where undersea detection units use White Rabbit. LHAASO, an air shower detection unit, consisting of 10000 detectors which are synchronized by 583 WR switches.

Another domain where White Rabbit has been gaining traction is the financial sector. The Deutsche Börse Group has been using White Rabbit to synchronize their trading network [13], [14]. Stock trading is highly dependent on accurate timing, as the trade execution order is based on bid price and time [15]. Stock exchanges also started offering timing as a service [14], [16], offering market participants to gain access to high precision timestamps for order executions. This enables insight into the strategies of other market participants trading strategies.

3.5. Performance

The first real-world application of White Rabbit was at the CERN Neutrino to Gran Sasso *CNGS* project [17], where a first WR performance survey was conducted. The experimental setup consisted of two timing devices, one of which was connected via Gigabit Ethernet over fiber to a WR grandmaster switch and the reference clock. The other was connected to a second WR switch and the reference clock. Total distance from the second timing device and the grandmaster switch was 16 km. Timestamps were taken for 31 d and the influence of temperature fluctuations of approximately 3.5 °C where taken into account. The average time difference between the two nodes was 0.517 ns with a standard deviation of 0.119 ns. Temperature fluctuations introduced only a small long-term timing drift.

Lipinski et al. [3] gives an overview of the performance of multiple WR installations, accuracy ranges from 150 ps to 8 ns. White Rabbit performance depends on the communication medium, achieving its highest performance when used with Ethernet over fiber. Using Ethernet over copper cable enables an accuracy of around 30 ns [9]. Experiments [18] to use WR over wireless bands showed that sub-nanosecond accuracy could be preserved, with slightly worse precision. Comparing this with standard PTP, where studies [1], [19] show an accuracy ranging from 1 ns to 800 ns, we see a performance improvement of 1 to 2 orders of magnitude. However, these comparisons should be taken with a grain of salt as experimental setup differs between these studies.

4. Related work

White Rabbit is an open hardware collaboration from CERN aiming to make hardware design, software and specification freely available. More information on WR can be found at the open hardware repository [6] where presentations, papers, information about hardware vendors and users, etc. are freely available.

A recent paper by Lipinski et al. [3] gives an overview of possible performance enhancements, benchmarks and users of the technology. Furthermore, he lists possible advanced use cases, for example using White Rabbit for low-latency event trigger distribution, fixed-latency data transfer or radio-frequency transfer.

5. Conclusion

In an evermore distributed world where time distribution becomes increasingly critical even at the network edges, White Rabbit is a technology particularly suited for this task. Based on existing technologies it enables fast and low-latency time synchronization. Additionally, the inclusion into the official PTP standard, promises an increasingly large adoption. Its transparent functionality enables easy integration into already existing networks.

References

[1] D. M. E. Ingram, P. Schaub, D. A. Campbell, and R. R. Taylor, "Performance Analysis of PTP Components for IEC 61850 Process Bus Applications," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 4, pp. 710–719, Apr. 2013.

[2] M. Lipiński, T. Włostowski, J. Serrano, and P. Alvarez, "White rabbit: A PTP application for robust sub-nanosecond synchronization," in *Control and Communication 2011 IEEE International Symposium on Precision Clock Synchronization for Measurement*, Sep. 2011, pp. 25–30.

[3] M. Lipiński, E. van der Bij, J. Serrano, T. Włostowski, G. Daniluk, A. Wujek, M. Rizzi, and D. Lampridis, "White Rabbit Applications and Enhancements," in *2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, Sep. 2018, pp. 1–7.

[4] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-2008)*, pp. 1–499, Jun. 2020.

[5] J. Ferrant, M. Gilson, S. Jobert, M. Mayer, M. Ouellette, L. Montini, S. Rodrigues, and S. Ruffini, "Synchronous ethernet: A method to transport synchronization," *IEEE Communications Magazine*, vol. 46, no. 9, pp. 126–134, Sep. 2008.

[6] "White Rabbit: Homepage," https://ohwr.org/project/white-rabbit/wikis/home, accessed: 2020-12-27.

[7] E. G. Cota, M. Lipiński, T. Włostowski, E. van der Bij, and J. Serrano, "White Rabbit Specification: Draft for Comments," Jul. 2011.

[8] J. Serrano, M. Cattin, G. Daniluk, E. Gousiou, M. M. Lipinski, and M. Lipiński, "The White Rabbit Project," p. 7, 2013.

[9] T. Włostowski, "Precise time and frequency transfer in a White Rabbit network," Master Thesis, Warsaw University of Technology, Warsaw, May 2011.

[10] "White Rabbit: Status," https://ohwr.org/project/white-rabbit/wikis/Status, accessed: 2020-12-27.

[11] "White Rabbit: PTP Integration," https://ohwr.org/project/wr-std/wikis/wrin1588, accessed: 2021-01-05.

[12] "White Rabbit: User Information," https://ohwr.org/project/white-rabbit/wikis/WRUsers, accessed: 2020-12-27.

[13] "Deutsche Börse Group - Ultra-accurate time distribution for Deutsche Börse's trading network," https://deutsche-boerse.com/dbg-en/products-services/insights-innovation-new-technologies-en/Ultra-accurate-time-distribution-for-Deutsche-B-rse-s-trading-network-246490, accessed: 2021-01-03.

[14] J. Lopez-Jimenez, J. L. Gutierrez-Rivas, E. Marin-Lopez, M. Rodriguez-Alvarez, and J. Diaz, "Time as a Service Based on White Rabbit for Finance Applications," *IEEE Communications Magazine*, vol. 58, no. 4, pp. 60–66, Apr. 2020.

[15] A. Lohr, "White Rabbit in Financial Markets," CERN, Geneva, Switzerland, Oct. 2018.

[16] "Data Shop Deutsche Börse AG - Your road to historical data - Data Shop Historical Data," https://datashop.deutsche-boerse.com/high-precision-timestamps, accessed: 2020-12-27.

[17] M. Lipinski, T. Wlostowski, J. Serrano, P. Alvarez, J. D. G. Cobas, A. Rubini, and P. Moreira, "Performance results of the first White Rabbit installation for CNGS time transfer," in *2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings*, Sep. 2012, pp. 1–6.

[18] J. E. Gilligan, E. M. Konitzer, E. Siman-Tov, J. W. Zobel, and E. J. Adles, "White Rabbit Time and Frequency Transfer Over Wireless Millimeter-Wave Carriers," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 67, no. 9, pp. 1946–1952, Sep. 2020.

[19] H. Liu, J. Liu, T. Bi, J. Li, W. Yang, and D. Zhang, "Performance analysis of time synchronization precision of PTP in smart substations," in *2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, Oct. 2015, pp. 37–42.

Intra-vehicular Data Sources

Paul Wiessner, Filip Rezabek, Kilian Holzinger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: wiessner@in.tum.de, frezabek@net.in.tum.de, holzinger@net.in.tum.de

***Abstract*—This article gives an overview of currently used sensors in autonomous driving. A focus is set to LiDAR type of RADAR, a Livox Mid-40 sensor discussed as an example. The conclusion considers an overview about parameters which are essential to differ several LiDAR devices.**

***Index Terms*—automotive, automotive data sources, autonomous driving, camera, lidar, radar, ultra sonic, Livox, Livox Mid-40**

1. Introduction

Autonomous driving is an emerging topic which gained more and more importance in the last years. It will take time to achieve a level of full driving automation at which a driver is not obligated to intervene or furthermore not able to intervene. However, the goal is clear, and a software is an important factor to reach it. But the software can be as good as delivered data produced by sensors.
In the following article we will investigate main sensors deployed in autonomous driving cars. Our focus will be set especially on LiDAR and Livox Mid-40, as a detailed example. There we will present its SDK, how the device could be set up and types of data exchanged. Moreover, several parameters listed in the article should be considered as essentials for proper application of a LiDAR in autonomous driving cars.

2. Sensors in Autonomous Driving

In new cars sold nowadays there is a number of sensors that are shored ex factory. Starting with simple assistance systems as a 'parking assistant' or 'blind spot monitor' and advancing to more complex systems as a 'lane keeping assistant' until full autonomous driving - every single assistant requires certain sensors. The most commonly used sensors are cameras, RADAR, LiDAR and ultra sonic, which will described one by one in the following.

2.1. Camera

Cameras are one crucial sensor in the system of autonomous driving. They are reliable and relatively cheap to produce and to build in. With a help of Artificial Intelligence surrounding objects can be identified and classified, road signs can be recognised. However, cameras face limitations such as dependency on weather, light conditions and primarily of clean lenses.
Often described as 'eyes of the car' cameras are the most accurate way to create a visual representation of the surrounding world [1]. For this representation multiple cameras are needed in the front, rear, left and right sides of a car. A sample calculation showing the amount of data created by a camera system for autonomous driving is presented in the following. The sample is based on a system of an Israeli company which provides a fully autonomous driving system based exclusively on 12 cameras [2]. Assuming we take a fictional camera with similar specifications as the 'MPC3' from Bosch [3]. Therefor we get the following specifications being important for data generation:

- Resolution: 2.6 MP HDR (2048 x 1280 pixels)
- Frame rate: 45 frames per second

Assuming a video stream with these specifications, a supposed dynamic range of 24 bit and approximately 20% protocol overhead, we get the following calculation:

- Pixel of each image:
  $2048 \times 1280 = 2621440px$
- Size per image:
  $2621440 \times 24bit \approx 62.9Mbit$
- Amount of data per second:
  $62.9Mbit \times 30 \approx 1.89Gbps$
- Including 20% protocol overhead: $1.89Gbp \times 1.2 \approx 2.26bps$

Given the specifications above one camera consequently requires approximately $2.26Gbps$ of bandwidth without any compression. Transmission of this amount of data is not convenient and thus compression should be used. Applying a modern compression algorithm can reduce the size by a ratio of up to 1:200 (using e.g. the lossy MPEG-4).
Referring to our example of a autonomous driving system with 12 cameras on board, without compression it would result in $28Gb$ the system has to progress every second.

2.2. RADAR

RADAR is a technology of which security in autonomous driving greatly benefits. RADAR is an acronym for "RAdio Detection And Ranging". It is used in autonomous cars among other systems for obstacle detection and Adaptive Cruise Control. It works by emitting an electromagnetic wave. This wave is reflected when meeting an obstacle and bounces back to the origin where it is measured. Thereby, it is able to measure the distance to an object, the approximate size and its roughly speed.

Compared to the other sensors it is inexpensive, but it is less angularly accurate than LiDAR as it may lose the sight of the target in curves and it may get confused if multiple objects are placed very close to each other. However, unlike LiDAR, RADAR is weather-independent, it is able to detects objects behind other objects and to determine relative traffic speed or the velocity of moving objects [4].

2.3. LiDAR

LiDAR is another sensor being seen as an indispensable technology for autonomous driving in order to reach Level 5 autonomy (see Figure 1). The typically used term LiDAR is an abbreviation for "Light Detection And Ranging". LiDAR is a technology similar to RADAR and it is able to optically measure distances or speed. Instead of radio waves, that are used for RADAR (see section 2.2), LiDAR uses laser for scanning the environment. This works by sending out rapid laser signals sometimes reaching up to 150.000 pulses per second. When the laser beam meets an obstacle it is reflected and bounces back. Nearby the laser source reside sensors measuring the time that the laser need to go to the obstacle and bounce back. With this information a quite precise three-dimensional model of the surrounding environment is created.
With LiDAR it is possible to detect objects within a range from just a few centimetres to up to several hundreds of metres.
Assuming the Livox Mid-40 LiDAR sensor, which we get to know better in Section 3, we can roughly calculate the amount of data which is produced. The necessary specifications are as follows:

- point rate: $100000\frac{pints}{s}$
- point size: $72bit$ to $104bit$

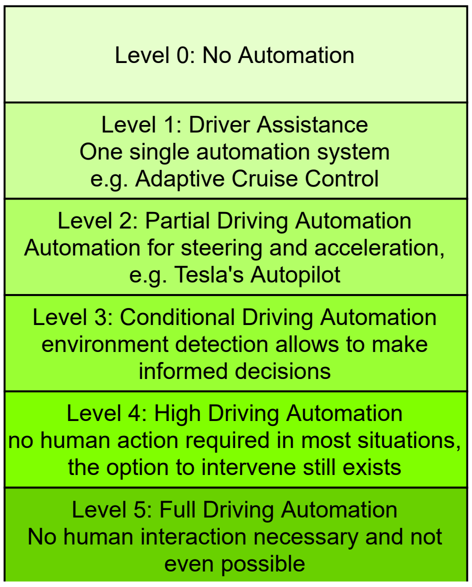
Further assuming a protocol overhead of about 20% we get the following:

- Minimum and maximum amount of data per second:
  $100000\frac{1}{s} \times 72bit = 7.2Mbits$
  $100000\frac{1}{s} \times 104bit = 10.4Mbits$
- Including 20% protocol overhead: $7.2Mbits \times 1.2 = 8.64Mbits$ $10.4Mbits \times 1.2 = 12.48Mbits$

Overall the LiDAR generates an output of $8.64Mbits$ to $12.48Mbits$ that is sent to the processing unit for further processing. Compared to the output of a camera, this amount is relatively small and compression is not necessarily needed for transmission.

2.4. Ultra Sonic

Ultra sonic sensors are mostly used in autonomous cars for systems like parking assistants and nearby obstacle detection [5]. Ultra sonic sensor sends out ultrasonic impulses that are reflected by nearby obstacles, the time between sending and receiving the reflection is measured. It is applicable within a range from a couple of centimetres to a couple of metres. Although the scope of ultra sonic sensor in autonomous driving is different from LiDAR they are comparable by ability to see objects through other


Level 0: No Automation

Level 1: Driver Assistance
One single automation system
e.g. Adaptive Cruise Control

Level 2: Partial Driving Automation
Automation for steering and acceleration,
e.g. Tesla's Autopilot

Level 3: Conditional Driving Automation
environment detection allows to make
informed decisions

Level 4: High Driving Automation
no human action required in most situations,
the option to intervene still exists

Level 5: Full Driving Automation
No human interaction necessary and not
even possible

Figure 1: Levels of Autonomous Driving

objects, being weather and day time independent, being relatively cheap and being equal in resolution. However, disadvantages should be considered as small objects as well as multiple fast moving objects cannot be detected and the field of view is more limited.

2.5. Other Sensors

There is a number of other sensors available and actively used in cars. Most of them are used for motor control to guarantee smooth operation. These sensors include e.g. mass air flow sensor, engine speed sensor, oxygen sensor etc.. Others are assigned to lower levels of autonomous driving such as wheel speed sensors, steering sensors, break sensors etc..

3. The Livox Mid-40

LiDAR is often said to be an indispensable technology to reach higher levels of autonomous driving. Nevertheless, there are hot tempered discussions whether to use LiDAR or rely other technologies such as RADAR or cameras, others see enormous potential in LiDAR. A broad range and high accuracy of depth perception, suitability for 3D mapping and a number of fields with huge and unrevealed potential.
However, in this section we go a bit more into details and examine the 'Livox Mid-40' - a LiDAR sensor from Livox.

3.1. Requirements

To begin with the LiDAR, a couple of requirements has to be complied with. Before all technical requirements, it has to be mounted properly, meaning there has to be enough space between the device and surrounding object to guarantee proper functioning of the fan. For technical requirements a few aspects have to be fulfilled. Firstly, the system (e.g. in an autonomous car) has to provide electrical tension between 10-16V direct current. Secondly, the

LiDAR requires an average power of 10W and at peaks in situations under extreme conditions (-20°C and during startup) up to 40W. Regarding the processing unit, the LiDAR provides a 100BASE-TX cable capable to transfer 100Mbit/s which has to be handled by the system.

3.2. Livox SDK

The Livox SDK is the Software Development Kit for the Livox Mid-40 and all Livox products applied to quickly connect to Livox sensors and receive data. It consists of Livox SDK communication protocol, Livox SDK core and Livox SDK API [6]. In the following, each of these points will be detailed.

3.2.1. Livox SDK Core. The structure of the Livox SDK Core can be described as in Figure 2 (based on [6]).

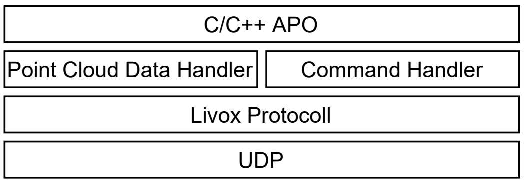


Figure 2: Livox SDK Core

As communication protocol between LiDAR sensor and Livox SDK the User Datagram Protocol (UDP) is applied. These serves being a basis for the Livox SDK Communication Protocol (see Section 3.2.1 which enables communication between a sensor and a user. The 'Point Cloud Data Handler' and 'Command Handler' support transmission of the correspondent data types defined in the Livox SDK Communication Protocol. And on top of that a C/C++ API provides convenient integration of C style functions into custom C/C++ programs.

3.2.2. Livox SDK Communication Protocol. The Livox SDK Communication Protocol enables communication between user programs and the LiDAR sensor. With its help, the user is able to set the LiDAR sensor in mainly three states. These cover 'normal', 'standby' and 'power-saving'. When mode switching is completed and the device is in LiDAR mode, there are additionally two states 'initializing' and 'error' (see Figure 3) [6].

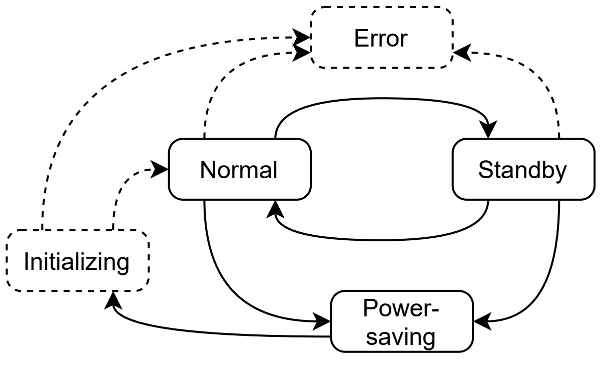


Figure 3: Operating States

It basically consists of two different packet types describing the different kinds of communication taking place between a user and a sensor. The two types are 'Control Command Data' and 'Sample Data'. The 'Control Command Data' type is used for the organizational part covering configuration and query of LiDAR parameters and status information [7]. The other part is covered by the 'Sample Data' type which transfers all kind of data generated by the LiDAR, e.g. point cloud data, time stamps or IMU data (not for Livox MID-40/70/100).
The model of communication is based on the master-slave principle with the LiDAR as slave and the user who acts as master receiving point cloud data.

3.2.3. Point Cloud Data. The main data for autonomous driving produced by the LiDAR sensor is the measurements of the surrounding environment. These dimensions, defining measured points and their reflectivity in a three-dimensional coordinate system, are then packed in the point cloud data format and sent to the processing master. The format can be seen in Figure 4 which is based on the documentation [6].

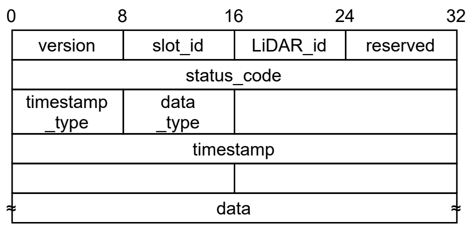


Figure 4: Point Cloud Data Format

Next to information about configuration and organisation, there is a timestamp type which should be highlighted at this point. The Livox SDK provides three types of timestamps: PTP (defined in IEEE 1588v2.0), GPS (requires PPS and UTC timestamp) and PPS (pulse per second). In case there are multiple synchronization sources available, the synchronization mode is selected in the order PTP > GPS > PPS. Talking about bandwidth, it is important to consider the data field or more precisely the amount of data fitting into that field. It mainly depends on the datatype transmitted in the packet. For the Livox Mid-40, there are two types that are transmitted, the Cartesian coordinate format with $104bit$ and the spherical coordinate format with $72bit$. As per header only 100 units of either type is transmitted, the amount of data is at most $10.4Kbit$. Including the header, the amount of data for the whole packet would rise by $96bit$. For the other fields and more information about the point cloud data format refer to the Livox SDK.

3.2.4. Livox SDK API. The Livox SDK API simply provides a set of C functions that can be accessed in C/C++ programs to get access to a Livox device.

3.3. Setup

All the physical hardware is located and provided by the 'Chair of Network Architecture and Services' in Garching. In the setup, there are mainly two components which can be seen in Figure 5.

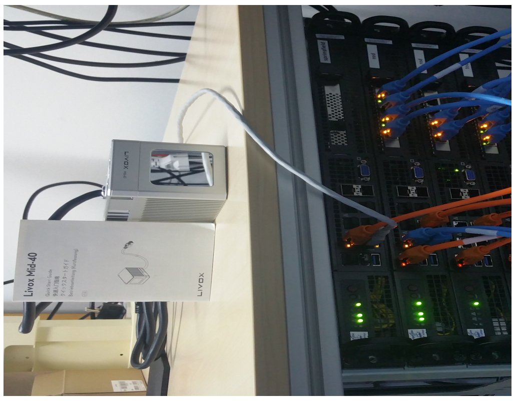
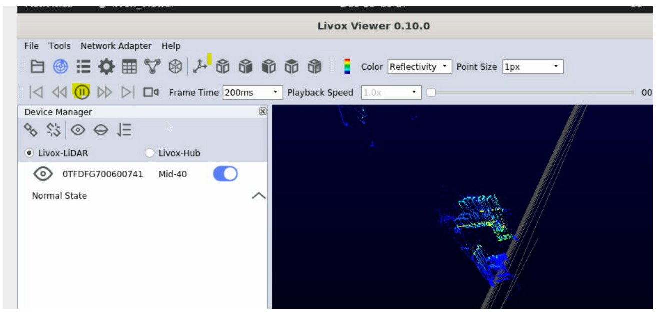


Figure 5: Setup

On the one side there is a host running an Ubuntu 20.04 image. Additionally, it has Livox Viewer 0.10.0 (64 bit) installed to process the data from the LiDAR.
The LiDAR from Livox of type ’Mid-40’ is on the other side, connected to the host with a cable of type ’100base10-tx’.

3.4. Livox Viewer

The Livox Viewer is a visualization software for the Livox LiDAR sensor. Its core function is to receive point cloud data recorded by the LiDAR, processes them and creates a three-dimensional visualization. Moreover, it is able to show and manipulate configurations such as the frame time and to store or display point cloud data. An example how it looks like can be seen in Figure 6.
The current version of the Livox viewer requires either


Livox Viewer 0.10.0

Color: Reflectivity
Point Size: 1px
Frame Time: 200ms
Playback Speed: 1.0x

Device Manager
(● Livox-LiDAR) ○ Livox-Hub
0TFDFG700600741 Mid-40
Normal State

Figure 6: Livox Viewer

Windows 7/8/10 (64 bit) or Ubuntu 16.04 (64 bit) and a dedicated graphics card is recommended, especially if multiple LiDAR sensors are in use.

4. LiDAR Comparison Parameters

As for all electronic devices, there are some parameters identifying and differing them from each other. It is also applied to LiDAR and to its broad field of application. In the following we will have a look on LiDAR in context of autonomous driving built-in in cars.

4.1. General LiDAR types

One can generally differ between two techniques that LiDARs use to detect their environment. On one hand, there is the spinning LiDAR able to spin 360° and thus detects its whole surrounding environment. Often used for developing prototypes, it is not common for series production, unlike, on the other hand, solid-state LiDARS. This name functions more like an umbrella term covering multiple types of solid-state LiDARS.

4.2. Mounting

In order to set a device properly into a car, several parameters have to be considered. Firstly, specific dimensions to integrate it into a car’s body as well as temperature requirements if autonomous driving functions should also work in extreme situations. Secondly, there should be followed several power requirements. Especially it has to meet parameters from a on-board system like the required voltage and the amount of power consumed by the LiDAR. Moving to processing of the provided data, the required bandwidth should be considered to function with the on-board system.

4.3. Specification

Diving more deeply into the specifications there are a couple of important points to examine.
In order to get most extensive and accurate data, there are crucial parameters to note such as visual field (FOV), detection range and maximum point rate. According accuracy, range precision, angular accuracy or beam divergence should also be considered. Using more parameters is not necessarily better, however, all depends on use cases and costs.

References

[1] Katie Burke, “How Does a Self-Driving Car See?” 2019. [Online]. Available: https://blogs.nvidia.com/blog/2019/04/15/how-does-a-self-driving-car-see/

[2] A. J. Hawins, “Watch Mobileye’s self-driving car drive through Jerusalem using only cameras,” 2020. [Online]. Available: https://www.theverge.com/2020/1/7/21055450/mobileye-self-driving-car-watch-camera-only-intel-jerusalem

[3] “Multi Purpose Camera.” [Online]. Available: https://www.bosch-mobility-solutions.com/media/global/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/multi-camera-system/multi-purpose-camera/summary_multi-purpose-camera_en.pdf

[4] Ann Neal, “LiDAR vs. RADAR,” April 2018. [Online]. Available: https://www.fierceelectronics.com/components/lidar-vs-radar

[5] B. S. Jahromi, “Ultrasonic Sensors in Self-Driving Cars,” 2019. [Online]. Available: https://medium.com/@BabakShah/ultrasonic-sensors-in-self-driving-cars-d28b63be676f

[6] Livox, “Livox SDK.” [Online]. Available: https://github.com/Livox-SDK/Livox-SDK

[7] Livox, “Livox SDK Communication Protokol.” [Online]. Available: https://github.com/Livox-SDK/Livox-SDK/wiki/Livox-SDK-Communication-Protocol#2-control-command

ISBN 978-3-937201-72-6
DOI 10.2313/NET-2021-05-1

ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)