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

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

Summer Semester 2020

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)

Summer Semester 2020

Munich, February 28, 2020 – August 16, 2020

Editors: Georg Carle, Stephan Günther, Benedikt Jaeger

Network Architectures
and Services
NET 2020-11-1

Proceedings of the Seminar
Innovative Internet Technologies and Mobile Communications (IITM)
Summer Semester 2020

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 SS 20
Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)
Munich, Germany, February 28, 2020 – August 16, 2020
ISBN: 978-3-937201-70-2

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2020-11-1
Innovative Internet Technologies and Mobile Communications (IITM) NET 2020-11-1
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 Summer Semester 2020. 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 Bernhard Vorhofer with the paper *Extending ZMap: Round-Trip Time Measurement via ICMP and TCP* and Simon Bachmeier with the paper *Network Simulation with OMNet++* .

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, November 2020

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@in.tum.de)
Technical University of Munich

Anubhab Banerjee (anubhab.banerjee@tum.de)
Technical University of Munich

Simon Bauer (bauersi@net.in.tum.de)
Technical University of Munich

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

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

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

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

Marton Kajo (kajo@net.in.tum.de)
Technical University of Munich

Holger Kinkelin (kinkelin@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

Johannes Schleger (schleger@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/ss20/seminars/`

Contents

The GDPR and its impact on the web

Daniel Anderson, Richard von Seck*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: ge72fat@mytum.de, seck@net.in.tum.de

***Abstract*—Processing of personal data is a key task for organizations and companies to optimize their business. To protect the consumers' privacy the General Data Protection Regulation (GDPR) was introduced in 2016 and became enforceable in 2018. The GDPR protects individuals by specifying how website operators can gain information about their customer. One core purpose is empowering people by giving them information on what their data is being used for. This paper introduces the tenets of the GDPR and presents the basic principles of data collection and processing, such as purpose limitation and data minimization. Furthermore, changes that have occurred on the web after the introduction of the regulation are discussed. A notable impact on web users is that they are now requested to give consent more often than before the law enforcement, less cookies are being collected, and users have the ability to inform themselves about data processing. However, the changes through the GDPR do not create more transparency on the web because the policies are too long and complex.**

***Index Terms*—gdpr, general data protection regulation, gdpr compliance**

1. Introduction

In the 21st century, internet is an essential part of most people's daily life. It is utilized for a multitude of different purposes, from purchasing clothes in a webstore to communicating with friends and family over social media to playing video games online. One thing that has become apparent in the last two years while browsing through the web is irritating pop-up windows or banners when visiting a website asking for consent to use cookies. Some might simply accept the use of cookies without giving it any further thought. Curious people might wonder what they are accepting and inform themselves what their data is being used for. For many people this is the first encounter where they become aware of data protection. Protection of personal data is specified to be a crucial right as stated in Article 8 of the Charter of Fundamental Rights of the European Union [1]: "Everyone has the right to the protection of personal data concerning him or her."
The GDPR [2] is a privacy and security law focusing on the protection of personal data within the European Union (EU) and the European Economic Area (EEA). Any information related to an identifiable natural person, i.e. an identification number or name, is considered personal data. The law protects every data subject no matter the nationality or residence. A data subject is an individual that organizations collect data about [3]. In this paper the terms "data subject", "user", "consumer", and "natural person" are used interchangeable. The GDPR applies to all organizations worldwide that collect or process personal data of EU citizens regardless of whether they have a presence in the EU [4]. There are two entities involved in the procedure of processing data. First, the controller which specifies the intent of the processing of personal data (Art. 4.7) and second, the processor that handles personal data for the controller (Art. 4.8). The regulation was introduced on 27 April 2016 and became enforceable on 25 May 2018. It replaced the 1995 Data Protection Directive (DPD) after Europe's data protection authority announced the DPD needed an update, following Googles lawsuit for scanning a user's emails in 2011 [5].
In the first part of this paper a brief overview over the core concepts and principles of the GDPR is given. The second part presents how the implementation of the data protection regulation has impacted the web and which changes have occurred.

2. Core Concepts

The following subchapters introduce key concepts and principles of the GDPR, show how consumers are protected by law, and specify special requirements that must be fulfilled by organizations to lawfully collect and process personal data.

2.1. Principles Regarding Personal Data

The GDPR consists of six core principles specifying how personal data must be collected and processed:

- Lawfulness, fairness, and transparency
- Purpose limitation
- Data minimization
- Accuracy
- Storage limitation
- Integrity and confidentiality

The first principle ensures that the data subject is always able to know what type of data is collected and what it is being used for. Transparency is further outlined in Section 2.6. Purpose limitation states that the collected information may only be used for the intended purpose and stored only for as long as necessary. Data minimization limits the data collection to only the relevant and necessary information. Only data that is relevant and necessary to fulfil the purpose should be gathered.

The principle accuracy specifies that all stored data must be accurate and updated. Therefore, false data must be deleted or corrected in order to ensure that the personal data is accurate. Storage limitation states that the information should not be stored for a longer time period than necessary. As soon as the data has been processed for the specified purpose and is no longer needed, it should be erased. Archiving in the public interest can however be a reason to store the data longer. The last principle ensures that all personal data is appropriately secured. It implies that data must be protected from unauthorized access, mislaying, or demolition. Thus, it is inevitable to take precautions in order to guarantee the security of the stored personal data (Art. 5) [3].

2.2. Lawfulness of Processing

The GDPR specifies several principles one of which is the lawfulness of processing personal data. The regulation states that the processing is illicit unless the GDPR states it contrarily. Consequently, one of the following reasons must apply for it to be lawful. The user has given consent (Art. 6.1.a). The processing of data is a necessity in order to perform or enter a contract (Art. 6.1.b). Another valid reason could be the controller having legitimate interest in processing the data (Art. 6.1.f), e.g. a company has discovered a security leak on their website and wants to inform users of this fault. Furthermore, data processing is also permitted in the case of it being a legal obligation (Art. 6.1.c), e.g. a messaging platform noticing suspicious interaction among users. Processing is also permitted if it is necessary to protect the data subject's vital interest or it is in the public's best interest (Art. 6). Vital interest is defined as being interests that are essential for the life of the data subject (Recital 46), i.e. a hospital checking an individual's medical background.

2.3. Consent

Consent is one of the six reasons defined in Article 6 of the GDPR as mentioned in Section 2.2 why permission for processing data is granted. Thus, one possible way to satisfy the GDPR requirements to process data is by getting consent from the data subject before processing their personal data. This can be done by simply having the user check of a tick-box. One of the core requisites is that consent must be a voluntary choice for the user, must be freely given, and shall be just as easy to withdraw at any time after it was granted (Art. 7.3). The request for consent shall be obviously distinguishable from others and presented in an appropriate way using foolproof language (Art. 7.2). The GDPR additionally protects children under the age of 16 by specifying that data collection is only lawful if authorized by a parent (Art. 8).

2.4. Special Cases of Data Processing

Next, the GDPR prohibits all processing of data belonging to special categories which consists i.a. of the race, ethnicity, political interest or religious belief of a natural person (Art. 9.1). Nevertheless, there are cases in which the processing is not prohibited. One example for this is if the data subject has given explicit consent for a specified purpose. This shows that even if the GDPR generally forbids the collection of some data, there are ways around it. The regulation further defines that personal data related to criminal conviction and offences is only to be processed under supervision of official authority (Art. 10).

2.5. Right to Erasure

The GDPR empowers the data subject by giving individuals the right to ask organizations to erase all the personal data that was collected. The right to erasure, also known as the right to be forgotten, forces the controller to delete personal data without "undue delay" if there is a legitimate reason, e.g. the data is no longer needed for the purposes which they were collected for or the user withdraws consent (Art. 17.1). Thus, if a user has the desire to have personal data removed, he can withdraw consent and request an erasure of his data. Also, the controller is obligated to notify the data subject and all other controllers in data processing about the erasure of the personal data if possible with reasonable effort (Art. 19).

2.6. Transparency for Data Subject

Transparency is defined so that any information that is brought to the public or to any person must be easily understandable, accessible, and succinct. The wording must be simple and comprehensible. If necessary, visualization should be used. Transparency is especially important in relation to the collection and processing of personal data of a data subject, as well as the reasoning and the identity of the controller. Especially when the data subject is a child, which means when the information targets children, the information must be communicated clearly and easy to understand. The information regarding privacy must be provided in a written document or electronically (Art. 12).

3. Impact on the Web

The GDPR regulations aim to impact the web in a way that it is more transparent. The following sections discuss different web interactions along with their technical implementations with the goal to determine if the transparency on the web has increased through the GDPR.

3.1. Privacy Policy

Firstly, many companies had to adapt their privacy policies in order to further be compliant with the requirements of the GDPR. Websites that did not have a privacy policy had to create one and if one already existed, they had to renew it based on the regulations. A factor that should not be left disregarded is that having a GDPR compliant privacy note does not necessary mean that it is transparent to a user. The length and difficulty to understand the provided information play an important role in comprehending the content. Sobers [6] measured the average reading time for privacy policies. An overview is presented in Figure 1. It shows the difference of reading time of the privacy policies across different websites from before the enforcement of the GDPR to afterwards. Eight out of the ten examples actually increased their wordcount of the privacy policy and thus the reading time has also gone up. Only Facebook and Reddit reduced the amount of words that are used in the privacy policy. In average the reading time increased from 17 minutes and 24 seconds to 20 minutes and 3 seconds. This refers to the average absolute reading time that a person needs to read through the entire privacy policy.

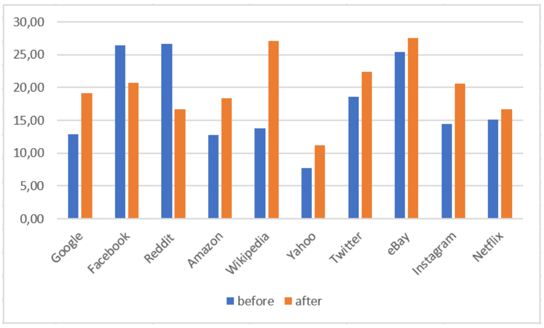


Figure 1: Reading Time in Minutes [6]

Furthermore, Sobers [6] analysed the reading difficulty of privacy policies. The reading level does not show a clear trend. The difficulty to understand the policy increased on five websites, decreased on two pages and remained at the same level on three sites. The platform eBay reached the highest reading level with 18 before the GDPR and 20 afterwards. This is at the reading comprehension level of a senior college student. However, it was not clearly stated which scale the author references for the comparison of the reading level. The goal to decrease privacy concerns and increase transparency was therefore not reached when looking at privacy policies [6].

3.2. Browser Cookies

A cookie is information about a natural person that is generated by a website and stored on the computer by the browser. They can track a user's activity in a browser and remember information such as shopping history. Since the GDPR there has been an increase in the amount of cookie consent banners on websites complying with Article 6.1.a. This provides reasons for data processing to be lawful, one of which is collecting consent from the user. Lawfulness of processing is further discussed in Section 2.3. Companies had to create consent forms in order to be allowed to further collect individuals' data and adjust the way they process data. This mostly applies to consumers from Europe. According to Dabrowski et al. [7] unrestricted use of tenacious cookies has become significantly less for EU web users. Their study collects data sets measuring cookie behavior using Alexa Top 100,000 websites before the adoption of the GDPR in 2016 as well as after introduction in 2018. Measurements show that only 26% of websites, which collect cookies, avoid collecting them without consent from an EU visitor. The GDPR even has an impact on users outside of the EU. When comparing the results from 2016 to the results from 2018 US consumers profit from a reduction of up to 46.7% of the amount of cookies collected. Therefore, the new cookie policies have in theory positively impacted the users' privacy all around the world. On another note, this is also perceivable for consumers since they are obliged to give their consent on most websites that accept the cookies in order to further be able to access the page. Users are found to be disturbed by the frequent occurrence of disclaimers. They also seem to have more privacy concerns because they are now aware that their activities are being monitored and their data is being used. Looking at this from a practical point of view, only very few users are actually willing to click on the banner and read through how the cookies are being used. This as well as the point that was already addressed in regard to the privacy policy does not really add to a higher transparency. If users are not able to understand what is written in the cookies or if they are confronted with a humongous amount of words that they are not willing to read, the people are not really empowered, and the process does not become more transparent [8].

3.3. Third Party Presence

In the business environment there is an interdependency between websites. An interdependency i.e. exists if one website uses the services of another webpage in form of third party presence. An example of a third party presence is Google Analytics. Google Analytics is a service that tracks website activities such as session duration and bounce rate [9]. For a person's privacy this means that another entity has access to their activities and thus it is interesting to determine if the third party presence has decreased since May 2018.

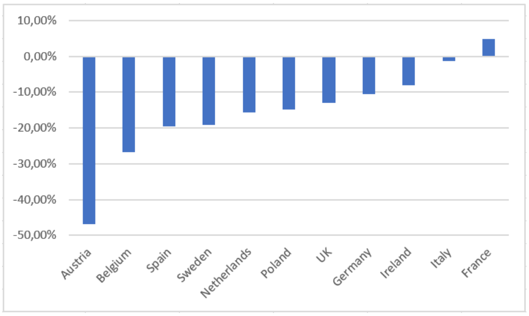


Figure 2: Comparing pre- and post May 25 average numbers of unique third parties for countries, listed as change in percent [10].

Sørensen and Kosta [10] show the percentage difference in the average numbers of unique third parties on websites before and after May 25th, 2018. An overview of the top ten countries with the strongest economies [11] is presented in Figure 2. Austria shows the biggest change with a decline of 46.79%. Whereas in France for example the average number of unique third parties actually grew 4.83%. It shows a decline in most countries and in the total number of third parties, but it must be noted that this is not a in depth inspection of each country. Therefore, it is not possible to conclude a significant impact of the GDPR on third party presence [10].

3.4. Newsletters

If a data subject had subscribed to a newsletter before the GDPR was enforced, the organization must have collected the data in a lawful way, following the rules presented in Section 2.3 to further send messages. Organizations must be able to prove that they have the user's consent or have another lawful reason for processing data so they can continue sending messages to the data subject. Recital 32 of the GDPR specifies that consent must be "freely given, specific, informative and unambiguous". If this is not the case, their personal data has to be deleted from the mailing list. Companies are forbidden to use personal data for any other reason than specified. Thus, previous to the law a company could for example use the contact information of a user that bought an eBook to send them recurring email campaigns. Now, they must add another checkbox where the user must agree that their information can be used for other marketing actions. If this is not agreed upon, no emails or letters can be sent to them [12].

3.5. Data Exchange

Companies like Google and Facebook provide free usage of their platforms in exchange for user data. They track activities like browsing history, likes, and purchases. There have been numerous scandals in the past concerning data privacy and protection. This data is then purchased by advertisers so that they can better direct their ads. The GDPR regulations influence this business model severely. In general, these companies now have different policies for the countries to which the GDPR applies to and all other countries. For the data collection and processing of Europeans the controller now has to follow the rules that were discussed above such as get consent from the data subject, prove the lawfulness of storing the data, and maintaining the data. A factor that is also influenced by this is the data transfer across borders. If the data is for example transferred to the US, a country which does not ensure sufficient data protection, they have to sign on to the Privacy Shield or assure adequate security in a contract [13].

3.6. Trust

The GDPR has not succeeded in increasing trust in the digital economy. More than 80% of European citizens have the feeling that they do not really have any control over the information they reveal on the internet [14]. This problem possibly arises due to the points that were discussed in relation to the privacy policy and cookies. Even though two thirds of Europeans are aware of the existence of the GDPR, there does not seem to be a strong relationship between the awareness of the data protection regulations and the actual feeling of safety in the web [14]. The reasoning for this might be that the GDPR is quite complicated to understand. However, it should be possible for every natural person affected by the GDPR to be able to understand its contents. Also, the awareness for the regulations should be raised so that more people can feel safer when browsing the internet.

4. Conclusion

The GDPR has definitely had an impact on the web. However, not all changes are positive. In theory, there has been a notable increase in transparency. Europeans now have the possibility to inform themselves about what data is collected and how it is processed. Also, less cookies are being collected. Nevertheless, all the transparency that is now given can lead to an incorrect sense of security because even though a website ensures you of your privacy, it does not necessarily mean that they actually follow this correctly. Also, even though it is now possible to be knowledgeable about one's privacy protection, this does not necessarily lead to more transparency in the web. The length and difficulty to read and understand privacy policies or cookies negatively impacts the transparency. It is furthermore important to create more awareness and understanding of the GDPR for it to be more effective and provide a sense of security to the users. For future research it would be interesting to get a better insight into the effect of third party presence and how the awareness of this can be raised for data subjects by the controller. Another interesting topic is to inspect whether the websites actually follow the data protection regulations that they claim to secure.

References

[1] European Parliament, "Charter of fundamental rights of the european union (2007/c 303/01)," https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:12012P/TXT&from=EN, 2007, online; accessed 26 March 2020.

[2] E. Parliament, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)," https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN, 2016, online; accessed 26 March 2020.

[3] "General data protection regulation faq," https://www.into.ie/app/uploads/2019/10/GDPR_FAQ.pdf, 2018, online; accessed 02 May 2020.

[4] C. Tankard, "What the gdpr means for businesses," *Network Security*, vol. 2016, no. 6, pp. 5–8, 2016.

[5] Ben Wolford, "What is gdpr, the eu's new data protection law?" https://gdpr.eu/what-is-gdpr/, online; accessed 28 March 2020.

[6] Rob Sobers, "The average reading level of a privacy policy," https://www.varonis.com/blog/gdpr-privacy-policy/, 2018, online; accessed 28 March 2020.

[7] A. Dabrowski, G. Merzdovnik, J. Ullrich, G. Sendera, and E. Weippl, "Measuring cookies and web privacy in a post-gdpr world," in *Passive and Active Measurement*, D. Choffnes and M. Barcellos, Eds. Cham: Springer International Publishing, 2019, pp. 258–270.

[8] O. Kulyk, A. Hilt, N. Gerber, and M. Volkamer, "This website uses cookies: Users' perceptions and reactions to the cookie disclaimer," https://www.researchgate.net/publication/325923893_This_Website_Uses_Cookies_Users'_Perceptions_and_Reactions_to_the_Cookie_Disclaimer, 2018, online; accessed 02 May 2020.

[9] B. Plaza, "Google analytics for measuring website performance," *Tourism Management*, vol. 32, no. 3, pp. 477–481, 2011.

[10] J. Sørensen and S. Kosta, "Before and after gdpr: The changes in third party presence at public and private european websites," 05 2019.

[11] Bruno Urmersbach, "Bip (bruttoinlandsprodukt) in den mitgliedsstaaten der eu in jeweiligen preisen im jahr 2018," https://de.statista.com/statistik/daten/studie/188776/umfrage/bruttoinlandsprodukt-bip-in-den-eu-laendern/, 2019, online; accessed 29 March 2020.

[12] Kyle McCarthy, "Examining the impact of gdpr one year in," https://www.act-on.com/blog/examining-the-impact-of-gdpr-one-year-in/, 2019, online; accessed 26 March 2020.

[13] K. Houser and W. Voss, "Gdpr: The end of google and facebook or a new paradigm in data privacy?" *SSRN Electronic Journal*, 07 2018.

[14] Eline Chivot and Daniel Castro, "What the evidence shows about the impact of the gdpr after one year," https://www.datainnovation.org/2019/06/what-the-evidence-shows-about-the-impact-of-the-gdpr-after-one-year/, 2019, online; accessed 28 March 2020.

WPA 3 - Improvements over WPA 2 or broken again?

Maximilian Appel, Dr.-Ing. Stephan Guenther*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: m.appel@tum.de, guenther@tum.de

***Abstract*—Due to the widespread usage of WiFi, securing them is and will continue to be an important task. After it was signed into IEEE 802.11 in 2004, WPA2 became the commonly used encryption standard for WiFi networks, replacing the originally as temporary solution conceived WPA. It's successor WPA3 was released in June 2018. At the time of writing it has not found widespread adoption yet. This paper aims to provide an overview on the designs of WPA2 and WPA3, including their currently known vulnerabilities. And tries to come to a conclusion on whether WPA3 is still a viable successor or if it has already been compromised beyond repair.**

***Index Terms*—wireless networks, WPA2, WPA3, Encryption, KRACK-Attack, Dragonblood-Attack**

1. Introduction

Created as a guideline for wireless connected networks in 1997, IEEE 802.11 also defined security protocols for such networks and as such has been revised multiple times in reaction to emerging technologies and attack methods. The original security mechanism WEP was replaced in 2003 in favor of the at them time new WPA. The main reason for this was the discovery of major weaknesses in the RC4 encryption algorithm, that WEP was based on. However, this was only an intermediate measure meant to strengthen security during the creation process for a full amendment to the the standard. The full standard was signed in 2004 as IEEE 802.11i, also more commonly known as WPA2. This amendment officially deprecated WEP and even forbids the implementation in new devices. However because of the higher hardware requirements of its successors it partially remains in use to this day. In 2018 WPA3 was announced as the replacement for WPA2, meant to solve known problems and vulnerabilities. The full protocol was released in June 2018 and is, at the time of writing, the currently recommended security standard for wireless networks. The rest of this paper first provides a detailed description of WPA2 and WPA3. This is followed by a brief analysis of the currently known vulnerability for each of them. It is then concluded with a discussion on whether or not WPA3 is still a viable security scheme. [1]

2. WPA2

Signed as IEEE 802.11i in 2004, WPA2 marked a large step forward not only in terms of security, but also in terms of hardware demands. The main reason for the latter is the utilization of Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP), which uses the Advanced Encryption Standard (AES) block cipher for its data encryption. Both of which are described in more detail in the following sections. Temporal Key Integrity Protocol (TKIP) is still available under WPA2, in order to provide backward compatibility to WPA capable devices that possess insufficient processing power for AES.

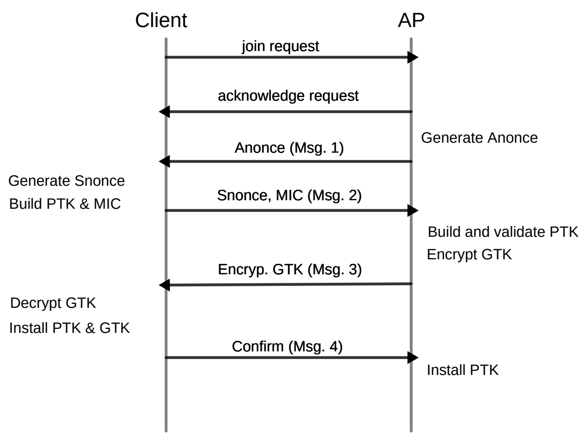


Figure 1: diagram showing the steps of the four-way handshake

2.1. Authentication

WPA2 handles authentication via two separate types of keys, which are used for the decryption and encryption of messages. Pairwise Transient Keys (PTKs) that are used for unicast messages and as such are only known to the AP and a single client. The other type is the Groupwise Transient Key (GTK), a single key that is used for multicasts and broadcast, and therefore is known by the AP and all clients in the network. In order to generate and distribute these keys WPA2 uses two separate handshake protocols. The so called four-way handshake is executed first and generates and distributes the PTK. A simple handshake that is secured with the individual clients PTK, is used to update and distribute the GTK from the AP to the clients. The four-way handshake (see figure 1) begins under the assumption that both the client and the AP possess a shared Pairwise Master Key (PMK), which consists of a PBKDF2 function value of the network's passphrase, the networks Service Set Identifier (SSID), and the Hash Message Authentication Protocol (HMAC) function used to stretch the passphrase. First the client sends a connection request to the AP, which is then answered with an acknowledgment. Afterwards the client generates a nonce (Anonce), a randomly generated value that prevents message replay attacks, and sends it to client. The client then generates a nonce of its own (Snonce) and uses it to generate the PTK, by concatenating both nonces, the PMK and the mac addresses of both AP and client. In the next step the client uses the PTK to generate a Message Integrity Code (MIC) and then sends the Snonce along it to the AP. The AP then uses the received Snonce to generate the PTK, in the same way the client did. After that it uses the PTK and the Snonce to derive a MIC, which is then compared to the MIC that was received with the Snonce. If either of the nonces was manipulated by an attacker, the MICs will not match and the handshake is aborted. Due to the random nonces role in the generation Process, the generated PTK will always be unique for the individual session. At the end of the four-way handshake, the AP uses the freshly established PTK to safely transmit the current GTK. This completes the client's authentication. [2]

2.2. AES

The Advanced Encryption Standard (AES) defines a secure block cipher encryption algorithm. AES was chosen in 2001 at the end of the AES selection process as the standard for safe encryption by the US government. AES supports various key sizes (128bit, 192bit, and 256bit) and handles data in blocks. The block's sizes are independent of the chosen key size. A complex algorithm is used to enlarge the initial key into several 128bit large keys. All but one of these keys are then used in separate encryption rounds, each of which consists of three substitutions and one permutation. The total number of rounds depends on the used key size (see table 1). The remaining unused key is later used to start the decryption process. [1], [3]

2.3. CCMP

Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP) is an implementation of the standards of the IEEE 802.11i amendment. It protects the confidentiality of the data by using AES in counter mode and uses CBC-MACs to assure authenticity and integrity of the messages. It therefore provides protection for confidentiality, authenticity and integrity. The protocol first takes a key, which in WPA2's case is either a PTK or the GTK, and additional data necessary for the protocol and runs them through AES in counter mode. Counter mode refers to a specific algorithm that turns a block cipher into a stream cipher. The in this way generated keystream is then combined with the plaintext in an XOR operation to build the encrypted ciphertext. (see figure 2) [1]

2.4. Vulnerabilities

This section provides a brief overview on WPA2s known weaknesses. For the sake of brevity, we are limiting this section to attacks that allow attacks on APs without prior knowledge of the password.We are also excluding the KRACK-Exploit, because patches that secure APs against it are available under WPA2.

TABLE 1: Number of encryption rounds for AES key sizes

AES key size	Number of rounds
128 bit	10
192 bit	12
256 bit	14

2.4.1. Deauthentication Attack.

Session management frames are system messages between clients and APs used for communication. Once type of management frame are de-authentication frames which are normally sent by client and AP to signal the end of a session. Under WPA2 session management frames are sent encrypted without authentication. By spoofing the clients and the AP's MAC address and then sending false Deauthentication frames, an attacker can cause both AP and client to cease communication with each other. [1]

2.4.2. Handshake Capture Dictionary Attack.

The four-way handshake generates the PTK by combining two randomly generated nonces with the otherwise completely static PMK. Because the nonces are sent in plaintext, an attacker can gain enough information to perform off-line dictionary and brute-force attacks against the passphrase by eavesdropping on an successful handshake. In theory this still requires an attacker to wait for a handshake that he can capture. However, by using the previously mentioned Deauthentication Attack to disconnect an already authenticated client the attacker is able force a the client to perform a four-way handshake, which he then can capture. [1]

2.4.3. PMKID Hash Dictionary Attack.

During the authentication phase of WPA2, but before the actual four-way handshake, the AP sends the client a Extensible Authentication Protocol over LAN (EAPOL) frame. This frame contains the titular Pairwise Master Key Identification (PMKID). The PKMID is a hash value derived from the PMK, a static String, the clients MAC address and the AP's MAC address. An attacker can use the PKMID to perform dictionary and brute-force attacks against the networks passphrase by simply calculating the hash with a candidate passphrase and comparing the result with the PKMID. [1]

3. WPA3

Published in 2018, WPA3 is mostly build upon its predecessor and as such only makes minor changes to the decryption standards. For the sake of brevity this paper only focuses on the big changes in the authentication protocol. This is then followed by a description of the relatively recently discovered vulnerabilities: the Dragonblood-Attacks.

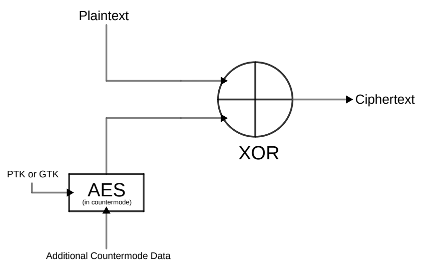


Figure 2: diagram showing the encryption process with CCMP

3.1. Authentication

WPA3 changes the authentication protocol by adding an additional layer of security in form of the Simultaneous Authentication of Equals (SAE) handshake, a variant of the Dragonfly Handshake. The Dragonfly Handshake was originally developed by Dan Harkins in 2008 as Password Authenticated Key Exchange (PAKE) for mesh networks and as such turns a shared plaintext password into a secure cryptographic key. Dragonfly supports Elliptic Curve Cryptography (ECC) and Finite Field Cryptography (FFC). Because ECC is the more common option and the general similarities between the two, this paper only contains a description of ECC.

3.1.1. Elliptic Curve Cryptography.

Dragonfly uses elliptic curves over a prime field (ECP groups) in its ECC mode. ECP groups are defined over a prime p and the parameters a and b for the polynomial

$$y^2 = (x^3 + ax + b) \bmod p$$

The key is generated out of the shared password by calculating a combined hash over the pre shared password, an increment counter and identities (IDs) of the client and the AP. In WPA3 the identities are the client's MAC address and AP's MAC address. That hash is then used as the x value in an attempt to find a corresponding y value. If that attempt is unsuccessful, the hash is recalculated with an increased counter. After this another attempt with the new x value is made. This strategy is repeated until a y value is found. The calculated point (x,y) is then used as the password (P) in the Dragonfly Handshake. [4]

3.1.2. The Dragonfly Handshake.

The handshake consists of two phases. The commit phase occurs first and can be initiated by both parties, although in WPA3-personal the client will always send the first commit, while in enterprise mode the radius server commits first. For the commits each of the peers chooses two random values within the interval [2,p] , a private r_i and a mask m_i such that $s_i = r_i + m_i \: \epsilon$ [2,p]. They then calculate the value $E_i = -m_i \cdot P$ and send it along with their respective s_i to each other. After having received the commit frames, both participants validate the received values and abort the entire handshake in the case of an incorrect value. In the confirm phase the parties use the exchanged data to calculate the secret point K with the formula $K = r_i(s_j P + E_j)$ ($_j$ denoting the parties own value and $_j$ denoting the received values) and hashes its x coordinate to get the key k. They then calculate a HMAC c_i over all the data that has been generated and exchanged during the handshake, utilizing k as key. The parties then exchange and check their corresponding c_i's in a confirm frame, which is discarded in the case of an unexpected value. If the values are correct, the handshake has succeeded and k is the resulting key (see figure 3. [4]

The key k is then used as PMK in WPA2s four way handshake, which is now more secure than before due to the much higher entropy of the PMK. This also enables backward compatibility with devices that are unable to perform the calculations necessary for the Dragonfly Handshake, by setting the AP into a Transition-Mode to merely advertise the Dragonfly Handshake as part of optional Management Frame Protection (MFP) to the clients, although actually all WPA3 capable devices are forced to use the MFP, despite it being advertised as optional. [1]

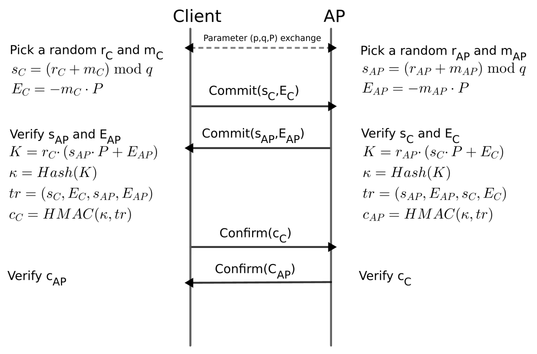

Client
AP
Parameter (p,q,P) exchange
Pick a random r_C and m_C
$s_C = (r_C + m_C) \bmod q$
$E_C = -m_C \cdot P$
Pick a random r_{AP} and m_{AP}
$s_{AP} = (r_{AP} + m_{AP}) \bmod q$
$E_{AP} = -m_{AP} \cdot P$
Commit(s_C,E_C)
Commit(s_{AP},E_{AP})
Verify s_{AP} and E_{AP}
$K = r_C \cdot (s_{AP} \cdot P + E_{AP})$
$\kappa = Hash(K)$
$tr = (s_C, E_C, s_{AP}, E_{AP})$
$c_C = HMAC(\kappa, tr)$
Verify s_C and E_C
$K = r_{AP} \cdot (s_C \cdot P + E_C)$
$\kappa = Hash(K)$
$tr = (s_{AP}, E_{AP}, s_C, E_C)$
$c_{AP} = HMAC(\kappa, tr)$
Confirm(c_C)
Confirm(C_{AP})
Verify c_{AP}
Verify c_C

Figure 3: a diagram detailing the WPA3 SAE Handshake between a connecting client and an AP. In theory either parties can initiated the handshake, but we assume the standard case of the Client sending the first commit. The Calculations shown assume that an ECP group is used.

3.2. Dragonblood

The Dragonblood attacks refer to a number of weaknesses in the WPA3 security scheme's personal mode. They were published by Mathy Vanhoef and Eyal Ronen in April 2019. Dragonblood consists of several different types of attacks.

3.2.1. Downgrade Attacks.

Downgrade attacks target WPA3 networks that are set into the earlier mentioned transition mode. Normally this mode allows devices that are incompatible with the SAE protocol to still connect to the AP under WPA2, while forcing all devices that are able to, to use WPA3. However, by setting up an impostor network with the same ID that only supports WPA2, even WPA3 capable clients are tricked into using WPA2. By catching parts of the WPA2 handshake they are able to once again perform dictionary and brute force attacks against the passwords of WPA3 networks. [4]

3.2.2. Security Group Downgrade Attack.

During the commit phase of the Dragonfly Handshake, the initiator, which in case of WPA3 is usually the client, sends his first commit frame with his preferred security group. If the AP doesn't support this specific security group he answers with a decline message, forcing the client to use a different possibly unsafer group instead. This is done until the AP accepts the offered group. By catching the clients commits and sending fake denial messages, an attacker is able to force the client into using a group of his choice. [4]

3.2.3. Cache-Based Side-Channel Attack.

These attack require an attacker to be able to observe the memory access pattern of one of the parties of the Dragonfly Handshake. The memory access patterns during the generation of a commit frame allow an attacker to gain information about the used password. This information can be used for dictionary attacks that compare the observed patterns with the expected patterns of a to be guessed password. [4]

3.2.4. Timing-Based Side-Channel Attack.

When using certain security groups, the time it takes for an AP to response to a commit frame depends on the used password. This leaked information allows an attacker to perform a variant dictionary attack, by comparing the expected time for a password with the AP's actual response time. [4]

3.2.5. Denial-of-Service Attack.

Due to the high computational cost in the Dragonfly Handshakes commit phase, an attacker can very easily overload an AP by sending bogus commit frames. This leads to high CPU usage on the AP, which in turn can cause delays or even prevention in the regular use of the AP. These attacks can be worsened, depending on if and how defenses against the previously described side channel attacks are implemented, due to the necessity of additional computations. [4]

3.2.6. Possible Fixes.

The downgrade attack against the transition mode, while in theory only temporary, is still highly problematic. Because it is to be expected that the adoption of WPA3 to the point of the full on deprecation of WPA2 will at least take several years. Until that point however most WPA3 networks will likely be used in transition mode. Meaning that unless this vulnerability is closed, all of these APs will be vulnerable to dictionary and brute-force attacks. As already mentioned and shown by the followup research made after the original publication, implementing WPA3 with defenses against the currently known side channel attacks without introducing new ones has proven rather difficult and tedious. They also have shown to increase WPA3's already high computational requirements even further, which poses problem for devices that are unable to implement them. [4]

4. Conclusion

WPA3 was meant to solve most of the vulnerabilities that WPA2 had. None of the known WPA2 specific attack methods went unaddressed and all of the vulnerabilities described in the WPA2 section 2.4 were fixed. Due to the additional security provided by the Dragonfly Handshake, it was considered to be almost impossible to crack the password of a WPA3 network. However Dragonblood revealed major flaws within the WPA3 security scheme that in our opinion cast serious doubt on its long term viability as a security standard. At the time of writing WPA3 is prone to implementation errors, which make side-channel attacks possible. In addition to that WPA3 also has two known conceptional faults, that make the side channel leaks even worse. Although technically temporary the downgrade attack against the transition mode remains especially worrisome, since transition mode can be expected to be the most common use case for at least the next several years. In total all of these vulnerabilities make dictionary and brute-force attacks on the passphrase once again possible, negating one of the biggest advantages WPA3 had over WPA2. All of this has lead us to believe that WPA3 in its current form should not be a long term solution and either has to amended in order to fix the currently known problems or possibly even abandoned in favor of an alternative improved scheme. Implementing additional defenses under WPA3 have proven to be problematic and even partially impossible on already existing WPA3 hardware. Meaning that expensive hardware upgrades might become necessary. All of this makes WPA3's future seem highly uncertain, as it will depend on the feasibility of possible solutions, which warrants further research. Despite these issues with WPA3 our conclusion is still that WPA3 is a considerable improvement over WPA2 in terms of security and should remain in service for the time being.

References

[1] T. H. Christopher P. Kohlios, "A Comprehensive Attack Flow Model and SecurityAnalysis for Wi-Fi and WPA3," *Electronics*, vol. 7, no. 11, 2018.

[2] M. Gast, *802.11 Wireless networks: the definitive guide, 2nd edn.* O´Reilly Media, Inc., 2005.

[3] "FIPs PUB 179: Announcing the ADVANCED ENCRYPTION STANDARD (AES)," November 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[4] M. Vanhoef and E. Ronen, "Dragonblood: Analyzing the Dragonfly handshake of WPA3 and EAP-pwd," in *IEEE Symposium on Security & Privacy (SP)*. IEEE, 2020.

Routing in Mobile Ad Hoc Networks

Christian Brechenmacher, Jonas Andre*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: brechenm@in.tum.de, andre@in.tum.de

Abstract—**This paper presents the different MANET routing types, introduces some specific routing algorithms, and draws a comparison between them. The paper focuses on the two main types of MANET routing, Table-Driven and On-Demand. It presents the Table-Driven DSDR protocol, and the On-Demand DSR protocol. The comparison will draw a spotlight on differences of performance in the categories overhead, availability and scalability. While Table-Driven protocols outperform On-Demand protocols on availability, as the MANET grows in size, the performance of Table-Driven protocols suffers as their rather large overhead worsens their scalability.**

Index Terms—**manet, routing, protocols, proactive, reactive, dsdv, dsr, ad hoc network, algorithm design and analysis**

1. Introduction

Mobile Ad Hoc Networks (*MANET*) consist of nodes that do not require the intervention of an access point. They are envisioned to be dynamic, multi-hop topologies that are composed of wireless links. Supporting this type of mobility requires routing protocols, as it ensures efficient use of the scarce capacity that wireless bandwidth has to offer. *MANET*s are incredibly useful, especially in situations when no other infrastructure is provided. As a result, there is a plethora of *MANET* routing algorithms.

*MANET*s are flexible alternatives to generic Wifi networks, not only in situations where critical infrastructure goes down, like in war zones, disaster areas or other emergencies; they are also very useful in planes, connecting cars and in other decentralized applications. The basic idea of the Ad Hoc Network in general is to enable transactions between different devices without the need for a router or similar base stations. Another difference to fixed networks is that the connection built by an Ad Hoc Network serves only one purpose and could only be temporary. Information in an Ad Hoc Network may be transferred over multiple nodes, if necessary, which is then called a multi-hop network. Ad Hoc Networks can be used locally, as a connection between several devices communicating wirelessly, or it may permit exogenous traffic to transit through one or several of the nodes [1].

2. Characteristics of *MANET*

Considering this, it becomes clear that *MANET* routing poses some unique challenges. The salient characteristics of *MANET* are [2]:

- Dynamic topologies: The nodes are free to move, which changes the network architecture unpredictably. The links between the nodes may consist of both unidirectional and bidirectional links.
- Bandwidth-constrained, variable capacity links: Wireless links have a lower capacity than hardwired ones. A lower thoughput because of multiple accesses, fading, noise, and interference conditions, may lead to congestion.
- Energy-constrained operation: As the network and its nodes are mobile, the energy resources it drains from may be exhaustible.
- Limited physical security: Because of *MANET*'s wireless connection, the danger of eavesdropping, spoofing or DOS-attacks is increased. However, its decentralized nature proves as a benefit in terms of robustness against single points of failure.

These characteristics provide a set of diverse prerequisites and performance concerns for routing designs, extending beyond those of the static, highspeed topology of wired networks. Routing is the strategy that oversees the way the nodes decide to forward packets between them. For most protocols, the manner in which they are sent boils down to the *Shortest Path Problem*. Each node should therefore maintain a preferred neighbour for each destination packets can be sent.

Due to the lack of one centralised router, in Ad Hoc Networks, nodes have to become aware of their topology themselves. New nodes therefore announce themselves and listen for announcements from their neighbours, hereby attaining knowledge about nodes nearby [2].

3. Related Work

As MANET is an extensive research field, there are many similar works available. Geetha Jayakumar and G. Gopinath [3] thoroughly describe the taxonomy of Ad Hoc Networks, before giving an outline of different Ad Hoc Network routing protocols. They then compare the benefits of the different routing types. Shima Mohseni et al. [4] also give a short overview over different Ad Hoc Network routing protocols, before discussing the discriminating factors between them.

Yuxia et al. [5] ran some simulations to determine how different protocols perform as the network's size increases, the results of which will also be used in this paper.

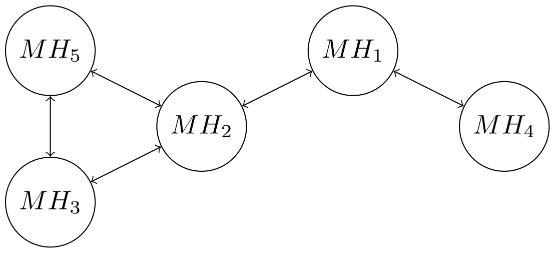


Figure 1: A bidirectional Ad Hoc Network

4. Table-Driven and On-Demand Routing

There are two main approaches to routing in *MANET*: *Table-Driven* (proactive) and *On-Demand* (reactive).

The basic idea of the proactive design is to operate each node as a specialized router, which periodically advertises its view of the interconnection topology with other mobile hosts within the network. The routing information is maintained on every node, even before it is needed.

A node contained in an environment maintained by a reactive routing protocol sends packets over routes that are determined in an On-Demand manner. There are two types of protocols needed for this approach to work [6]:

- *Route Discovery*: a route between two certain nodes is not yet known, meaning it is not yet in the sending node's routing table. A broadcast with a route discovery packet is sent through the network, and once a route has been discovered, a connection is established and data can be transmitted.
- *Route Maintenance*: as one of the characteristics of *MANET* is its dynamic topology, the occurence of link breaks must be taken into account. A node should therefore be able to confirm that a packet has been orderly received by its next-in-line node. If that is not the case, the node can then choose another route that is already known, or can invoke route discovery to find another path.

5. The DSDV Routing Protocol

One proactive protocol is the *Destination-Sequenced Distance-Vector* routing protocol (DSDV) [7]. Its basic idea is to display a shortest path for each node contained in the *MANET* by managing routing tables stored at each station of the network. Each of these tables lists all possible destinations and the number of hops over auxiliary stations required to get there. Each entry in these tables is tagged with a sequence number (see tables 1 and 2), which stems from the route's destination and is, usually, always even. It guarantees the freshness of the information by increasing the sequence number when initializing a new update period. To remain a consistent picture of the network's topology, these updates have to be frequent, and immediate when significant new information arises. As there is no time synchronization between the hosts of the network, the update periods are not coordinated between them, although some regularity can be expected.

The routing information is advertised by broadcasting packets with the network information and is transmitted in *Network Protocol Data Units* (NPDU). They also contain information about the broadcaster and an incremented sequence number in the header of the packet. The extend of the routing information depends on the type of update that is occurring: one type carries all the available information, called the *full dump*. That includes the destination, the metric – which informs about the number of hops needed until the destination is reached – and the sequence number.

The other is called an *incremental* and only contains the information that has changed since the last full dump occured.

TABLE 1: Advertised route table of MH_3 (full dump)

Destination	**Metric**	**Sequence number**
MH_1	2	S380_MH_1
MH_2	1	S256_MH_2
MH_3	0	S468_MH_3
MH_4	3	S176_MH_4
MH_5	1	S324_MH_5

Table 1 depicts the advertised full dump of mobile host 3 (MH_3) from figure 1. It is visible that, for instance, MH_1 with its sequence number S380 is two hops away.

A full dump requires, even for small networks, multiple NPDUs, while an incremental regularly only requires one NPDU. Because of its size, full dumps can be scheduled to happen infrequently when the topology of the network is not changing. As change becomes more frequent, full dumps can be scheduled to occur more often to decrease the size of the next incremental.

Routes advertised in a broadcast are updated in the routing tables of the receiver and readvertised in the broadcast of the receiver. Firstly, the receiver integrates the new information in its forwarding information table. That includes the destination, the next hop, the metric, the sequence number, the install number – which informs about the time the station became visible to the current station – flags, and a pointer that points to alternative routes. Table 2 shows such a table from the perspective of MH_3. Picking up on our earlier example, it is now additionally visible that the next hop on the route to MH_1 would be MH_2. We can see that MH_1 was the last host that became visible to MH_3, as it has the highest install number. In this example, all pointers would point to null structures, as there are no routes that compete with each other, or that are in any way likely to be superseded, if they were, for instance, broken links.

If a route's sequence number is more recent then before, it is always used. The number of hops required for one route, received by the broadcast, is increased by one. If the route now requires fewer hops than before and the sequence number is recent, it is updated in the routing table of the receiver. Alternatively to updating the table entries, which effectively discards old information, it is also possible to just create a new entry, and use this one preferably. This, however, results in a bigger overhead, and is therefore not recommendable if the network is stable.

If a link to another router is broken, it can be detected through the layer-2 protocol, or can just be inferred if there has been no communication from this host in a while. Its sequence number is then updated and its metric is set to infinity. Sequence numbers indicating a broken link with an infinite metric are always odd numbers. That way they

TABLE 2: All forwarding information of MH_3

Destination	**NextHop**	**Metric**	**Sequence number**	**Install**	**Flags**	**Stable_data**
MH_1	MH_2	2	S380_MH_1	T003_MH_3		Ptr1_MH_1
MH_2	MH_2	1	S256_MH_2	T001_MH_3		Ptr1_MH_2
MH_3	MH_3	0	S468_MH_3	T001_MH_3		Ptr1_MH_3
MH_4	MH_2	3	S176_MH_4	T002_MH_3		Ptr1_MH_4
MH_5	MH_5	1	S324_MH_5	T002_MH_3		Ptr1_MH_5

are superseded by a new sequence number if a connection towards that host can be established again. A broken link, as well as information about a host which was previously unreachable, is important information which needs to be broadcasted immediately.

However, this is not the case for a normal update. In a suboptimal setting, a router can receive a route with an updated sequence number, but with a bad metric, first, before it receives another route with the same sequence number and a better metric. Were it to immediatly broadcast the new route, it would lead to a burst of route broadcasts from that receiver, which resolves in a chain reaction of broadcasts from all the hosts in the network. A solution to this problem is to delay the broadcast of the router, but already use the new information for itself to forward received packages. That concludes that two tables have to be maintained: one for the purpose of routing, the other one for broadcasting. The host also has to maintain a history of the average time it takes from the first route to the best route to arrive. Based on that, a host may be able to predict how long it has to wait before advertising its routes.

6. The DSR Routing Protocol

In the reactive *Dynamic Source Routing* routing protocol (DSR) [6], route information is maintained in a *Route Cache*. If a destination can not be found in there, the Route Discovery protocol is initiated. All possible routes to a target are contained in the Route Cache in order to dynamically replace them if one link breaks, and not having to initiate a Route Discovery immediately. All routes have to be maintained, a task taken over by the Route Maintenance protocol. The route of a specific packet is carried in its header. This provides control over its path, for example ensuring loop prevention.

6.1. Route Discovery

The Route Discovery is initiated by the node willing to send data (initiator). It broadcasts a *Route Request* to all nodes that are in range. Every Route Request contains the following information: the initiator and the target of the Route Request, a unique identifier that is set by the initiator, and a record of the nodes through which this Route Request has already passed.

If the receiver of such a Route Request is the intended target, it generates a *Route Reply* that contains the record of nodes through which the Route Request has passed. Upon receiving the Route Reply, the initiator of the request stores the information in its Route cache and can now send packets to the target.

If the receiver is not the intended target, it compares the identifier and initator of the Route Request with that of Route Requests it has recently seen. If it finds a match, or if the record of nodes already passed contains this node, the Request is discarded silently. The node then looks up its Route Cache for a route to the target. If it finds one, it generates a Route Reply itself, and adds the route from its Route Cache to the record of the Route Request. Here, it has to eliminate possible loops. For example, if the record shows the route A→B→C with destination E and node C has cached the route C→B→D→E, the loop B→C→B is eliminated to generate a Route Reply with the route A→B→D→E. Otherwise, the receiver adds itself to the record, and then broadcasts the Route Request to be processed further.

A Route Request generally has a hop limit that is used to limit the number of nodes allowed to forward the request. This can be used to limit the spread of a request through the network. If a request with a low hop limit does not find its target, the hop limit is increased and the Route Reply is sent again.

The Route Reply can be sent back in two different ways: The target can use its own Route Cache and utilize a route stored from it. If a route to the initiator is not present, the target has to initiate a Route Discovery to the initiator himself. To avoid infinite Route Discoveries, however, the target has to store the Route Reply to the initator in its Route Request. The other way is to simply reverse the sequence of nodes of the Route Request and use this as the source route of the Route Reply.

Routes can also be discovered by transmitting nodes that overhear packets that are sent over an unknown route. However, one has to keep in mind that in a network containing mainly unidirectional links, routes can only be cached from the current route forward. If a network contains mainly bidirectional links, routes should be cached in both directions.

While the Route Discovery is ongoing, the initiating packets must be stored in a buffer. Every node therefore maintains a ‘Send buffer’ that contains every packet that could not yet be transmitted because there is no route available. Each packet is tagged with its arrival time and is discarded after a specific amount of time passes. If the buffer is overflowing nevertheless, the packets will have to be deleted with some replacement strategy, like FIFO. While a packet resides in the Send Buffer, its node should, at a limited rate, initiate new Route Discoveries. The initiation should be limited as the target node could not be available at the moment.

6.2. Route Maintenance

Each node is responsible to ensure that the link to its next hop is working. It can do so by the following means:

- The acknowledgement is an existing part of the MAC layer in use (e.g. link-layer acknowledgement defined in IEEE 802.11 [8]).
- Passive Acknowledgement: the node overhears its next hop send the package itself to another node.
- Explicit request: the node can issue an explicit request to receive a software acknowledgement by the next hop, either directly or via a different route. Upon receiving an acknowledgement from the next hop, the node may not require another for a specific amount of time. The requesting node issues a request a specific number of times, before it assumes the link as broken and sends a *Route Error* message to the sender of the packet. The node receiving a Route Error should update its Route Cache accordingly and use a different route, or initiate a Route Discovery for the target node.

7. Comparison

Table 3 depicts the performance metrics Availability of Routes, Delay of Route Acquisition, Control Overhead, Required Bandwidth, Memory Requirements and Scalability, that will now be inspected further.

TABLE 3: Pro- and reactive routing compared

	AoR	DoRA	CO	RB	MR	S
Proactive	o	+	–	–	–	–
Reactive	o	–	+	+	+	o

+: efficient, –: non-efficient, o: context-dependent

Availability of Routes – The availability of routes is one of the major differences between the proactive and reactive routing: In proactive routing, the routes are always available, whereas in reactive routing they have to be determined when needed.

Delay of Route Acquisition – As all the routes are already known in proactive routing, the delay is usually low, while in reactive protocols, the delay when acquiring a new route might larger.

Control Overhead – As proactive networks maintain their routes proactively, their control overhead is considerably higher than reactive ones.

Required Bandwidth – A consequence of the large control overhead is that the required bandwidth for proactive protocols is higher than for reactive protocols.

Memory Requirements – That depends on the routes kept in the routing cache using a reactive protocol, but it is normally lower than in proactive protocols. Specifically, the DSDV protocol has to maintain two tables of routing information alone, while the DSR protocol only has one. It can be smaller than that, too, like in the reactive AODV [9] protocol, where it is not possible to have two routes to the same destination, also they expire after a certain amount of time and are discarded.

Scalability – Proactive routing protocols are not very suited for large networks, as every node needs to keep an entry for every node in the network in their tables. In larger networks, the overhead of the reactive DSR protocol can also increase significantly, as it keeps the route in the header of each of its packets. Protocols like AODV only have their target in the header of its packets, making the scalability to bigger networks much better for reactive protocols.

Yuxia Bai et. al. [5] have carried out simulations between the DSDV, FSR, AODV and DSR algorithms, that showed that both proactive and reactive protocols are very competitive. However, each protocol has its unique weakness. They looked at throughput, packet delivery ratio, and average end to end delay. Concerning throughput, the DSR algorithm was the worst performer, while, with increasing network size, it became the best performer concerning the packet delivery ratio. The AODV protocol has been proven to be the best choice for throughput and average end to end delay as the network size increases, while the perfomance of the other protocols worsened.

They concluded that, while in a small network the proactive algorithms are competitive, as the network grows larger, the reactive algorithms become the dominant players.

8. Conclusion and Future Work

In this paper, we provided an overview of the different *MANET* routing types and several routing protocols. A comparison between these two types has been drawn, outlining their strengths and weaknesses, features and characteristics.

It has been shown that, while there is no ‘perfect’ protocol for every circumstance, there are different protocols best suited for different requirements. However, it also became clear that reactive routing protocols are better suited for managing bigger *MANET*s.

In the future, *MANET*s will have an increasingly big influence on mobile computing. A possible next step is the integration of *MANET* as a common expansion of wireless networks and fixed network architectures, a move that could see benefits like less centralized routing, resulting in higher speed, more mobility, and cheaper maintenance costs.

References

[1] https://techterms.com/definition/adhocnetwork, [Online; accessed 15-March-2020].

[2] S. Corson and J. Macker, *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*, January 1999.

[3] G. Jayakumar and G. Gopinath, “Ad hoc mobile wireless networks routing protocols - a review,” *Journal of Computer Science*, vol. 3, no. 8, pp. 574–582, 2007.

[4] A. P. Shima Mohseni, Rosilah Hassan and R. Razali, “Comparative review study of reactive and proactive routing protocols in MANETs,” 2010.

[5] Y. M. Yuxia Bai and N. Wang, “Performance comparison and evaluation of the proactive and reactive routing protocols for MANETs,” 2017.

[6] D. A. M. David B. Johnson and Y.-C. Hu, “The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR),” 19th July 2004.

[7] C. Perkins and P. Bhagwat, “Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers,” pp. 234–242, 1994.

[8] IEEE Computer Society LAN MAN Standards Committee, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11,” 1997.

[9] E. B.-R. C. Perkins and S. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*, July 2003.

Graph Databases for TLS Scans

Johannes Kunz, Markus Sosnowski*, Patrick Sattler†
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: johannes.f.kunz@tum.de, sosnowski@net.in.tum.de, sattler@net.in.tum.de

***Abstract*—In the analysis of big-data, graph databases get a lot of attention due to their capability of dealing with large, unstructured, and rich data. The existing graph storage and analysis systems have different strengths and weaknesses depending on the use-case.**

In this paper, we will examine the three popular graph databases Neo4j, Apache Giraph, and MsGraphEngine in the context of storing and analyzing TLS scans. We will compare their storage architecture and graph processing capabilities to conclude that MsGraphEngine and Apache Giraph are better suited for our large-scale data analysis than Neo4j, where MsGraphEngine can be best adapted to our needs.

***Index Terms*—graph database, tls scan, big data, olap**

1. Introduction

In the field of graph processing, traditional database systems such as Relational Database Management Systems (RDBMS) and diverse NoSQL stores face problems due to the irregularity, richness, size, and the structure of the data [1]. A plethora of new databases called Graph Databases (graph DBs) have been developed, thus, specialized on storing and processing graphs. They can help reveal new information in the data by efficiently applying graph algorithms.

Graph DBs differ in data architecture and access, data distribution, query language and execution, and support for different transactions; see Section 2. Hence, the preferable graph DB is specific to the use-case. In this paper, we analyze and compare three popular graph DBs, namely *Neo4j*, *Apache Giraph*, and *MsGraphEngine* in the context of processing Transport Layer Security (TLS) scans. TLS is a widespread security protocol on the internet.

In Section 2 we explain the fundamentals of graph databases. We then define our problem and derive requirements on the Graph DB in Section 3. Finally, we analyze our candidates and compare them according to our requirements in Sections 5 and 6, respectively.

2. What are Graph Databases?

Graph Databases are specialized databases for handling graphs. They can cope with data-inherent properties that other DBs struggle with. In contrast to tabular data, graphs have an irregular structure, e.g. different incoming and outgoing numbers of edges per node [2]. Often, nodes and edges contain rich data such as labels and properties, which are inconsistent in size. Moreover, graphs can grow large in size, as we will see in Section 3.2, while modifications are made and information is ever-changing. On top, graph algorithms require irregular access to nodes and edges and, therefore, a low latency [1]. Ultimately, Graph DBs are designed to efficiently run graph algorithms.

Due to the requirements, a variety of different approaches have emerged, based on which graph databases can be distinguished. In the following, we will briefly explain these key concepts.

2.1. Graph Models

There exist three graph models, of which variants are implemented by graph DBs. The most common is the *Labeled Property Graph (LPG)* model [1]. The *Resource Description Framework (RDF)* and the *Hyper Graph* model are used by fewer databases and do not concern the DBs discussed here.

A Property Graph is defined as the tuple $(N, E, \rho, \lambda, \sigma)$ [3]. Let N and E denote finite sets of nodes (also called vertices) and edges such that $N \cap E = \emptyset$. Then, the function $\rho : E \to (N \times N)$ maps the edges to their corresponding start and end node. Additionally, the graph contains rich data associated to nodes and edges. Such data can be labels and properties, defined by the functions $\lambda : (N \cup E) \to \mathcal{P}^+(L)$ and $\sigma : (N \cup E) \times P \to \mathcal{P}^+(V)$, respectively. Here, L, P, and V are sets of labels, property names, and property values, respectively. The operator $\mathcal{P}^+(\cdot)$ denotes the power set excluding the empty set.

2.2. Storage Architectures

The storage architecture determines the efficiency and scalability of graph operations. Its index structures must provide an efficient way of querying the elements of the graph, while maintaining modifiability when it scales [1].

Some Graph DBs are based on more fundamental databases such as *key-value stores* or *wide-column stores*; see [1]. *Native* graph DBs use specially adapted storage architectures for graphs.

2.3. Types of Transactions

There are two types of transaction which a graph DB can be optimized for, namely Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP).

With OLTP, the user performs many smaller transactional queries that are processed interactively on the client for more in-depth analysis. This requires a low latency for transactions. The queries are local in nature, i.e., do not affect the graph on a global scale [1]. The graph database mainly acts as a storage system.

Conversely, OLAP is mostly executed on the server, i.e. the server performs exhaustive graph algorithms, while the client awaits the result. The requests are fewer, but often span the whole graph. The main goal is a high throughput, which is often achieved by a high degree of parallelization on the server [1].

2.4. ACID

A database complies with *ACID* if it ensures the following properties. Transactions are always performed completely or not at all (Atomicity). Data always remains consistent, i.e., a completed transaction always produces valid output (Consistency). Concurrent transactions cannot see intermediate results (Isolation). The result of completed transactions persists (Durability) [4]. This also applies to graph DBs.

3. Problem Statement

This section describes the dataset that needs to be processed and the resulting requirements on the database.

3.1. Input Data

We use the TLS scanner *goscanner*, see [5], to scan the internet for servers using the TLS protocol. The output are *.csv* tables containing information such as used certificates, ports, and protocols about all servers. The servers' IP is a unique identifier. An exemplary scan is shown in Table 1.

TABLE 1: Exemplary TLS scan

Host	Port	Server Name	Protocol
2a00:1450:4001:81f::200e	443	google.com	TLSv1.2
172.217.22.78	443	google.com	TLSv1.3
192.30.253.113	443	github.com	TLSv1.3

We want to store this table together with additional information as a directed graph connecting servers, server properties, domain names, certificates, and certificate authorities. We use labeled edges to express relations. An exemplary graph is shown in Figure 1.

In the future, scanners may gather even more information, such as more server properties or direct relations between servers. This input data structure is therefore not fixed, but rather the current state.

3.2. Requirements on the Graph Database

In the following, we derive a list of requirements on the graph DB based on the properties of our dataset.

3.2.1. Scalability. TLS scans easily reach sizes up to 200 GB with over 400M entries. Thus, the graph DB must be able to efficiently deal with large numbers of nodes and edges, while the data quantity per node is comparatively low (limited to mostly the node name and a few properties).

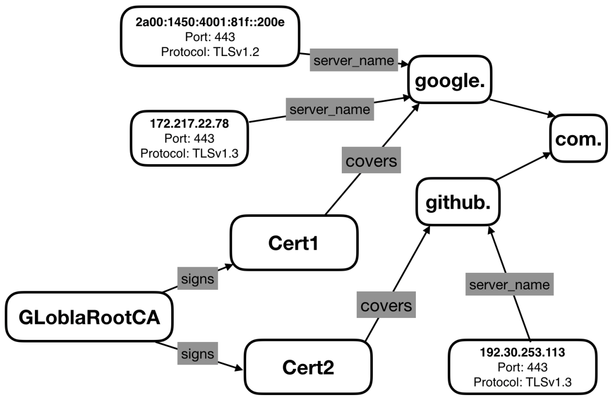


Figure 1: Exemplary graph structure of a TLS scan corresponding to Table 1

3.2.2. Fast Loading. TLS scans are performed on a frequent basis. Once a new scan is ready, the old one is obsolete. Thus, the most recent scan shall be inserted into the graph DB as quickly as possible to have more time for analysis. Due to the huge data size, the insertion speed is a relevant parameter.

3.2.3. Adaptability. As explained above, our input data structure is not fixed. If we want to focus on different aspects in our analysis, we may need to change the graph structure or add additional information. Thus, the graph DB must to adapt to our needs.

3.2.4. OLTP and OLAP. As we focus on graph analysis, OLAP will be more suited for us than OLTP. Due to the scale of the dataset, we need a system designed for high throughput.

However, the support of OLTP could be useful as the client requests the server for information via an API. Thus, the graph algorithm running on the client can be written in the programming language we prefer. Also, it requires less computational power of the server and is good for locally inspecting the data, e.g., getting information about one specific TLS server.

4. Related work

There exists a variety of surveys on graph DBs, each specialized on different aspects. In 2019, Besta et al. [1] published an exhaustive survey covering general design, data models and organization, data distribution, and transactions and queries of graph DBs. Over 40 databases were compared and classified accordingly. Focusing on scalability and performance, Barpis and Kolovos [6] investigated the performance of graph DBs compared with RDBM Systems in the context of model-driven engineering. In 2015, Kaliyar [7] published a brief overview over popular graph DBs, such as Neo4j, DEX, HyperGraphDB, and Trinity, concentrating on data modeling. A benchmark of graph algorithms performed on different graph DBs was published in 2013 by McColl et al. [2]. In 2018, Patil et al. [8] reviewed both implemented and theoretical computational techniques for graph DBs.

So far, no survey compares the above-mentioned graph databases Neo4j, Apache Giraph, and MsGraphEngine based on the requirements stated in Section 3.2, i.e, the special case of large-scale data sets with a possibly changing data structure, the need for fast loading, and OLAP.

5. Analysis of the Graph Databases

In the following, we introduce the three graph DBs Neo4j, Apache Giraph, and MsGraphEngine.

5.1. Neo4j

The database Neo4j is one of, if not the most popular Graph DB, hence, why we have chosen this DB for comparison in our paper. According to its creators, Neo4j benefits from their first-mover advantage and a thriving community. It claims to scale well, to support highly parallel graph processing and to have high loading speeds [9]. It is implemented in Java.

5.1.1. Storage Architecture. Neo4j stores edges, nodes, and properties in *records* of fixed size. These are addressable, contiguous blocks of memory. Records storing a node or an edge are called *node records* and *edge records*, respectively. Properties are stored in *property records* and can hold up to four properties. For large property values, there is an additional dynamic store [1].

These three types of records are used to store a property graph (see Section 2.1) as depicted in Figure 2. Let n1 and n2 denote two nodes, which are connected by the edge e2. Both nodes have further connections to nodes that are not explicitly depicted for simplicity.

Node records contain a reference to the edge record of the first edge that is attached to the node. In our example the record of n1 points to the record e2. It also stores labels and a pointer to a property record [1]. For administration purposes, the node record keeps flags, which are omitted in the figure.

Edge records contain pointer to the node records of the start and end node. It stores one label and a pointer to a property record. Each edge record belongs to the adjacency lists (AL) of the start and end node. These adjacency lists are implemented as doubly linked lists [1]. Hence, an edge record also stores forward and back pointers for both ALs. Due to these linked lists, Neo4j supports *index-free* adjacency., i.e., no special index structure is required for querying the adjacencies of a node.

Due to the index-free adjacency, Neo4j deals well with large graph sizes. No index structure needs to be kept up-to-date.

The storage is *disk-based*, meaning that not the entire graph is kept in the RAM [10].

5.1.2. Data Distribution and Types of Transactions. Neo4j does not support distributed data across servers. The data may be replicated but cannot be segmented [1].

It fully supports ACID in all transactions [10] and can be used for both OLAP and OLTP [1].

5.2. Apache Giraph

Apache Giraph is an open-source implementation of Google's Graph Analysis System Pregel. It was famously

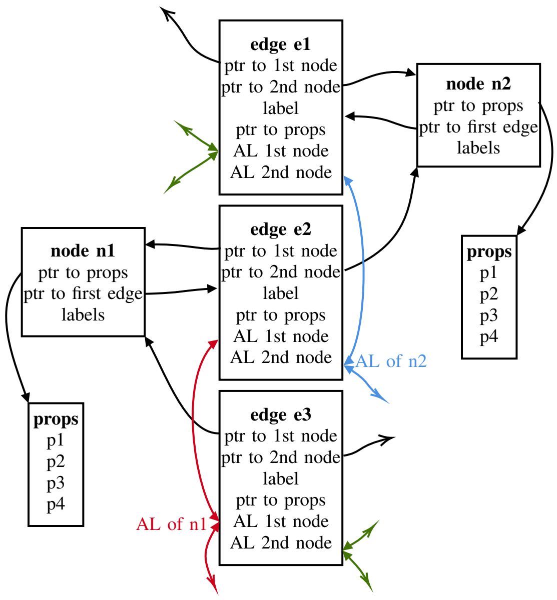


Figure 2: Storage architecture of Neo4j

used by Facebook to process a large-scale Graph with a Billion nodes and more than a Trillion edges within minutes [11], making it an interesting candidate for us.

Giraph is much more than just a Graph storage system. It was designed to perform exhaustive graph analysis on distributed servers and to interface with a variety of services in Apache's framework *Hadoop*; see [12].

5.2.1. Distributed Computing. The system Hadoop by Apache is the basis for Giraph's capability to deal with big data. It is an open-source implementation of the *MapReduce* programming model developed by Google. Its core idea is to split the computation in a *Map* and a *Reduce* function that both accept and generate key/value pairs. In the first step, Map generates a set of intermediate values with intermediate keys. The intermediate values are grouped by the intermediate keys. In the second step, the Reduce function reduces each group to a more simple value. Both steps can be parallelized on multiple machines [13]. Giraph uses a variant of MapReduce, also called *map-only* [12], meaning that there are no Reduce steps between Maps.

However, graph algorithms are often hard to parallelize. Therefore, the algorithms are executed in *super-steps*. Each super-step is parallelized on the machines. Once the machines have finished the super-step, the next one is started. This is also known as the *Bulk Synchronous Parallel* programming model. The programming is *vertex based*, meaning that nodes are the fundamental entities of processing. In between two super-steps, nodes can send messages to other nodes [10].

Hadoop manages the exchange of messages, code, and other information between the many workers and the master that controls the computation.

5.2.2. Storage Architecture. Nodes, edges, and messages are stored as Java objects and are serialized to byte arrays. To avoid memory management issues and to minimize Java's expensive garbage collection, Giraph performs parts of the memory management on its own [12].

Nodes are stored in objects instantiated by the class *Vertex*. Each vertex object has a unique ID, a value (a generic), and a set of outgoing edges; see [14].

The outgoing edges of a node are stored in an object that can be an instance of different classes implementing the *OutEdges* interface. According to the use-case, the user can choose between different implementations. In case one wants to efficiently iterate over all outgoing edges, a byte array or a linked list may be most efficient. For random access, on the other hand, a hash map for indexing the edges may be more suitable. The user is also allowed to create own implementations of the OutEdges interface [12].

Objects of the class *Edge* store a pointer to the target node and a value that is a generic.

Without different specification, Giraph stores the entire graph in the RAM of the distributed machines and is, thus, *memory-based* [15].

5.2.3. Types of Transactions. Due to its nature, Giraph supports OLAP but is not suited for OLTP. It is optimized for graph analysis rather than for storing graphs. Hence, there is no point in investigating the compliancy with ACID.

5.3. MsGraphEngine (Trinity)

Microsoft's Graph Engine Trinity is a distributed graph processing framework based on a globally addressable key-value store. It can be customized using the Trinity Specification Language (TSL) [16].

5.3.1. Storage Architecture. The core element of Trinity is a key-value store, where values are addressable system-wide across several machines. The values (called *cells*) are binary blobs of variable size that can hold arbitrary data. Their usage is specified via the Trinity Specification Language. Thus, the user can customize the graph schema and adapt to the needs. Cells may hold node or edge data or associated rich data [17]. A communication framework passes messages between machines, which can also be adjusted with TSL.

The key/value store is partitioned in trunks of fixed size with individual hash tables for addressing. Each machine keeps multiple trunks. The entire store is loaded in the RAM [17].

5.3.2. Distributed Query Execution. In Trinity, there exist three types of workers: slaves, proxies, and a client. Slaves store trunks and process and answer incoming messages. Proxies only deal with messages and do not store graph data. They can be used for gathering and relaying results from slaves to the client. The client is a worker that interfaces with the user [17].

The programming model is vertex centric. As with Giraph, consecutive super-steps are performed. At each step, a node receives messages from a fixed set of nodes, which have been sent in the previous super-step [17].

5.3.3. Types of Transactions. There is no inherent mechanism ensuring ACID. However, Trinity provides measures such as spin locks for each storage cell to guarantee consistency [17].

It can be used for both OLAP and OLTP.

6. Comparison of the Graph Databases

In this section, we compare our candidates based upon our requirements.

Scalability. All three graph DB discussed above are scalable in size in terms of their graph model implementation. There are no index structures that would become much more inefficient with growing size. However, with bigger the graph size, more storage space and computational power is needed. This is where Giraph and Trinity outperform Neo4j. They are specifically designed to run distributed graph analysis taking advantage of the RAM memory and the computing power of several machines. Neo4j can run distributed, but with copies of the data set, improving availability rather than storage capacity. It uses disc memory, which is cheaper and persistent but not as fast as the in-memory storage of Giraph and Trinity. The performance of the latter is throttled by the network speed of the cluster [17].

Adaptability. Neo4j is designed for storing labeled property graphs, i.e., its graph model is fixed. This can lead to both memory overhead and design constraints if the data does not suit the model. In Giraph the user has a certain freedom by choosing generic types, selecting different implementations (see OutEdges) and writing own classes. Trinity has even more degrees of freedom. Its language TSL lets the user define the entire graph schema and communication protocols. Thus, there is less storage overhead than in Giraph [17]. However, the user has to manually set up the database in TSL. Additionally, Giraph stores Java runtime-objects including their meta-data, causing storage overhead [17]. Again, Trinity or Giraph may be the better choice.

Note that Trinity has a more restrictive programming model than Giraph. Between super-steps, nodes can send messages to arbitrary nodes in Giraph, whereas the receivers are fixed in Trinity. However, this increases the efficiency of Trinity's inter-machine communication [17].

ACID, OLAP, and OLTP. Giraph and Trinity do not have built-in support for ACID, whereas Neo4j does. We can use Neo4j for both OLTP and OLAP. However, the benefit of OLAP may be limited due to the lack of processing power of a single machine. Trinity is designed to handle both use-cases well. It is a hybrid of Neo4j and Giraph, so to speak, as Giraph is only useful with OLAP.

Input Loading. Solely based on the storage architecture, it is hard to estimate, which of the three contestants will have the fastest loading time for our data set.

7. Conclusion and future work

We have compared the graph databases Neo4j, Apache Giraph, and MsGraphEngine (Trinity) in the context of storing and analyzing TLS scans. In summary, Neo4j is most suited if we want to store and locally query a smaller graph without much analysis. Conversely, Apache Giraph is not suited for storing graphs but specialized for distributed and parallel analysis of big data. Trinity serves both and can be adjusted as needed. Future work could include implementing and benchmarking graph algorithms on the three DBs to be sure about the best choice.

References

[1] M. Besta, E. K. Peter, R. Gerstenberger, M. Fischer, M. Podstawski, C. Barthels, G. Alonso, and T. Hoefler, "Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries," *ArXiv*, vol. abs/1910.09017, 2019.

[2] R. McColl, D. Ediger, J. Poovey, D. Campbell, and D. A. Bader, "A Brief Study of Open Source Graph Databases," *ArXiv*, vol. abs/1309.2675, 2013.

[3] R. Angles, "The Property Graph Database Model," in *AMW*, 2018.

[4] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Comput. Surv.*, vol. 15, no. 4, p. 287–317, Dec. 1983.

[5] O. Gasser and P. Sattler, "goscanner," https://github.com/tumi8/goscanner, 2020, [Online; accessed 03-May-2020].

[6] K. Barmpis and D. S. Kolovos, "Evaluation of Contemporary Graph Databases for Efficient Persistence of Large-Scale Models," *Journal of Object Technology*, vol. 13, pp. 3: 1–26, 2014.

[7] R. k. Kaliyar, "Graph databases: A survey," *International Conference on Computing, Communication & Automation*, 2015.

[8] N. Patil, P. Kiran, N. Kavya, and K. Patel, "A Survey on Graph Database Management Techniques for Huge Unstructured Data," *International Journal of Electrical and Computer Engineering*, vol. 81, pp. 1140–1149, 04 2018.

[9] N. Inc., "Top Ten Reasons for Choosing Neo4j," https://neo4j.com/top-ten-reasons/, 2020, [Online; accessed 20-March-2020].

[10] S. Sakr, F. Orakzai, I. Abdelaziz, and Z. Khayyat, *Large-Scale Graph Processing Using Apache Giraph*. Springer International Publishing, 2017.

[11] Facebook, "Scaling Apache Giraph to a trillion edges," https://www.facebook.com/notes/facebook-engineering/scaling-apache-giraph-to-a-trillion-edges/10151617006153920/, 2013, [Online; accessed 04-May-2020].

[12] C. Martella, *Practical graph analytics with Apache Giraph*. Apress, 2015.

[13] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, 2004, pp. 137–150.

[14] T. A. S. Foundation, "Apache Giraph Parent 1.3.0-SNAPSHOT API," http://giraph.apache.org/apidocs, 2019, [Online; accessed 21-March-2020].

[15] S. Heidari, Y. Simmhan, R. Calheiros, and R. Buyya, "Scalable Graph Processing Frameworks: A Taxonomy and Open Challenges," *ACM Computing Surveys*, vol. 51, pp. 1–53, 06 2018.

[16] Microsoft, "Graph Engine," https://www.graphengine.io/, 2017, [Online; accessed 21-March-2020].

[17] B. Shao, H. Wang, and Y. Li, "Trinity: A Distributed Graph Engine on a Memory Cloud," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13. New York, NY, USA: ACM, 2013, pp. 505–516.

Session: Secure Messenger with additional Measures for Metadata Protection

Johannes Martin Löbbecke, Holger Kinkelin*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: loebbeckejohannes@hotmail.de, kinkelin@net.in.tum.de

***Abstract*—With the widespread use of end-to-end encryption in popular messengers, the security of direct messages has improved significantly over recent years. However, due to the lacking protection of metadata, like a user's IP address and contact list, the privacy of users is not completely protected. The open-source application Session aims to provide the features expected from a modern chat application while implementing additional measures to protect the user's metadata. This is achieved by using public keys as an identifier and operating on a decentralized anonymous network to send end-to-end encrypted messages. This paper presents the techniques and approaches that Session uses to provide additional privacy protection in comparison to other popular messengers.**

***Index Terms*—privacy protection, end-to-end encryption (E2EE), metadata, decentralised, anonymous network**

1. Introduction

Privacy protection in communication software has been gaining increasing awareness since the revelations of widespread mass surveillance in 2013. Since then, many new messenger applications with additional security measures have been developed and old applications have implemented security protocols to protect their user's privacy. In particular the use of *end-to-end encryption (E2EE)* protects the contents of messages.

However, while E2EE secures the contents of the messages it does not completely protect the user's privacy from attacks that target metadata that can be directly connected to a person like IP addresses and phone numbers as well as quantity and time of messages. Especially protocols like VoIP are susceptible to tracking and timing attacks, even if they are encrypted and transmitted over anonymity networks.

Session is an open-source messenger application, that attempts to provide further protection against these and other issues while providing all the common features of popular chat services like group chats, video calls, and multi-device access. This is achieved by using existing protocols, like the Signal protocol, and operating on Service Nodes in a decentralized permissionless Network.

This paper will present the central protocols and techniques that Session is built upon in Section 2, give an overview of the measures in current messenger applications in Section 3, and then explain details of Sessions implementation in the Sections 4 and 5. If not otherwise cited, the Session implementation details are based on the Session white paper found at [1] as well as an informal conversation with Session developers. The economic viability of the Loki Blockchain, that Session is based upon, has been formally analyzed using game theory by Brendan Markey-Towler [2] and further details regarding the staking requirement can be found in the research by Johnathan Ross et al. [3].

2. Background

To develop a secure Messenger with additional privacy protection measures, basic systems to protect the content of messages as well as hide the IP addresses during routing are needed. The protocols, techniques, and network presented in this section have been formally analyzed and tested and can be used as a strong foundation in any secure messaging service.

2.1. Signal Protocol

The Signal Protocol (originally known as TextSecure Protocol) is a non-federated open-source cryptographic protocol for E2EE. It can be used to protect the contents of messages, voice calls as well as video calls while guaranteeing important like properties of deniable authentication, perfect forward secrecy, integrity, destination validity, and others. In particular, the properties perfect forward secrecy and deniable authentication guarantee, that every time a new session is created, a new key is generated for that particular communication session, which means that a single compromised communication session key, does not compromise the content of past and future conversations. Furthermore, deniable authentication, ensures that a sender can authenticate messages for a receiver, such that the receiver cannot prove to a third party that the authentication ever took place.The Signal Protocol uses the well established elliptic curve known as Curve25519 as a basis for its key generation and the key exchange protocol known as Elliptic-curve Diffie-Hellman. These keys are often called X25519 keys. [4]–[6]

Since the code for the Signal Protocol is open-source, it has been formally analyzed and found to be cryptographically sound [6], which has further increased its widespread use in several chat applications (see Table 1).

2.2. Onion routing

Onion routing is a well-reviewed technique for protecting metadata during network communication. The message that is being transmitted is protected by multiple layers of encryption, akin to the layers of an routing. Every hop removes a layer of encryption and therefore knows only the next hop, while the actual sender and receiver stay anonymous, since every intermediate node only knows the immediately preceding and following nodes. However this technique does not protect against traffic analysis and the exit node can be a single point of failure. The weaknesses of onion routing have been analyzed multiple times, especially regarding TOR (See Section 2.3). [7], [8]

2.3. TOR

TOR, derived from the name of the original project "The onion Router", is a free and open-source server application licensed under the BSD 3-clause license. TOR is an altruistic network, that allows users to protect their privacy through onion routing Protocols running on volunteer-operated servers. Similarly, TOR can circumvent censorship, by providing access to otherwise blocked destinations and content. TOR has been both extensively peer-reviewed as well as analyzed in formal studies, and can, therefore, be a good inspiration or building block for further software. [7]–[9]

3. Overview of popular secure messenger applications

This section presents an overview of popular messenger applications that use different forms of E2EE encryption to provide certain levels of privacy protection.

3.1. WhatsApp

At 1.600 Million monthly active users, as of October 2019 [10], WhatsApp is the most popular Messenger application. It provides users with the expected features in sending messages, creating group-chats, and sharing different forms of media like pictures and videos.

Messages on WhatsApp are encrypted using the Signal Protocol presented in Section 2.1. WhatsApp is closed-source and has not allowed independent code audits, which makes formal analysis and testing difficult, therefore raising privacy concerns.

3.2. Signal

Signal is an open-source messaging service developed by the Signal Foundation and Signal Messenger LLC. Unlike commercial software like WhatsApp, Signal relies on donations and support for its commercial viability.

Signal uses the Signal Protocol as presented in Section 2.1 and provides further privacy protection by encrypting the sender's information in the message, and only storing the login dates, rather than more detailed information. While providing significant protection, Signal still has several vulnerabilities. In particular through the use of a phone number as identifier for the users as well as centralized servers. (see Section 5.2) [11], [12]

TABLE 1: Overview

Name	**E2EE**	**reviewed**	**‘Identification**
WhatsApp	Signal Protocol	No	Phone Number
Signal	Signal Protocol	Yes	Phone Number
Telegram	MTProto	partly	Phone Number

reviewed applications have allowed independent code audits and have been formally tested

3.3. Telegram

Telegram, unlike Signal, is only in part open-source, with its client-side code being open to the public, while the server-side code is closed-source. While all server-client communication is in default encrypted, E2EE is only optional for messaging and video chat between users. [13]

Telegram has received notable criticism, for using its untested E2EE algorithm known as MTProto and making its use optional between users, as well as storing critical information like contacts, messages, and media on their centralized Server. [14], [15]

4. Session Basics

This section presents the basic functions and building blocks of the Session messenger.

4.1. Protections

The goal of Session's design is to provide the following protections to its users.

- **Sender Anonymity:** Personal identity of the Sender is only known to the recipients of the messages, while their IP address is only known to the first hop in the onion routing path.
- **Recipient Anonymity:** The IP address of the recipient is only known to the first hop in the onion routing path.
- **Data Integrity:** Session ensures the integrity of the messages both regarding modification and corruption. Messages where this integrity is violated, are discarded.
- **Storage:** For the duration of their *time to live(TTL)*, messages are cached and delivered to clients.
- **E2EE:** Messages maintain the properties of Deniable Authentication and Perfect Forward Secrecy.

4.2. Incentivized Service Nodes

In a centralized Network the central authority is always a single point of failure, which is why, with the rise of Bitcoin, decentralized Networks have become increasingly popular, with projects exploring applications in different fields. In a permissionless network new users can join at any time and provide additional nodes. [16]

However, many of these projects have struggled with similar problems like overloaded servers, as well as security concerns regarding attacks like Sybill attacks. Here an attacker creates multiple anonymous nodes and can, therefore, use traffic analysis to deanonymize users and access private data. [7], [17] Session aims to prevent these problems by incentivizing good node behavior and creating a financial precondition for their *Loki Service Network*. This network integrates a blockchain and therefore requires anyone wishing to host a server for Session, to go through a staking transaction, during which an operator has to lock an amount of cryptocurrency assigned to the node.(equivalent value of 7.420 USD as of 10/02/2020) [18]

This ensures, that whenever a buyer decides to run a new Service Node, the supply of Loki is decreased, since it is locked during the staking mechanism and therefore removed from the market. This means that the financial resources to acquire enough Service Nodes for a Sybill attack are significant and increase exponentially with the scale of the attack.

Furthermore, the use of a blockchain integrated network allows for a reward system, where whenever a new block is mined, the service node is paid with a part of the block reward. This incentive system has been formally analyzed using game theory by Brendan Markey-Towler. [2] Combined with a consensus-based testing suite, it ensures a high standard of operation and honest node behavior, while being an alternative approach to altruistic networks like TOR or I2P.

4.3. Onion Requests

Service Nodes provide Session with access to a distributed network with a high standard of operation, but to transmit messages while protecting the user's identity from third parties as well as protecting the contents of the messages it still needs encryption and message routing. Session, therefore, uses an onion routing Protocol(See Section 2.2) referred to as *Onion Requests*. The purpose of Onion Requests is to further protect the IP addresses of Session users, by creating randomized three-hop paths through the Service Node Network. For this purpose, every Session client has access to a Service Node list, containing the IP address of each Service Node, as well as the corresponding storage server ports and X25519 keys. This list is fetched on the first launch and then kept up to date through periodic queries on multiple Service Nodes. On application startup, the Session client should use this list to choose three random nodes to establish an onion routed path. After testing whether this path creation was successful, by sending down a request and waiting for a response, the path should persist through multiple requests. In case a response is not achieved, the client will try to create a new path.

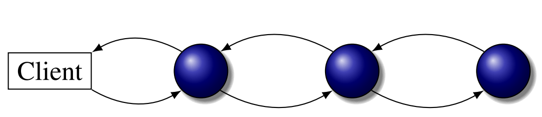


Figure 1: Onion Requests

5. Extending for Messenger app functionality

Through Onion Requests and the Loki Service Network, Session has access to an anonymous network as well as bandwidth and storage space. This section presents central services, that are built on top of the foundation, to allow Session to provide the features expected from a modern chat application.

5.1. Storage

Users of modern chat applications expect message transmission to occur both for synchronous as well as asynchronous communication. To reliably provide this service with an app running on a decentralized network, an additional storage level that provides redundancy is needed, to deal with unexpected operational problems like software bugs. For this purpose, session combines its incentivized Service Nodes with a secondary logical layer, called swarms.

These swarms are created by grouping together Service Nodes and replicating messages across them. To protect from malicious node operation, the initial swarm a node joins is determined by an algorithm that gives minimal influence to the Service Node operator. With nodes joining and leaving the network, the compositions of the swarms have to naturally change during operation:

- **Starving swarm:** In the case, that a swarm needs more nodes, it can "steal" nodes from a different swarm, which has more than its minimal amount of n needed for operation (N_{min})
- **All swarms at N_{min}:** The nodes of the starving swarm will be redistributed among all other swarms
- **Oversaturated swarms:** In the case that multiple nodes will join the network while all swarms are already at maximum capacity (N_{max}), a new swarm will be created from a random selection of N_{target} nodes, where N_{target} is defined as $N_{min}>N_{target}>N_{max}$ to ensure the new swarm is neither under nor oversaturated.

Furthermore, to ensure intended node operation, changes to swarm composition remain synchronized by pushing data records to new members and redistribution of data records by leaving nodes.

5.2. Identification

As can be seen in Table 1, most widespread messenger applications rely on a phone number or email address as an identifier. While this approach has obvious usability advantages like ease of social networking or recovery of login data, it is problematic regarding security and privacy. High-level actors(a person or group, with a lot of resources and access rights) like government institutions or service providers can compromise user accounts tied to a phone number. Furthermore, a high-level actor can in many states directly connect a phone number to more personal information like passport, social security number and more. Even a low-level actor(who has limited resources) can access databases that collect leaked data sets and use them for spoofing attacks like SIM swapping attacks. Therefore, Session instead uses X25519 public-private key pairs for identification and new key pairs can be generated within the application at virtually any time. Upon first launch a user is presented with a generated pair and encouraged to write down their long term private key for potential account recovery in case their device is lost or for other reasons out of service.

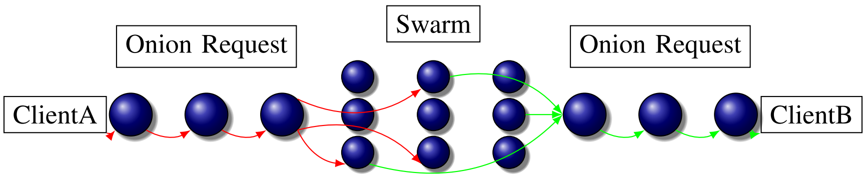


Figure 2: Asynchronous Routing

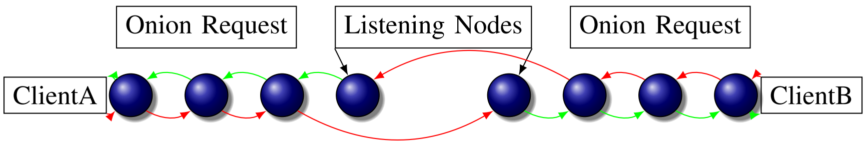


Figure 3: Synchronous Routing

5.3. Message Routing

Through the usage of swarms, as explained in Section 5.1, as well as the public X25519 keys as long-term identification, messages can now be addressed to other Session users, but the actual messages still need to be transmitted. To ensure that the user experience is similar to other modern chat applications, message routing needs to handle both synchronous and asynchronous routing.

5.3.1. Asynchronous Routing. When a user sends a message and either sender or recipient are offline as well as in the default case, Session uses asynchronous routing (Further details on determination in Section 5.3.2) as illustrated in Figure 2. For asynchronous routing the sender‘s client will use the recipients identification key (see section 5.2), to determine the recipients swarm through the deterministic mapping between Service Nodes and long-term-public keys. The sender then serializes the message using the Protocol Buffer method and packs it in a wrapper containing the recipient’s public key, a timestamp the TTL as well as the proof of work. By requiring proof of work, spam attacks become more computationally expensive and, therefore, less practical. Onion Requests create the path towards the recipient’s swarm, where the package is spread to three different Service Nodes inside the swarm. These Service Nodes then spread the message to the other Service Nodes in the swarm. By targeting multiple Service Nodes and spreading the message amongst all members of the swarm, enough redundancy is introduced, to ensure that the message will not be lost for the duration of its TTL and can, therefore, be received whenever the recipient connects to the network.

5.3.2. Synchronous Routing. Asynchronous routing is also used in the default case since the Protocol Buffer of any asynchronous message also contains the online status of the sender as well as a specified Service Node in their swarm, on which they are listening. This means a recipient’s client can then set up a synchronous routing through the specified listening node, by exposing its listening node in the network as illustrated in Figure 3. Once set up the two clients can now simultaneously use Onion Requests to send messages through the network to the conversation partner’s respective listening nodes. Since messages are not propagated through the swarms, there is no redundancy being created and no proof of work is sent. To prevent lost messages, acknowledgments are sent back after every received message. If a message times out, if for example the recipient went offline, the message is resent using the default asynchronous routing.

5.4. Modifications to the Signal Protocol

While onion routing hides the user’s IP addresses via Onion Requests, the actual content of the messages still needs to be encrypted. Session uses the Signal Protocol as described in Section 2.1 as its basis and adds some slight modifications to adapt it to a decentralized network. Additional information is added to each message, to ensure correct routing and verifying correct message creation. Furthermore the sharing of prekey bundles is instead conducted by a "friend request" system. Whenever a user first initiates communication with a new contact, a friend request will be sent. This friend request contains a short written introduction as well as the sender’s prekey bundle and metadata like the sender‘s display name and public key, which as explained in the previous sections can be used to reply to the sender. These friend requests are themselves encrypted for the recipient’s public key using the Elliptic-curve Diffie–Hellman protocol, that the Signal Protocol is based on. In the case that the recipient accepts the friend request, they can then use the prekey bundle to exchange messages, encrypted with the Signal Protocol as desired.

6. Conclusion

Surveillance on the internet is constantly increasing both by individuals as well as states and corporations. In a time where many countries and regimes are working increasingly towards blocking free communication and expression, it has become more important then ever to not just protect the contents of conversations via techniques like E2EE, but also the information if, when, and between whom the conversation took place.

By building on the Signal application with the Signal Protocol and using onion routing like in TOR, Session combines formally tested measures for security and privacy protection, with a decentralized network. The incentive structure designed around the Loki Service Network presents a commercially viable approach to anonymous networks and allows for protection against common problems like Sybill attacks and overloaded servers. By extending their systems with an additional logical layer in swarms, Session can use both asynchronous and synchronous routing, to provide the functionality expected from a modern chat application.

Further extensions that allow for the use of multi-device, attachments and group chats, were out of the scope this paper, but are described in the Session white paper as found on their GitHub page and in [1]. Some features like video chats are as of May 2020 still under development.

References

[1] K. Jefferys, M. Shishmarev, and S. Harman, "Session: A Model for End-To-End Encrypted Conversations With Minimal Metadata Leakage," 2020, [Online; accessed 25/03/2020].

[2] B. Markey-Towler, "Cryptoeconomics of the Loki network," 2018, [Online; accessed 25/03/2020].

[3] K. Jefferys, J. Ross, and S. Harman, "Loki Cryptoeconomics: Alterations to the staking requirement and emission curve." 2019, [Online; accessed 25/03/2020].

[4] M. D. Raimondo, R. Gennaro, B. Dowling, L. Garratt, and D. Stebila, "New approaches to deniable authentication." Journal of Cryptology, May 2009.

[5] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk, and T. Holz, "How secure is textsecure?" in *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, March 2016, pp. 457–472.

[6] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A formal security analysis of the signal messaging protocol," in *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, April 2017, pp. 451–466.

[7] A. Sanatinia and G. Noubir, "Honey Onions: a Framework for Characterizing and Identifying Misbehaving Tor HSDirs," [Online; accessed 22/03/2020].

[8] TorProject.org, "Toor Project: FAQ," [Online; accessed 26/03/2020].

[9] "Tor Project: Overview," https://2019.www.torproject.org/about/overview.html.en, [Online; accessed 18/03/2020].

[10] J. Clement, "Most popular global mobile messenger apps as of October 2019, based on number of monthly active users," https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/, 2019, [Online; accessed 12/03/2020].

[11] Signal.org, "Technology preview: Sealed sender for Signal," https://signal.org/blog/sealed-sender/, 2018, [Online; accessed 15/03/2020].

[12] M. Lee, "Battle of the secure messaging apps: How signal beats whatsapp," *The Intercept*, 2016.

[13] "Telegram F.A.Q," https://telegram.org/faq#q-why-not-open-source-everything, [Online; accessed 22/03/2020].

[14] "why telegrams security flaws may put irans journalists at risk," https://cpj.org/blog/2016/05/why-telegrams-security-flaws-may-put-irans-journal.php, [Online; accessed 22/03/2020].

[15] W. William Turton, "why you should stop using telegram right now," https://gizmodo.com/why-you-should-stop-using-telegram-right-now-1782557415, [Online; accessed 22/03/2020].

[16] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *2017 IEEE International Conference on Software Architecture (ICSA)*, 2017, pp. 243–252.

[17] D. McIntyre, "Ethereum Classic Gas System Economics Explained," [Online; accessed 25/03/2020].

[18] Developers, "Loki documentation, loki service node staking requirement," https://docs.loki.network/ServiceNodes/StakingRequirement/, [Online; accessed 17/03/2020].

Extending ZMap: Round-Trip Time Measurement via ICMP and TCP

Bernhard Vorhofer, Patrick Sattler*, Johannes Zirngibl*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: bernhard.vorhofer@tum.de, sattler@net.in.tum.de, zirngibl@net.in.tum.de

***Abstract*—ZMap is a high-performance network scanner for Internet-wide network surveys. Due to its modular architecture, it allows researchers to conduct a variety of different types of scans and can be extended with custom modules.**

The aim of this paper is to outline the process of how we implemented two modules for ZMap and the IPv6-enabled ZMapv6 for measuring round-trip time. The first sends ICMP echo request packets, the other is based on the existing SYN scan module and allows RTT measurement using TCP.

We give a brief overview of ZMap's implementation and discuss key architectural aspects that influenced our module design. Furthermore, we present the results of multiple tests we conducted using the new modules and compare them to an earlier study involving large scale round-trip time measurement. The experiments suggest that the implemented modules are working correctly. However, we found a discrepancy between ZMap's reported values and timestamps captured during the measurement using a network analyzer.

***Index Terms*—Network scanning, measurement, round-trip time**

1. Introduction

ZMap is a network scanner for Internet-scale network surveys. As Durumeric et al. have shown in [1], this tool is capable of scanning the entire IPv4 address space in under 45 minutes. ZMap was extended with IPv6 support along with three new modules by Gasser et al. [2], resulting in ZMapv6. The added IPv6-enabled modules allow scanning using ICMP echo request, TCP SYN or UDP messages.

In this paper, we present two new modules for ZMap and ZMapv6. One of them, the ICMP time module, has been ported from the existing IPv4 version. The other is based on the existing SYN scan module and allows RTT measurements via TCP without using the TCP Timestamps option. These modules, along with the existing ICMP time module, enable large-scale round-trip time measurements using ZMap. This can be useful in a variety of applications, for example for finding bottlenecks in large networks.

The ICMP time module provides the same functionality as the existing IPv4 version of this component, but does so using IPv6. Since the basic structure of the ICMPv6 header is identical to its version four counterpart, porting the ICMP time module to version six of the Internet Protocol was matter of refactoring the existing code. Therefore, the implementation details of this module are not described any further in this paper. However, we discuss the results of tests we performed using the ICMPv6 time module in Section 4.3.

The TCP time module accomplishes two things at once. First, it performs a standard SYN scan, determining whether a specific port is open for each target host. Second, the round-trip time is determined for each responding host by measuring the delay between the transmission of a SYN segment and the arrival of the corresponding SYN-ACK segment.

One noteworthy piece of related work is D. J. Bernstein's description of SYN cookies [3], a method of embedding information in the TCP sequence number field originally intended as a countermeasure against SYN flood attacks. ZMap uses a similar technique in its TCP time modules to include additional data in probe packets.

In the following sections, we first give a brief overview of ZMap's architecture before discussing the implementation details of the TCP time module in Section 3. Finally, we present the results of experiments we conducted in Section 4.

2. ZMap primer

This section gives a brief introduction to ZMap and its modular architecture, mostly summarized from the original paper by Durumeric et al. [1].

ZMap owes its exceptional speed at least in part to the stateless nature of its architecture – the tool was specifically designed with performance in mind. Target addresses are selected using a cyclic multiplicative group, which distributes them randomly across the IPv4 address space. This also eliminates the need for storing addresses already scanned because each address in the address space is reached exactly once.

To distinguish scan responses from background traffic, probe modules embed scan- and host-specific validation data in headers of probe packets (where possible). This allows responses to be validated on receipt without keeping additional per-connection state. ZMap generates a message authentication code (MAC) of the destination address for probe modules to use as validation data.

2.1. Architecture

ZMap's modular architecture can be split into three basic parts: the scanner core, output modules and probe modules. The scanner core handles packet transmission

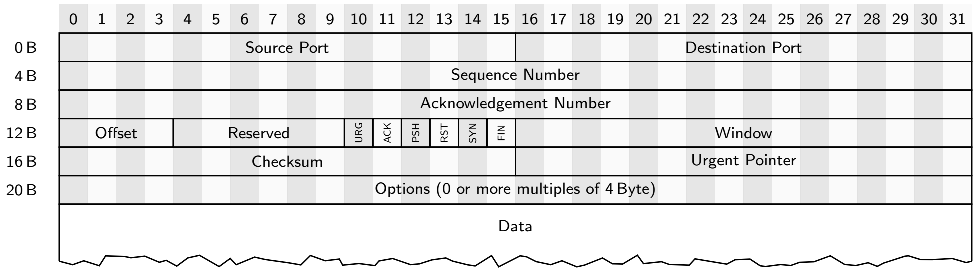


Figure 1: TCP Header

and receipt, address generation and other fundamental tasks. It also invokes probe- and output modules via callbacks, for example to generate packets before they are sent or to validate packets after they have been received. Output modules allow for multiple forms of scan output, e.g. CSV files, JSON files or writing entries to an in-memory Redis database. Since output modules are not immediately relevant for this work, we will not elaborate on the details of their implementation. Probe modules provide multiple different scan methods, for example TCP SYN scans or ICMP echo scans. Their implementation is further discussed in Section 2.3.

2.2. Transmission and reception

To avoid unnecessary kernel overhead, ZMap makes use of raw sockets for packet transmission. Consequently, probe packets are assembled on the data-link layer, which allows for the Ethernet header to be cached and reused because the source (local network interface) and destination (gateway) MAC addresses remain unchanged during a scan. Packet reception is achieved using libpcap, a C/C++ library for traffic capture. Furthermore, this library provides a first filter for incoming packets, so traffic clearly not related to the current scan is discarded before it reaches the active probe module.

2.3. Probe modules

Probe modules are responsible for constructing probe packets to be sent to and validating responses received from target hosts. The scanner core provides the active probe module with a single buffer which holds the data to be sent. Before a scan commences, the probe module populates the empty buffer with any necessary headers. Values set at this stage (e.g. source and destination MAC addresses in the Ethernet header) are not modified during the rest of the scan, as mentioned before.

For each scan target, the probe module in use fills all host-specific header fields in the buffer with data provided by the scanner core. This includes the network layer addresses as well as validation data. It is desirable to include as much of the validation string as possible in the probe packet while ensuring it is placed in header fields which allow recovery of the validation data from the probe response. The reason for this is that more validation data decreases the probability of a false positive, i.e., an incoming packet unrelated to ZMap being erroneously classified as a probe reply. After the probe module has populated the buffer, the scanner core handles transmission of the packet.

Whenever a packet is received, the scanner core calls two probe module callbacks in succession. First, the incoming message is validated, i.e., validation data is extracted from the header and compared to the expected validation string provided by the scanner core. Second, and only if validation succeeded, the received packet is processed. The primary purpose of this step is to pass values from the received packet to the active output module for formatting.

3. TCP time module

The TCP time module extends the existing TCP SYN scan module with the capability of measuring round-trip time during a partial three-way handshake. This is achieved in a manner similar to measuring RTT using ICMP: a timestamp is included in the probe packet in a way which allows it to be recovered from the response, thus allowing the time difference between transmission of the probe packet and reception of the response to be determined without keeping additional state.

In our research, we found two viable methods of measuring round-trip time using TCP. The first, which was employed for the TCP time module, utilizes the sequence number field (see Figure 1 for reference) to embed a timestamp in the TCP header. The second makes use of a TCP option intended for precisely this use case.

3.1. Embedding and recovering the timestamp

The problem we faced when first attempting to implement RTT measurement without using the TCP Timestamps option was that the sequence number field provides only four bytes of space for the timestamp, but the timestamp structure used in other modules is eight bytes in size. As a consequence of this, we decided to reduce the size of the timestamp by using a relative timestamp instead of an absolute one and decreasing its resolution from one microsecond to one millisecond.

Furthermore, since the original TCP SYN scan module utilizes both the source port and the sequence number fields for validation data, the amount of validation data included in the probe packet needed to be reduced in order to accommodate the additional timestamp data. With the sequence number field occupied by the timestamp, this leaves only the source port field available for validation data. Because we chose to only allow ports from the standard Linux ephemeral port range (32 768 to 61 000), the effective validation string length (binary logarithm of the number of available ports) is approximately 14.8 bit, even though the source port field is two bytes in size.

When the probe module is initialized before the scan starts, a timestamp is stored in a global variable which all of the relative timestamps included in probe packets use as a reference. For each probe packet, a 32 bit value representing the number of milliseconds passed since the scan started is used as the initial sequence number. In contrast to the `timeval` struct used in the ICMP time module, this timestamp provides only millisecond instead of microsecond resolution. With the information about libpcap's internal timestamp precision found in [4], we conclude that the prospect of achieving microsecond precision without employing specialized hardware is questionable at best. Coupled with the fact that most round-trip time measurement tools we have used (most notably *ping*) report results in millisecond precision, this supports our decision to reduce the timestamp resolution. When a SYN-ACK message is received, the timestamp can be recovered by subtracting 1 from the acknowledgement number field in its header.

3.2. Alternative method for RTT measurement

An alternative method of measuring round-trip time using TCP is by using the aforementioned TCP Timestamps option [5]. It was designed specifically for measuring round-trip time, among other uses. Although this would simplify the task at hand, the method we implemented in the TCP time module offers a significant advantage: it does not require any TCP extensions to be supported and enabled. Kühlewind et al. [6] tested a selection of popular web servers for support of certain TCP options and extensions. The results of their tests from 2012 show that in total, roughly 80 % of responding hosts supported the timestamp option.

Additionally, some administrators might decide to disable this option for security reasons. The reason for this is that, as Beverly et al. [7] describe, TCP timestamps can be used to estimate the uptime of a host, potentially allowing attackers to discern whether a security patch is installed or not. With the technique employed in the TCP time module, measurements can be made regardless of TCP timestamp support on the target host.

3.3. Implications of using less validation data

When reducing the amount of validation data included in the probe packet TCP header, the question of whether the remaining data is sufficient for discriminating between scan responses and background traffic arises. For this application, there is only one specific scenario we need to be concerned about: an unsolicited incoming TCP segment leading to a false positive and, consequently, an incorrect round-trip time value. More specifically, only an incoming SYN-ACK or RST message can lead to such an error, because any other type of message will be discarded by libpcap before it reaches the validation function due to the packet filter set up for the module.

Since a SYN-ACK is received only during an outgoing connection attempt in normal operation, it is safe to assume that these messages would only be received due to another program in the process of establishing a TCP connection while a scan is being performed. Outgoing TCP connections are usually assigned an ephemeral port from the same range ZMap uses for probe packets. Even though it is possible that a program is assigned the exact port that would lead to a false positive in terms of response validation, the probability of this happening should be rather low considering the large number of ports available. To further reduce the odds of such an error occurring, network activity unrelated to ZMap should be reduced on the scanning machine where possible.

Another case that should be considered is the effect a scan has on existing TCP connections. If a connection to a target host exists on the same port ZMap uses for its probe of that specific host – which is unlikely, as mentioned above – can this lead to an RST message? According to the TCP protocol specification [8], a SYN received in one of the synchronized states does not lead to a reset. Rather, an ACK message containing the next expected sequence number would be sent. Since such a packet would be discarded by the libpcap filter, existing connections do not interfere with the network scanner.

4. Test results

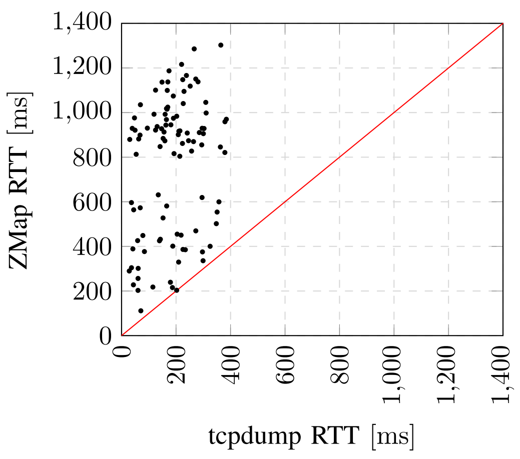


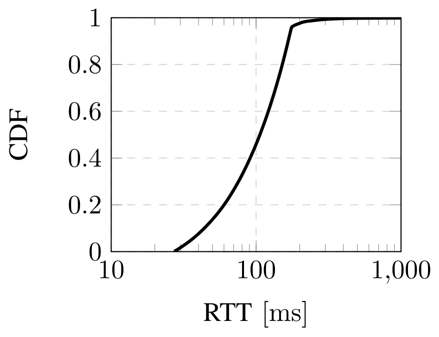
Figure 2: tcpdump vs. ZMap ICMPv4 RTT (red line represents both measurements reporting the same value)

In order to confirm the functionality of the newly implemented modules, we conducted several tests and recorded more than 600 000 responses. We compare our results to those presented by Padmanabhan et al. [9] by plotting the cumulative distribution function (CDF) of the RTT measurements.

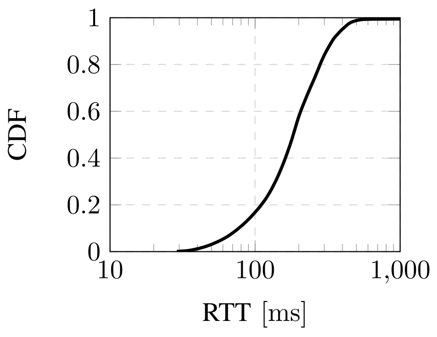
4.1. Testing methodology

Two series of ZMap measurements were conducted to verify the functionality of the implemented modules. For the TCP module test, probes were sent to randomly selected hosts and nearly 400 000 replies have been recorded. For ICMPv6, hosts from the Alexa[1] Top 1 Million web server list (which is not offered anymore at the time of writing) were probed and approximately 250 000 replies have been recorded. These two tests have been carried out from inside the university network.

1. https://www.alexa.com/topsites



(a) ICMPv6



(b) TCP (IPv4)

Figure 3: Cumulative distributions of ZMap RTT measurements

During our tests, network traffic was captured using tcpdump, an open-source network protocol analyzer. This allowed us to cross-reference the timestamps of outgoing and incoming packets as recorded by tcpdump with ZMap's round-trip time measurements. Naturally, the expected result would be the tools reporting similar values, but we found that a significant number of Zmap's measurements diverge substantially from the RTT values indicated by the captured timestamps (see Section 4.2).

4.2. Validation of RTT measurements

To assess the overall quality of ZMap's round-trip time measurements, we conducted an additional test using the existing ICMP time module. ZMap was configured to probe randomly selected hosts until 100 echo replies were collected. As mentioned before, tcpdump was capturing all network traffic throughout the test so we could compare the RTT calculated using the timestamps in the capture file with ZMap's reported values. Even though libpcap timestamps may not be the most precise [4] (tcpdump captures packets using libpcap), they suffice for a rough estimate of the actual RTT.

Figure 2 shows the results of this test. It is clear that ZMap tends to report higher RTT values than those measured by tcpdump. Furthermore, there is a considerable inconsistency in the ZMap measurements – numerous values deviate significantly from those obtained from the capture file. Closer examination of the timestamps revealed that the receive timestamps are the cause of this inaccuracy, which leads us to believe that somewhere in the receiving process, a variable amount of delay causes unreliable receive timestamps. Further investigation to find the source of this delay is out of scope for this paper.

4.3. ICMPv6 time module

For the ICMPv6 test, around 200 000 probe replies were collected. The cumulative distribution function in Figure 3a was plotted using a subsample of roughly 1000 data points. Systematic sampling was used to select the samples, i.e., the replies were ordered by round-trip time before values were selected at regular intervals to yield the desired sample size.

The function has similar characteristics to that in [9, Figure 7]. It seems that our dataset contains a larger portion of comparatively low values, though it should be noted that our sample size is several orders of magnitude smaller. Nevertheless, we believe this shows that the ICMPv6 time module is working as intended.

4.4. TCP time module

Around 400 000 replies were collected for the TCP over IPv4 test, roughly 1000 of which were selected (using systematic sampling) to plot the CDF in Figure 3b. This graph shows even more similarity to that in [9, Figure 7], with significantly more values above 200 ms. Keeping in mind the restricted sample size and the aforementioned variability of receive timestamps, we believe this shows that the values reported by the TCP time module are plausible.

5. Conclusion and future work

With the new modules we implemented for ZMap, two additional scan types now have the capability to report round-trip time measurements – ICMPv6 and TCP. This allows researchers to collect one more metric from large-scale network surveys, determining not only reachability but also response time.

During our tests, we discovered an unexpected disparity between RTT values reported by ZMap and measurements we recorded simultaneously using tcpdump. This was observed with all of the modules, including the existing ICMP time module. Before the source of this measurement error is found and eliminated, we are unable to reliably determine the precision of the implemented modules' measurements. The reason for this is that ZMap's measurements can not be directly compared to values calculated using the captured timestamps. However, we compared the distributions of round-trip time measurements to [9, Figure 7] and found that they exhibit very similar characteristics, which suggests that the modules themselves do indeed work as intended.

Other than finding and resolving the cause of the observed measurement error, possible future work includes implementing a ZMap module for RTT measurement using the TCP Timestamps option. Although we decided against employing this protocol extension for our TCP time module, it might be useful in some situations to have both methods at one's disposal.

References

[1] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide Scanning and Its Security Applications," in *22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX Association, Aug. 2013, pp. 605–620.

[2] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle, "Scanning the IPv6 Internet: Towards a Comprehensive Hitlist," *CoRR*, vol. abs/1607.05179, 2016.

[3] D. J. Bernstein, "SYN cookies." [Online]. Available: http://cr.yp.to/syncookies.html

[4] "Manpage of PCAP-TSTAMP." [Online]. Available: https://www.tcpdump.org/manpages/pcap-tstamp.7.html

[5] D. Borman, B. Braden, V. Jacobson, and R. Scheffenegger, "TCP Extensions for High Performance," Internet Requests for Comments, RFC Editor, RFC 7323, September 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7323.txt

[6] M. Kühlewind, S. Neuner, and B. Trammell, "On the State of ECN and TCP Options on the Internet," in *Passive and Active Measurement*, M. Roughan and R. Chang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 135–144.

[7] R. Beverly, M. Luckie, L. Mosley, and K. Claffy, "Measuring and Characterizing IPv6 Router Availability," in *Passive and Active Measurement*, J. Mirkovic and Y. Liu, Eds. Cham: Springer International Publishing, 2015, pp. 123–135.

[8] J. Postel, "Transmission Control Protocol," Internet Requests for Comments, RFC Editor, RFC 793, September 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc793.txt

[9] R. Padmanabhan, P. Owen, A. Schulman, and N. Spring, "Timeouts: Beware Surprisingly High Delay," in *Proceedings of the 2015 Internet Measurement Conference*. Association for Computing Machinery, 2015, pp. 303–316.

User Localization in 5G Mobile Networks

Simone Zügner, Marton Kajo*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: ge56ced@mytum.de, kajo@net.in.tum.de

***Abstract*—5G technology enables accurate user location within centimeter range, which has led to the development of a multitude of new use cases. At first, this paper presents several use cases, like emergency response, transport and indoor navigation. Then the fundamental localization techniques proximity, trilateration and fingerprinting are presented. After that, three different categories of localization architectures are explained. Furthermore, user localization in 4G and 5G cellular networks is explored. 5G user localization benefits from increased bandwidth, smaller cells, device-to-device communication and multipath-assisted algorithms. Finally, open research topics like machine learning, heterogeneous networking protocols and beamforming schemes are discussed.**

***Index Terms*—User Localization, 5G, Context-Awareness, Tracking, Navigation, Positioning, Localization Architecture**

1. Introduction

The increasing ubiquity of the 5G network stirrs expectations towards user location estimation. Positioning with 5G has many benefits such as high coverage, high accuracy, low latency, low energy requirements and scalability. Historically, the main reason for the standardization of localization services in cellular networks were emergency calls (see 2.1). The standardization group 3rd Generation Partnership Project (3GPP) was formed as a worldwide organisation that develops protocols for mobile telephony [1]. Release 9 of the 3GPP was the first release to contain positioning protocols requiring network operators get the accurate position of emergency callers [2]. It was released in 2009. Further releases have improved the user localization. Release 17, which is scheduled for delivery in 2021, will include 5G Core Location Services [3]. 5G will improve accuracy through enhancements like high density of base stations, high signal bandwidths, device-to-device communication and millimeter wavelength technology [4] (see 6).

2. Use Cases

Different categorizations of use cases are possible: Bartoletti et al. [2] mention four different categories of use cases: regulatory and mission critical, location-based services, industry and eHealth, and transport. Laoudias et al. [5] choose the use case categories consumer, networking, industrial, health care, public safety and emergency response. Three use cases - emergency response, transport and indoor navigation - will be explored further in this paper.

2.1. Emergency Response

Nowadays, the majority of wireless 911 emergency calls are made indoors. Consequently, in 2015, the U.S.A. Federal Communications Commission (FCC) introduced new requirements for network operators to improve location determination for indoor as well as outdoor calls. The operators need to implement the following location rules: within 6 years, they must provide 50 meter horizontal accuracy for 80% of all wireless 911 calls and implement several requirements for provision of vertical location information [6]. For emergency responders, it is better to know the exact vertical position (the correct floor) and get the horizontal position slightly wrong than to have an inaccurate vertical position (the wrong floor) and have the exact horizontal position [5]. Emergency response service also includes sending an alert to nearby emergency responders via their phones and the localization service of emergency equipment outside hospitals [2].

2.2. Transport

The tracking of assets and freights leads to higher transportation efficiency. The localization of vehicles is needed for traffic monitoring, management, and control. For example, when the vehicle position is known, drivers can be taxed through road-user charging [2]. Although Global Navigation Satellite Systems (GNSS) are often used in vehicles, they have difficulties in non line of sight environments like urban areas or areas with dense foliage. Therefore, they are complemented with other positioning systems, for example radars, cameras and sensors [4]. For automotive use cases, the automotive industry expects location accuracy within a 10cm range for self- or assisted-driving applications. Automotive use cases include automated driving, road safety and traffic efficiency services, digitalization of transport and logistics, intelligent navigation, information society on the road and nomadic nodes [7]. For an unmanned aerial vehicle, accurate positioning is critical [2]. In the future, a combination of unmanned aerial and ground vehicles will be deployed in automated security and surveillance systems. They will be also used in military applications, remote monitoring and data acquisition applications and applications in photography, inspection and surveillance missions [8].

2.3. Invented Use Case: Indoor Navigation Systems

Outdoor navigation systems are frequently used today. However, studies have shown that the average American spends 86.9% of his time indoors [9]. With improvements in vertical localization, indoor navigation systems, that guide people to their indoor destination, can be developed. This development will be especially useful for visually impaired people. Examples of buildings that might deploy indoor navigation systems are airports, office buildings, hospitals, hotels, universities and other government buildings, shopping malls and museums.

3. Fundamental Network Localization Techniques

The prevailing technology for outdoor localization, tracking, and navigation is the satellite system GNSS. Under ideal conditions GNSS delivers high accuracy (within a few meters). The disadvantages of GNSS receivers are high energy requirements, high time-to-first-fix and accuracy degradation in urban and indoor environments. The time it takes for the GNSS receiver to estimate the user location after it is turned on is called time-to-first-fix [5]. Because not all devices feature GNSS chips, other user localization methods that rely on wireless communication networks are also worth considering.

A rough estimation of the user location in wireless communication networks can be done with a technique called proximity. The location is estimated as the known location of the transmitter associated with the end device. Cell ID is a representative method for proximity. It estimates the user location as the location of the closest base station. Its accuracy depends on the density of transmitters [5]. Cell ID is the backup option when other methods can not be applied [1].

Trilateration is a method of determining the relative positions of three or more points by treating these points as vertices of a triangle or triangles of which the angles and sides can be measured [10]. Challenges of the trilateration approach are coverage, inter-cell interference, multipath channel and synchronization [1]. Possible measurements for trilateration are Time of Arrival, Time Difference of Arrival (TDoA), Direction of Arrival, Angle of Arrival and Received Signal Strength [2]. In order to improve system availability or localization accuracy, custom hybrid solutions can be implemented [5]. A combination of trilateration with GNSS is the most common hybrid solution [1].

Another popular technique is called fingerprinting or Radio Frequency Pattern Matching [1]. It addresses the problem of inaccuracy due to signal reflection and diffraction in urban areas. A database called radiomap is created, in which fingerprints (location-tagged signatures) are stored together with the corresponding location. The user can be located by finding the best match for a certain signal measurement, such as Received Signal Strength or time delay, with the fingerprints of the radiomap through pattern recognition [2].

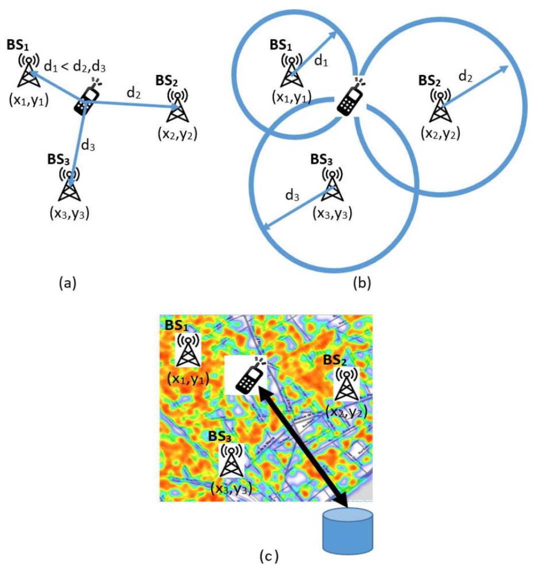


Figure 1. Illustrations of (a) proximity, (b) trilateration and (c) fingerprinting [5]

4. Localization Architectures

This subsection presents the three most common architectures, categorized by who estimates the location. Different aspects of a localization architecture, such as security, privacy and availability need to be considered. Nowadays, GNSS is regarded as critical infrastructure due to its use in essential systems like banking. In the future, indoor positioning will likely become part of the critical infrastructure as well. Therefore, service level availability is a concern. For example, emergency callers urgently need availability of the service and its location estimate. The service availability can be improved by redundancy, the location estimate by appropriate crowd-learning techniques [5]. Privacy is important for the user as well as the network operator: on the one hand, the user has the right that his location data is treated with confidentiality. The operator, on the other hand, has the right to keep the information about his network structure private [1].

In UE-based architectures, the user equipment (UE) estimates the location with assistance data from the the network. This approach is suitable for large user bases and situations in which location-awareness of the device is needed. Its advantages are low-cost and low-latency delivery. This approach is also more secure, because the network only provides assistance data. It can be implemented with radiomaps. Per building, a radio map can be obtained from a Content Delivery Network. With the radiomap, the UE is able to estimate its position within the building [5].

UE-assisted architectures are useful when the UE does not need to know the location information. An example is object tracking, where the location of the object only matters to the controllers. The UE measures certain signals, such as Wifi or Bluetooth signal strength and sends this data to the network. The UE does not need to be sophisticated, because it does not need to perform any calculations or store radiomaps. The network then does the location estimation. For each location event, a transaction is needed, which leads to a higher cost compared to the UE-based approach [5].

In network-based architectures, the network does the location estimation. This can be done, for example, by installing Bluetooth sniffers in a building. The disadvantages are the power and network connectivity requirements of the sniffers and the limited number of trackable devices. However, the advantage of this approach is its passiveness, meaning the UE does not need to do any measurements or calculations [5].

5. User Localization in 4G Cellular Networks

Release 9 supports three different localization methods: assisted GNSS, observed TDoA, and enhanced Cell ID. GNSS needs at least four satellites with clear line of sight to get a 3D position. Assisted GNSS tries to overcome that problem by having the network provide assistance data to the GNSS receiver [11]. With observed TDoA, the UE measures the time interval between the reception of downlink signals of two different neighbor base stations. The observed TDoA between base station 1 and base station 2 is the difference t2-t1, where t1 is the time of receiving the signal from base station 1 (t2 respectively) [11]. In order to calculate the position, the observed TDoA from at least three different pairs of base stations are needed [11]. Enhanced Cell ID uses the geographical coordinates of the closest serving base station and one of three additional measurements: the distance from one base station, the distance from three base stations or the Angle of Arrival from at least two base stations. The first two approaches are UE-assisted, the third one is network-based [11]. Observed TDoA is UE-assisted, whereas uplink TDoA is network-based [5].

Uplink TDoA, a method which calculates the position of the user by evaluating the time difference of LTE uplink signals sent to the base stations by the UE, was added in Release 11 [5]. Release 12 introduced Radio Frequency Pattern Matching. Release 13 added an observed TDoA enhancement, terrestrial beacon systems, WLAN, Bluetooth and barometric pressure sensor positioning. A terrestrial beacon system consists of ground-based transmitters, which complement GNSS by sending positioning signals. In barometric pressure sensor positioning, a barometer measures the air pressure. This approach has a vertical positioning accuracy below one meter. [1].

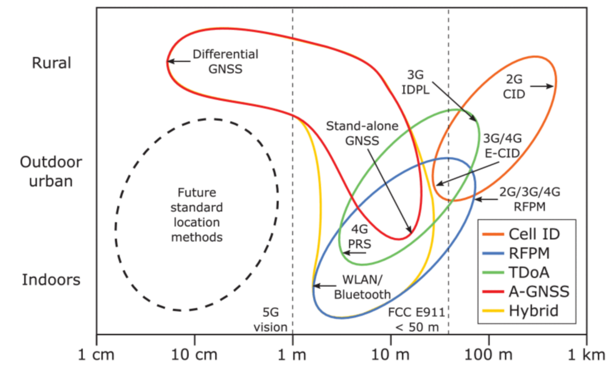


Figure 2. Expected horizontal accuracy of cellular mobile radio localization methods for indoor, outdoor urban and rural scenarios [1]

6. User Localization in 5G Cellular Networks

Release 15 supports assisted GNSS, WLAN, Bluetooth and barometric pressure sensor positioning techniques and TDoA on an LTE carrier. Release 16 reintroduces some positioning methods from LTE - such as observed TDoA, uplink TDoA, enhanced Cell ID - and introduces new methods such as Multicell Round Trip Time, uplink Angle of Arrival and downlink Angle of Departure. Whether Release 17 will introduce new positioning methods depends on analysis of the accuracy of Release 16 [12].

Localization possibilities will increase because of the introduction of new 5G features like mmWave technology. mmWave frequency is defined as 24-52.6 GHz [12]. mmWave technology has the twofold advantage of increased frequency and bandwidth. Furthermore, mmWave technology leads to a better resolution of multipath components. One challenge of mmWave technology is increased path loss, because the path loss is in proportion to the square of the carrier frequency. Other challenges are difficulties in diffracting around obstacles and penetrating through solid materials. For example, a brick wall causes 178 dB attenuation at 40 GHz [13].

Multipath-assisted localization techniques offer centimeter range accuracy [1]. Algorithms, for example Channel-SLAM, exploit multipath propagation to estimate the position of the UE. SLAM means Simultaneous Localization and Mapping. Channel-SLAM treats multipath components as signals emitted from virtual transmitters and can accurately estimate the location of the UE even if only one physical transmitter is available. Another advantage of Channel-SLAM is that it needs no prior information like for example a fingerprint database. It only needs to know the physical transmitter position, the initial receiver position and the moving direction [14].

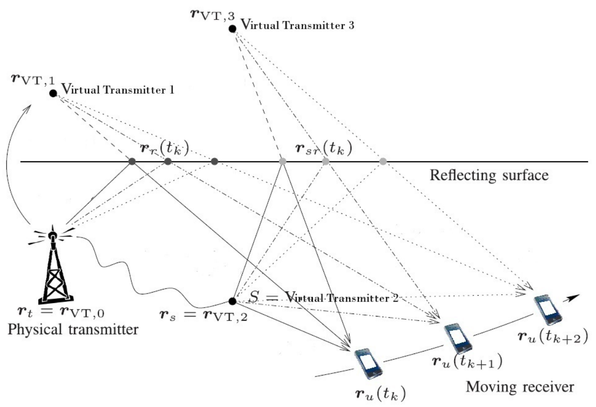


Figure 3. Illustration of Channel-SLAM [14]

Channel-SLAM takes into account reflection on a smooth surface and scattering. Scattering takes place at a fixed point S at position r_s. r_t is the position of the physical transmitter. $r_u(t_k)$ denotes the position of the user equipment at time t_k with $k = 0, \ldots, \infty$. The position of virtual transmitter 1 is constructed by mirroring the physical transmitter position at the smooth surface. Virtual transmitter 2 is constructed at the position of the fixed point S, where the signal is scattered. A combination of scattering and reflection is also possible: the signal is first scattered at S and then reflected on the surface. Therefore, virtual transmitter 3 is constructed by mirroring the scattered signal at the surface [14].

With the emergence of 5G ultra-dense networks, cooperative positioning will be achieved through D2D communication over directional mmWave networks [13]. It will be used in wireless sensor networks and ultra-wide bandwidth networks. Observed TDoA accuracy is enhanced through D2D cooperative positioning. The user equipment, base stations, access points and any object capable of emitting and receiving radio signals are called nodes. In D2D cooperative localization, the nodes get their location information in relation to one another and can calculate their absolute position with the help of global reference information [15]. This method is suitable for short and medium range [13]. D2D communication will lead to high robustness and accuracy below one meter. The hybrid fusion of multiple sensors serves the same goal [1]. Smaller cells like picocells (range under 100 m) and femtocells (WiFi-like range) enable more accurate line of sight localization [15].

7. Future Research

(Statistical) machine learning will play an important role in future localization systems. With further research, machine learning technologies will be able to estimate missing or corrupted data. Machine learning has already been used for simultaneous localization and mapping. Heterogeneous networking protocols pose a challenge because they operate according to different standards and on different frequencies. Many different IoT standards such as Bluetooth, ZigBee, SigFox, LoRa, Narrow Band IoT exist. Research questions include how location services can remain accurate while switching communication protocols and how to make switching between the different standards efficient. 5G cellular network will rely on New Radio, which is a wireless access technology that was standardized in Release 15 of the 3rd Generation Partnership Project. It is mainly unexplored how characteristics of New Radio can be best utilized. Another research area is the limitations of low latency communication. Accuracy within centimeters will presumably be reached through ultra-reliable low latency communication. The UE will be able to send and receive messages within milliseconds [5]. Another open research topic for high data rate wireless systems are different beamforming schemes and the potential of beamforming for the improvement of the localization quality. Challenges of beamforming schemes are the uncertainty of parameters (for example channel states) and the non-convexity of the optimization [16].

8. Conclusion

5G enables more accurate location services. In this paper, different user localization use cases were explored. Fundamental network techniques like proximity, trilateration and fingerprinting were presented. Three different kinds of localization architectures were classified. After that, user localization in 4G cellular networks was briefly analyzed. Then this paper explored different features of 5G: mmWave technology, D2D communication and multipath-assisted algorithms like Channel-SLAM. These features will lead to centimeter level accuracy in user localization. While this development has many advantages, like better emergency response, it also leads us to the question of user privacy and data protection. With increasing technological possibilities to track user equipment with centimeter level accuracy, many questions about data privacy arise, like what data will be stored, where will the data be stored, for how long will the data be stored, who has access to the data, for which purposes is the data used, how is the data protected? These questions should be the topic of an ongoing public debate.

References

[1] J. A. L.-S. J. A. del Peral-Rosado, R. Raulefs and G. Seco-Granados, "Survey of cellular Mobile Radio Localization Methods: From 1G to 5G," in *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, 2018, pp. 1124–1148.

[2] S. Bartoletti, A. Conti, D. Dardari, and A. Giorgetti, "5G Localization and Context-Awareness."

[3] 3gpp Release 17. [Online]. Available: https://www.3gpp.org/release-17

[4] S. K. J. A. del Peral-Rosado, J. A. López-Salcedo and G. Seco-Granados, "Feasibility Study of 5G-based Localization for assisted Driving," *International Conference on Localization and GNSS (ICL-GNSS)*, pp. 1–6, 2016.

[5] C. Laoudias, A. Moreira, S. Kim, S. Lee, L. Wirola, and C. Fischione, "A Survey of Enabling Technologies for Network Localization, Tracking, and Navigation," in *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, 2018, pp. 3607–3644.

[6] F. C. Commission. (2015, Feb) Wireless E911 Location Accuracy Requirements. [Online]. Available: https://apps.fcc.gov/edocs_public/attachmatch/FCC-15-9A1_Rcd.pdf

[7] 5G-PPP, "5g Automotive Vision," *Eur. Commission, ERTICO ITS Europe, Brussels, Belgium, White Paper*, pp. 1620–1635, 2015.

[8] M. F. Ayub, F. Ghawash, M. A. Shabbir, M. Kamran, and F. A. Butt, "Next Generation Security and Surveillance System using Autonomous Vehicles," *Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS), Wuhan*, pp. 1–5, 2018.

[9] The National Human Activity Pattern Survey. [Online]. Available: https://indoor.lbl.gov/sites/all/files/lbnl-47713.pdf

[10] Definition Trilateration. [Online]. Available: https://www.dictionary.com/browse/trilateration

[11] E. Z. Mike Thorpe, "White Paper LTE Location based Services Technology Introduction," 2013. [Online]. Available: https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_common_library/dl_brochures_and_datasheets/pdf_1/LTE_LBS_White_Paper.pdf

[12] M. B. A. Ghosh, A. Maeder and D. Chandramouli, "5G Evolution: A View on 5G cellular Technology beyond 3GPP Release 15," *IEEE Access*, vol. 7, pp. 127 639–127 651, 2019.

[13] J. Qiao, X. S. Shen, J. W. Mark, Q. Shen, Y. He, and L. Lei, "Enabling device-to-device communications in millimeter-wave 5G cellular networks," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 209–215, 2015.

[14] C. Gentner, T. Jost, W. Wang, S. Zhang, A. Dammann, and U. Fiebig, "Multipath assisted positioning with simultaneous localization and mapping," *IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 6104–6117, 2016.

[15] P. Zhang, J. Lu, Y. Wang, and Q. Wang, "Cooperative Localization in 5G Networks: A Survey," 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959517300346#br000045

[16] A. L. B. Zhou and V. Lau, "Successive Localization and Beamforming in 5G mmWave MIMO Communication Systems," *IEEE Transactions on Signal Processing*, vol. 67, no. 6, pp. 1620–1635, 2019.

Network Simulation with OMNet++

Simon Bachmeier, Benedikt Jaeger*, Kilian Holzinger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: Simon.Bachmeier@in.tum.de, jaeger@net.in.tum.de, holzinger@net.in.tum.de

***Abstract*—**
In view of the current development, computer networks are increasing in size and complexity. For development reasons simulations became more important in this area. OMNeT++ is a discrete event simulation environment. It was available and present for over 20 years and designed not as a network simulator, but for all-purpose use. This results in its area of application being not only in computer science fields. Over the years OMNeT++ has built a revenue for being able to simulate all kinds of areas. From queuing to wireless networks simulation, everything is possible. In this paper the OMNeT++ framework is presented with a detailed view into network simulation. Also, OMNeT++ is compared to other popular network simulators, where its strengths and weaknesses are discussed.

***Index Terms*—OMNet++, Network Simulation, NS-3, INET, MiXiM**

1. Introduction

The Internet backbone changed from a connection between research communitys to commercial use. This means already after eight and a half years of its lifetime, it developed from six nodes with 56 kbps links to 21 nodes with multiple 45 Mbps links. During the early stages, the Internet already grew to over 50,000 networks, including outer space and all continents [1]. 2016 the LTE networks nearly had 1.5 billion users with peak data rates of 450 Mbps [2]. The number of deployed devices using the Internet by 2020 has been estimated to be as high as 20.8 billion [3]. These numbers show that it is crucial that networks are functioning without errors. For development reasons testing is needed when working with networks. Testing with real networks can be very expensive and does not scale, therefore network simulations became very popular. Much hardware is needed for an experiment. The size of the current networks makes it hard to build a real simulation. A simulation breaks down the functionality of a network and is able to build large networks. A simulator capable of such is OMNeT++. OMNeT++ stands for Objective Modular Network Test bed in C++ [4]. It was introduced in 1997 [5] and is a C++ based discrete event simulator. OMNeT++ software is free for academic and non-profit use. Its commercial version OMNEST can be obtained on the official website [6]. As Vargas states, OMNeT++ can be seen as a gap filler [5]. It positions itself between open-source, research orientated software and expensive commercial alternatives. To get a better understanding of OMNeT++, Section 2 gives an overview. OMNeT++'s approach to simulation is quite different from other simulators like NS-3 [7] or JiST [8]. OMNeT++ does not provide simulation components directly, but gives a framework, which can be seen as a basic building block. It also differs in its modularity, components can be reused in all kinds of simulations. In Chapter 3 there are examples and an explanation of extensions for OMNeT++, so called frameworks, related to network simulations. These frameworks, e.g. INET [9] or MiXim [10] are developed independently and follow their own update cycles. Over the years a many simulation models, have been developed in a broad field, e.g. IPv6-, peer-to-peer-, storage area-, and optical networks. Also companies like IBM, Intel and Cisco are using OMNeT++ successfully in commercial projects or for in-house research [5]. In this paper the author gives an overview of OMNeT++, in order to understand why it was able to remain and be popular over all those years since existing. Therefore, it is important to understand how it works and what makes it different from its competitors.
The papers's structure is divided into to the following sections: Section 2 gives an overview of OMNeT++, to provide a basic understanding. How network simulation is done, is shown in Section 3. Finally, Section 4, compares OMNeT++ to other simulators in its class.

2. Structure of OMNeT++

OMNeT++ is a multipurpose network simulation framework. It provides a building kit for many kinds of simulations. At its core are reusable modules, which is the reason why OMNeT++ is considered a component architecture. The structure can be compared to LEGO blocks [11], which is a good illustration. When there are well implemented modules, they can be reused in various ways in all kinds of simulations, just like LEGO blocks. The following sections describe the functionality of OMNeT++'s internal and external structure.

2.1. Modeling and Modules

OMNeT++'s architecture is built of modules communicating by messages. There are two types of modules, the active modules, referred to as simple modules and compound modules. Simple modules can be grouped into compound modules, the second module type. Compound modules group other modules an can be therefore seen as passive only representing the internal functionalities. Hierarchy levels are not limited, that is why OMNeT++'s

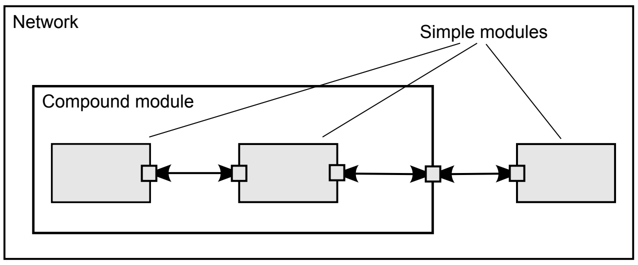


Figure 1: Simple and compound modules. Arrows represent connections between modules via gates [11].

structure is considered hierachical [11]. The entire simulation model represented by a network, therefore is a compound module. Simple modules are written in C++. The communication of modules is done by messages. In order to send messages, simple modules have input and output gates for sending and receiving. It is also possible to send messages directly to their destination, but normally it is done using gates. In a simulation, messages can represent customers in queues or packages in a network. Input and output gates can be linked by so-called connections. Compound modules also have such gates and connections to communicate with external modules. Connections can only be established in a single level of module hierarchy i.e sub modules can have connections to the compound module gate or other sub modules inside a compound module , but not to the outside. That is needed to maintain the reusability of modules. These connections can be seen as an own entity with parameters like bit error- rate or delay. Connections with predefined parameters, so called termed channels, can also be reused [11].

2.2. Design of a Model

An OMNeT++ model has a starting module, the system module. From the system module at the top level there are nested modules with a hierarchical structure with no boundaries to the bottom. With no limits in depth the user has the opportunity to build every structure he wants. The model structure is written in OMNeT++ network description language, short NED. The simple modules at the bottom are programmed in C++ using the simulation class library provided by OMNeT++. Listing 1 shows a C++ code example by TU-Ilmenau [12].

```
#include "omnetpp.h"
class MyModule : public cSimpleModule
{
    //a macro that calls constructor
   and sets
    //up inheritance relationships:
    Module_Class_Members (MyModule,
   cSimpleModule, 0)
    //user-specified functionality
   follows:...
};
Define_Module (MyModule);//announce
    this class as a module to OMNeT
```

Listing 1: Example of a simple module's C++ code. The class MyModule describes the functionality of a simple module. User can specify how to react on messages.

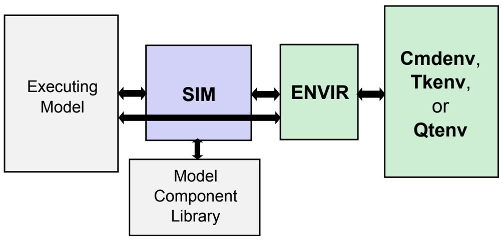


Figure 2: Logical architecture of an OMNeT++ simulation programm [11].

2.3. Architecture

The previous sections gave a short overview of OMNeT's internal structure. Figure 2 gives a understanding of the high level and logical architecture. ENVIR, CMDENV and TKENV describe the user interface libraries. This is the environment where the simulation is executed. It also describes the origin of input data and where simulations result go to. ENVIR is a library, which presents itself as a connection, containing all common user interface code. CMDENV and TKENV are ENVIR based libraries that have a specific user interface implementation. The all-purpose interface is CMDENV, known as the command line user interface. It is a small portable and fast interface that works on all platforms, which provides a console e.g. linux and is mainly used for batch execution. TKENV is a graphical user interface, for debugging and visualization of simulations. This environment describes the simulation's execution, from debugging to visualizing. It is possible to embed a OmNeT++ simulation into a larger application by replacing the user interface. The component library is built by the code of compiled simple and compound modules [5]. At the beginning of the simulation execution phase, the simimulation kernel builds the simulation model, which is the model created for the simulation. [13]

3. Network Simulation with OMNeT++

As stated in the previous sections OMNeT++ itself is merely a framework. That means in the base version only supports routing messages [5]. This feature traversions of a network and building a graph data structures. In addition, a user could further use graph algorithms to traverse it. But with nothing more provided the user has to implement a whole network with protocols and network associated structures himself. With the design of OMNeT++, however, a simulation can be easily extended by a network framework. Network simulation with OMNeT++ is simplified by extending it with existing frameworks such as INET [9] elaborated in section 3.1 or, specialized on mobile and wireless networks, the MiXiM framework, elaborated in section 3.2 [10]. Importing a framework is similar to importing libraries in Java [11]. Framework files are added to the project and can be accessed via the import [11].

3.1. INET

INET is a model library, which is specialized in designing new protocols [14]. With INET the user can build

hosts, routers and other network devices by using reusable components. INET provides components that are easy to modify and understand so that users can build their own components [14]. Some of INET's features are: IPv4/IPv6 network stacks, pluggable protocol implementation for various layers, wired/wireless interfaces and many more. In the following a project by Mohammed Alasmar and George Parisis will be presented which uses INET [15]. It uses the OMNeT++ with the INET framework to evaluate modern data center transport protocols.
Data center networks (DCNs) are getting more important, because they are used for the infrastructure of search services such as Google, social networks, cloud services and video streaming (e.g. Disney+). Alasmar et al. state that OMNeT++ in addition to INET is perfect for simulating and evaluating data transport protocols in DCNs [15]. The reason lies within INET's flexibility. New protocols can be easily added to the project and network devices (e.g. switches, routers etc.) are already provided. For example, there are simple module implementations for TCP and UDP, which users can use and extend. Within this project, a simulation model for the neighbor discovery protocol (NDP) and a FatTree network topology generator was built. Also, a data center environment that can be used to evaluate and compare data transport protocols, such as NDP and TCP, was developed. This outlines the possibilities with OMNeT++, not only creating simulations, but also creating a framework for other evaluations. In the simulations specialized on NDP protocols they used OMNeT++'s save tool command line to process and compare the results [15].

3.2. MiXiM

Another example for a network simulation framework which runs on OMNeT++ is MiXiM (mixed simulator). This framework is specialized for mobile and wireless networks. It offers detailed models of radio wave propagation, interference estimation, radio transceiver power consumption and wireless MAC protocols [10].
Following there is a short example of a simulation using the MiXiM framework, in order to improve the Quality of Service (QoS) in a wireless sensor network (WSN) [16]. This paper was written by Kodali et al [16]. The reason behind it was, that in a real time environment WSN are often unreliable and inaccessible, which has a bad effect on the QoS. In the paper the authors discuss adjustments of power, range and bit rates to attain adaptive topology control (ATC), in order to maintain the QoS of a WSN, which have a broad spectrum of usage. For example, it is used by the military or for weather monitoring, with the core concept of sensing, computing and communication. MiXiM was chosen because it supports the mobility framework which is used for the WSN network. The performance of Carrier Sense Multiple Access (CSMA) was measured in packet end-to-end delivery ratio and throughput. As Kodali et al state in their conclusion, they were able to improve the network performance by 29%, by varying the range and power adaption [16].

3.3. Recent Use of OMNeT++

This section provides an overview of projects featuring OMNeT++.

- UAV (Unmanned aerial vehicle) Swarm Network: The paper, deals with a swarm intelligence-based damage-resilient mechanism for UAV swarm networks. The authors build a simulation in OMNeT++, which presents a mechanism to recover a unified UAV swarm network after suffering damage [17].
- Secure Data Transmission and Sinkhole Detection: OMNeT++ is used for evaluating better methods for improved security in a multi-clustering wireless sensor network by homomorphic encryption and watermarking [18].
- Novel Segment Based Safety Message Broadcasting in Cluster-Based Vehicular Sensor Network: With multiple vehicles sending messages in vehicular sensor networks, data collisions occur more often, leading to corrupted packages. To counteract, OMNeT++ ist used for validation [19].

4. Comparison to other Network Simulators

Over the last years computer hardware has significantly improved. Also, networks as a topic got more attention, with the world being more connected than ever. The complexity of systems and networks topology is increasing drastically. To implement new network structure or protocols, it is indispensable to test, before actually deploying them. For example, when an untested defective new structure is pushed into an established network, fixing it will be costly and effort to fix that, while the system is still running. There are two kinds of test environments. A real system with existing hardware or a simulation. As already mentioned, the increased complexity makes it very expensive to build real systems for testing. Therefore, simulations have gained a lot of attention over the past years. Two network simulators which gained a lot of popularity during the last years are OMNeT++ and NS-3 [20]. In the following sections a comparison between these two simulators takes place. Also taking OMNeT++ design decisions, described in the previous sections, into account described by A. Vargas and R. Hornig [5] and comparing them to other simulators. After the comparison to NS-3, the paper presents a short comparison to other popular simulators.

4.1. Comparison to NS-3

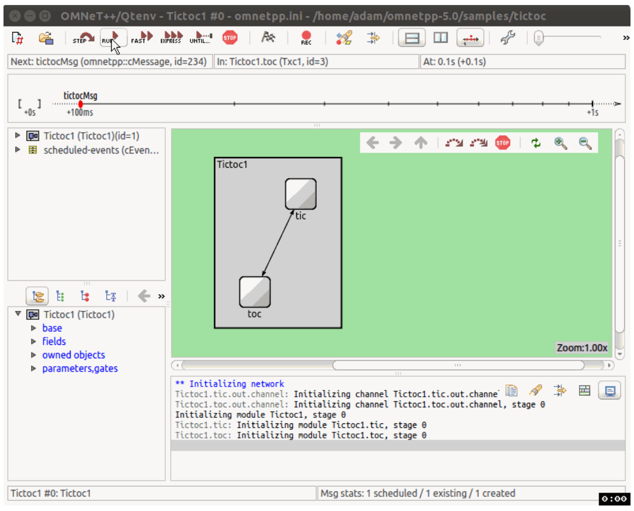
NS-3 (network simulator version 3) is the successor of the of NS-2, which is not supported anymore nor has backwards-compatibility. It is an open source discrete event network simulator [21], that tries to maintain an open environment for researchers to share their projects. The simulator itself is a C++ library and an integrated development environment is needed. During a simulation the network data traffic can be collected in a pcap-file. Then it is possible to analyze the file via wireshark or similar software. Wenhan Yao compared these two simulators in his master thesis [21]. Other sources are the general comparisons to other simulators in various scenarios [22]–[24]

4.1.1. Structural Differences of the Simulators. In OMNeT++, a NED file is used to describe the structure of a model, while the behavior of modules is written in C++. For simulations, the user has to choose a user interface (e.g. CMDENV,TKENV) see Figure 2 and define the starting parameter in an `omnetpp.ini` file. Then the result of a simulation is written into vector and scalar files, which can be evaluated by external software. For a simulation in NS-3, the user starts with a script written in C++ or Python. The script defines the topology of the network and how the simulations is run. A network is described by its nodes, connections and devices and their usage [21]. The models, which can be used for simulation, can be found in the NS-3 library. All the parameters of simulation are also written into the script, for example data rate, IP address and positioning of nodes. The resulting data can go to a pcap file, which can also be analyzed.
Both simulators are using C++ to describe their modules. NS-3's connection to modules is more direct than OMNeT++'s. In NS-3, the script can directly access the modules from the library, while in OMNeT++ the NED files are interposed. In other words, in OMNeT++ you need NED files to create and access modules, which makes the simulation more complex. OMNeT++ has visualization options already included. This simplifies network simulation decisions for less experienced users, compared to NS-3 with no included visualization. It is possible, though, to visualize the output files from NS-3, in order to have a visualization of graphs. This works similar to OMNeT++, where the output files can be processed by various programms [21].

4.1.2. Differentiation between OMNeT++ and NS-3. The GUI is one of OMNeT++'s main advantage, compared to NS-3. It can picture signals e.g TCP packages and components of a network structure. An example of a GUI screen can be seen in Figure 3. In simulations OMNeT++ always uses more computer resources than NS-3 [21], [25]. Also, OMNeT++ takes longer to execute. Another core difference is that NS-3 was built for network simulation, while OMNeT++ is multipurpose. Therefore, it is less workload for the user to create a simulation in NS-3, because the whole structure can be written in one script rather than having to implement modules, NED files and specify user interfaces. Both simulators deliver good performances, with similar workloads on the testing system and messages throughput. An outstanding feature is OMNeT++ visualization. The integrated GUI and the possibility to visualize the simulation, debugging process or even the structure of a simulation are one of the outstanding features of OMNeT++ [21], [22], [24]. But it is important to mention that both simulators are giving good simulation results for similar simulations in execution time and workload of the system [20]–[22], [24].

4.2. OMNeT++ and Other Simulators

This section provides a summary of the comparison from OMNeT++, NS-3, NS-2, SimPy and JiST, done by Weingartner et al [20]. SimPy incorporates a different


OMNeT++/Qtenv - Tictoc1 #0 - omnetpp.ini - /home/adam/omnetpp-5.0/samples/tictoc
STEP
RUN
FAST
EXPRESS
UNTIL...
STOP
REC
Next: tictocMsg (omnetpp::cMessage, id=234)
In: Tictoc1.toc (Txc1, id=3)
At: 0.1s (+0.1s)
tictocMsg
+0s
+100ms
+1s
Tictoc1 (Tictoc1)(id=1)
scheduled-events (cEven...
Tictoc1
tic
toc
Zoom:1.00x
Tictoc1 (Tictoc1)
base
fields
owned objects
parameters,gates
Tictoc1 #0: Tictoc1
Msg stats: 1 scheduled / 1 existing / 1 created
0:00

```
** Initializing network
Tictoc1.tic.out.channel: Initializing channel Tictoc1.tic.out.channe
Tictoc1.toc.out.channel: Initializing channel Tictoc1.toc.out.channel, stage 0
Initializing module Tictoc1, stage 0
Tictoc1.tic: Initializing module Tictoc1.tic, stage 0
Tictoc1.toc: Initializing module Tictoc1.toc, stage 0
```

Figure 3: Example of a GUI in an OMNeT++ Simulation [26].

simulation approach as a process-oriented discrete-event simulator and is written in Python [20], [23]. Another different simulation approach is presented by JiST. It is a high performance Java based simulation environment and just a simulation kernel [5], [20]. The authors built equal simulation scenarios for all the simulators, to compare them to each other. Their result was that only SimPy had slightly higher loss rates, still acceptable results thou. Afterwards they went to performance testing of the different simulators under equal conditions. Overall, the conclusion was that NS-3 has the best performance, while JiST advantages are in simulation run time. But OMNeT++ is the only simulator providing a GUI. JiST and NS-3 simulations are source code developed. The authors wrote that all of them are equally suited for network simulations, but it depends on the situation which one to choose [20], [23], [24].

5. Conclusion

Varga and Hornig describe in their paper the core elements of OMNeT++'s design, which is also crucial for large scale simulation [5]. The hierarchical structure with its reusable components, takes a huge part in its success. It means less work for the user by just reusing finished modules as much as possible. Another key point which differentiates OMNeT++ to many other simulators is its GUI and the possibility to visualize the simulation, debugging and the structure. Debugging consumes a lot of time in projects. Visualization often helps and reduces time consumption. Simulations are modular customizable allowing a user to embed an emulation into a larger application. This integration opens up many areas of application. As other simulators, OMNeT++ can produce output files to have a better analysis with other programs. However, the structure of OMNeT++ with its layers makes small simulations often more complex than they need to be. For example, when a user just wants to create a test simulation, where two nodes send each other a package, in OMNeT++ you need to define the kernel and structure of al parts from the system . The NED language was designed to scale well. With increasing complexity in networks, however, it reaches its limits. Therefore, improvements to the software itself had to be made, where the user has to adjust too. Furthermore, the environment is clearly divided in parts like NED modules which requires more computer resources for network simulations. Overall, however, OMNeT++ follows a clear structure, which makes it easy to handle, also within large simulations.

References

[1] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, and S. Wolff, "A brief history of the internet," *Computer Communication Review*, vol. 39, pp. 22–31, 10 2009.

[2] D. López-Pérez, D. Laselva, E. Wallmeier, P. Purovesi, P. Lundén, E. Virtej, P. Lechowicz, E. Malkamaki, and M. Ding, "Long term evolution-wireless local area network aggregation flow control," *IEEE Access*, vol. 4, pp. 9860–9869, 2016.

[3] Z. Yan, H. Li, S. Zeadally, Y. Zeng, and G. Geng, "Is dns ready for ubiquitous internet of things?" *IEEE Access*, vol. 7, pp. 28 835–28 846, 2019.

[4] "OMNet++ Homepage," http://www.omnetpp.org, [Online; accessed 8-May-2020].

[5] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," 01 2008, p. 60.

[6] "OMNEST Homepage," https://omnest.com/, [Online; accessed 24-May-2020].

[7] "NS-3 Homepage," https://www.nsnam.org/, [Online; accessed 24-May-2020].

[8] "JiST Homepage," http://jist.ece.cornell.edu/, [Online; accessed 24-May-2020].

[9] "INET Framework," https://inet.omnetpp.org/, [Online; accessed 11-May-2020].

[10] "MiXiM Framework," http://mixim.sourceforge.net/, [Online; accessed 11-May-2020].

[11] "OMNet++ Simulation Manual," https://doc.omnetpp.org/omnetpp/manual/, [Online; accessed 9-May-2020].

[12] "C++ code snippet," https://www.tu-ilmenau.de/fileadmin/media/telematik/lehre/Projektseminar_Simulation_Internet_Protokollfunktionen/Material/03_OmNet__.pdf, [Online; accessed 27-May-2020].

[13] [Online; accessed 15-August-2020]. [Online]. Available: https://ewh.ieee.org/soc/es/Nov1999/18/userif.htm

[14] "INET introduction," https://inet.omnetpp.org/Introduction, [Online; accessed 12-May-2020].

[15] M. Alasmar and G. Parisis, "Evaluating modern data centre transport protocols in omnet++/inet," in *Proceedings of 6th International OMNeT++ Community Summit 2019*, ser. EPiC Series in Computing, M. Zongo, A. Virdis, V. Vesely, Z. Vatandas, A. Udugama, K. Kuladinithi, M. Kirsche, and A. F\"orster, Eds., vol. 66. EasyChair, 2019, pp. 1–10. [Online]. Available: https://easychair.org/publications/paper/4xwP

[16] R. Kodali and M. Vijay Kumar, "Mixim framework simulation of wsn with qos," 05 2016, pp. 128–131.

[17] M. Chen, H. Wang, C. Chang, and X. Wei, "Sidr: A swarm intelligence-based damage-resilient mechanism for uav swarm networks," *IEEE Access*, vol. 8, pp. 77 089–77 105, 2020.

[18] H. A. Babaeer and S. A. AL-ahmadi, "Efficient and secure data transmission and sinkhole detection in a multi-clustering wireless sensor network based on homomorphic encryption and watermarking," *IEEE Access*, pp. 1–1, 2020.

[19] I. S. Alkhalifa and A. S. Almogren, "Nssc: Novel segment based safety message broadcasting in cluster-based vehicular sensor network," *IEEE Access*, vol. 8, pp. 34 299–34 312, 2020.

[20] E. Weingartner, H. vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *2009 IEEE International Conference on Communications*, 2009, pp. 1–5.

[21] W. Yao, "Analyse und vergleich der netzsimulatorenns-3 und omnet++," p. 74, 05 2018, [Online; accessed 21-May-2020]. [Online]. Available: http://midas1.e-technik.tu-ilmenau.de/~webkn/Abschlussarbeiten/Masterarbeiten/ma_yao_sw.pdf

[22] Xiaodong Xian, Weiren Shi, and He Huang, "Comparison of omnet++ and other simulator for wsn simulation," in *2008 3rd IEEE Conference on Industrial Electronics and Applications*, 2008, pp. 1439–1443.

[23] V. Oujezsky and T. Horvath, "Case study and comparison of simpy 3 and omnet++ simulation," in *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, 2016, pp. 15–19.

[24] A. Zarrad and I. Alsmadi, "Evaluating network test scenarios for network simulators systems," *International Journal of Distributed Sensor Networks*, vol. 13, no. 10, p. 1550147717738216, 2017. [Online]. Available: https://doi.org/10.1177/1550147717738216

[25] S. B. A. Khana and M. Othmana, "A performance comparison of network simulatorsfor wireless networks," pp. 1–6.

[26] [Online; accessed 11-June-2020]. [Online]. Available: https://docs.omnetpp.org/tutorials/tictoc/part2/

State of the Certificate Transparency Ecosystem

Nikita Blagov, Max Helm*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: n.blagov@tum.de, helm@net.in.tum.de

***Abstract*—Certificate Transparency (CT) can improve end user's security advancing the already established HTTPS; it is an emerging, though not yet fully developed technology. Its imperfections bring about new potential threats such as Partitioning Attacks, leaking of domain names, or overloaded logs. In this paper, we study how CT has been deployed so far and which browsers already support it. We take a closer look at logs and their requirements, as well as available HTTP headers assisting CT. We further study the aforementioned threats and demonstrate currently available countermeasures.**

Index Terms—**certificate transparency, log, browser, deployment, expect-ct, threats, equivocation, gossiping, leaking domain names, wildcard certificates, label redaction**

1. Introduction

HTTPS ensures confidentiality on networks by encrypting communication. In a TCP connection, a client's browser performs a handshake with a webserver during which an SSL certificate is submitted, proving authenticity of the domain (represented as 'B' in Figure 1). Website owners request this certificate from a Certificate Authority (CA) ('A' in Figure 1). [1]

In 2011, however, hackers compromised the Dutch CA DigiNotar and issued a fraudulent certificate targeting Google users in Iran [2]. Subsequently, "the certificate was revoked and the offending CA was removed from client trust stores" [1], causing many Dutch websites using a certificate from DigniNotar becoming inaccessible. On numerous occasions, similar incidents were not detected for several weeks causing a great deal of damage [3].

To better cope with such incidents, Certificate Transparency (CT) has been developed [1]. It aims to provide "an open auditing and monitoring system that lets any domain owner or CA determine whether certificates have been mistakenly or maliciously issued" [3], ultimately protecting end users.

This paper provides an overview of background information, CT's current deployment, contemporary threats and their countermeasures. The rest of this paper is structured as follows. Chapter 2 presents important background information and introduces the key concepts. Chapter 3 focuses on CT's current deployment. How successfully has CT been developed so far? Is it already supported by every browser? How does this support look like? Which requirements must logs meet to become trusted? Can HTTP headers assist if CT is not supported by default? Chapter 4 focuses on contemporary threats and unintended uses of CT. What are Partitioning Attacks and is there any available countermeasure today? Why is the leaking of domain names treacherous and can it be prevented? What if logs become overloaded? Has it ever happened at all and how could it be averted? Related work can be found in Chapter 5.

2. Background

CT extends the existing TLS system by three new actors: logs, monitors, and auditors. Figure 1 illustrates the existing components in blue and supplementary ones in green.

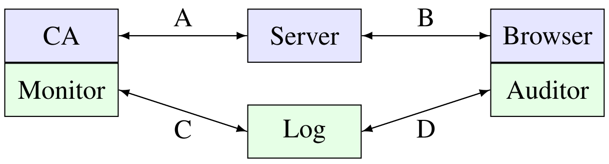


Figure 1: Components of CT [4]

Logs are independently operated, publicly auditable records of certificates. They are append-only: once a certificate has been added to a log, it cannot be deleted, modified, or retroactively inserted.

Logs are formed by a binary Merkle Hash Tree (with leaves and nodes). Figure 2 exemplifies such a tree. Each leave represents a value of the corresponding appended hashed certificate. Leaves are paired with other leaves forming nodes, which can be paired too forming further nodes. Nodes also represent hash-values. The root hash, comprising all nodes and leaves, is called the Merkle Tree Hash (MTH). It assures logs are append-only since any change of log's entries leads to a different MTH. Logs regularly sign it together with other information such as the Merkle Hash Tree's size and version. A signed MTH is called Signed Tree Head (STH). [4]–[7]

"A certificate is considered included in a log when it is covered by an STH" [5]. "When a log receives a certificate, it replies with a Signed Certificate Timestamp (SCT)" [1].

An SCT is a promise to add the certificate within a time known as Maximum Merge Delay (MDD). There are three ways to deliver an SCT either embedded as an X.509v3 Extension, as an TLS Extension, or via OCSP Stapling. The first method is most widely used and oftentimes the only supported. The CA logs the certificate

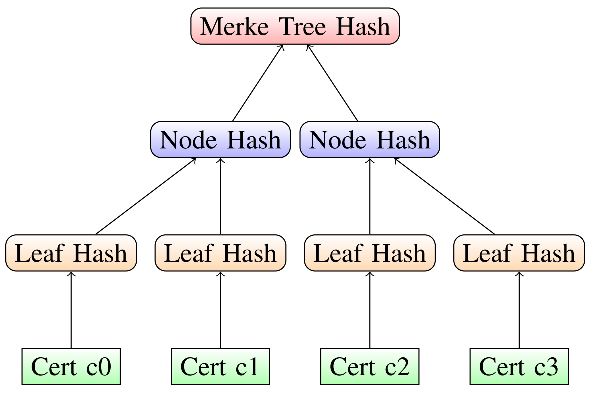


Figure 2: Merkle Hash Tree: Log's Structure [6], [8]

and embeds the SCT, thus it does not require any server modification. [4]

In TLS Extension, domain's server itself submits the certificate to the log and obtains the SCT itself [4]. It is considered the fastest method but requires modifying the server, thus less supported [9]. OCSP Stapling is more complex, very little supported and disabled in Chrome [5].

Both certificates and logs can lose their trust. Misissued or bogus certificates are revoked. Misbehaving logs can be fully distrusted, disqualified, or frozen. A frozen log remains in read-only mode but cannot add any new certificates or sign any new STHs, though existing STCs remain being accepted. Logs face similar consequences if they are disqualified, except that non-embedded SCTs must be replaced. In case a log is fully distrusted, none of its issued SCTs will be accepted anymore. [1]

Monitors detect suspicious certificates in logs and verify the visibility of all certificates ('C' in Figure 1). They are typically run by CAs and since they maintain a log's complete copy, they can "act as a backup read-only log". [4]

Auditors are typically included in client's browsers and verify SCT's validity [6]. They also assure that a "certificate represented by a given SCT has in fact been logged" by requesting an inclusion proof [1]. In addition, they can request a consistency proof to verify that log's entries have neither been modified nor retroactively deleted and logs only present consistent views ('D' in Figure 1) [1].

3. Deployment

CT was first standardized in June 2013 with RFC 6962 [7]. Today, CT is already supported in more than 60% of HTTPS traffic [1]; still not every browser is capable of effectively handling it. Overall, CT is an emerging, yet not fully developed technology.

In the following, we show which of the major four browsers provide support. We take a more detailed look at Chrome for two reasons. First, Google has been the pioneer in CT deployment and its enforcement. Secondly, Chrome was by far the most widely used browser worldwide with over 65% in May 2020. Safari browser was on the second place with 18%, followed by Firefox and Edge both with around 3% [10]. We examine Chrome's and Safari's policies for CT and trusted logs and finally introduce two HTTP headers: Require-CT and Expect-CT.

3.1. Browser Inclusion

Fearing to break too many websites, browsers enforced CT gradually. At first, Google required all Extended Validation (EV) certificates issued following January 2015 to be CT-logged. In September 2017, Chrome also began to provide support for the Expect-CT HTTP header. In July 2018, they enforced CT for all certificates issued since May to comply with Chromium's CT policy. If during a TLS handshake the certificate is missing, expired or is not logged, the connection fails and users are shown a security warning. [1], [11]

Each certificate must be accompanied by at least two SCTs from diverse qualified logs, one from a Google log and one from a non-Google Log. This forces a potential attacker to compromise multiple different logs simultaneously [12]. Currently qualified logs are listed in [13].

If at least one of the SCTs is delivered using TLS extension or OCSP Stapling, the aforementioned criteria is enough; otherwise, the number of required logs depends on certificate's validity period, as listed in Table 1. Should it be shorter than 15 months, 2 distinct (qualified) logs are sufficient. One more additional log is required in case the certificate's lifetime is between 15 and 27 months. The policy argues against exceeding 27 months; however, it does not prohibit it. Supposing the validity period is longer than 27 months but maximum 39, 4 different logs are needed; all longer valid certificates must be logged in 5 diverse logs. [12]

TABLE 1: Number of Embedded SCTs in Chrome [12]

Lifetime of Certificate	Number of SCTs from distinct logs
< 15 months	2
>= 15, <= 27 months	3
> 27, <= 39 months	4
> 39 months	5

Apple quickly followed with its Safari enforcing all certificates issued subsequent to October 15th, 2018 to be CT-logged. Its CT policy comprises the same criteria as Chrome's [14]. Microsoft announced their aspiration for CT checking in Edge [15], though there is no current support yet [16]. Neither is any available for Mozilla Firefox [17], their experimental implementation for telemetry caused performance issues [18]. The middle column of Table 2 displays the aforementioned four browsers and whether they enforce CT.

TABLE 2: Browsers' CT-Support [12], [14], [16], [17], [19], [20]

Browser	CT Enforcement	ExpectCT Support
Chrome	yes	yes
Safari	yes	no
Firefox	no	no
Edge	no	yes

3.2. Log Requirements

In the following, we explain which criteria need to be met for a log to become and remain trusted in Chrome and

Safari. It is important to note that both Google and Apple can always withdraw their trust in a previously accepted log again for any reason. Overall, their log policies are almost identical. [21], [22]

3.2.1. Inclusion Request Requirements. When applying for log inclusion, its operators must provide a log's description, policies for accepting and rejecting certificates, as well as log's MDD. They must further list all accepted root certificates, log's public key and its URL for the public HTTP endpoint. They must finally provide contact information for people both operating and representing, Apple requires two contacts respectively. [21], [22]

An example (Nimbus 2018) for an inclusion request can be found in [23].

3.2.2. Monitoring: Ongoing Requirements. Both Google and Apple monitor the log after accepting its inclusion request to verify it conforms to their CT policies. They expect the following criteria to be met at any time. [21], [22]

Logs must comply with RFC 6962 and implement its CT correspondingly, e.g., being append-only and publicly auditable. They must always provide consistent views to different parties, i.e., avoiding equivocation (see Section 4.1). Their MDD must not exceed 24 hours. Both Google and Apple further require an uptime of 99%, though both measure it individually. [21], [22]

Apple instructs to "trust all root CA certificates included in Apple's trust store" and to "accept certificates that are issued by Apple's compliance root CA to monitor the log's compliance with these policies" [22]. Google does similarly and also issues own certificates for monitoring purposes that must be accepted when requested [21].

3.3. HTTP Headers

A site operator can utilize an HTTP header to necessitate all certificates for its domain being logged even if CT is not enforced by default in the browser. This has been particularly important for certificates issued prior to May 2018 as these certificates are not required to be CT-logged to be shown as valid in Chrome [19]. A common header is Expect-CT, an alternative is Require-CT. The latter one, however, is unofficial and relatively unpopular; while 7,300 domains were using the Expect-CT header, and only 8 were using Require CT as shown by Gasser et al. in [5]. Expect-CT may become outdated by June 2021 since certificates issued before May 2018 will all expire until then [19].

Expect-CT is formed by three directives: a compulsory max-age, as well as an optional enforce and report-uri [24]. "The following example specifies enforcement of Certificate Transparency for 24 hours and reports violations to foo.example" [19].

```
Expect-CT:
max-age=86400, enforce,
report-uri="https://foo.example/report"
```

Most commonly, however, only the reporting feature is used by setting max-age to zero and leaving out the enforce directive [5]. To the best of our knowledge, Expect-CT support is currently only implemented in Chrome and Edge, not, however, in Safari or Firefox, as displayed in the right column of Table 2 [19], [20].

4. Threats and Countermeasures

New technology often has weaknesses at the beginning that offer attackers new possibilities. Next, we introduce three threats arising with CT: equivocation, leaking of domain names, and overloaded logs. In addition, we demonstrate potential countermeasures.

4.1. Equivocation and Gossiping

Ideally, even if an attacker uses a bogus certificate, as long as the end user pays attention to warnings and uses a browser supporting CT, a conventional man-in-the-middle (MITM) attack is useless. The connection fails prior to causing any harm. Since the certificate is logged, CT monitors can quickly detect it and prompt its revocation. If the attacker is capable, however, to prevent monitors and auditors to detect the rogue certificate by hiding its entry in the log, they would impede any security measures. Monitors simply cannot ascertain the rogue certificate and the corresponding isolated (browser-) auditor is provided a fraudulent, though in itself consistent view. Such an attack relies on a second Merkle Hash tree version within a log with a different STH. [6]

It is called 'Split-View', 'Partitioning Attack', or 'Equivocation' [5], [6], [25].

To successfully counteract it, there is a technology called 'Gossiping' whereby auditors gossip about a log. They exchange STHs or SCTs to verify their view is consistent with others' view [26]. Unfortunately, it has "next to no deployment in the wild" yet [5]. Gossiping faces two major challenges. First, a defined and scalable mechanism must be standardized among all clients; secondly it entails a high risk of uniquely identifying (fingerprinting) clients by means of gossiped STHs or SCTs, and thus violating their privacy [27].

There are various experimental gossiping-mechanims; however, it is important to note that they "are drafts and are subject to rapid and substantial change" [6]. We demonstrate two of them, STH Pollination and Google's current experimental Minimal Gossip, an extension and refinement of the prior.

4.1.1. STH Pollination. In STH Pollination, both clients and auditors submit STHs to and receive them from the servers representing STH pools. Clients and auditors do not communicate directly. [26]

Since "an STH uniquely represents an entire tree version", it suffices for a consistency verification. "STHs are normally not considered privacy sensitive, as long as they are shared by a large set of clients". Therefore, there are two restrictive measures. First, only fresh, maximum 14 days old STHs are gossiped; older ones are discarded. Secondly, participating (gossiped) logs are prohibited to issue (sign) new STHs more frequently than once per hour; otherwise, they will be ignored. In effect, the two measures guarantee having at most 336 pollinated STHs for each log at any time. These are finally verified by auditors and monitors for consistency. [6]

4.1.2. Minimal Gossip.

Google's Minimal Gossip approach implements an altered version of STH Pollination. It focuses on minimal modification of already existing CT and introduces a new synthetic certificate chain conjoining a root and a leaf certificate. The root certificate identifies the gossiper (usually an auditor) and "indicates minimal gossip use". Pollinated (destination) logs add it to their acceptable roots, thus trusting the gossiper. Each STH received from a source log is embedded in a separate leaf which is then added to the destination log. The leaf certificate is valid for only 24 hours and marked as critical to avoid being mistakenly recognized as a valid conventional certificate. The verification can then be performed using goshawk. [27]

Goshawk "scans a destination log for gossiped STH values and checks consistency against the source logs" [28].

4.2. Domain Names Leaking and Label Redaction

Logs can easily be misused by attackers to discover new domain names that would otherwise remain secret. The new information aids targeting victims precisely, thus, exposing the whole CT ecosystem to risk. [29]

Likewise, logs complicate keeping new products confidential prior to their public release. Let us assume, the company Guenther GmbH ('guenther.eu') is developing a new product 'Moepi' and dedicates an individual domain, 'moepi.guenther.eu', to it. Competitors could discover and ascertain Moepi's development by simply scanning a log since logged domain labels, in this case 'moepi', are publicly readable. [30]

We look at three countermeasures: starting with the least applicable wildcard certificates, following with name-constrained intermediate CA certificate, and finishing with the most applicable mechanism called label redaction. The more applicable a mechanism is, the higher its complexity. [31]

4.2.1. Wildcard Certificates.

In wildcard certificates, a wildcard "*" label effectively hides the private domain. For instance, the domain 'secret.example.com' can be replaced by '*.example.com'; 'moepi.guenther.eu' by '*.guenther.eu'. [31]

Wildcard certificates, however, face two limitations. First, they cannot be used when dealing with EV certificates. Secondly, they can only cover one level of subdomain, conforming to RFC 2818. Consequently, '*.example.com' could not replace 'top.secret.example.com' since it involves 2 levels of subdomains. [31], [32]

A special type are partial-wildcard certificates, e.g., '*p.example.com', which covers both 'top.example.com' and 'flop.example.com'. Partial-wildcard certificates, however, are disabled in major browsers and should, therefore, not be used. [32]

4.2.2. Name-Constrained Intermediate CA Certificate.

Another way is logging a name-constrained intermediate CA certificate instead of the "end-entity certificate issued by that intermediate CA" [31].

Such an intermediate CA certificate comprises a name constraint extension [see RFC 5280] with two core components, permitted and excluded subtrees. A permitted subtree explicitly defines an allowed namespace, whereas an excluded subtree disallows certain namespaces. Assuming we want to allow 'example.com' and its subdomains (e.g., 'top.secret.example.com'), we would enter 'example.com' (for the domain itself) and '.example.com' (for all subdomains) under permitted subtrees. Likewise, to disallow 'bad.example.com' and its subdomains, we would add 'bad.example.com' and '.bad.example.com' under excluded subtrees. [31], [33], [34]

The following criteria must be met to log a name-constrained intermediate CA certificate. First, it must contain a non-critical extension indicating the acceptability of not logging certificates issued by the according intermediate CA. Secondly, there must be at least one DNS-name defined in permitted trees; and finally, excluded subtrees must disallow the full range of IPv4 and IPv6 addresses. [31]

This method is included by default in CT 2 [see RFC6962-bis], a currently developed improved version of CT [35], [36]. Its latest version is 34 [37].

4.2.3. Label Redaction.

Label redaction can effectively hide non-wildcard domain-names by replacing them with redacted labels. In case of Guenther GmbH, 'moepi.guenther.eu' would be redacted to '(redacted).guenther.eu'. [30]

Each redacted label is determined, inter alia, using a hash function and a Base 32 Encoding function (see RFC 4648). Using hashing, legitimate domain owners can verify that each redacted label corresponds with the original label. A redactedSubjectAltName extension helps reconstructing the certificate by imparting which labels have been redacted. [31]

Nevertheless, label redaction is not absolute, as it may be forbidden by client's policies [31].

Furthermore, it requires domains owners to include new domains (new products) as subdomains of an existing domain. Thus, label redaction would not be sufficient provided Guenther GmbH wants to allocate an independent domain for Moepi, such as 'moepi.com'. It would require redacting the domain to '(redacted).com' which will be rejected by logs. Likewise, wildcard certificates ('*.com') also fail in this particular situation. [30]

4.3. Overloaded Logs and Log Splitting

Large CAs submit certificates to only a small number of CT logs, leaving them with a massive number of certificates [29]. This can lead a log to performance issues due to overload, which may result in disqualification or being frozen [29]. E.g., Google's log Aviator, which was frozen on November 30th, 2016 due to failing to add a certificate within the MDD [38].

Today, log operators often split their logs into several, one for each year, as can be seen in [39]. It reduces the number of certificates per log significantly. A potential disqualification affects only the logs expiring in the same year and not all together, thus, reducing the overall damage. The year represents the year of certificate's expiration, e.g., Nimbus 2018 promising to log only certificates expiring in 2018 and being frozen straight afterwards. [23]

Nevertheless, one may criticize these measures as not thoroughgoing enough. Despite log splitting, one Nimbus

CT log encountered performance issues in 2018 during an update, risking being disqualified [40]. Nimbus and Google Argon are among the largest CT logs. The final (fixed) STH of 'Nimbus 2019' has had over 493 million entries, while the one of Google Argon 2019 has had over 857 million entries [41], [42]. One potential solution could be CAs distributing "their logging load more evenly among logs and log operators" [29].

5. Related Work

Scheitle et al. analyze the deployment of CT until the first half of 2018 and define new threats of certificates; they do not, however, study possible countermeasures [29]. Gasser et al. observe CT logs and search for violations of its requirements. They also examine HTTP headers and gossiping [5]. Stark et al. investigate challenges facing in deployment and its adoption on the web until 2019 [1].

6. Conclusion

In this paper, we provided an overview of CT's current deployment state; how Google and Apple have progressively implemented it so far, while others fall behind. We studied how CT can be unintendedly used by attackers or competitors to discover new domain names and how the use of wildcard or intermediate CA certificates, as well as label redaction can hinder them. We illustrated equivocation as a threat to the CT ecosystem and presented Google's experimental Minimal Gossip, an extension of STH Pollination, as a potential solution. Further research could be done examining gossiping's impact once it has been officially deployed and standardized.

References

[1] E. Stark, R. Sleevi, R. Muminovic, D. O'Brien, E. Messeri, A. P. Felt, B. McMillion, and P. Tabriz, "Does Certificate Transparency Break the Web? Measuring Adoption and Error Rate," in *2019 IEEE Symposium on Security and Privacy*. Los Alamitos, CA: IEEE Computer Society, 2019.

[2] H. Adkins, "Google Online Security Blog: An update on attempted man-in-the-middle attacks," https://security.googleblog.com/2011/08/update-on-attempted-man-in-middle.html, 2011, [Online; accessed: 7/06/2020].

[3] "What is Certificate Transparency? - Certificate Transparency," https://www.certificate-transparency.org/what-is-ct, [Online; accessed: 5/06/2020].

[4] "How Certificate Transparency Works - Certificate Transparency," https://www.certificate-transparency.org/how-ct-works, [Online; accessed: 5/06/2020].

[5] O. Gasser, B. Hof, M. Helm, M. Korczynski, R. Holz, and G. Carle, "In Log We Trust: Revealing Poor Security Practices with Certificate Transparency Logs and Internet Measurements," in *Passive and active measurement*, ser. LNCS sublibrary. SL 5, Computer communication networks and telecommunications, R. Beverly, G. Smaragdakis, and A. Feldmann, Eds. Cham, Switzerland: Springer, 2018, pp. 173–185.

[6] J. Gustafsson, "Certificate Transparency in Theory and Practice," Ph.D. dissertation, 2016, http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A909303&dswid=3471, [Online; accessed: 17/06/2020].

[7] B. Laurie, A. Langley, and E. Kasper, "RFC 6962 - Certificate Transparency," https://tools.ietf.org/html/rfc6962, 2013, [Online; accessed: 11/06/2020].

[8] "How Log Proofs Work - Certificate Transparency," https://www.certificate-transparency.org/log-proofs-work, [Online; accessed: 29/06/2020].

[9] C. Nykvist, L. Sjöström, J. Gustafsson, and N. Carlsson, "Server-Side Adoption of Certificate Transparency," in *Passive and active measurement*, ser. LNCS sublibrary. SL 5, Computer communication networks and telecommunications, R. Beverly, G. Smaragdakis, and A. Feldmann, Eds. Cham, Switzerland: Springer, 2018, pp. 186–199.

[10] "Browser Market Share - May 2020 - NetMarketShare," https://netmarketshare.com/?options={"filter":{},"dateLabel":"Custom","attributes":"share","group":"browser","sort":{"share":-1},"id":"browsersDesktop","dateInterval":"Monthly","dateStart":"2020-05","dateEnd":"2020-05","segments":"-1000"}, 2020, [Online; accessed: 10/06/2020].

[11] D. O'Brien, "Certificate Transparency Enforcement in Google Chrome–Google Groups," https://groups.google.com/a/chromium.org/forum/#!msg/ct-policy/wHILiYf31DE/iMFmpMEkAQAJ, 2018, [Online; accessed: 11/06/2020].

[12] "Certificate Transparency in Chrome," https://github.com/chromium/ct-policy/blob/master/ct_policy.md, 2019, [Online; accessed: 11/06/2020].

[13] "Qualified Logs Chromium," https://github.com/chromium/ct-policy#chromium-certificate-transparency-policy, 2020, [Online; accessed: 11/06/2020].

[14] "Apple's Certificate Transparency policy," https://support.apple.com/en-gb/HT205280, 2019, [Online; accessed: 11/06/2020].

[15] R. Jha, "Certificate Transparency," https://docs.microsoft.com/en-us/archive/blogs/azuresecurity/certificate-transparency, 2018, [Online; accessed: 12/06/2020].

[16] B. Li, J. Lin, F. Li, Q. Wang, Q. Li, J. Jing, and C. Wang, "Certificate Transparency in the Wild," in *CCS'19*, L. Cavallaro, J. Kinder, X. Wang, J. Katz, L. Cavallero, and X. Wang, Eds. New York, NY: Association for Computing Machinery, 2019, pp. 2505–2520.

[17] "Implement Certificate Transparency support (RFC 6962)," https://bugzilla.mozilla.org/show_bug.cgi?id=1281469, [Online; accessed: 11/06/2020].

[18] "Certificate Transparency Signature Verifications Negatively Impact TLS Handshake Performance," https://bugzilla.mozilla.org/show_bug.cgi?id=1353216, [Online; accessed: 11/06/2020].

[19] "Expect-CT," https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expect-CT, 2020, [Online; accessed: 11/06/2020].

[20] "when will Mozilla support Expect-CT, a new security header," https://support.mozilla.org/de/questions/1268433, [Online; accessed: 30/06/2020].

[21] "Certificate Transparency Log Policy," https://github.com/chromium/ct-policy/blob/master/log_policy.md, 2017, [Online; accessed: 11/06/2020].

[22] "Apple's Certificate Transparency log program," https://support.apple.com/en-om/HT209255, 2019, [Online; accessed: 15/06/2020].

[23] "Issue 780654: Certificate Transparency - Cloudflare "nimbus2018" Log Server Inclusion Request," https://bugs.chromium.org/p/chromium/issues/detail?id=780654#c14., 2017, [Online; accessed: 11/06/2020].

[24] E. Stark, "Expect-CT Extension for HTTP," https://github.com/bifurcation/expect-ct/blob/master/draft-stark-expect-ct.md, 2017, [Online; accessed: 14/06/2020].

[25] R. Dahlberg, T. Pulls, J. Vestin, T. Høiland-Jørgensen, and A. Kassler, "Aggregation-based gossip for certificate transparency," 2019.

[26] L. Nordberg, D. Gillmor, and T. Ritter, "Gossiping in CT - draft-ietf-trans-gossip-05," https://tools.ietf.org/html/draft-ietf-trans-gossip-05, 2018, [Online; accessed: 16/06/2020].

[27] "Google/Certificate-Transparency-Go," https://github.com/google/certificate-transparency-go/blob/5690bed5b44db4a1b17fae99187a8bf4dc69830a/gossip/minimal/README.md, 2019, [Online; accessed: 15/06/2020].

[28] “Command goshawk,” https://godoc.org/github.com/google/certificate-transparency-go/gossip/minimal/goshawk, [Online; accessed: 18/06/2020].

[29] Q. Scheitle, O. Gasser, T. Nolte, J. Amann, L. Brent, G. Carle, R. Holz, T. C. Schmidt, and M. Wählisch, “The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem,” in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC ’18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 343–349.

[30] “CA/CT Redaction,” https://wiki.mozilla.org/CA/CT_Redaction, [Online; accessed: 01/07/2020].

[31] R. Stradling, “Certificate Transparency: Domain Label Redaction draft-strad-trans-redaction-01,” https://tools.ietf.org/html/draft-strad-trans-redaction-01, 2017, [Online; accessed: 11/06/2020].

[32] “Wildcard certificate,” https://en.wikipedia.org/wiki/Wildcard_certificate, [Online; accessed: 30/06/2020].

[33] T. Quinn, “Name Constraints in x509 Certificates,” https://timothy-quinn.com/name-constraints-in-x509-certificates/, [Online; accessed: 30/06/2020].

[34] V. Podāns, “X.509 Name Constraints certificate extension – all you should know,” https://www.sysadmins.lv/blog-en/x509-name-constraints-certificate-extension-all-you-should-know.aspx, [Online; accessed: 30/06/2020].

[35] E. Messeri, “Overview of changes between RFC6962 and RFC6962bis,” https://www.certificate-transparency.org/ct-v2-rfc6962-bis, [Online; accessed: 30/06/2020].

[36] B. Laurie, A. Langley, E. Kasper, E. Messeri, and R. Stradling, “Certificate Transparency draft-ietf-trans-rfc6962-bis-14,” https://tools.ietf.org/html/draft-ietf-trans-rfc6962-bis-14, [Online; accessed: 30/06/2020].

[37] ——, “Certificate Transparency Version 2.0 draft-ietf-trans-rfc6962-bis-34,” https://tools.ietf.org/html/draft-ietf-trans-rfc6962-bis-34, [Online; accessed: 30/06/2020].

[38] “Issue 389514: Certificate Transparency: Inclusion of Google’s "Aviator" log,” https://bugs.chromium.org/p/chromium/issues/detail?id=389514, 2014, [Online; accessed: 20/06/2020].

[39] “Certificate Transparency Monitor - Graham Edgecombe,” https://ct.grahamedgecombe.com/, 2020, [Online; accessed: 15/06/2020].

[40] Brendan McMillion, “Post-Mortem: Nimbus issuing bad SCTs - Google Groups,” https://groups.google.com/a/chromium.org/forum/#!searchin/ct-policy/Nimbus$20issuing$20bad$20SCTs%7Csort:date/ct-policy/E88pjOZzkIM/-uphL2fqBQAJ, 2018, [Online; accessed: 06/06/2020].

[41] “Cloudflare Nimbus 2019 - Graham Edgecombe,” https://ct.grahamedgecombe.com/logs/53, [Online; accessed: 20/06/2020].

[42] “Google Argon 2019 - Graham Edgecombe,” https://ct.grahamedgecombe.com/logs/43, [Online; accessed: 20/06/2020].

A Survey on Threshold Signature Schemes

Sinan Ergezer, Holger Kinkelin*, Filip Rezabek*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: sinan.ergezer@tum.de, kinkelin@net.in.tum.de, frezabek@net.in.tum.de

***Abstract*—Threshold signatures, an alternative of current signature systems, provide better security than its formers. Nevertheless, constructing a threshold signature scheme from a non-threshold one is a complex process that requires two main challenges to be solved. The first challenge, namely the key generation part can be centralized or decentralized. The solution to the second challenge, distributed signing, is affected by the signature scheme being used. In this paper, these two main challenges will be analyzed by giving insights on the signature schemes ECDSA, BLS, and Schnorr.**

***Index Terms*—threshold crypto schemes, key generation, distributed signing**

1. Introduction

With the current development in the cryptocurrency industry, the necessity of secure digital signatures has also grown proportionally. As of 2018, the amount of cryptocurrency daily stolen is equivalent to 1.7 billion dollars. [1] That means, there are some huge problems regarding the authentication of the parties in a digital transaction. There are numerous ways for authentication in a digital currency transaction. [2]

The most widely used way for authentication is, single signature, which only has one approver and one private key taking part in the transaction; that also makes it the least secure option, since single point of failure/attack is present. That means, if the signing party goes unavailable, the signing cannot be succeeded. Similarly, an adversary can directly forge the signature if he/she can compromise the signing party. As a solution, multisignature was proposed. In multisignature, there are multiple approvers, having a signature each. This is inefficient, since multiple are needed for every transaction, but also insecure since information of the other parties that involved are revealed by the time of every signing. This led to an inevitable upgrade of multisignature: the threshold signature system, which eliminates single point of failure and attack. In a threshold cryptosystem, there are also multiple parties involved in the signature, but one single key is shared among them. This makes it hard to forge the signature since an adversary must compromise the threshold number of parties. In a threshold signature system, not all of the parties need to be present during the signing phase, if a pre-determined number t of n parties are available, signing can be successfully done.

The rest of the paper is structured as follows:

1) Background section 2
2) Threshold Key Generation section 3
3) Comparison and Use Cases of Threshold Signature Schemes section 4
4) Conclusion section 5

2. Background

Before diving into the threshold signature schemes, we will introduce the general concept of a digital signature scheme to the reader.

2.1. Structure of a Digital Signature

A digital signature scheme can be divided into three main algorithms: key generation, signing and verification [3]. In key generation, a pair of keys are generated: a signing key(private key) which we sign the message with and a public key to verify this message. Let us express the the output of key-gen algorithm as a tuple (sk, pk). The signing function sig takes the secret key sk and a message m as an input. It outputs the signed message m' as follows: $sig(sk, m) \rightarrow m'$. And lastly the verification function $ver(pk, m, m')$ verifies whether the signature is valid. That means, decryption of m' with the help of pk should be equal to message m.

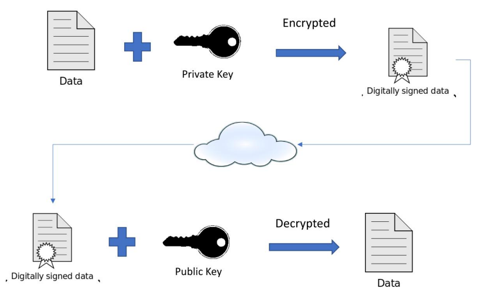


Figure 1: Digital Signature [4]

Before getting started with the digital signature schemes, the concept of elliptic curves should be introduced to the reader since it plays a significant role in the signature schemes ECDSA and BLS.

2.2. Elliptic Curves

An elliptic curve is an algebraic structure that is used for cryptographic purposes. It is a plane curve over a finite field, which has the mathematical term $y^2 = x^3 + ax + b$, and as a result of its term, it is symmetrical about the x-axis [5]. The elementary operation over an elliptical curve is the point addition. To add two points P and Q that are on the curve following steps are done [6] :

1) Determine the curve between the two points
2) Make the curve intersect with the ellliptic curve to find the third point R'
3) Reflect the the third point to find the resulting point R

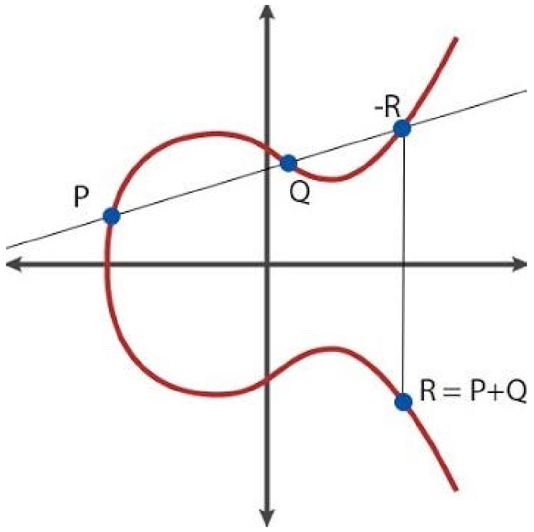


Figure 2: Point Addition in an Elliptic Curve [7]

2.3. Discrete Logarithm Problem

The mathematical property of elliptic curves enables us to get to every other point in the curve if we add any chosen point on the curve to itself enough times. We call this arbitrarily chosen starting point the base point or the generator G of the elliptic curve. That way, every point P can be written as a multiple of G: $P = nG$ where n is a positive integer. As a result, multiplying the generator n times should have had $\mathcal{O}(n)$ complexity, since we are doing n point additions. But this is not the case, since with the help of the property $nP + rP = (n + r)P$ we only have $\mathcal{O}(\log n)$ complexity. We have an efficient method for calculation nG as it has $\mathcal{O}(\log n)$ complexity, but no feasible method is known for deducing n from P and G. This problem is known as the "Discrete Logarithm Problem" in cryptography [8].

2.4. Elliptic Curves and Public Key Cryptography

In a cryptographic scheme that uses an elliptic curve, the public key is computed using xG where x is the private key and G the generator. We know due to the Discrete Logarithm Problem that there is no efficient method for finding the private key from the public key. If the private key were a 256-bit integer, it would take on average 2^{128} calculations for finding it. This leads us to one of the core principles of public key cryptography: It should be computationally infeasible to figure out the private key given the public key [9].

2.5. Digital Signature Schemes

The key generation and signing functions of the signature Schemes ECDSA, BLS and Schnorr are presented in the following, where the verification functions are omitted, due to brevity. ECDSA [10]:

- Key-Gen : Random integer x on the elliptic curve, which is the private key. Public key := $y = g^x$
- Signing:
  - Hash the message : $h = H(m)$
  - Generate a random nonce k
  - let $R = kG$ and take the x-coordinate of it
  - let $s = k^{-1}(h + r * x) mod n$ where $k^{-1}(mod n)$ is modular inverse function Our signature σ is the pair (r, s)

BLS: [11]:

- Key-Gen : Random integer x on the elliptic curve, which is our private key. Public key : $y = g^x$
- Signing : The bitstring of the message m is hashed using a hash function : $h = H(m)$ The signature is: $\sigma = h^x$

Schnorr: [12]:

- Key-Gen: Random x from Schnorr group, which is our private key. Our public key is:$y = g^x$
- Signing:
  - let $r = g^k$ where k is a random integer from the Schnorr Group
  - let $e = H(r||m)$: the bitstring r and the message m is concatenated and then hashed
  - let $s = k - xe$. Our signature σ is the pair (s, e)

3. Threshold Signatures

The most complex challenge of a threshold scheme is the key generation part. Before explaining the centralized and the decentralized approaches, the concept of secret sharing will be introduced. Secret sharing helps a party to distribute it's secret among other parties. In the centralized approach, the secret is the private key that is distributed among other parties by a chosen party. In the decentralized approach, there is no such authority, the generated key share of each party is distributed among every other party.

3.1. Shamir's Secret Sharing

Shamir's Secret Sharing is based on the idea that t points are sufficient to uniquely define a polynomial of degree $t - 1$. F is a finite field of size q, where q is a prime number [13].

Start by choosing $t-1$ positive integers as coefficients to our polynomial f : $a_1, .., a_{t-1}$ with every a_i being an element of our finite field F. Let a_0 be the private key.

Construct for $1 <= i <= n$, $s_i = (i, f(i) mod q)$: Every party i is given its secret key share sk_i. With any t parties of n, it would be possible to reconstruct our polynomial through interpolation [14]. Thus, it will be possible to acquire the secret S. This is also the threshold property: with our threshold number of parties being present, the private key will be reconstructed.

3.2. Verifiable Secret Sharing(VSS)

In Shamir's Secret Sharing there is no verification ensured for the secret that is shared. Or in other words, one cannot ensure that the secret share is valid. In VSS, the party which distributes the key shares also gives a commitment along with the key shares [15]. Every party can verify their share with the help of this commitment.

3.3. Centralized Key Generation Approach

In the centralized approach, a single party known as "dealer" generates public and private key pairs, whereas the private key is our private key from the previous secret sharing algorithms [16].The most naive secret sharing method to use in the centralized approach is Shamir's Secret Sharing, which is acceptable if the dealer is a "trusted authority", meaning that it is not assumed to be malicious. If the dealer is untrusted, a secret sharing method with additional verification functionality is needed, which makes VSS a more secure choice.

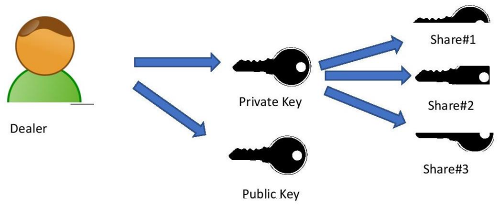


Figure 3: Centralized Key Generation

3.4. Decentralized Key Generation Approach

In the decentralized key generation, there is not any authority who generates the private key. In this approach, each player of the threshold signature computes their own share. This is done using a distributed key generation algorithm known as DKG [17]. A DKG algorithm consists of two phases: the sharing phase and the reconstruction phase.

1) Sharing phase: Every party does a secret sharing of their randomly chosen secret x using VSS to the other parties. In the end, each party has a share S_i of the secret S, (the private key), which is a linear combination of the shared values.
2) Reconstruction phase: Each party declares its secret share, which is reconstructed with the help of a reconstruction function Rec and a consequent broadcast of the public key is followed.

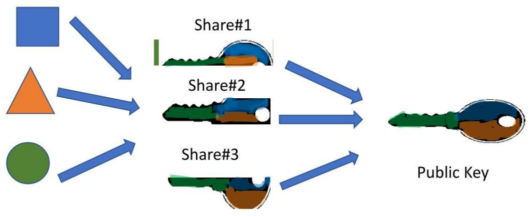


Figure 4: Distributed Key Generation

4. Comparison and Usage of Threshold Signature Schemes

4.1. Trade-Off Between the Two Approaches

In the centralized version, as aforementioned, the dealer can be trusted or untrusted. Verifying the key shares through VSS seems to solve the security problem, but it does not provide a full security. In the centralized approach, there is always a "single point of attack/failure", which is in the side of the authority. If the authority goes unavailable, the key generation can not be made. Or an access to the authority's side by an adversary could mean a reveal of the private key. Those both scenarios are meant to be tackled by the decentralized approach, which has no single point of failure. It could be said that the centralized approach is not complex but insecure because participating parties rely on one single party. On the other hand, the implementation of DKG is more complex, lasting multiple computational steps. Also, communication overhead is present between the parties, which could cause a problem for devices with limited computational power. It could nevertheless be said that this computational burden could still be tolerated since key generation is not a process that is repeated frequently. Generally, if no security concerns are encountered, the same key can be used for signing different messages.

4.2. Solution to the Distributed Signing Problem

In the signing phase of a threshold signature scheme, every party should sign the message m with their own share of the secret key obtained from the KeyGen phase [2].The most important part is that each party should never reveal their secret share and nor should the secret key be reconstructed at any point, which would cause a direct single point of attack.

4.3. Difficulty of Transforming Signature Schemes into Their Threshold Versions

Implementing a distributed signing protocol proved to have technical difficulties [18]. Threshold ECDSA signatures need additional cryptographic primitives such as Paillier encryption in the signing phase. This complexity is present in the signature schemes coming from the DSA family [19]. There is no standardized solution for the threshold signing protocol of ECDSA. Also, there is not

property	ECDSA	BLS	Schnorr
sig. aggregation	no	yes	yes
determinism	no	yes	yes

TABLE 1: Table for Property Comparison of Each Threshold Scheme

any known version of threshold ECDSA that does not use multiple rounds, in other words, the parties should exchange secret information with each other multiple times.

BLS and Schnorr can easily be transformed into threshold versions since they are suitable for signature aggregation in contrary to ECDSA. That means, with their current schemes, every party signs the message using their own key share sk_i. The resulting signature can then be additively combined into one signature.

4.4. Property Comparison of Each Threshold Scheme

A table that compares the properties of the threshold schemes is presented above.Signature aggregation is already presented in subsection 4.3. Another property is determinism, which means: "For any given public key and message there can be only one valid signature" [20].

4.5. Use Cases of Threshold Signatures

Threshold Signatures are still a prototyped technology which does not have any standardized usage, although there is still ongoing research and development regarding the adoption of threshold signatures in various fields.

4.5.1. Bitcoin Transactions. A standard Bitcoin transaction uses single signature ECDSA with being behind today's signature technology. The adoption of multisignature wallets is also significantly low, as of 2016, multisignature wallets are being used in only 11 percent of the transactions [21]. Although the cryptocurrency industry is unable to catch the current signature technology, there is a continous research on the compatibility with Bitcoin: in well-known papers of Gennaro et al [18] and Goldfeder et al. [22], threshold ECDSA signature schemes are proposed for usage in Bitcoin. Despite the difficulty in the adaptation of ECDSA into a threshold scheme as stated in subsection 4.3, the reason why ECDSA is selected is convenience. ECDSA is the standard signature scheme for Bitcoin.

4.5.2. Certificate Authorities. Certificate Authorities are trusted third parties, who validate the identity of internet entities by digitally signing certificates [23]. Since two parties rely on these digital certificates when communicating with each other, an adversary that can forge the signature of that authority could directly influence the communication. Using threshold signature mechanism would be a more secure choice, as presented in the paper of Zhou et. al [24].

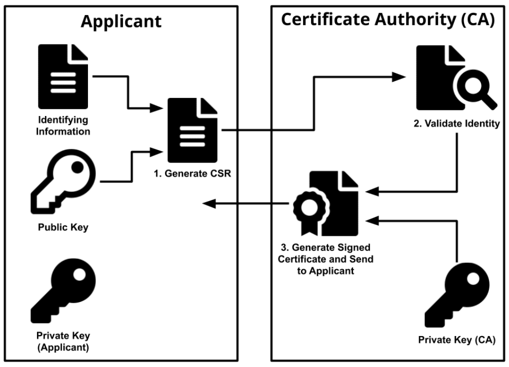


Figure 5: Certificate Signing Mechanism [23]

5. Conclusion

In threshold signatures, there are two main challenges to tackle. The first one, also the most complex one, is the key generation part. The key generation has two main approaches. In the centralized approach, a trusted authority exists, who generates the private key and distributes it among the parties who will sign the message. In this approach, the parties do not have any computational burden but it is not secure enough due to having a single point of failure. In the decentralized approach the single point of failure is eliminated, but this time each party directly takes part in the key generation. The complexity of distributed signing problem varies because of the nature of the scheme being used. It is a known fact that threshold ECDSA scheme is difficult to implement, whereas threshold BLS and threshold Schnorr do not bring any implementation difficulties by dint of signature aggregation property. Even so, the standard threshold signature schemes to be used and the approaches going to be taken to solve the two mentioned challenges are still unknown. Threshold signature systems remain to be a prototype.

References

[1] G. Chavez-Dreyfuss, "Cryptocurrency thefts, scams hit $1.7 billion in 2018: report ," *Reuters*, 2019, [Online; accessed 16-August-2020].

[2] S. Alex. (2020) Threshold Signature Explained— Bringing Exciting Applications with TSS . https://medium.com/@arpa/threshold-signature-explained-brining-exciting-apps-with-tss-8a75b43e19bf. [Online; accessed 16-August-2020].

[3] R. Gennaro and S. Goldfeder, "Fast multiparty threshold ecdsa with fast trustless setup," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1179–1194. [Online]. Available: https://doi.org/10.1145/3243734.3243859

[4] https://thecybersecurityman.com/2017/12/12/understanding-the-cia/, [Online; accessed 16-August-2020].

[5] Y. El Housni, "Introduction to the Mathematical Foundations of Elliptic Curve Cryptography," 2018.

[6] B. Bill, "Adding Points in Elliptic Curve Cryptography ," *Medium*, 2019, [Online; accessed 16-August-2020].

[7] https://hal.archives-ouvertes.fr/hal-01914807/document, [Online; accessed 16-August-2020].

[8] D. J. Pantůček, "Elliptic curves: discrete logarithm problem ," *Trustica*, 2018, [Online; accessed 16-August-2020].

[9] M. D. Ryan, “Computer Security lecture notes,” https://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/public_key.html, 2008, [Online; accessed 16-August-2020].

[10] S. Nakov, “ECDSA: Elliptic Curve Signatures ,” 2019, [Online; accessed 16-August-2020].

[11] S. Varadaraj, “The BLS Signature Scheme — A short intro,” 2018, [Online; accessed 16-August-2020].

[12] “Schnorr Signature,” [Online; accessed 16-August-2020].

[13] E. Rafaloff, “Shamir’s Secret Sharing Scheme ,” 2018, [Online; accessed 16-August-2020].

[14] B. Archer and E. W. Weisstein, “Lagrange Interpolation ,” [Online; accessed 16-August-2020].

[15] B. Schoenmakers, *Verifiable Secret Sharing*. Boston, MA: Springer US, 2005, pp. 645–647. [Online]. Available: https://doi.org/10.1007/0-387-23483-7_452

[16] C. Li and J. Pieprzyk, *Conference Key Agreement from Secret Sharing*. Springer,Berlin,Heidelberg, 2001, vol. 1587.

[17] A. Kate, Y. Huang, and I. Goldberg, “Distributed key generation in the wild.” *IACR Cryptology ePrint Archive*, vol. 2012, p. 377, 2012.

[18] R. Gennaro, S. Goldfeder, and A. Narayanan, *Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security*. Springer, 2016, pp. 156–174.

[19] P. MacKenzie and M. K. Reiter, “Two-party generation of dsa signatures,” in *Advances in Cryptology — CRYPTO 2001*, J. Kilian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 137–154.

[20] A. Block, “Secret Sharing and Threshold Signatures with BLS,” 2018, [Online; accessed 16-August-2020].

[21] O. Ogundeji, “Multisig Best for Bitcoin Wallet Security But Only 11 Percent Use It ,” *CoinTelegraph*, 2016, [Online; accessed 16-August-2020].

[22] S. e. a. Golfeder, “Securing Bitcoin wallets via a new DSA/ECDSA threshold signature scheme .”

[23] A. Russell, “ What is a Certificate Authority (CA)?” 2019, [Online; accessed 16-August-2020].

[24] L. Zhou, F. B. Schneider, and R. Van Renesse, “Coca: A secure distributed online certification authority,” *ACM Trans. Comput. Syst.*, vol. 20, no. 4, p. 329–368, Nov. 2002. [Online]. Available: https://doi.org/10.1145/571637.571638

A survey on Multi-Agent Systems (MAS)

Angjeliqi Anxhela Gjikopulli, Anubhab Banerjee*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: anxhela.gjikopulli@tum.de, anubhab.banerjee@tum.de

Abstract—**Multi-agent systems (MAS) got the attention of scholars and experts from various disciplines, from computer science to civil and electrical engineering, due to its capability of solving complex problems. The problem solving skill is achieved through the subdivision of the problem into smaller tasks and allocating them to specific agents within the system. Despite finding multiple applications across various disciplines, MAS still face numerous challenges related to the organizational issues among agents, security, task allocation, and so on. This paper provides a discussion on the diverse aspects of MAS starting with a definition of agents and their features. In addition, it tackles into details the application of MAS in three different fields: Cellular and Mobile Networks, Smart Cities and Ambient Technologies and Social Networks, the challenges faced in those fields and possible future works in respective fields.**

Index Terms—**Multi-Agent Systems (MAS), MAS applications,**

1. Introduction

Multi-agent systems is a subfield of Distributed Artificial Intelligence (DAI) [1] and has experienced rapid growth due to its ability to address complex computing problems. Multi-Agent Systems (MAS) are systems composed by an accumulation of autonomous entities, known as agents. Agents are computer systems with the important capabilities of learning and making autonomous decisions, and also interacting with neighbouring agents [2]. Their learning capabilities enable them to autonomously decide what they need to do in order to satisfy their design objectives [3] and solve their allocated tasks. The ability of interacting with other agents or with the environment enables them to learn new actions or contexts. The flexibility that such capabilities give to MAS, makes them suitable for problem-solving in a variety of disciplines, from computer science to civil or electrical engineering. At the same time there are various common challenges in building effective MAS, regardless of the application discipline, such as security [3], coordination within the system, task allocation, fault detection etc. In order to better understand and solve these challenges, it is neccessary to well-define agents and their features.

The content of this paper is organized as follows. First, we provide a theoretical overview of MAS, starting with a definition of agents and their features and the advantages of MAS to other systems in Section 1. Next, in Section 2, we give a detailed overview of three main areas of MAS applications, including the current developments and challenges in each field and some possible future work to improve each application. Finally, the conclusion of this MAS survey is given in Section 3.

2. Introduction to Agents

The most commonly agreed definition of agent is the one by Russell and Norvig [4] who define it as a flexible autonomous entity capable of perceiving the environment through the sensors connected to it. Unfortunately, the provided definition does not cover all the characteristics that an agent possesses. In this section we break down the definition presented in [2] where the authors define an agent as "an entity which is placed in an environment and senses different parameters that are used to make a decision based on the goal of the entity. The entity performs the necessary action on the environment based on this decision".

- Entity- the type of agent; could be a software, a hardware (sensor) or a combination of both (robot).
- Environment- the place where the agent senses the information for later decision making. The environment itself has multiple features that could affect the complexity of the system, mentioning here accessibility, determinism, dynamism and continuity [3].
- Parameters- the different types of data that an agent is responsible of sensing.
- Action- each agent is responsible of performing a set of continuous or discrete actions that result in changes to the environment.

When put together, *entities* sense and collect *parameters* from the *environment* or from neighbouring agents, in order for the agent to build up knowledge about the environment. Once the knowledge is built, the inference engine decides on the *actions* that need to be taken by the agent. Multi-Agent Systems (MAS) consist of numerous agents collaborating to solve a complex task.

2.1. Overview of MAS Features

MAS share numerous features and characteristics as outlined in [2]:

Leadership: As a leader in MAS we refer to a coordinator like role, that guides the other agents with the main target being the achievement of the system functions. Based on the presence of this role MAS follows two structural paradigms, leaderless and leader-follow. A leaderless system, as the name denotes, operates without this role resulting in its agents having to function on an autonomous basis. In a leader-follow MAS, in most cases, the source of the actions is the leader agent. In case of multiple leaders they communicate with each other to guide the followers.

Decision function: In an MAS decisions can be linear or non-linear. Linearity is mainly determined by the source upon which its information is used to perform decisions. If the decisions of an agent are based on direct information then the MAS is linear. In case the agent's decisions are not affected by incoming information then this is considered as a non-linear MAS.

Heterogeneity: Based on the characteristics of its agents, a MAS is defined as either homogenous or heterogeneous. In a homogeneous MAS all agents have shared characteristics whereas in a heterogenous one the agents do not share the same functionalities.

Agreement parameters: In multiple MAS applications its agents must agree on certain metrics in order to further define the decision making progress. The number of these metrics determines the classification of a MAS namely: first, second or higher order.

Delay consideration: Given the co-dependence of the agents in various scenarios, delays might be faced. If the agents of a MAS expect delay and thus operate based on that, then the system is a system with delay. On the contrary, if the agents of a system cannot operate based on delays then the MAS is considered to be one without delay.

Topology: Based on the typology, agents can be divided into dynamic and static judging from their location and their relations to each-other. As dynamic, they move within a system or even in cases leave and rejoin. As static, they remain in place.

Data transmission frequency: Agents can either send data continuously resulting in a time-triggered MAS or requiring that other agents notify them to send information resulting in an event-triggered MAS.

Mobility: With the dynamicity of agents as the main focus, we can classify them as static or mobile. Static agents remain in their designated position whereas dynamic ones move based on the needs of the system.

2.2. MAS Advantages to Other Systems

MAS is comparatively assessed against other systems such as Object-oriented programming and Expert systems by Wooldridge [3], Zhao et al. [5], and Sadeghi et al. [6]. Expert systems observe and sense the environment, and then perform tasks based on the acquired knowledge. MAS is different due to the way it communicates between its agents in order for knowledge to be accumulated and tasks to be performed. An example of that would be having different results due to the sheer difference in the number of observers. In object-oriented programming objects deliver information and take decisions only from items immediately related to it by public functions. As a result, the frequency of interaction cannot be controlled, whereas in MAS agents can control the flow of information. Additionally, contrary to object orientation, agents in MAS can receive and propagate information from multiple sources.

3. MAS Applications

In this section we give an overview of some of the MAS applications: Smart Cities and Ambient Technologies (section 2.1), Social Networks (section 2.2) and Cellular and Mobile Networks (section 2.3). For each application we include the existing work suggested by the literature, possible voids in the topic and potential improvements or applications for future researchers to investigate further.

3.1. Smart Cities and Ambient Technologies

The Smart City (SC) refers to the place and territorial context where the planned usage of human and natural resources is properly managed and integrated through already available information and communications technology(ICT) [7]. In a city, properly integrated ICT within a network of fixed and mobile telecommunications, can improve the quality of life, can reduce unemployment, boost urbanization while being environmentally and socially sustainable and reduce extra costs and encourage development [8]. One of the technologies for a smart city grid that helps to achieve the mentioned scopes, is Distributed Intelligent Agents. The common goal of these systems is to always keep the network operational and the quality standards of the highest possible service. An ICT infrastructure with such purposes must be capable of providing instant bi-directional communications among devices, which can only be achieved through a distributed architectural framework. In the following section we address some of the main challenges of these systems, such as having increased costs, pollution and congestion etc.

3.1.1. Existing Research Works. In order to address the challenge of increasing costs, pollution and congestion due to unorganized distribution of freight, Khayyat and Awasthi [9] proposed an agent-based method which sheds light on a permanent solution by using six agents known as RFIDG (which stands for retailer, supplier, carrier, network, and city agents). The idea behind this proposal is the usage of RFID tags by the RFIDG agent, to easily handle and manage the supply of resources. When buying goods from a retailer or supplier, the buyer sends the request to one of their agents and upon receiving the request the same agent searches the database for the requested goods. As a next step, the goods are sent to the carrier agent which will ship them to the customer. In order to deliver the goods, the carrier agent follows the optimal path, which is determined by the network agent. Lastly, the city administrator agent informs both the supplier and the customer about relevant policies for the shipping goods.

Controlling the transport system and managing the ongrowing traffic in metropolitan cities is another important challenge. A proposed solution through agent-based methods from Hager [10] suggests that parameters such as fare and people satisfaction are used for analyzing the transportation model. This would be feasible through two groups of agents: travellers and vehicles. The agents would then share knowledge about traffic which is used by other agents at the same time, to decide on the shortest and fastest way toward their destination.

Similarly, MAS is applied in building management. The suggested literature [11] reviews an agent-based method for heating management in buildings, through heterogeneous heating appliances and sensors distributed around the building. The demand agents check the building temperature and pass the collected data to the heater, buffer and heat pump agents. The data from the demand agents are later used by the latter agents to adjust the temperature in the building if needed. After testing this method in an apartment and comparing the results with the centralized heating method, it was noticed that the daily consumption was significantly reduced when using the agent method.

3.1.2. Possible Future Works.

With all the technological advancements in almost every industrial field, the number of integrated sensors into building materials is increasing by the day. Market-leading companies in glass and aluminium processing, engaged in the production of doors, windows and facades have started integrating sensors capable of capturing data related to air humidity levels, pollution indexes, noise levels etc. in the outdoor environment as well as in the inside spaces. Yet, these data are currently being used only for product optimization purposes. Our recommendation would be that there is a great research potential in further looking into how using these agents in the context of MAS could be useful for optimizing pollution and traffic levels in cities, as well as for weather forecasting purposes.

3.2. Social Networks

The number of social networks is increasing on a daily bases in terms of users, daily session-time spent and even in terms of available networks in the market. When put it in a bigger scale, this growth would be attributed to the continuously increasing number of Internet users. According to the literature [11], a social network is comprised of a set of autonomous social actors like users, groups and services, all of which act as agents. The complexity of social networks comes from the large number of users leaving or entering the network, which comes with an exponential growth of connections. Thus, the need to effectively manage such systems arises and MAS is seen as a potential solution to overcoming the complexity of social networks [11]. Through the years, researchers have come up with the three following main views for understanding social networks from a MAS point of view [11]:

1. *The structure-oriented view*: the main focus is on analysing the characteristics of the social actors' interaction of a network's topological structure.

2. *The actor-oriented view*: the main focus is on analysing the characteristics and effects of the behaviour of social actors within the social network environment.

3. *The actor-structure crossing view*: both structure oriented and actor-oriented views are in focus and researchers study how the behaviour of actors shape the network structures and how the network structure can shape user behaviours.

All three of the views find different applications among disciplines, but in this paper, we focus in some of the application areas of the actor-oriented view, where the structure of social networks is not quite strengthened.

3.2.1. Existing Research Works.

Many researchers argue that social networks are an effective structure for multi-agent systems [11], [12]. One of them is also Gatti et al. [13], who proposes a similar approach on how MAS can be used to predict various aspects and functionalities of a social network. The researchers propose the introduction of agents throughout the social network that will collect information related to user behaviour. After performing topic and semantic classification to build specific user profiles, this information can be used to forecast future user behaviours, related to their activities within the network. Yet, social networks do not necessarily need to occur in the cyberspace. Ma and Zhang [14] considered the school system as a social network. The authors attempted the use of MAS in order to adequately allocate funds to various extra-curricular school programs. They modelled every role such as professor and student or other authorities within the system as an agent. Then, the system would assess, based on academic performance, whether the funds were adequately allocated.

3.2.2. Possible Future Works.

The opportunities that come with implementing multi-agent systems in the context of social networks are endless. When focusing in the cyberspace, due to the undeniable growth of the smart-devices industry, everyone who owns such a device transmits endless amounts of data on a daily basis. In the context of search engines or other mobile applications, these data could be used to improve the overall user experience through further personalization and content improvement. To summarize, our recommendation would be to further look into how MAS could be useful for optimizing user experience in the cyberspace.

3.3. Mobile and Cellular Network Applications

Mobile Applications are amongst the most important utility tools in our everyday life and have tremendously upgraded our daily communication. The continuously increasing number of mobile network users and the extensive online activities performed daily, require the implementation of new technologies to improve the quality of the already offered services in mobile and cellular networks. The range of available agents in this context extends from weak (reactive) agents that respond in a stimulus response manner and have minimal need for a representational model of the agent environment, to strong and highly sophisticated agents that need to support rational reasoning in collaborative contexts in order to maintain complex mental states [15]. Multi-agent system approaches have proven to be particularly effective when applied for systems that are comprised by many dynamically interacting components [16], although in the mobile computing arena the utilized agents are of a weak variety.

On the other hand, when it comes to cellular networks, although wireless communication is extensive nowadays, there are still numerous challenges faced starting from the wide variety of data rates, to the high traffic data asymmetry, high energy consumption, low latency requirements in crowded areas etc. [17], [18]. The fifth-generation wireless communication systems (5G) is expected to address some of these challenges due to the integration of different types of cellular networks into a holistic network system, with the primary goal of providing quality of service to its users in an energy efficient manner [19]. Yet, with the emergence of 5G as heterogeneous networks, wireless radio spectrum resources are in high demand, causing a shortage of resources. This means that it is very important to look into new methods of improving the spectrum utilization [20]. One of the most promising solutions that is based on multi-agent architecture, is explained below.

3.3.1. Existing Research Works. *Cognitive Radio (CR) Spectrum Sensing Framework*: According to a 2014 study [20], changing the design of traditional Cognitive Radio Spectrum Sensing based on a multi-agent architecture, would meet the 5G network requirements. Alleviating the problem of scarce radio spectrum is attempted through the Cognitive Radio technology, by using radio spectrum resource management techniques [21]. Spectrum sensing itself is one of the key functions of cognitive radio that prevents the harmful interference between secondary users and primary users and improved the spectrum utilization by identifying the available spectrum [22]. In the traditional technology, the secondary user has the cognitive capability that performs spectrum sensing. The new approach consists on removing this capability from second users and assign the spectrum sensing duty to a new communication entity known as a spectrum agent (SA), with the ultimate goal of reducing energy consumption and wasteful resources, and improving the overall spectrum efficiency [20]. The introduction of SA as a third agent would not only reduce the design complexity of network equipment and improve spectrum utilization in 5G networks, but it would also minimize the cost of hardware. The spectrum agent (SA) itself is thought as independent of the number of users and as part of its function it collaborates with other detection points for an efficient collaborative spectrum sensing. According to the researchers, spectrum agents could be several tiny-base or micro-base stations.

3.3.2. Possible Future Works. Although the proposed multi-agent framework brings great improvements into the spectrum utilization and not only, there are still some voids that remain open for further research. Some of these issues would include: SA deployment, SA detection period and delay, SA scalability, Inter-SA handoff and sensing tasks assignment.

4. Conclusion and future work

Multi-agent systems have a wide application domain due to their high flexibility and as such, the challenges faced are also numerous. In this survey we discussed the diverse aspects of MAS and gave an overview of few MAS applications and challenges encountered in Cellular and Mobile Networks, Smart Cities and Ambient Technologies and Social Networks. We concluded each application with a discussion on the possible future studies that could improve the current voids in the respective applications. We believe that there is great potential in further researching the implementation of MAS in the mentioned areas. We expect this article to serve as an insightful resource to further investigate the possibilities of extending MAS applications beyond their current scope.

References

[1] B. Parasumanna Gokulan and D. Srinivasan, *An Introduction to Multi-Agent Systems*, 07 2010, vol. 310, pp. 1–27.

[2] A. Dorri, S. S. Kanhere, and R. Jurdak, “Multi-Agent Systems: A Survey,” *IEEE Access*, vol. 6, pp. 28 573–28 593, 2018.

[3] M. Wooldridge, *An Introduction to MultiAgent Systems*. John Wiley & Sons, 1995.

[4] S. J. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*. Egnlewood Cliffs, NJ, USA: Prentice-Hall, 2009.

[5] Y. Zhao, G. Wen, Z. Duan, X. Xu, and G. Chen, “A new observer-type consensus protocol for linear multi-agent dynamical systems,” *Asian Journal of Control*, vol. 15, no. 2, pp. 571–582, 2013.

[6] P. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC), “Security and privacy challenges in industrial Internet of Things,” *Asian Journal of Control*, vol. 52, pp. 1–6, 2015.

[7] J. N. Rathod and A. Zaveri, “Use of Multiple Agents for Making the Smart City,” *International Journal of Advance Engineering and Research Development*, vol. 2, pp. 225–234, 2015.

[8] E. Pipattanasomporn, H. Feroze, and S. Rahman, “Multi-Agent Systems in a Distributed Smart Grid: Design and Implementation,” 2009, unpublished.

[9] M. Khayyat and A. Awasthi, “An intelligent multi-agent based model for collaborative logistics systems,” *Transportation Research Procedia*, vol. 12, pp. 325–338, 2015.

[10] K. Hager, J. Rauh, and W. Rid, “Agent-based modeling of traffic behavior in growing metropolitan areas,” *Transportation Research Procedia*, vol. 10, pp. 306–315, 2015.

[11] Y. Jiang and J. C. Jiang, “Understanding social networks from a multi-agent perspective,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2743–2759, 2014.

[12] E. Franchi and A. Poggi, *Multi-Agent Systems and Social Networks*, 01 2011.

[13] M. Gatti, “Large-scale multi-agent-based modeling and simulation of microblogging-based online social network,” *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pp. 17–33, 2013.

[14] L. Ma and Y. Zhang, “Hierarchical social network analysis using multi-agent systems: A school system case,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1412–1419, 2014.

[15] I. Ganchev, S. Stojanov, M. O’Droma, and D. Meere, “(an infostation-based multi-agent system supporting intelligent mobile services across a university campus,” *Journal of Computers*, vol. 2, no. 3, May 2007.

[16] G. O’hare and M. O’grady, “Gulliver’s genie: a multi-agent system for ubiquitous and intelligent content delivery,” *Computer Communications*, vol. 26, no. 11, p. 1177–1187, 2003.

[17] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M. Uusitalo, B. Timus, and M. Fallgren, “Scenarios for 5g mobile and wireless communications: The vision of the metis project,” *Communications Magazine, IEEE*, vol. 52, pp. 26–35, 05 2014.

[18] S. Chen and J. Zhao, “The requirements, challenges, and technologies for 5g of terrestrial mobile telecommunication,” *IEEE Communications Magazine*, vol. 52, no. 5, pp. 36–43, 2014.

[19] C. Wang, F. Haider, X. Gao, X. You, Y. Yang, D. Yuan, H. M. Aggoune, H. Haas, S. Fletcher, and E. Hepsaydir, “Cellular architecture and key technologies for 5g wireless communication networks,” *IEEE Communications Magazine*, vol. 52, no. 2, pp. 122–130, 2014.

[20] Z. Zhang, W. Zhang, S. Zeadally, Y. Wang, and Y. Liu, "Cognitive radio spectrum sensing framework based on multi-agent arc hitecture for 5g networks," *IEEE Wireless Communications*, vol. 22, no. 6, pp. 34–39, 2015.

[21] G. Ding, Y. Jiao, J. Wang, Y. Zou, Q. Wu, Y. Yao, and L. Hanzo, "Spectrum inference in cognitive radio networks: Algorithms and applications," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 150–182, 2018.

[22] I. F. Akyildiz, B. F. Lo, and R. Balakrishnan, "Cooperative spectrum sensing in cognitive radio networks: A survey," *Physical Communication*, vol. 4, no. 1, p. 40–62, Mar 2011.

I8-Testbed: Introduction

Michael Haden, Benedikt Jaeger*, Sebastian Gallenmüller*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: michael.haden@tum.de, jaeger@net.in.tum.de, gallenmu@net.in.tum.de

Abstract—**There are different ways for testing of network technologies. In this rapidly changing area the reproducibility of the performed experiments is especially important in order to be validated for correctness by other scientists. In this paper we present our I8-testbed, its architecture, tools and workflow for fully automated and reproducible network experiments. Special attention is paid to its main part the internally developed plain orchestrating service which manages the testbed orchestration, execution and evaluation to achieve reproducible experiment results. An important aim of this paper is to be a starting point and to facilitate the entry for new users into the creation of own experiments on the testbed.**

Index Terms—**testbeds, measurement, pos, reproducibility**

1. Introduction

In experimental research the right testing infrastructure plays a crucial role in achieving reproducible results. There are various ways for an implementation depending on the requirements of the particular research focus of the chair. We present the testbed of our chair and its main part the pos, the plain orchestrating service. It executes the fully automated workflow for execution and evaluation of network experiments. So far only a brief web documentation of different parts of the testbed and some example scripts exist [1] which in our experience causes a prolonged learning period for a new user. Therefore it is the motivation of this paper to provide a more detailed documentation that gives an overview and acts as an entry point for new users of the testbed to minimize the time spend learning to create own experiments.

The goals of this paper are therefore to provide a general understanding of the architecture of our specific testbed for a new user, how the experiments are automated, orchestrated, executed and what distinguishes it from different approaches. Moreover this paper should provide a good overview of the different functionalities of the testbed orchestration service pos, when each function is used and how a typical experiment workflow looks like. Finally after reading this paper a new user should be able to understand the functioning of other testing scripts and finally to create experiments on their own.

The paper is structured as follows. Section 2 explains basic terms for the understanding of the further sections. Section 3 reviews how the testbed compares to other testbeds. In Section 4 we provide an overview of our testbed architecture including pos. In Section 5 a typical experiment workflow is shown on basis of an example script, before the paper is concluded in Section 6.

2. Background

The ACM distinguishes experiment results into 3 main categories [2]:

Repeatability means that the same experiment result can be repeatedly obtained by the same researcher using the same measurement setup each time. As the weakest classification every well-controlled experiment should fulfill this condition.

Reproducibility has been achieved by an experiment if the same results can also be obtained by a different researcher but using the same measurement setup. This presupposes that the original setup is made available for other researchers by publishing all used tools and scripts or by providing access to the infrastructure that was initially used.

Replicability is the final goal for every experiment result. Different researchers obtain the same measurements by using their own self developed experiment setup which differs from the initial setup.

3. Related work

In this section we provide an overview of the differences between our testbed and some others that specify on reproducible research mainly based on the research done by Nussbaum [3].

A main characteristic of our testbed are the dedicated hosts that are used for every test node together with the live images that are booted for every experiment. This puts it in a common ground with Emulab which also uses a cluster of bare metal systems that are allocated exclusively to users. In contrast to PlanetLab which uses a container technology to share one host between different users resulting in changing testing conditions.

Chameleon, CloudLab and Grid'5000 also support a customized configuration by users like our testbed but not all provide information about its implementation.

Compared to Planet-Lab where the Internet has a direct impact on the experiment result and thus making the results not reproducible [4] Emulab and our testbed are in no influence by outside networks. That means that all traffic which is needed for the experiment needs to be generated by a load generator like MoonGen [5] on one of the test nodes in the testbed.

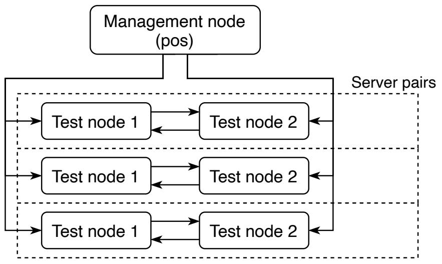


Figure 1: Testbed architecture

4. I8-Testbed

The chair infrastructure we use for our network experiments consists of multiple testbeds with different points of focus. [1] One testbed with a focus on testing Software Defined Networking (SDN) for instance is built up of 15 test nodes (servers) having different network connectivity. [5] Each testbed is structured in a management node and the different test nodes as seen in Figure 1. A dedicated bare-metal server controlled over the Intelligent Platform Management Interface (IPMI) is used for every node instead of using virtualization to prevent a possible distortion of the results by the simultaneous use of a server by several users. IPMI allows the management and control of the server hardware over network also if powered off or unresponsive and therefore eliminates the need for being physically present in front of the server. To encourage reproducibility new images of the desired operating systems are live booted for every test to get reproducible results. A reboot therefore leads to a fresh operating system and a loss of all data on the node but also forces users to plan out their experiments including configuration and result gathering as scripts. Which in turn has the advantages that every experiment by itself is fully repeatable and every test node stateless which results in less administrative workload to maintain each system. Each authorized user has a home directory and remote ssh access to the management node and root access on the test nodes which enables others to replicate the experiments. On the management node runs the management tool pos (plain orchestrating service) daemon which follows the overall sequence of allocating the needed hosts, orchestrating the experiments and collecting the experiment data. The pos daemon offers a REST-API which can be accessed by using the python library `posapi` or its command line interface `pos cli`, both accessible from the management node. They allow reservation, allocation, configuration and management of test nodes and are used to launch commands on them. From the test nodes themselves the communication with the pos daemon happens through the python library `postools` or via the `pos_*` commands on the command line. They are used to synchronize test nodes, start, kill or wait for an additionally process in the background or transferring test results or files that are needed for the experiment between the management node and the test nodes. The configuration of the experiments happens over variables that are set for the experiment to make the configuration easier.

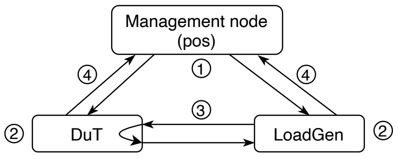


Figure 2: Typical experiment setup [5]

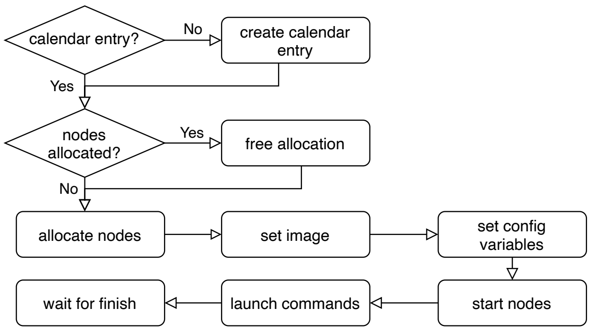


Figure 3: Typical node allocation workflow

5. Workflow of the I8-Testbed

This section provides a high-level overview of the workflow of a typical experiment from setup, execution to evaluation, first from the management node side as seen in Figure 3 and then from the side of the testing nodes as seen in Figure 5. A typical experiment setup consists of two test nodes, one of which is the LoadGen with a load generator like MoonGen [5] and the other the DuT as seen in Figure 2. In the following example we will explain an experiment to measure the impact of restricting the CPU frequency on the network latency between two nodes. After the management node has setup the nodes ① and the experiment is configured ② , one node, the LoadGen, will generate traffic that is sent to the other node, the DuT ③ , who forwards the received traffic back to the sender. The latency of the packets is measured and the results are transferred back to the management node ④ . This setup is often used in various ways to benchmark software like firewalls for latency or data rate of the connection. The corresponding commands that are referenced in the following sections for the management node and the two test nodes are to be found in Listing 1, Listing 2 (DuT) and 3 (LoadGen) respectively. The references per line of the test node scripts follow the pattern (DuT, LoadGen) or (D) (L) e.g. (7,4) or (D7) (L4) when referencing to line 7 in the DuT and line 4 in the LoadGen script.

5.1. Management node

Access to testbed. To get access to the chair infrastructure including the testbed, a chair account is needed which will be created together with the user's advisor. The account is created and the user gets the corresponding password. Due to security reasons the infrastructure of the chair itself is only accessible mostly via ssh key instead of 'password only'. Therefore it is necessary to upload a ssh key to the ssh gateway where it is added into the central ldap.

Reservation. To allow a simultaneous use of the different nodes of a testbed by several users and a better organization of the resources over time a reservation for needed

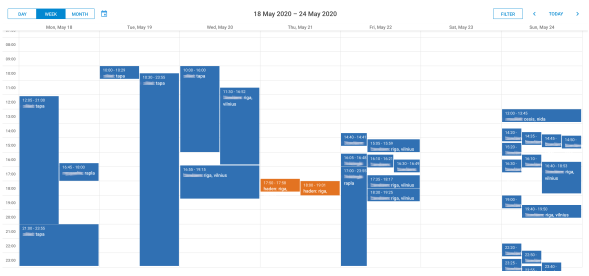


Figure 4: Web calendar reservation [6]

nodes is mandatory for every experiment. This reservation can be done by authorized users through a calendar web view as seen in Figure 4 where all reserved nodes and their users are listed, by using the pos cli on the used testbed itself or by specifying a duration when creating the allocation. The reservation can be extended later on. To get familiar with the system it is desirable to use the nodes with 1 Gbit/s connection because of their low usage. The nodes are getting reserved for the time of the calendar entry during which a claim of the node allocation (which follows shortly) by other users (except admins) is prevented. After the reservation has expired the allocation can be freed by everyone.

Node allocation. With a valid calendar entry the allocations of needed nodes can now be freed if still allocated by their last user (1) and a new allocation can be created with the pos cli (3). An allocation ID and the path to a new corresponding directory for this experiment will be created and returned. By using the ID more nodes can be added later. Admins can also free any reservation by using the `--force` flag.

Configuration. The configuration of the experiment works by using YAML formatted files that are referenced at with the pos cli (4). For most use-cases only simple key value configuration is needed. The variables of the configuration will then be loaded to the specified nodes where they can then be accessed by the nodes with the pos cli. The variables can be either set as local or global.

Image. Because of the live booted operating system the desired image of each node needs to be selected and started (7). There are multiple testbed images available including the most current versions of standard distributions which are updated regularly. Usually a regular Debian image is enough for the experiments. Custom images are also supported by using mandelstamm [7] which is a collection of scripts for building custom images for pos-controlled testbeds.

Booting. The management node can now transfer the selected image to the nodes and boot them by using PXE boot. When `reset` is specified the nodes will be stopped if running and then booted (9).

Command launching. The experiment itself on the test nodes consists of bash or python scripts which contain the commands that shall be executed on the nodes. These scripts are selected via the pos cli (11), automatically transferred to the defined nodes and executed. The commands can be launched as blocking, non-blocking or queued. Now the test nodes run the experiment as described in the next section. When invoked as non-blocking or queued a command ID is returned which can be used to wait for the commands to finish (13).

After the experiment is done the used nodes are stopped and their allocation will be freed (14). The experiment results are now available on the management node where they can be further processed and analyzed.

Evaluate results. The results are placed in the directory as stated at the allocation with a root directory that is named with a unique timestamp of the experiment and under that is the configuration of the experiment and the uploaded results which make then the experiment easier to reproduce. On the second level are folders for configuration, each node that was involved in the experiment and energy measurements if the node supports them. `Config` contains values that define the configuration of the nodes like kernel version, physical address and the configuration variables that were set for the experiment. This is essential to replicate the results with identical resources and configuration. The test node directories contain the program code that was used for the experiment, the produced output of the program and the actual results that were specified to be uploaded. If energy measurements were started during the experiments they are also in a separate directory.

Listing 1: Experiment script on management node [8]

```
1  pos allocations free NODE1
2  pos allocations free NODE2
3  pos allocations allocate NODE1 NODE2
4  pos allocations variables NODE1 PATH
5  pos allocations variables NODE2 PATH
6  pos allocations variables NODE2 PATH --as-global
7  pos nodes image NODE1 debian-buster
8  pos nodes image NODE2 debian-buster
9  pos nodes reset NODE1
10 pos nodes reset NODE2
11 pos commands launch --infile PATH NODE 1 --
     queued
12 pos commands launch --infile PATH NODE 2 --
     queued
13 pos commands await ID
14 pos allocations free ALLOCID
```

5.2. Test node

The scripts on both nodes are now executed and start their procedure.

Setup system. First, the shell is configured to log every command to be able to retrace the sequence of events later (1) and exit when an error occurs (2). Afterwards, various system specific variables of the DuT are logged like the current kernel version (3) that might be of interest in the experiment evaluation later.

Both nodes then clone the program code that is used for the experiment (8,5), in this case *libmoon* and the tool *MoonGen* which builds upon it. A specific branch is checked out by fetching the corresponding variable that was set beforehand (10,7) and the hash of the exact git commit is written back (11,8) to the management node. Important configuration variables like the exact commit

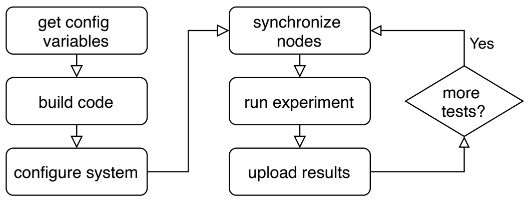


Figure 5: Typical experiment workflow

that is used are again logged for future reference. It should be noted that the access of allocated variables creates traffic on the management interface and should therefore be only used between measurements or during setup to not distort the experiment results. Now the code on both nodes can be freshly built (12,9) and the DuT is configured for the experiment by disabling turbo boost and locking the frequency of the CPU (14) on a value specified in the configuration. Additional variables like the network interfaces that are to be used and the measurement period are fetched. By specifying the minimum and the maximum transfer speed that should be tested in combination with the delta that shall be used between the tests, the LoadGen calculates the individual speed steps that are to be tested in this experiment (L14).

Synchronize nodes. The steps are then synchronized with the DuT by setting the corresponding variable from the LoadGen (L15) and initiating a sync on both nodes by using an instruction that will only return when all nodes called it (20,16). This ensures that the DuT can receive the variable of the steps (21) that was set from the LoadGen which completes the experiment setup.

Run experiment. Now the actual experiment can be started which is divided into multiple test runs with a different transmission speed as specified in the speed steps. Each step begins by specifying a location of the experiment results on the LoadGen named after the current step (L19) and a synchronization of the two nodes (24,20). The freshly built programs can then be launched in the background with the defined configuration (25,22). The LoadGen starts a Lua script that generates traffic with a specified packet rate on a defined port, receives it again on a different one and writes the latency of the received packets to a file. The DuT also executes a Lua script of libmoon that simply forwards the traffic that it receives back to the transmitter. When running the scripts a command id is specified, named after the current step, which is later used as reference point to the launched command.

Upload results. After the measurement period has expired (L24) the LoadGen kills the program per command id (L25) and uploads its results back to the management node (L26). A sync is then initiated (2,27) which informs the DuT that this test run is finished and the program on its side can also be killed (D28). The experiment is now repeated with the rest of the steps that are to be tested after which the script is done and exits.

Listing 2: dut.sh [9]

```
1  set -e
2  set -x
3  pos_set_variable host/kernel-version $(uname -v)
4  pos_set_variable host/os $(uname -o)
5  pos_set_variable host/machine $(uname -m)
6  GIT_REPO=$(pos_get_variable git/repo)
7  DUT=libmoon
8  git clone --recursive $GIT_REPO $DUT
9  cd $DUT
10 git checkout $(pos_get_variable git/commit)
11 pos_set_variable git/commit-hash $(git rev-parse
       --verify HEAD)
12 ./build.sh
13 ./setup-hugetlbfs.sh
14 echo 1 >   /sys/devices/system/cpu/intel_pstate/
       no_turbo
15 echo $(pos_get_variable cpu-freq) > /sys/devices
       /system/cpu/intel_pstate/max_perf_pct
16 echo $(pos_get_variable cpu-freq) > /sys/devices
       /system/cpu/intel_pstate/min_perf_pct
17 RUNTIME=$(pos_get_variable runtime --from-global
       )
18 PORT_TX=$(pos_get_variable port/tx)
19 PORT_RX=$(pos_get_variable port/rx)
20 pos_sync --tag sync_steps
21 STEPS=$(pos_get_variable --remote --from-global
       steps)
22 pos_sync
23 for STEP in $STEPS; do
24     pos_sync
25     pos_run dut$STEP -- ./build/$DUT examples/l2
           -forward.lua \
26     $PORT_TX $PORT_RX
27     pos_sync
28     pos_kill dut$STEP
29 done
```

Listing 3: loadgen.sh [10]

```
1  set -e
2  set -x
3  GIT_REPO=$(pos_get_variable git/repo)
4  LOADGEN=MoonGen
5  git clone --recursive $GIT_REPO $LOADGEN
6  cd $LOADGEN
7  git checkout $(pos_get_variable git/commit)
8  pos_set_variable git/commit-hash $(git rev-parse
       --verify HEAD)
9  ./build.sh
10 ./setup-hugetlbfs.sh
11 RUNTIME=$(pos_get_variable runtime --from-global
       )
12 PORT_TX=$(pos_get_variable port/tx)
13 PORT_RX=$(pos_get_variable port/rx)
14 STEPS=$(calc_steps)
15 pos_set_variable --as-global steps $STEPS
16 pos_sync --tag sync_steps
17 pos_sync
18 for STEP in $STEPS; do
19     OUTFILE="/tmp/histogram${STEP}.csv"
20     pos_sync
21     sleep 2
22     pos_run loadgen$STEP -- ./build/$LOADGEN
           examples/l2-load-latency.lua \
23     $PORT_TX $PORT_RX -r $STEP -f $OUTFILE
24     sleep $RUNTIME
25     pos_kill loadgen$STEP
26     pos_upload $OUTFILE
27     pos_sync
28 done
```

6. Conclusion

The I8-Testbed with pos simplifies the whole testing workflow and eases the realization of reproducible experiments by providing a reliable orchestration for performing and evaluating network experiments. A main problem was that the introduction to reproducible research on our testbed for new users may be difficult because no good entry point and overview was present. We believe we have improved it by starting with presenting the meaning of different reproducibility terms and the characteristics and differences of our testbed in comparison to others. By proving a thorough explanation of the general architecture of the testbed, pos and the interaction between parts of the testbed in combination with a typical workflow we think that a new user gets a good entry and overview into working with our testing infrastructure. Moreover a typical example script that is the foundation for most of the experiments was provided and explained in detail which gives a good overview over all functions of pos, should enable a new user to understand additional scripts and finally implement own experiments on the basis of this one on the testbed.

References

[1] "I8-testbeds Wiki," https://gitlab.lrz.de/I8-testbeds/wiki/-/wikis/home, 2020, [Online; accessed 03-June-2020].

[2] "Artifact Review and Badging," https://www.acm.org/publications/policies/artifact-review-badging, 2020, [Online; accessed 08-June-2020].

[3] L. Nussbaum, "Testbeds support for reproducible research," in *Proceedings of the Reproducibility Workshop*, ser. Reproducibility '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 24–26. [Online]. Available: https://doi.org/10.1145/3097766.3097773

[4] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using planetlab for network research: Myths, realities, and best practices," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, p. 17–24, Jan. 2006. [Online]. Available: https://doi.org/10.1145/1113361.1113368

[5] 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.

[6] "baltikum Calendar," https://kaunas.net.in.tum.de/, 2020, [Online; accessed 15-June-2020].

[7] "mandelstamm - Create OS images for the pos-controlled testbeds," https://gitlab.lrz.de/I8-testbeds/mandelstamm, 2020, [Online; accessed 08-June-2020].

[8] "simple-moongen setup.sh," https://gitlab.lrz.de/I8-testbeds/pos-examples/-/blob/master/tutorials/simple-moongen/bash/setup.sh, 2020, [Online; accessed 15-June-2020].

[9] "simple-moongen dut.sh," https://gitlab.lrz.de/I8-testbeds/pos-examples/-/blob/master/tutorials/simple-moongen/bash/commands/dut.sh, 2020, [Online; accessed 15-June-2020].

[10] "simple-moongen loadgen.sh," https://gitlab.lrz.de/I8-testbeds/pos-examples/-/blob/master/tutorials/simple-moongen/bash/commands/loadgen.sh, 2020, [Online; accessed 15-June-2020].

Network Simulation with ns-3

Niklas Kuse, Benedikt Jaeger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: niklas.kuse@tum.de, jaeger@net.in.tum.de

Abstract—**As computer networks become more complex, agile and flexible methods for research are required, allowing for a faster development of new network technologies. Network simulation provides this flexibility. This paper gives an overview on the network simulator ns-3. In addition to that, ns-3 is compared to OMNeT++ another well known network simulator and other measurement methods using real or emulated hardware. This paper references current projects which were realized using ns-3 to give an overview of the various application areas of ns-3.**

Index Terms—**network simulation, ns-3, network emulation, comparison, testbed**

1. Introduction

Today's global network infrastructure is growing faster and faster. In order to keep up with the growing data demand, new and improved techniques to send data through those networks are required. That can be achieved through more and optimized research in this field. A popular approach is network simulation, providing a universal applicable and cost effective way. Cost-effective in terms of manpower, material and working time as development using network simulation brings benefits for all these aspects. In addition, research results using network simulation are deterministic and can be validated by others, by sharing the simulation setup. Another way is to build a network topology using real world hardware. This is not as flexible and cost effective as network simulation. This paper gives an overview on the network simulator ns-3 including its main structure. In addition to this, other in the research well known approaches are named and compared to ns-3. To help to decide if ns-3 is the best research tool for a specific question, this study will provide an overview on current research using ns-3 in their evaluation.

The structure of this paper is as follows: Section 2 provides related literature. Section 3 explains the main structure and design goals of ns-3. To get an idea of the overall performance of ns-3, Section 4 compares ns-3 to OMNeT++, real and emulated hardware and gives insights into the implementation of ns-3. Section 5 names work of different research areas all using ns-3. Finally, Section 6 summarizes the main statements of this study and states further work that can be done to support the conclusions drawn.

2. Related Work

ns-3 is not the only network simulator available. The book [1] by Wehrle et al. provides a good overview on contemporary simulators. The book contains code snippets explaining many use cases. Additionally, the authors provide insights into the implementation of the simulators ns-3, OMNeT++ and others. This allows a better understanding of the use cases the simulators are aiming at.

To get more information on the performance of the simulators ns-3 and OMNeT++, the author of this paper recommends taking a look into the original papers by Weingartner et al. [2], Rehman Khana et al. [3] and Khan et al. [4]. These papers do not only compare ns-3 and OMNeT++, but other simulators as well, allowing to get a better overview of all simulators available. All three papers determine ns-3 as the most performant simulator of the ones tested.

ns-3 can also be used in cooperation with other software. Gawłowicz et al. picture a good example on such cooperation in [5]. With their paper they contribute ns3-gym. A framework composed of ns-3 and OpenAI supporting the research for better networking protocols using reinforcement learning.

3. Structure and Functionality of ns-3

ns-3 is a network simulator targeting research and educational usage [6]. The simulator is implemented in C++ and provides bindings for Python, backing much of the C++ API [7]. Therefore, simulations for ns-3 have to be written in one of these languages as well. Allowing executed tests to be more easy to debug [8, section 2.1] and more realistic than simulations realized in a higher abstraction level like the predecessor ns-2 does. Simulations are more realistic, because the C++ code will be executed and no extra level of complex interpretation has to take place. By avoiding this extra level of abstraction, the implemented code of the simulation is closer to the real world implementation used later. Simulation is realized by processing a discrete list of all events about to occur, sorted ascending by their occurrence time [8].

ns-3 provides several model types allowing the user to specify all the different components of a network. To run the simulation, the user has to create all network parts needed for the simulation with respect to the following categories:

- Nodes, used for physical network systems e.g. smartphones or switches.

- Devices, representing the actual part of a network node that is responsible for network communication like the ethernet card.
- Channels, illustrating cabels and other mediums used to transfer data in the real world.
- Protocols, describing the actions carried out on each network packet.
- Headers, organizing the data stored in packet headers as required by network standards.
- Packets, used to depict the data passed over the network with header information and payload.

Besides these components ns-3 provides more objects helping with the simulation by providing random numbers or storing additional required information [8]. As described above, the simulator works by processing a sorted list of events. To achieve this, the network topology, that is about to be simulated, has to be created using the network parts named above. Initial events have to be declared that will start the simulation and may result in the creation of further events. To schedule events the event occurrence time, an event handler and all its needed parameters are handed over to the simulator. Once the simulation is started, the simulator iterates over the list of sorted upcoming events. For each event, the simulation time is adjusted with the event time and afterwards the corresponding handler is called to process it. Events that get triggered by this handler during simulation, will be added into the simulators list of upcoming events and processed accordingly. As the simulator hops from event to event, the processing time is not in correlation with the real time. The simulation stops after a specified simulation time, or if the list of upcoming events in the topology runs out [8]. Because ns-3 is aimed towards research, it is important that test results can be gathered in any way required. To fit this constraint, ns-3 includes a tracing subsystem, allowing to measure and log any data. Data collection is supported in common data formats like *pcap* that can be analysed with packet analyzer software like Wireshark. For a more customized data collection, the trace sinks, parts of the tracing subsystem for data consumption, can be edited, to support any format of data output to meet the users need. Besides these analysis tools, ns-3 ships with software supporting the visualised debugging. The AnimationInterface can be incorporated in the siumlatiom to collect test result data for debugging. After finalizing the simulation, this data can be viewed as a graphic, showing the packet flow in the simulations topology [8]. All these features enhance ns-3 to be a good simulator to fit many research and educational needs.

4. Comparison with other Methods

Besides ns-3, other network simulators like OMNeT++ and even other approaches using emulated or real hardware exist. This section will evaluate the major differences to these concepts.

4.1. OMNeT++

OMNeT++ is a discrete event based simulator like ns-3. Just as ns-3, OMNeT++ is written in C++, but simulations are modelled using NED, a description language translated into C++ during simulation. OMNeT++ does not focus on network simulation only, allowing other event based simulations to be modelled and executed as well [2]. This is because OMNeT++ was originally not designed as a network simulator. Nevertheless, simulations are modelled using different approaches. Both tools can run similar simulations as shown by Weingartner et al. in [2]. Due to this, comparing the performance of both is crucial to decide which one to use for research. As shown in Figure 1, both tools perform simulations in linear computation time regarding the network size with ns-3 being slightly faster than OMNeT++. The second performance test executed by the authors of [2] investigated the memory usage. Their results, as shown in Figure 2, indicate a better memory usage of ns-3 compared to OMNeT++.

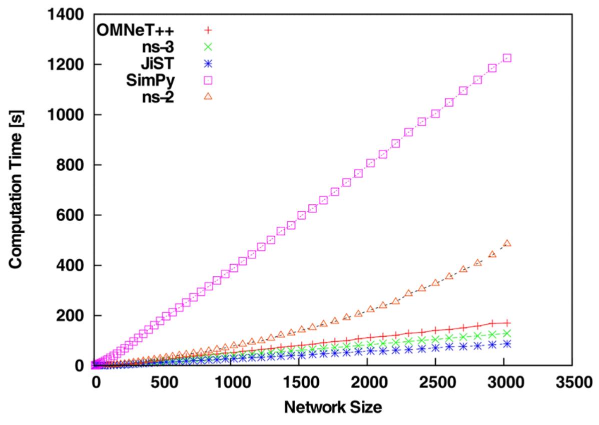


Figure 1: Simulation runtime vs. network size [2]

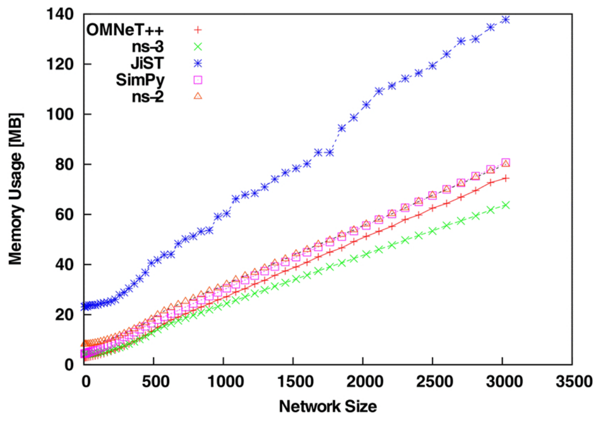


Figure 2: Memory usage vs. network size [2]

Comparing these results to the ones measured by Rehman Khana et al. in [3] and Khan et al. in [4] for ns-3 and OMNeT++ shows that ns-3 is able to maintain its advantage over OMNeT++ in different simulations. When evaluating these performance measurements today, it should be kept in mind that the three tests have been performed between 2009 and 2014. Today's versions of both simulators might behave different due to OS and software optimizations.

Another advantage of ns-3 is the implementation of packets. As packets in a network are carrying payload, this has to be possible in the simulation as well. For many simulation use cases the actual payload itself does not matter, but only its size. Therefore, dummy payload is only saved as a number representing the size used. As packets are subject to changes in a real world network, packets in ns-3 have to be as well. E.g. protocol headers are added or removed in different processing stages. Because ns-3 only passes pointers of packets around, changing the data for this pointer would as well result in changes at earlier stages of the packet, which are not intended. To avoid this, ns-3 enables a copy-on-write policy. This means, packets are duplicated, if a function writes to the memory at a pointer it has received from others [8]. Copy-on-write allows the programm to copy as little memory as possible. This again results in an optimized memory usage. In OMNeT++ packets are represented by C++ classes containing all carried information. In addition, each packet contains a pointer to its payload (the wrapped packet). Contrary to ns-3, OMNeT++ includes a copy-on-access policy to reduce the copy of unused packets [9, section 3.3.5].

4.2. Real Hardware

In addition to running network simulators, measurements can be performed using real hardware composed in a testbed. Using real world hardware allows the researcher to execute protocols and other software to be tested in their native environment. This results in more accurate test results, containing information a simulator could not determine [10] like random interference on wireless networks. Most simulators make assumptions (on the environment or based on random numbers) while simulating the network topology [11]. As network simulators are mainly constructed for research, they allow tests to be deterministic [11]. This constraint is highly required in scientific research allowing other researchers to verify the claimed statements. In contrast to this, testbeds using real hardware can not fit this constraint. As described by the author in [11] testbeds using wireless communication have to be shielded against external radiation. As this might be hard in larger scale networks, the author describes that their testbed runs tests during the night when less devices are emitting. In addition to that, their testbed allows to be reset to any point of the experiment, if unexplainable results are gathered afterwards. Adding such features allows the test results to be more reproducible, but not deterministic. Another drawback using real hardware is that test enviroments can not be scaled as easy as in network simulators. In order to scale networks, additional hardware has to be bought and added, resulting in network simulation to be more affordable [3]. These expenses require experiments to be planned more accurately in advance to minimalise experiment costs. Additionally, these expenses make it harder for other researchers to execute and verify the tests. Observing experiments in testbeds can be challenging as well, because test results can not be transferred using the same communication channel as the test data. Using the same channel, the test results would interfer with the test experiment resulting in non-determinism. This requires the data to be gathered and evaluated after the completion of the test or adding another communication channel [11]. Besides testing in testbeds, a researcher can perform active or passive measurements in a real world network e.g. the internet. Using active measurements, the topology of a network part is analysed by sending packets to other machines and determining the transfer rate or elapsed time. In contrast to this, passive measurement describes oberserving the traffic and packets, passing a specific node without sending packets [12]. As these measurements do not require to build a new network, it comes with advantages in costs. In addition, results are gathered on a real network making them representative. On the other hand, results can vary according to the workload in the network.

4.3. Emulated Hardware

Besides network simulation and testbeds using real hardware, there also is a method taking parts of both. Research approaches using emulated hardware would result in a simulator controlling the experiment, but sending the data over real, physical hardware [11]. Emulated hardware allows tests to be more affordable than using only real hardware. Nevertheless it has the same drawback with respect to non-determinism, because data is transferred using the same real world channels as approaches using only real hardware. ns-3 supports such a research method, as it can by used as the simulator sending its data over the hardware [8].

5. ns-3 in Research

As mentioned above, ns-3 is a network simulator aiming towards research. This section provides an overview of recent work using ns-3. All these approaches have some specific requirements that differ from the classic ones which most common networks do not have. As ns-3 is able to match all these requirements, this overview should help to decide, whether ns-3 is the correct tool for a research task or not. **Quantum key distribution.** ns-3 is used by Mehic et al. in their paper [13] providing a module for quantum key distribution. Quantum key distribution requires a specific topology containing two communication channels. One referred to as 'Quantum channel' is used for the transfer of quantum key material. The other one is used for the transmission of encrypted data and verification. As described by the authors, the quantum channel always has to be a point-to-point connection. Additionally the network has to meet some specific security requirements. Quantum key distribution on real hardware is limited to a transmission distance of about 100km using optical fibres. Building a testbed using this technology is expensive as well. As a result of this and the fact that ns-3 can deal with all requirements, follows that ns-3 is a good choice for such research tasks [13]. It should be mentioned that the module described by the authors is made available with their paper. **Electric vehicle.** According to Sanguesa et al. in [14], network simulation becomes more and more important for electric vehicle development. This is, because simulating scenarios for vehicular networks is too expensive to be done on real hardware. As cars become more and more

interconnected, new and improved ways for this communication have to be found. This connection takes place to enable better traffic management or autonomous driving. ns-3 provides a perfect base for the development of such protocols. Electric cars are dependent on electric energy for all their operations. To provide a realistic simulation of these vehicles, the authors of [14] provide a module for energy consumption measurement in their paper. This module provides a helper class to create an energy consumption model for each node in the ns-3 simulation. This module takes into account all important information like weight or battery level during its evaluation. To proof the good accuracy of their module, the results are compared to a similar simulation using the traffic simulator SUMO. For taking into account that the nodes can move around dynamically as it might be required, the next paper FlyNetSim provides a good approach.
FlyNetSim. ns-3 in not only a stand-alone network simulator, but rather capable to incorporate with other software, as shown in [15]. The authors use ns-3 in cooperation with Ardupilot, an open source software for unmanned vehicles navigation. With their paper they provide FlyNetSim. A simulator capable of simulating multiple unmanned aerial vehicles. ns-3 enables their tool to provide simulations for different communication ways between the vehicles, using LTE or device-to-device communication, taking into account that the vehicles are moving and distances between nodes are changing. The authors choosed ns-3 because of its interfaces for external systems. A challenge the authors had to deal with is the different time management used by ns-3 and Ardupilot. As mentioned, ns-3 is a discrete event simulator. As Ardupilot performs operations in real time, incorporating both software is challenging. However, binding ns-3 to a real-time scheduler allowed both ns-3 and Ardupilot to incorporate with only a lack of accuracy in situations with many events in a short amount of time [15].
ns3-gym. ns-3 can not only work in cooperation with other software, as described above, but rather simulations can be controlled by external software, allowing for even more complex cooperations. The authors of [5] have created a framework that combines ns-3 and OpenAI to support research for optimized networking algorithms using machine learning based on reinforcement learning (RL). Therefore, they incorporate ns-3 and OpenAI using a custom middleware. This middleware is used, to enable OpenAI to control specific nodes in the simulation. As RL works by breaking the situation into steps executed on after another, the middleware allows to start and stop the simulation after such step in order to report the state back to OpenAI. This allows OpenAI, to evaluate and improve the learned after each step. In addition, ns3-gym allows to evaluate knowledge gained from simulations to be executed on real hardware afterwards, to improve know-how in real world situations.

6. Conclusion and Further Work

A main goal of network simulation is to provide a platform to develop and test new communication technologies. ns-3 fits this requirement by providing a framework that can be extended by anybody allowing it to become more and more robust. Another advantage of ns-3 is the simplicity to scale topologies and execute deterministic simulations. Both are constraints to develop new robust and reliable technologies. Approaches using emulated or real hardware are not able to match. On top of that, ns-3 uses good approaches to reduce mistakes done by the developer during the simulation development. This avoids the occurrence of memory leaks during simulation execution and results in an optimized memory usage. Comparing ns-3 to OMNeT++ shows that it is more performant than the latter, even though both are implemented using C++. As shown in Section 5 modules for many research fields are available for ns-3 simplifying its usage. In the authors opinion, this enables ns-3 to be considered in many research evaluations using any kind of network. Thus, many research fields can profit from the benefits ns-3 provides.

As mentioned in Section 4.1 the cited performance evaluations of OMNeT++ and ns-3 were all performed between 2009 and 2014. Due to the possible performance improvements in OS and simulator, these results might not be representative anymore. To get insights into their today's performance, some additional test would be required. To aquire such information, performance test using today's OS and simulator versions and the ones of the cited paper could be performed. Running the same simulation structure in the old and current version would allow to determine the performance improvements done in the last years, resulting in a better understanding of today's accuracy of the results drawn from the cited papers. Back then, the cited papers were produced five years apart and the performance comparison was quite similar. This leads us to the conclusion that the performance comparison would not differ significantly today.

References

[1] K. Wehrle, M. Güneş, and J. Gross, *Modeling and tools for network simulation*. Berlin, Heidelberg: Springer, 2010.

[2] E. Weingartner, H. vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *2009 IEEE International Conference on Communications*, June 2009, pp. 1–5.

[3] A. u. Rehman Khana, S. M. Bilalb, and M. Othmana, "A performance comparison of network simulators for wireless networks," 2013.

[4] M. A. Khan, H. Hasbullah, and B. Nazir, "Recent open source wireless sensor network supporting simulators: A performance comparison," in *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, Sep. 2014, pp. 324–328.

[5] P. Gawłowicz and A. Zubow, "Ns-3 meets openai gym: The playground for machine learning in networking research," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWIM '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 113–120. [Online]. Available: https://doi.org/10.1145/3345768.3355908

[6] "Ns-3 homepage," https://www.nsnam.org/, [Online; accessed 06-June-2020].

[7] "Using python to run ns-3," https://www.nsnam.org/docs/manual/html/python.html, [Online; accessed 14-June-2020].

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

[9] A. Varga, *OMNeT++*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 35–59. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_3

[10] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Ya Xu, and Haobo Yu, “Advances in network simulation,” *Computer*, vol. 33, no. 5, pp. 59–67, 2000.

[11] B. Blywis, M. Guenes, F. Juraschek, and J. H. Schiller, “Trends, advances, and challenges in testbed-based wireless mesh network research,” *Mobile Networks and Applications*, vol. 15, no. 3, pp. 315–329, 2010. [Online]. Available: https://doi.org/10.1007/s11036-010-0227-9

[12] V. Mohan, Y. R. Janardhan Reddy, and K. Kalpana, “Active and passive network measurements: A survey,” *International Journal of Computer Science and Information Technology*, vol. 2, pp. 1372–1385, 2011. [Online]. Available: http://ijcsit.com/docs/Volume%202/vol2issue4/ijcsit2011020402.pdf

[13] M. Mehic, O. Maurhart, S. Rass, and M. Voznak, “Implementation of quantum key distribution network simulation module in the network simulator ns-3,” *Quantum Information Processing*, vol. 16, no. 10, p. 253, 2017. [Online]. Available: https://doi.org/10.1007/s11128-017-1702-z

[14] J. A. Sanguesa, S. Salvatella, F. J. Martinez, J. M. Marquez-Barja, and M. P. Ricardo, “Enhancing the ns-3 simulator by introducing electric vehicles features,” in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, July 2019, pp. 1–7.

[15] S. Baidya, Z. Shaikh, and M. Levorato, “Flynetsim: An open source synchronized uav network simulator based on ns-3 and ardupilot,” in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWIM ’18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 37–45. [Online]. Available: https://doi.org/10.1145/3242102.3242118

An Overview of OS Fingerprinting Tools on the Internet

Ruoshi Li, Markus Sosnowski, Patrick Sattler*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: ruoshi.li@tum.de, sosnowski@net.in.tum.de, sattler@net.in.tum.de

Abstract—**Operating System (OS) fingerprinting is an important technique that can be used to identify OS. Currently, there are many fingerprinting tools on the internet. This paper offers an overview of some popular tools for OS fingerprinting including Nmap, Xprobe, p0f, Satori etc. We introduce the differences between their mechanisms and information gains. We also introduce some papers that show the automation of OS fingerprinting is possible. The accuracy and efficiency of the fingerprinting technique can be improved using machine learning.**

Index Terms—**operating system fingerprinting**

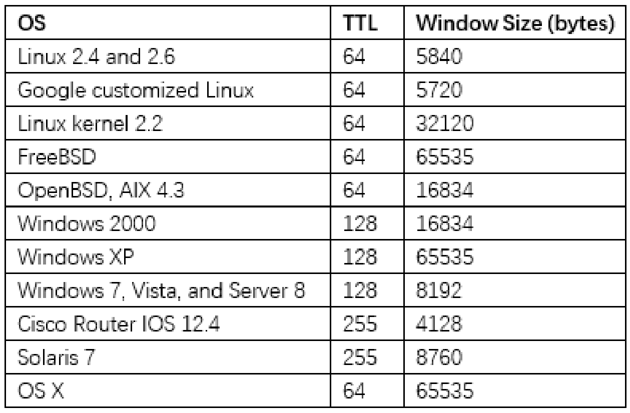
1. Introduction

Operating system (OS) fingerprinting can recognize OS that is running on the host. It can be used in the enterprise network to detect malware, which works through virtual machines [1]. OS fingerprinting is also an important skill in the penetration test, which is performed to assess system security. Since different OSs have their unique vulnerabilities, the attack method can be determined after the OS is identified. Another use of OS fingerprinting is to detect outdated OS versions that contain vulnerabilities [2]. It is also useful in generating network statistics and research.

This paper provides an overview of some popular OS fingerprinting tools. In section 2 of this paper, some necessary background knowledge is introduced. In section 3, the mechanism of some popular tools for OS fingerprinting and the information that can be gained with them are briefly introduced. These include active fingerprinting tools (Nmap, Xprobe2, RING), passive fingerprinting tools (p0f, Ettercap, Satori, NetworkMiner) and other tools that can be used for fingerprinting (SinFP, ZMap, Scapy). Since most of them work manually, some research focusing on automation of OS fingerprinting is also introduced in section 4. These papers show that the process of gathering, normalizing information, and classifying OSs can be automated with higher accuracy. Finally, a brief conclusion is in section 5.

2. Background

The name of OS fingerprinting vividly describes the way it works. Just like we can uniquely identify a human through the human fingerprint, we can identify an OS by the information contained in the packets it sends. For example, the values of Time-To-Live (TTL) field are one of the indicators. While in the RFC 791 a prescribed value for TTL has not been defined, different OSs then determine their own TTL values [3]. A recommended default TTL is given in RFC 1700 [4], but it is not followed in most implementations. A signature can then be generated uniquely from integrating the different values for every single OS. Fig.1 shows the TTL values and window size values of popular OSs. Through comparing the generated signature to the signatures stored in the database, an OS can be identified.



OS	TTL	Window Size (bytes)
Linux 2.4 and 2.6	64	5840
Google customized Linux	64	5720
Linux kernel 2.2	64	32120
FreeBSD	64	65535
OpenBSD, AIX 4.3	64	16834
Windows 2000	128	16834
Windows XP	128	65535
Windows 7, Vista, and Server 8	128	8192
Cisco Router IOS 12.4	255	4128
Solaris 7	255	8760
OS X	64	65535

Figure 1: Popular OSs' Time-To-Live (TTL) and window size values [1]

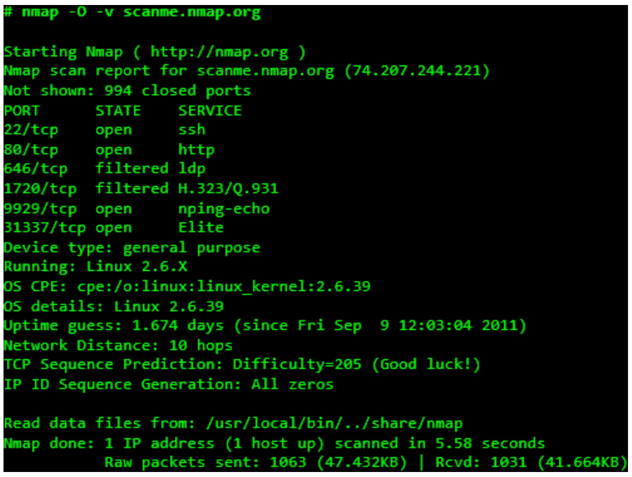
OS fingerprinting has two types: active and passive. With active OS fingerprinting, specific packets are sent to the target host, and the information inside the response packets is analyzed. Passive OS fingerprinting sniffs the network traffic rather than sending packets to the hosts. According to their different ways of working, their advantages are complementary and their information gains are also different. Active OS fingerprinting has higher accuracy and is easier to operate. In contrast, passive OS fingerprinting is more concealed and can reduce the likelihood of being discovered.

3. Different kinds of fingerprinting tools

In this section, popular tools are categorized and introduced based on active fingerprinting, passive fingerprinting and other types.

3.1. Active fingerprinting

We introduce Nmap, RING, Xprobe and its evolution Xprobe2 as examples of active fingerprinting tools.



```
# nmap -O -v scanme.nmap.org

Starting Nmap ( http://nmap.org )
Nmap scan report for scanme.nmap.org (74.207.244.221)
Not shown: 994 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
80/tcp    open     http
646/tcp   filtered ldp
1720/tcp  filtered H.323/Q.931
9929/tcp  open     nping-echo
31337/tcp open     Elite
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.39
OS details: Linux 2.6.39
Uptime guess: 1.674 days (since Fri Sep  9 12:03:04 2011)
Network Distance: 10 hops
TCP Sequence Prediction: Difficulty=205 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/local/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 5.58 seconds
           Raw packets sent: 1063 (47.432KB) | Rcvd: 1031 (41.664KB)
```

Figure 2: The output of OS detection with verbosity [5]

3.1.1. Nmap. Network Mapper (Nmap) is the most popular tool that can be used for active OS fingerprinting. Nmap has two effective OS detection methods. One is OS detection with verbosity, and the other is using a version scan to detect the OS. We can receive the output shown in Fig.2 with the following information [5]:

1. Device type. This field indicates the type of device, e.g. router, firewall.

2. Running. This field shows the OS Family and OS generation.

3. OS CPE. This field shows the Common Platform Enumeration (CPE) representation of the OS or the hardware. CPE is a naming standard for IT products and platforms, which is machine-readable [6].

4. OS details. This field represents the detailed description for each matched fingerprint.

5. Uptime guess. This is a guess because Nmap can't ensure its accuracy.

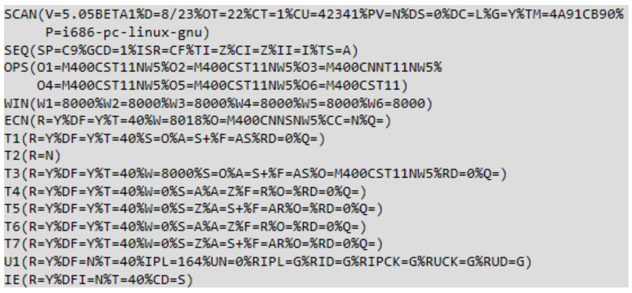
6. Network Distance. Nmap can compute the number of routers between it and a target host.

7. TCP Sequence Prediction. This field shows the Initial Sequence Number (ISN) generation algorithm and the difficulty of how hard it is to attack the host using blind TCP spoofing.

8. IP ID Sequence Generation. This field indicates the algorithm of how the host generates the lowly 16-bit ID field in IP packets.

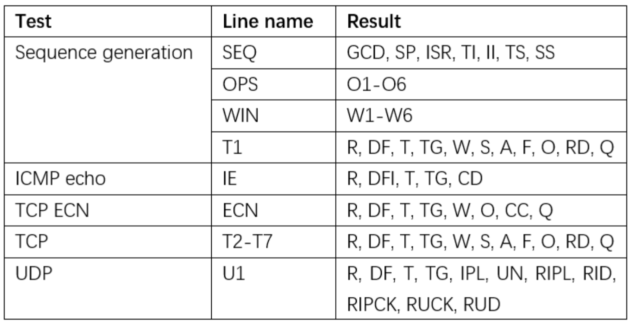
In the case of no fingerprinting matching, the "Service Info" field might reveal the type of OS and Nmap provides the most similar result. Using a fuzzy approach can let Nmap guess more aggressively and offer a list of possible matches with their confidence level in percentage. Nmap also offers detailed information about the host with subject fingerprints, when it cannot identify the host or when we force it to print. An example of a subject fingerprint is shown in Fig.3.

The scan line of the subject fingerprint describes the conditions of the scan, which helps to integrate the fingerprints in the database. Nmap conducts five response tests through sending up to 16 special designed probes (13 TCP, 2 ICMP, 1 UDP). Below the scan line, the response lines are related to the sent probes. The gained information in each line according to the tests is shown in Fig.4.



```
SCAN(V=5.05BETA1%D=8/23%OT=22%CT=1%CU=42341%PV=N%DS=0%DC=L%G=Y%TM=4A91CB90%
     P=i686-pc-linux-gnu)
SEQ(SP=C9%GCD=1%ISR=CF%TI=Z%CI=Z%II=I%TS=A)
OPS(O1=M400CST11NW5%O2=M400CST11NW5%O3=M400CNNT11NW5%
    O4=M400CST11NW5%O5=M400CST11NW5%O6=M400CST11)
WIN(W1=8000%W2=8000%W3=8000%W4=8000%W5=8000%W6=8000)
ECN(R=Y%DF=Y%T=40%W=8018%O=M400CNNSNW5%CC=N%Q=)
T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=Y%DF=Y%T=40%W=8000%S=O%A=S+%F=AS%O=M400CST11NW5%RD=0%Q=)
T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(R=Y%DFI=N%T=40%CD=S)
```

Figure 3: A subject fingerprint [5]



Test	Line name	Result
Sequence generation	SEQ	GCD, SP, ISR, TI, II, TS, SS
	OPS	O1-O6
	WIN	W1-W6
	T1	R, DF, T, TG, W, S, A, F, O, RD, Q
ICMP echo	IE	R, DFI, T, TG, CD
TCP ECN	ECN	R, DF, T, TG, W, O, CC, Q
TCP	T2-T7	R, DF, T, TG, W, S, A, F, O, RD, Q
UDP	U1	R, DF, T, TG, IPL, UN, RIPL, RID, RIPCK, RUCK, RUD

Figure 4: The result obtained by each line classified by the test

In the sequence generation test, six TCP probes are sent. In the response line "SEQ", seven results are shown. Among them, the first three results (TCP ISN greatest common divisor (GCD), TCP ISN counter rate (ISR), TCP ISN sequence predictability index (SP)) are related to the ISN generation algorithm. TI, II and SS (Shared IP ID sequence Boolean) are related to the IP ID sequence. TS is about the TCP timestamp option algorithm. The response line "OPS" contains the TCP options of the six received packets (O1-O6) and the "WIN" line includes the TCP initial window sizes of them (W1-W6). The line "T1" indicates values in the received packets that responses to the first probe. The results are introduced together with the TCP test below.

Next is the ICMP echo test. SS represents the result of testing whether the IP ID sequence is shared between TCP and ICMP. During the test two ICMP echo requests are sent to the host. R represents responsiveness. The value is true when reply is received. DFI indicates the Don't Fragment bit for ICMP. T means initial Time-To-Live, and TG represents its guess if Nmap can't get the value of T. CD represents ICMP response code.

And then is the TCP explicit congestion notification (ECN) test. ECN is a method that intends to reduce network congestion without dropping packets and improve network performance [7]. Nmap sends an SYN packet that sets the ECN CWR (Congestion Window Reduced) and ECE (ECN-Echo) congestion control flags in this test. The value of CC (explicit congestion notification) indicates the setting of CWR and ECE congestion control flags in the response SYN/ACK packet.

Afterwards is the TCP test. Six TCP probe packets with specific TCP options data are sent to the host. Except for R, T, TG, W and O which have already introduced above, DF (Don't Fragment), S (TCP sequence number), A (TCP acknowledgement number), F (TCP flags), RD

(TCP RST data checksum), and Q (TCP miscellaneous quirks) are also recorded as results. For RD, Nmap checks if the host returns data like error message in the reset packets. Q indicates the result for checking two quirks in some implementation.

Finally, in the UDP test, a UDP packet is sent to a closed port. IPL (IP total length) indicates the length of the response packet. UN (Unused port unreachable field nonzero) represents the result of checking whether the last four bytes in the header is zero or not. RIPL (Returned probe IP total length value) indicates the value of the returned IP total length. RID (Returned probe IP ID value) represents if the probe IP ID value has been altered by the host. RIPCK (Integrity of returned probe IP checksum value) shows if the IP checksum still matches the packet, although it has been altered by network hops during transit. RUCK (Integrity of returned probe UDP checksum) shows if the UDP checksum remains the same. RUD indicates the integrity of returned UDP data.

3.1.2. RING.

Nmap has a great performance under the condition of one opened, one closed TCP port and one closed UDP port. RING is a tool as a patch against Nmap that can be used when there's only one opened port. RING exploits the mechanism of packet retransmission in TCP three-way handshake, which has not been considered in other tools. An OS's signature can be established by forcing timeouts and then measuring the delay between the packet retransmission or analyzing the information such as TCP flags, sequence number or acknowledge number [8].

3.1.3. Xprobe.

Xprobe is based on analysis of ICMP instead of TCP. Xprobe has advantages when the differences between the TCP implementation are subtle, for example, some Microsoft based OS. The number of datagrams that Xprobe sends is very small. Thus Xprobe is stealthier. It can send only one datagram and receive the corresponding reply and then recognize up to eight different OSs [9].

Xprobe uses the following methods to fingerprint OS: ICMP error message quoting size, ICMP error message echoing integrity (based on the different implementations of IP total length field, IPID, 3bits flags and offset fields, IP header checksum, UDP header checksum, precedence bits issues with ICMP error messages), DF bit echoing with ICMP error messages, the IP time-to-live field value with ICMP messages, using code field values different from 0 with ICMP echo requests, and TOS echoing [10].

3.1.4. Xprobe2.

The tools like Nmap and Xprobe rely on a static decision tree to perform the results of identification. This approach reduces accuracy because of network topology or the nature of the fingerprinting process itself. For instance, a packet might be affected while in transit, or the user can alter some characteristics of a TCP/IP stack's behaviour [11]. To improve the problem, Xprobe2 utilizes a fuzzy approach with OS fingerprinting. The fuzzy approach is a matrix-based fingerprinting matching approach using the OCR (Optical Character Recognition) technique. The results are shown in a matrix that includes the scores (from 0 (NO) to 3 (YES)) of each test for each OS, and a total score will be calculated for each OS. However, Nmap has already implemented this fuzzy approach [12].

Xprobe2 allows users to modify the modules used for testing. In general, there are seven modules loaded: ICMP echo, time-to-live distance, ICMP echo, ICMP timestamp, ICMP address, ICMP info request and ICMP port unreachable. After the tests, Xprobe2 offers a list of possible OSs with their corresponding probability in percentage sort from the highest to the lowest.

3.2. Passive fingerprinting

The most well-known passive fingerprinting tool p0f and other tools such as Ettercap, Satori and NetworkMiner are introduced in this section.

3.2.1. p0f.

The name of p0f is the acronym for passive OS fingerprinting. Almost all passive fingerprinting tools today reuse p0f for TCP-level checks [13]. Though p0f can never be as accurate as Nmap, because its analysis is based on the packet sent by the host itself, it is still an ideal and precise tool and can be helpful when Nmap is confused.

Except source IP address, port, the TCP flag, OS, network distance, uptime, the output also offers a raw signature. The 67-bit signature consists of information of window size (16 bits), initial time-to-live (8 bits), maximum segment size (16 bits), "Don't fragment" flag (1 bit), window scaling option (8 bits), sackOK option (1 bit), nop option (1 bit), initial packet size (16 bits) [12].

3.2.2. Ettercap.

Ettercap is an open-source tool and is "a multipurpose sniffer/interceptor/logger for switch LANs" [14]. It is popular for detecting man-in-the-middle attacks and enables OS fingerprinting. As a generalist, Ettercap's function of OS fingerprinting is not as precise as p0f. This information can be gained by Ettercap's fingerprinting: IP address, port, network distance, hostname, device type, fingerprint and OS.

3.2.3. Satori.

Satori is a tool based on the analysis of DHCP. The advantage of DHCP is unicast. One of the methods is to use the difference of actual time, seconds elapsed, transaction ID fields of the captured packages to recognize OSs. Another method, also the main method of Satori, is to analyze the parameters of DHCP's option 55, which indicates the parameter request list. Other options can also be used to help to analyze specific OS, for example, option 51 (IP Address Lease Time) and option 57 (Maximum DHCP Segment Size) for Linux [15]. The lease of DHCP can also be exploited to recognize OSs [16]. As the result of OS fingerprinting, Satori offers IP address, fingerprint and a list of possible OSs with their weight up to 12. The higher the weight, the bigger the possibility.

3.2.4. NetworkMiner.

NetworkMiner is an open-source tool for Windows that can be used as a passive network sniffer/packet capturing to detect OSs [17]. NetworkMiner uses the database from p0f, Ettercap, and Satori. It also makes use of the MAC-vendor list from Nmap. The information is shown according to hosts but not to packets or frames on NetworkMiner. IP address, MAC address, vendor, hostname, OS, time-to-live, network distance and open TCP ports of each host can be gained using NetworkMiner. The results of OS fingerprinting are listed according to different databases, containing guesses with their confidence interval in percentage.

3.3. Others

There are other types of tools that also can be used for OS fingerprinting. Some of them work with the help of the functions of the tools that we have introduced in section 3.1 and 3.2. Some combine active and passive fingerprinting. We briefly introduce them in this section.

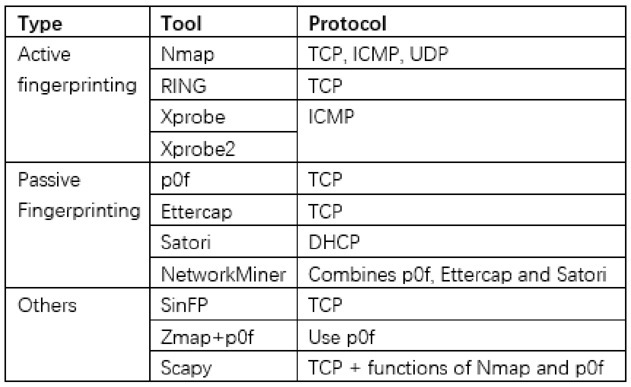
3.3.1. SinFP. SinFP is a hybrid OS fingerprinting tool that is active and passive. It is designed for addressing the increasingly strict limitations and accurately identify OSs in worst network conditions. For active OS fingerprinting, only three standard requests will be sent to the target, and these standard-compliant frames ensure that the responses are replied from the right target but not the devices in-between. After receiving the responses, the active signature with 15 elements (5 elements for each of the three packets) can be established. The 5 elements are respectively: a list of constant values, TCP flags, TCP window size, TCP options and MSS (Maximum Segment Size). For passive OS fingerprinting, SinFP implements the method that modifies a passive signature and then compares it with active signatures. The database of SinFP has strict conditions for each signature. The principle is that the response is not from devices in-between and the target has at least one open TCP port [18].

3.3.2. ZMap+p0f. ZMap is a free and open-source network scanner that has a very fast scanning speed. ZMap can scan the whole public IPv4 address space within 45 minutes with a gigabit Ethernet connection [19]. To achieve OS fingerprinting, one possible way is to run ZMap and to launch p0f in the background at the same time.

3.3.3. Scapy. Scapy is a program that can be used for packet manipulation. It supports active and passive OS fingerprinting by using the functions of Nmap and p0f. On the other hand, it can actively fingerprint Linux kernel 2.4+ through establishing a TCP three-way handshake with the target and then sending a segment with no TCP flags and arbitrary payload. If the response set the flag ACK, then the target is Linux server (2.4+ kernel) because no other well-known current OS accepts this kind of segment and this behaviour is hard to alter for Linux. [20]

4. Machine Learning applied to OS fingerprinting

Machine learning can realize the automatic process of OS fingerprinting. In [21], Aksoy et al. develop a mechanism that classifies OS with high accuracy using machine learning. They collect header information of IP, ICMP, UDP, DNS, HTTP, IGMP, TCP, FTP, SSH and SSL protocol manually to train the classifiers, instead of using



Type	Tool	Protocol
Active fingerprinting	Nmap	TCP, ICMP, UDP
	RING	TCP
	Xprobe	ICMP
	Xprobe2	
Passive Fingerprinting	p0f	TCP
	Ettercap	TCP
	Satori	DHCP
	NetworkMiner	Combines p0f, Ettercap and Satori
Others	SinFP	TCP
	Zmap+p0f	Use p0f
	Scapy	TCP + functions of Nmap and p0f

Figure 5: A brief conclusion of the protocols that are used by each tool

available OS fingerprinting approaches. The classifiers can successfully detect OSs automatically. Since the approach sniffs packets passively, it is seen as passive fingerprinting, but the approach can also apply for active fingerprinting.

In [22] Schwartzenberg automates the process of updating the database and recognizing the new OSs with the help of machine learning. The database of p0f has been updated manually all the time, and this is a complicated work. The accuracy of p0f decreases because the database has not been updated for new OSs. The approach in [22] attempts to address this problem. The automatic process of extracting the characteristics and normalizing the information is proved to be possible. In addition to this, after using the traditional approach to identify OS, the unrecognizable OS can be classified with higher accuracy using the process that implemented machine learning and trained by existing system set.

Anderson and McGrew [2] present a new approach that integrates TCP/IP, HTTP and TLS features to identify OSs. This approach uses a machine-learning classifier and can identify minor versions with an accuracy of 97.5%. Even under the condition of applying obfuscation, the performance of this approach is robust. The accuracy decreases to 94.95% against an obfuscation level of 25%.

5. Conclusion

According to the introduction above, we can find out that Nmap sends more probes and gained more information than others. Therefore, the accuracy of Nmap is higher. Different from Nmap, Xprobe2 is based on ICMP and implements a fuzzy approach. RING is more like a patch for Nmap, it detects OSs in another point of view. But since Nmap is popular, there are already many measures against Nmap. Hence, the tool like SinFP helps to steer by the limitations. In the field of passive OS fingerprinting, p0f is already a very mature tool. Other passive OS fingerprinting tools reuse p0f more or less. Fig.5 briefly concludes the protocols that these tools use for fingerprinting. Using the machine learning technique, the OS fingerprinting process can be automated to increase accuracy and efficiency.

This paper only offers a part of an overview of TCP/IP OS fingerprinting. For instance, Nmap has specific tests for IPv6 fingerprinting. The mechanism and the information gained by it differ. Further work can focus on a wider overview of fingerprinting tools, that use the characteristics of IPv6 to recognize OS.

References

[1] R. Tyagi, T. Paul, B. S. Manoj, and B. Thanudas, "Packet inspection for unauthorized os detection in enterprises," *IEEE Security Privacy*, vol. 13, no. 4, pp. 60–65, 2015.

[2] B. Anderson and D. McGrew, "Os fingerprinting: New techniques and a study of information gain and obfuscation," in *2017 IEEE Conference on Communications and Network Security (CNS)*, 2017, pp. 1–9.

[3] J. Postel, "Internet Protocol," Internet Requests for Comments, RFC Editor, RFC 791, September 1981. [Online]. Available: https://tools.ietf.org/html/rfc791

[4] J. K. Reynolds and J. Postel, "Assigned Numbers," Internet Requests for Comments, RFC Editor, RFC 1700, October 1994. [Online]. Available: https://tools.ietf.org/html/rfc1700

[5] G. F. Lyon, "Chapter 8. remote os detection," https://nmap.org/book/osdetect.html, [Online; accessed 18-June-2008].

[6] "About cpe - archive," https://cpe.mitre.org/about/, Mar 2013, [Online; accessed 18-June-2008].

[7] D. B. K. Ramakrishnan, S. Floyd, "The Addition of Explicit Congestion Notification (ECN) to IP," Internet Requests for Comments, RFC Editor, RFC 3168, September 2001. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3168.txt

[8] F. Veysset, O. Courtay, and O. Heen, "New tool and technique for remote operating system fingerprinting," 2002.

[9] O. Arkin and F. Yarochkin, "X remote icmp based os fingerprinting techniques," August 2001.

[10] O. Arkin, "A remote active os fingerprinting tool using icmp," vol. 27, no. 2, pp. 1–6, April 2002.

[11] O. Arkin and F. Yarochkin, "A 'fuzzy' approach to remote active operating system fingerprinting," August 2002.

[12] C. Peikari and A. Chuvakin, *Security Warrior*. USA: O'Reilly & Associates, Inc., 2004.

[13] M. Zalewski, "p0f v3 (version 3.09b)," https://lcamtuf.coredump.cx/p0f3/, 2012, [Online; accessed 19-June-2008].

[14] A. Ornaghi and M. Valleri, "Ettercap," https://www.ettercap-project.org/, [Online; accessed 19-June-2008].

[15] S. Alexander and R. Droms, "DHCP Options and BOOTP Vendor Extensions," Internet Requests for Comments, RFC Editor, RFC 2132, March 1997. [Online]. Available: https://tools.ietf.org/html/rfc2132

[16] E. Kollmann, "Chatter on the wire: A look at dhcp traffic," September 2007.

[17] N. AB, "Networkminer," https://www.netresec.com/?page=NetworkMiner, [Online; accessed 18-July-2008].

[18] P. Auffret, "Sinfp, unification of active and passive operating system fingerprinting," *Journal in Computer Virology*, vol. 6, pp. 197–205, August 2010.

[19] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX Association, Aug. 2013, pp. 605–620. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric

[20] P. Biondi, "The art of packet crafting with scapy," https://0xbharath.github.io/art-of-packet-crafting-with-scapy/, [Online; accessed 18-July-2008].

[21] A. Aksoy, S. Louis, and M. H. Gunes, "Operating system fingerprinting via automated network traffic analysis," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 2502–2509.

[22] J. Schwartzenberg, "Using machine learning techniques for advanced passive operating system fingerprinting," 2010.

Overview of extra-vehicular communication

Felix Myhsok, Holger Kinkelin*, Filip Rezabek*
**Chair of Network Architectures and Services, Department of Informatics*
Technical University of Munich, Germany
Email: felix.myhsok@tum.de, kinkelin@net.in.tum.de, frezabek@net.in.tum.de

***Abstract*—Extra-vehicular communication is the key element of connected mobility. Therefor the identification of vehicles and the ability to authenticated information from vehicles need to be accomplished while preserving high privacy standards.**

In this paper an architecture for vehicle-to-everything communications is presented at the example of the European ETSI C-ITS standard. A public key infrastructure is thereby the commonly trusted approach to secure communications without compromising the entities privacy.

***Index Terms*—vehicle-to-everything communications, V2X, etsi c-its, public-key-infrastructure, PKI**

1. Introduction

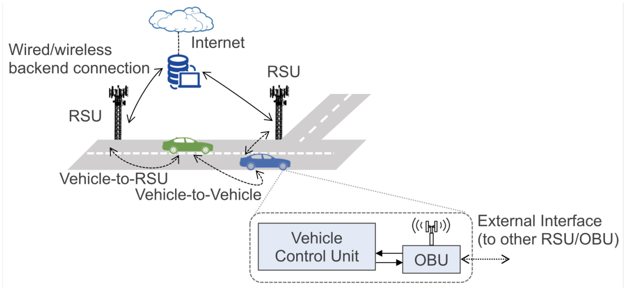
The future of connected mobility and transport is based on extra-vehicular communications. By giving vehicles the ability to communicate and exchange information with their surroundings, improvements at mobility and road traffic can be achieved. Areas of improvement are mainly road-safety, efficiency and environmental pollution [1]. The concept of vehicles communicating with their environment is summarized in the term *vehicle-to-everything (V2X)* (also car-to-everything (C2X)) communication. V2X combines multiple communication applications such as *vehicle-to-vehicle (V2V)*, *vehicle-to-Infrastructure (V2I)*, and *vehicle-to-pedestrians (V2P)*. In each of them, a vehicle communicates with a surrounding entity using a data connection to share and collect information about the environment. This information can then be used to improve the decision-making process of the driver or in an autonomous concept of the vehicle itself. Some use cases as described by [2] are:

- emergency brake lights
- emergency warnings
- collision/intersection warnings
- road work warnings
- lane change assistance
- traffic light optimized speed
- cooperative automated cruise control

1.1. Communication architecture

In this section, we will describe the communication scheme of V2X followed by an overview of the transmission technology.

To equip a vehicle with V2X communication capabilities, it requires an explicit communication interface as part of its hardware. This external interface, called *on-board unit (OBU)*, connects intra-vehicular computing units to the outside world. Communication partners for OBUs are in most scenarios other OBUs (V2V communication) or infrastructure at the roadside, referred to as *roadside units (RSU)*(V2I communication). A graphic representation of the general communication scheme can be found in Figure 1. RSUs, like traffic lights or construction sites, can also serve as a gateway to other communication infrastructure. Thus communication partners are not necessarily located aside of the road, for example like databases or authorities. In this paper, these various communication partners like OBUs and RSUs, are referred to as other entities. [3]


Internet
Wired/wireless backend connection
RSU
RSU
Vehicle-to-RSU
Vehicle-to-Vehicle
Vehicle Control Unit
OBU
External Interface (to other RSU/OBU)
Figure 1: V2X Communication Scheme [3]

In the currently existing concepts, there are two different approaches concerning the underlying technology for wireless transmissions: WiFi-based or cellular-based networks. Both approaches are briefly outlined in the following Sections 1.1.1 and 1.1.2. In Chapter 2 and 4, we focus on the WiFi-based approach of the European ETSI C-ITS standard.

1.1.1. WiFi-based communication. In this approach, the technology used to establish wireless communication is based on WiFi - more specific, on the IEEE 802.11p standard [4]. Communications via WiFi are often summarized under the term *Dedicated Short Range Communication (DSRC)*. The entities in the network communicate over a wireless adhoc network using the 5.9 GHz frequency. Therefore, every entity has an antenna to send, receive, or forward messages. Since the signal range of WiFi is usually limited to a few hundred meters, connections between two entities are mostly of shorter duration and the vehicle is not always connected to the network. Since the WiFi-technology is widely known and used, there are already different standards for V2X communications in production using this technology. The most most popular are:

- ETSI Cooperative Intelligent Transport Systems (C-ITS) standard in Europe [5]
- Wireless Access for Vehicular Environment (WAVE) standard in the United States [6]
- ITS Connect standard in Japan (operates on the 700MHz band) [7]

1.1.2. Cellular-based communication. In cellular vehicle to everything (C-V2X) communications the network connection is established using Long Term Evolution (LTE) (3GPP Release 12) or 5G (Release 16) cellular networks. In this architecture, the vehicle communicates in most cases with base stations which provide a high ground coverage. Additionally, through the LTE-PC5 interface (also LTE-Sidelink) entities have the ability to communicate directly without using a base station [6].

This paper is structured as follows: Chapter 2 explains security challenges in V2X communications. In Chapter 3 the general concept of a public key infrastructure is summarized briefly before in Chapter 4 the concrete public key infrastructure used in the ETSI C-ITS standard explained and compared to the US WAVE standard.

2. Security challenges

In this chapter we outline the most important security challenges and threats for V2X communications. However the focus of this paper is on not authenticated messages and tracking of vehicles.

By establishing V2X communications, a vehicle relies not only on information obtained by itself (e.g. through sensors at the vehicle) but also on information generated by others. If this received information does not correspond to the reality it can cause severe damage to the driver, the vehicle or others. For example if the vehicle receives false information and as a result initiates an emergency breaking, this could lead to collisions and traffic jams. The source of this false information can either be from a malfunctioning entity in the network or a malicious entity. We consider an entity, which tries manipulate other entities by sending false information, as an attacker. In this paper we focus on malicious attacks.

Attack scenarios, like the ones described in [3], can be clustered in three categories by analyzing the underling attack strategy.

2.1. Denial of Service Attacks on the communication channel

The basic principle of *denial-of-service (DoS)* attacks, is to overload the receiving entity with more messages than it can process. Due to the lack of resources, important data can then be lost or not processed in time. These attacks can happen on different layers like simple physical frequency jamming or by acting as a router in an ad hoc network and dropping packets, as in a JellyFish Attack. They can also be spread over multiple nodes (distributed DoS [DDoS]) to increase the number of messages send and to bypass security measures. [8] DoS attacks are mostly geographically limited. For this reason, they only affect a limited amount of entities. There are existing techniques and concepts how to reduce impact of DoS attacks, which are explained in the further reading [9].

2.2. Insertion of not Authenticated Packets

If entities are not identifiable, attackers can take on any appearance they want. Thus they can send faked messages to manipulate other entities without being detected. This makes proper responses and prosecution of attackers more difficult. It also leads to multiple attack scenarios, which are explained in the following.

In **sybil attacks**, one attacker has more than one identity. Thus, he is able to send bogus information, e.g. about the traffic situation, to other entities. It can also be used to boost the trustworthiness of malicious entities or lower it for legitimate entities to increase the impact of false information. [3] [8]

Message replay attacks are used to reveal conditions or services at the receiving end. In general, an attacker records a valid message but resends it to a different time or location. For example, the attacker records the message send by a vehicle when it accesses a restricted area, for instance, a parking deck. Without security measures, an attacker could then just transmit this message again to gain access to this parking deck. [3] [8]

By using **false data injection attacks**, attackers can send bogus data to other entities to influence their behaviour. Thereby, the attacker simply alters the real-world situation. E.g. he transmits that he is 20 meters away when he is actually 200 meters away. Using this method, an attacker could affect e.g. the road traffic or trigger emergency brakes. [3] [8]

It is possible to counter these attacks with adding a unique identity to every entity in the network to sanction them for false behaviour. Furthermore it needs to be possible to authenticate if an entity belongs to the identity it is using. This can be achieved through cryptographic scheme with digital signatures as shown in Chapter 4. This, however, comes at the cost of privacy since tracking is possible. Nevertheless, authenticated messages do not provide complete security. Despite authentication, valid messages can be replayed by attackers or false information can be transmitted in an authenticated message if the entity is compromised. [3] [8]

2.3. Tracking of Vehicles

In a V2X communications architecture, privacy protection must be a vital part. In this paper we focus on privacy issues caused by identification of entities based on sent messages. Attackers can track the digital signatures broadcasted in messages. Other tracking methods like radio fingerprinting or mobile phone tracking are out of scope. [8]

Due to missing privacy protection, **identity revealing attacks** can be facilitated, where attackers are able to identify the vehicle driver. It allows the attackers to gather personal information about the driver (e.g. personal activities) which can lead to personal profiling. [8]

Another attack which can be prevented through privacy measures is the **location tracking** of a vehicle. This attack tracks movements and current position of the vehicle and can, for example, be used in combination with identity revealing to track a person's movements. [8]

3. Background: General architecture of a PKI

This chapter provides background information for a better understanding of a public key infrastructure and asymmetric encryption schemes.

A *public key infrastructures (PKI)* main purpose is to give every participant a digital identity and ensure the authenticity of it.

Therefore, a PKI delivers certificates for every entity based on a signing process with asymmetric keys and digital signatures. Every entity owns a unique pair of keys, which consist of a public key and a private key. The public key is accessible for everyone while the private key is kept secret by the owning entity. A *certification authority (CA)* serves as a trusted third party and issues certificates to all participants in the network. Certificates Certificate bind the identity of a participant to the key that belongs to the participant. To prove authenticity and validity of the certificate, the issuing CA signs the certificate using its own private key. [10]

To ensure that the CA is trustworthy, the CA also owns a certificate. This certificate is created and signed by another CA. Through this process certificate chains are build up. Every chain has its root in one common *Root Certification Authority (RCA)*. A RCA is an anchor that needs to be trusted by everyone in the certificate chain. [10]

When two entities are now communicating, the sender signs (encrypt) the data with his private key. The receiving entity can then use the public key from the sender's certificate to decrypt the message. This allows the recipient to verify the identity of the sender. Since the public and private key is a unique combination, only the certificate corresponding participant can successfully sign the data with its private key. Furthermore, the recipient can verify whether the certificate of the sender is valid by analyzing the certificate of the issuing authority. [10]

If a participant e.g. behaves incorrectly or the private key of the participant is compromised, the participant needs to be excluded from the PKI. Therefore his certificate gets revoked. The mostly used approach therefore is a *Certificate Revocation List (CRL)*. This CRL contains all revoked certificates and allows to check if a specific certificate is revoked. Revocation of certificates is also a task of a CA. [10]

4. The PKI in the C-ITS standard

A PKI can solve attack scenarios caused by unauthorized entities by giving every entity a unique identification. Furthermore, the PKI which is used for V2X, was also specifically designed to protect privacy to encounter tracking of vehicles. [11] In this section, we briefly explain the functionality of the PKI proposed in the C-ITS standard and the used certificates. Some aspects are fairly similar to the approaches proposed in the WAVE standard and some differences are outlined in Section 4.3.

In the V2X context the PKI main goals are to issue valid certificates to every entity, to minimize the abuse of issued certificates and to exclude malicious entities of the network. Therefor the private key needs to be securely stored inside the vehicle and the certificate is appended to every outgoing message. [11]

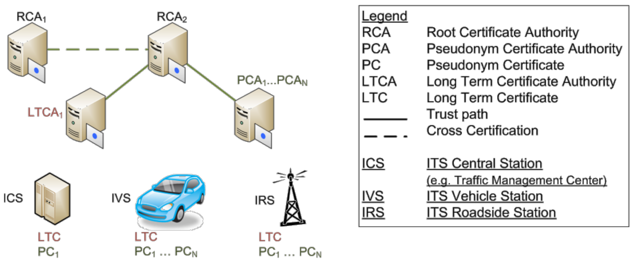


Figure 2: General PKI Structure [11]

4.1. The PKI of the C-ITS

This section explains the PKI which was created by the *Car 2 Car Communication Consortium (C2C-CC)* [11] and got adopted by the European ETSI C-ITS standard for V2X communications.

4.1.1. Structure. We describe the structure of the PKI from top down, explaining functionality and components layer by layer. The structure described in the following is corresponding to the one given in Figure 2.

At the highest level of the PKI proposed by the C2C-CC is the RCA. Its main task is to control and manage the CAs on the layer below. Therefore the RCA issues certificates for the underlying CAs with a long validity. If there are multiple RCAs it is possible that they cross-sign their certificates to increase their trust level. Cross certification is only possible between RCAs and not between other CAs on the lower layer. Below the RCA, there are two kinds of Sub-CAs, *Long-Term Certification Authorities (LTCA)* (also Enrolment Authorities) and *Pseudonym Certification Authorities (PCA)*(also Authorization Authority). [11]

Entities need to have a long term identity to identify and authenticate them inside the PKI. Therefor every entity owns unique a *Long-Term Certificate (LTC)* (also Enrolment Credentials) which gets issued to the entity by a LTCA. To prevent tracking and traceability of entities, LTCs are not used for communication between two entities. Instead entities use pseudonyms identities which can not be mapped to the LTC. These pseudonymous identities are realized through *pseudonymous certificates (PC)*(also Authorization Tickets) which disguise the individual identifiers of the entity, such as MAC-Address and the network layer identifier. These PCs (usually distributed in a set) get issued to the entity by PCAs. In contrast to LTCs, PCs are short-lived, which means their validity is limited to a couple of minutes to a few hours. If a single PC is used to often it enables tracking again, since the identifiers of the PC can be tracked. Therefor an entity stores a large amount of valid PCs inside the vehicle. As a result PCs need to be exchanged and renewed often. [11]

The last layer of the PKI hirarchy are the actual entities. Each of them owns as described one LTC and multiple PCs. [11]

4.1.2. Issuing & renewing of Certificates. This section briefly explains how the issuing of certificates is handled in the PKI and how they are renewed.

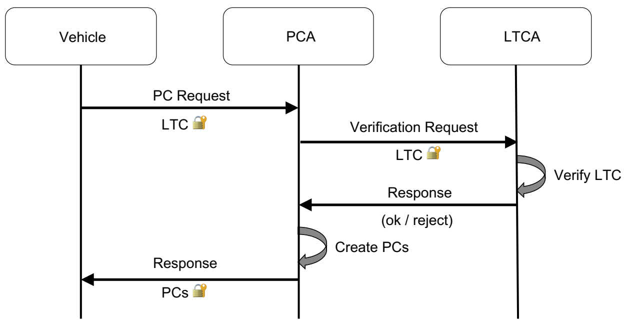


Figure 3: Issuing PCs Flowchart

The first LTC of an entity could be installed by the manufacturer. Since the LTC is valid for longer periods of time, it is not renewed very often. When a LTC needs to be renewed, the entity sends its LTC (encryped with the public key of the LTCA) to the LTCA and receives in a response the new LTC. [12]

The lifetime of PCs and thereby the number of renewing operations is based on three major factors: the number of PCs an entity uses silmultaneously, the lifetime of one PC (e.g. 10min or 1h) combined with the decision if they can be reused and how many usable PCs an entity has to store. In the C-ITS standard these parameters are not further specified yet and left up to the manufacturer. [12]

Since PCs need to be renewed often than LTCs, this process must be more flexible. The process of issuing PCs to an entity, which is described in the following, is also shown in Figure 3.

To get a new set of PCs, an entity sends a request to a PCA. This request includes the LTC (encrypted with the public key of the LTCA), the ID of the corresponding LTCA, public keys and the current position. The for this region responsible PCA, sends a request to the LTCA stated in the entity's request with the received LTC. By analyzing the LTC, the LTCA then permits the PCA to issue new PCs if the entity is a valid part of the network. The PCA generates a set of PCs which are encrypted, using the received public keys, and sends these new PCs back to the original entity. [11] [12]

4.1.3. Revocation.

If an entity gets identified as malicious or defect, it is the responsibility of the PKI to ensure the reported entity gets excluded from the network. The detection of the malicious entities is not part of this paper.

To exclude entities, the PKI implements a Certification Revocation List. The CRL contains the LTCs of the reported entities and is collectively managed by the LTCAs. When an entity is reported, the LTCA, in collaboration with the PCA, identifies the LTC of this entity and adds it to the CRL. When the malicious entity requests new PCs, the LTCA compares the given LTC with the CRL and is able to reject the request if the entity is reported. If an entity does not have valid PCs, it is not trusted in the network communication and therefore excluded. [11] [12]

By using this approach, an entity can send valid and authenticated messages until it runs out of valid cached PCs. Without valid PCs the messages send by the entity are not authenticated and discarded at the receiving end. This can cause problems, depending on the specific design of the PCs and the renewing process, because there could be a significant amount of time between detecting a compromised entity (adding the LTC to the CRL) and excluding this entity from the network (entity runs out of PCs). In this period of time the entity could cause severe damage in the network. [11] [12]

In a situation where a CA is compromised, for example, if the private key got exposed, all the certificates issued by this CA needs to be revoked. Otherwise an attacker could produce valid certificates for malicious entities, which can not be detected by the PKI. [13]

To revoke all certificates issued by a CA, CRLs are also put to use. The PKI administrator appends the certificate of the compromised CA to a CRL. This CRL only contains revoked CAs and gets actively distributed to all PKI participants. When a certificate, issued by the compromised CA, gets checked by an entity through the certificate chain, the entity compares the certificate of the CA with the distributed CRL and is able to verify if it is valid. This expensive process only needs to be executed rarely due to the fact that compromised CAs occurs seldom. [13]

TABLE 1: Certificate types in the PKI [14]

Name	Quantity	Size	Lifetime
RCA Certificates	20	126 Byte	up to 15 years
LTCA Certificates	up to 1000	126 Byte	up to 15 years
PCA Certificates	up to 2000	126 Byte	up to 5 years
LTC	1	125 Byte	up to 10 years
PC	1500/Year	124 Byte	up to 1 years

4.2. Certificates

In this section, we explain general design ideas and possible options for certificates. The exact definition of the certificates is not part of this paper.

4.2.1. Format & Types.

For the PKI in the V2X network, multiple types of certificates are needed. According to the specifications made by the ETSI C-ITS standard [14], there are five different types of certificates which are listed in the Table 1 along with some estimations of the amount stored in an entity, the size of one certificate and the lifetime, defined by the C2C-CC in [11]. All certificates follow the same structure based on the ExplicitCertificate defined in the IEEE standard 1609.2 clause 6.4.6 [15]. The technical differences of each certificate are defined in [16].

4.2.2. Cryptographic algorithms.

The cryptographic algorithms used in the certificates and the V2X communications are also vitally important to ensure secure communications.

In the C-ITS standard Elliptic Curve Digital Signature Algorithms (ECDSA) are intended for signing data [16].

For sending encrypted data, the Elliptic Curve Integrated Encryption Scheme (ECIES) is specified by the the IEEE standard 1609.2 [15].

Both use one of the elliptic curves NIST P-256 (specified in FIPS 186-4) or brainpoolP256r1 (specified in RFC 5639) [15].

4.3. Comparision with the WAVE standard

In the following, we point out major differences of the European C-ITS and the US WAVE standard. We especially focus on the key-generation and the revocation process. A more detailed comparison between the different standards can be found in [6].

In comparison to the European standard, The WAVE standard distributes tasks among several smaller independent authorities as in contrast to the three authorities (RCA, LTCA, PCA) in the European standard. As a result, power is widely distributed in the system and abuse is more difficult. This ensures no one has enough information to do harm or breach the privacy of entities. [17]

Furthermore, the WAVE standard uses a new cryptographic construct for the key generation, called *butterfly key expansion*. The basic principle behind it, is that the entity generates one key pair and sends the public key (also public seed) along with one expansion function to the certificate issuing authority. The authority can now apply the received function to the public key to generate multiple public keys which are used to create certificates. The entity also uses an expansion function on the private key to generate multiple private keys. These separately generated public and private keys fit together to an asymmetric key pair and can be used for signing and encryption. [18] Using this approach, the number of messages send between the entity and the issuing authority is drastically reduced as shown in [19].

As a result, a different revocation process has been established. This is explained in detail in [18]. In summary, multiple authorities cooperate to identify the LTC. Furthermore, it is possible to revoke the PCs of the revoked entity, based on the seed value used for the certificate generation. This poses a major difference to the C-ITS approach.

5. Conclusion and further work

V2X communications is a promising concept to improve our daily mobility on many levels and we are getting closer to a connected mobility every day. The European WLAN based ETSI C-ITS standard gives manufacturers and developers a fundamental structure of how communications should be established but leaves some aspects up to the manufacturer. We outlined that the concept of a PKI with LTCs and PCs seems suitable for establishing trust and validate messages in a V2X network. It should be emphasized that privacy has been an important design goal from the beginning. Concerning the revocation process, new concepts as in the WAVE standard show efficient techniques, which could be part of a future adaption. Since mobility is not limited to specific regions in the long run, the different standards should be compatible on a basic level without major adjustments at the entity.

References

[1] U. D. of Transportation, "How connected vehicles work," https://www.its.dot.gov/factsheets/pdf/connected_vehicles_work.pdf, [Online; accessed 11-June-2020].

[2] G. Pocovi, M. Lauridsen, B. Soret, K. I. Pedersen, and P. Mogensen, "Automation for on-road vehicles: Use cases and requirements for radio design," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, 2015, pp. 1–5.

[3] M. Hasan, S. Mohan, T. Shimizu, and H. Lu, "Securing vehicle-to-everything (v2x) communication platforms," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020.

[4] I. of Electrical and E. Engineers, "Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, 2016.

[5] A. Festag, "Cooperative intelligent transport systems standards in europe," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 166–172, 2014.

[6] T. Yoshizawa and B. Preneel, "Survey of security aspect of v2x standards and related issues," in *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2019, pp. 1–5.

[7] ARIB, "700 MHz Band Intelligent Transport Systems," Association of Radio Industries and Businesses, Standard ARIB STD-T109, 2013.

[8] A. Ghosal and M. Conti, "Security issues and challenges in v2x: A survey," *Computer Networks*, vol. 169, p. 107093, 2020.

[9] I. Aad, J. Hubaux, and E. W. Knightly, "Impact of denial of service attacks on ad hoc networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 791–802, 2008.

[10] J. Weise, "Public key infrastructure overview," *Sun BluePrints OnLine, August*, 2001.

[11] N. Bißmeyer, H. Stbing, E. Schoch, S. Gtz, and B. Lonc, "A generic public key infrastructure for securing car-to-x communication," in *18th World Congress on Intelligent Transport Systems featuring ITS America's annual meeting and exposition 2011*, 10 2011.

[12] J. P. Monteuuis, B. Hammi, E. Salles, H. Labiod, R. Blancher, E. Abalea, and B. Lonc, "Securing pki requests for c-its systems," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–8.

[13] Working Group 2, Task Force 1, "Draft report on european security mechanism," C-Roads, Tech. Rep., 2019.

[14] ETSI, "Intelligent transport systems (its); security; its communications security architecture and security management," European Telecommunications Standards Institute, Standard TS 102 940, 2018.

[15] IEEE, "Ieee standard for wireless access in vehicular environments–security services for applications and management messages," *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)*, pp. 1–240, 2016.

[16] ETSI, "Intelligent transport systems (its); security; security header and certificate formats," European Telecommunications Standards Institute, Standard TS 103 097, 2017.

[17] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn, "A security credential management system for v2v communications," in *2013 IEEE Vehicular Networking Conference*, 2013, pp. 1–8.

[18] B. Brecht, D. Therriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, and R. Goudy, "A security credential management system for v2x communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3850–3871, 2018.

[19] B. Hammi, J. P. Monteuuis, H. Labiod, R. Khatoun, and A. Serhrouchni, "Using butterfly keys: A performance study of pseudonym certificates requests in c-its," in *2017 1st Cyber Security in Networking Conference (CSNet)*, 2017, pp. 1–6.

Stochastic Network Calculus Latency Bounds

Curt Polack, Max Helm*, Dominik Scholz*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: curt.polack@tum.de, helm@net.in.tum.de, scholz@net.in.tum.de

Abstract—

The question of how to calculate latency and buffer bounds in complex networks is becoming ever more relevant in today's interconnected world. A number of approaches to this problem have been developed; this paper provides an introduction to the concepts as well as the application of stochastic network calculus and shortly introduces two alternative methods for calculating relevant bounds in networks: deterministic network calculus and classic queuing theory. This paper also provides an overview of various open-source tools that use these different approaches and, using two different reference topologies, compares them regarding important factors such as the tightness of bounds. While the tested tools perform similarly in uncongested networks, specific tools, such as DISCO SNC, can provide tighter bounds and greater functionality. The results of one tool, DISCO DNCv2, could not be reliably compared with those of the others.

Index Terms—**stochastic network calculus, queuing theory, network performance evaluation, scheduling**

1. Introduction

The increasing dependence on fast, reliable networks in academia and elsewhere necessitated the formalization of those networks. Quality of Service (QoS) guarantees regarding delay, throughput, and packet loss are especially important when measuring the performance of networks and ensuring their QoS [1] . The required guarantees vary depending on the needs of end-users and the contracts they have with their respective network provider. A user who, for example, is interested in real-time communication with other users would be less interested in the lower bounds on throughput and more interested in the upper bounds on delay; on the other hand, a user who must send large files would be more concerned with the lower bounds on throughput.

Network calculus was first introduced in 1991 by Cruz in two related papers, [2] and [3], as a new way to determine the relevant bounds inside a communication network. Since this initial publication, network calculus has developed in two branches: stochastic and deterministic network calculus. In short, deterministic network calculus provides the worst-case bounds for a given network while stochastic network calculus provides bounds based on statistical distributions and a certain level of acceptable exceedance of required bounds [4]. Deterministic network calculus is often used to model networks in which there is no tolerance for packet loss or long delays (e.g. critical infrastructure). Stochastic network calculus can be employed to model networks which can tolerate a certain amount of loss or delay in order to more closely mirror the real-life requirements of many networks and users. It can, therefore, be used to more accurately and tightly find bounds / guarantees in networks that are inherently statistical in nature [4].

2. Background

This section provides an overview of the fundamentals of (stochastic) network calculus and its similarities and differences with other theories.

2.1. Network Calculus

In order to be effectively used for network analysis (i.e. deriving relevant bounds), a theory must be characterized by the following five properties [4]:

1) Service Guarantees - Stochastic service guarantees (e.g. backlog, delay) can be derived for single nodes using a specific traffic model and a specific server model.
2) Output Characterization - The output of a server can be modeled using the same traffic model as the input.
3) Concatenation - The concatenation (i.e. convolution) of multiple servers can be modeled using the same server model.
4) Leftover Service - The service available to a traffic flow can be modeled using the same server model if multiple flows are simultaneously using the service.
5) Superposition - The superposition of multiple traffic flows can be modeled using the same traffic model.

These properties, especially the third and fourth properties, allow for the concatenation of multiple service and arrival flows; this, in turn, reduces the necessary calculations and can significantly improve the results when compared to node-to-node analysis [4].

Network calculus characterizes networks using two curves: service (server) and arrival (traffic). The arrival curve describes the traffic sent using its upper bound while the service curve defines a lower bound that a server provides [4]. Often an "envelope process" is used to describe these curves; the envelope process simply refers to a function which deterministically bounds the process but is not necessarily tight. One of the main advantages of network calculus is that service curves can be concatenated (i.e. convoluted) using min-plus algebra and thus more effectively analyzed.

2.2. Mathematical Basics & Notation

This section provides an overview of the notation used and, partially, the fundamental mathematical concepts as described and used in [4] and [5].

$\mathbb{F}$ is used to denote the set of non-negative wide-sense increasing functions in $\{a() : \forall 0 \leq x \leq y, 0 \leq a(x) \leq a(y)\}$ and for which it holds $\forall x < 0 : a(x) = 0$. This set of functions is used to characterize arrival and service curves as a function of time t.

Min-plus algebra is used to perform operations on flows. An important algebraic structure when using min-plus algebra is

$$(\mathbb{F} \cup \{+\infty\}, +, \wedge)$$

which is a commutative diode with the zero element $+\infty$ and the identity element 0 for all $x \geq 0$ and $+\infty$ otherwise [6]. The min-plus convolution and deconvolution of two functions, a and b, are respectively defined as follows:

$$(a \otimes b)(x) = \inf_{0 \leq x \leq y} [a(x+y) - b(y)]$$

$$(a \oslash b)(x) = \sup_{y \geq 0} [a(x+y) - b(y)]$$

$A(t)$ refers to the (cumulative) arrival process and $A*(t)$ to the (cumulative) departing traffic process of a (lossless) server. The backlog $B(t)$ is then defined as $A(t) - A*(t)$. A_h^i and $A*_h^i$ refer respectively to the arrival and departure models of flow i in network element h.

2.3. Traffic Models

The traffic model originally introduced in [2] is the (σ, ρ) traffic characterization and is deterministic. σ refers to the burstiness and ρ to the rate of the traffic flow. A popular implementation of this traffic flow is referred to as a token bucket; a traffic flow / arrival curve A is bounded by the (σ, ρ) model if the following condition holds for all $0 \leq s \leq t$ [4]:

$$A(s,t) \leq \rho \cdot (t - s) + \sigma$$

In stochastic network calculus, this model is expanded and depicted using the $(\sigma(\theta), \rho(\theta))$ traffic characterization [4]. Using the moment generating function (MGF) of the arrival curve, a bound can be derived for this traffic characterization [4]; an arrival curve A is bounded by the $(\sigma(\theta), \rho(\theta))$ model for some θ if the following condition holds for all $0 \leq s \leq t$:

$$\frac{1}{\theta} log E[e^{\theta A(s,s+t)}] \leq \rho(\theta) \cdot (t) + \sigma(\theta)$$

Stochastic network calculus can then be used to find and optimize θ by defining, for example, the maximum allowed backlog and the probability with which this bound must be respected [5]. There are a number of further variations of the stochastic arrival curve introduced in [4] including the traffic-amount-centric (t.a.c.) arrival curve, the virtual-backlog-centric (v.b.c.) arrival curve, and the maximum-backlog-centric (m.b.c.) arrival curve.

2.4. Similarities and Differences with Queuing Theory

Queuing theory was first introduced in 1909 by A.K. Erlang and is a branch of mathematics that focuses on the modelling of the act of waiting in line [7]; queuing theory was thus not developed with modern communication / packet networks in mind. One of the most common proposals to model arrival and service curves is the M/M/1 model using a Poisson distribution. Although many networks can be modeled using queuing theory, multiple of the properties, especially 3 and 4, described in Section 2.1 cannot, in general, be concluded for queuing theory [4]. In addition, queuing theory focuses on the calculation of the average case and not the worst case, as in network calculus [7].

3. Application and Comparison

This section provides an overview of a diverse set of tools as well as their defining characteristics and defines two reference topologies. The aforementioned tools are then used to evaluate both reference topologies; the results of all evaluations are subsequently analyzed and compared with one another.

3.1. Tools

The tools which are introduced in this section are all open-source and were developed primarily for use in research. Table 1 provides an overview of the supported network topologies (Tandem, Tree, and Feed-forward) as well as important network calculus operations (Convolution and Deconvolution).

TABLE 1: Supported Topologies and Operations

	Tandem	**Tree**	**Feed-forward**	$\otimes$	$\oslash$
DISCO SNC	✓	✓	✓	✓	✗
DISCO DNCv2	✓	✓	✓	✓	✓
SNC MGF Toolbox	✓	✓	✗	✓	✓
OMNeT++	✓	✓	✗	✗	✗

3.1.1. The DISCO Stochastic Network Calculator. The DISCO Stochastic Network Calculator (DISCO SNC) is an open-source tool developed by researchers at the Distributed Computer Systems (DISCO) chair of the University of Kaiserslautern [8]. It is a modular program developed primarily in Java for the application of stochastic network calculus and also provides a graphical user interface (GUI) in order to "make the SNC accessible even for SNC-inexperienced users" [8]. It supports a number of traffic characterizations including exponential, exponentially bounded burstiness, and token bucket.

3.1.2. Stochastic Network Calculus Moment Generating Function Toolbox. The Stochastic Network Calculus Moment Generating Function Toolbox (SNC MGF Toolbox) is an open-source tool developed by Paul Nikolaus, a researcher at the DISCO chair of the University of Kaiserslautern [9]. It is a library developed in Python for the application of stochastic network calculus and does

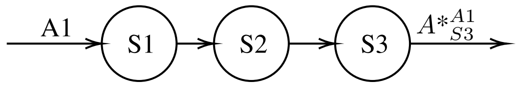


Figure 1: Topology 1 - Tandem

not provide a GUI. It supports a number of traffic characterizations, including Markov modulated on-off traffic, exponentially bounded burstiness, and token bucket.

3.1.3. The DISCO Deterministic Network Calculator v2. The DISCO Deterministic Network Calculator v2 (Disco DNCv2) is the second-generation open-source tool developed by researchers at the DISCO chair of the University of Kaiserslautern for analyzing networks using deterministic network calculus [10].

3.1.4. Objective Modular Network Testbed in C++. Objective Modular Network Testbed in C++ (OMNeT++) is a discrete event simulator developed primarily in C++ and used for, among other things, modeling communication networks [11]. OMNeT++ was first introduced in 1997 and has since been expanded by multiple libraries, including one for the simulation of queuing networks, and a comprehensive GUI.

3.2. Reference Topologies

In order to compare the tightness of the bounds calculated by the aforementioned tools, this section defines two relatively simple network topologies which can be modeled in every tool. Figure 1 models a tandem (chain) network with three service curves; the arrival curve enters the first node and is processed by all three service curves. Figure 2 models a (fat) tree network with two arrival curves entering the two lowest nodes respectively and both departing curves being sent to the root node / service curve. S1, S2, and S3 provide a constant service rate of 5, 4.9, and 4.5 respectively in both topologies. The bounds on the departure curve(s), $A*^{A1}_{S3}$ for the tandem topology as well as $A*^{A1}_{S1}$ and $A*^{A2}_{S1}$ for the tree topology, are of interest for the analysis. The arrival curve(s) A1 and A2 are characterized by an exponential distribution with $exp(\lambda)$; multiple values for λ are tested. For both stochastic network calculus tools, the tolerance for the delay as well as the backlog bound was set to .05. The graphed results for the tree topology are cumulative results of both $A*^{A1}_{S1}$ and $A*^{A2}_{S1}$.

The arrival curve(s), which can be represented by the exponential distribution with the parameter λ, are bounded by the $\sigma(\theta), \rho(\theta))$ traffic characterization for all $\theta < \lambda$ with $\sigma(\theta) = 0$ and $\rho(\theta) = \frac{1}{\theta}(\frac{\lambda}{\lambda - \theta})$ [12]. This can be used to provide a deterministic bound on the stochastic distributions for DISCO DNCv2.

3.3. Configuration of Tools

This section documents the methods used to configure each tool and derive relevant bounds using the tandem topology and backlog bound as an example.

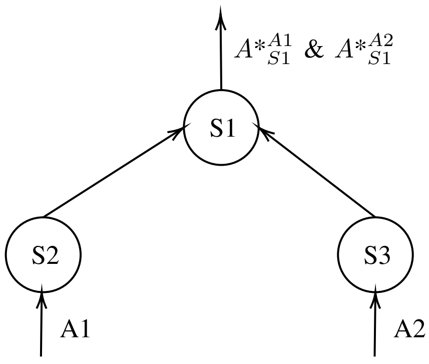


Figure 2: Topology 2 - Fat Tree

3.3.1. DISCO SNC. The following configuration can be loaded directly into the DISCO SNC GUI for $\lambda = .3$.

```
# Configuration of Network
# Interface configuration. Unit: Mbps
I v1, FIFO, CR, 5
I v2, FIFO, CR, 4.9
I v3, FIFO, CR, 4.5

EOI
# Traffic configuration. Unit Mbps or Mb
# One flow with the route v1->v2->v3 with priorities and characterization
F F1, 3, v1:1, v2:1, v3:1, EXPONENTIAL, .3
EOF
```

3.3.2. SNC MGF Toolbox. This Python file for $\lambda = .3$ can be run in the root folder of the project once all dependencies have been added.

```
if __name__ == '__main__':
    print("Tandem␣Performance␣Bounds:\n")

    DELAY_PROB_BOUND = PerformParameter(perform_metric=PerformEnum.DELAY_PROB,
    value=.05)

    Server1 = ConstantRateServer(rate=5.0)
    Server2 = ConstantRateServer(rate=4.9)
    Server3 = ConstantRateServer(rate=4.5)

    ConvolvedServer = Convolve(Convolve(Server1, Server2), Server3)

    TandemTopology = SingleServerMitPerform(arr_list=[DM1(lamb=.3)],
                                            server=ConvolvedServer,
                                            perform_param=DELAY_PROB_BOUND)

    #Grid search for param between 0.1 and 5.0 with granularity 0.1
    print(Optimize(TandemTopology, number_param=1,
                   print_x=True).grid_search(bound_list=[(0.1, 5.0)],
                   delta=0.1))
```

It is important to note that SNC MGF Toolbox does not support FIFO multiplexing / scheduling and instead always uses arbitrary scheduling. Arbitrary scheduling provides a worst-case bound on any scheduler, including FIFO.

3.3.3. DISCO DNCv2. This Java file for can be run for $\lambda = .3$ and $\theta = .1$ in the root folder of the project once all dependencies have been added.

```
public void run() throws Exception {
    ServiceCurve service_curve_1 = Curve.getFactory()
    .createRateLatency(5.0, 0);
    ServiceCurve service_curve_2 = Curve.getFactory()
    .createRateLatency(4.9, 0);
    ServiceCurve service_curve_3 = Curve.getFactory()
    .createRateLatency(4.5, 0);

    ServerGraph sg = new ServerGraph();

    Server s0 = sg.addServer(service_curve_1, Multiplexing.FIFO);
    Server s1 = sg.addServer(service_curve_2, Multiplexing.FIFO);
    Server s2 = sg.addServer(service_curve_3, Multiplexing.FIFO);

    sg.addTurn(s0, s1);
    sg.addTurn(s1, s2);

    ArrivalCurve arrival_curve = Curve.getFactory().createTokenBucket(15, 0);
    sg.addFlow("f0", arrival_curve, s0, s2);
    CompFFApresets compffa_analyses = new CompFFApresets( sg );
    TotalFlowAnalysis tfa = compffa_analyses.tf_analysis;

    tfa.performAnalysis(sg.getFlow(0));
    System.out.println("delay␣bound:␣" + tfa.getDelayBound());
    System.out.println("backlog␣bound:␣" + tfa.getBacklogBound());
}
```

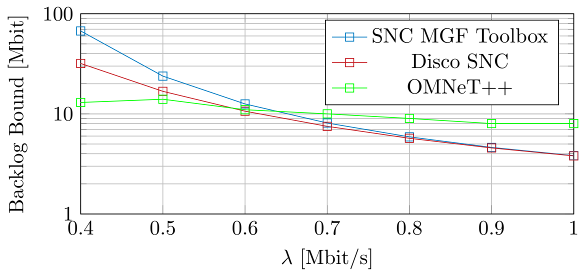


Figure 3: Backlog Bound - Tandem Topology

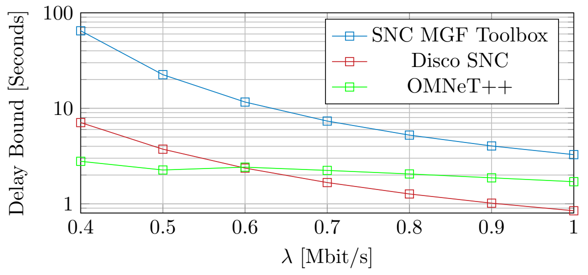


Figure 4: Delay Bound - Tandem Topology

DISCO DNCv2 provides, depending on the choice of θ, a bound of either infinity or 0; this is caused by the lack of burstiness of the arrival curve and latency of the service curves, which cannot be represented in the other programs. The results produced by DISCO DNCv2 are, therefore, not depicted.

3.3.4. OMNeT++. The variable “interArrivalTime“ of the source refers to the time between generated jobs and is set to exponential(λ). The variable “serviceTime“ refers to the time required by the queue to serve a job and is set to is $\frac{1}{5}$, $\frac{1}{4.9}$, and $\frac{1}{4.5}$ for the three queues respectively. The variable which represents the backlog bound for the entire system is the sum of the variable “queueLength:max“ for all queues. The delay bound is represented by the variable “lifeTime:max“ of the sink. Each simulation was run for 2,000,000 events.

3.4. Results & Analysis

Figures 3 and 4 depict, respectively, the backlog and delay bounds of the calculations and simulations for the tandem topology. Figures 5 and 6 depict, respectively, the backlog and delay bounds of the calculations and simulations for the tree topology.

The data, produced by the most uniform configuration of the heterogeneous tools possible, shows that the values for both the backlog and delay are similar for less congested networks, i.e. networks with larger λ values. In more congested networks, however, DISCO SNC almost always provides significantly tighter bounds. The values produced by the OMNeT++ simulations are frequently smaller than those calculated by the other tools as it does not necessarily provide a worst-case bound (with certain tolerance) but the worst-case values found in simulations.

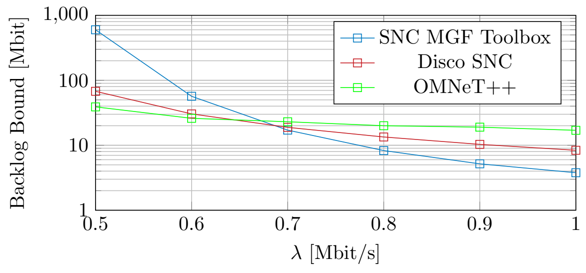


Figure 5: Backlog Bound - Tree Topology

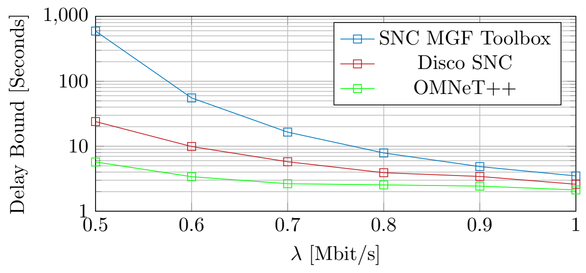


Figure 6: Delay Bound - Tree Topology

4. Conclusion & Further Work

This paper described the basics of stochastic network calculus as well as its similarities and differences with deterministic network calculus and queuing theory. Four tools used for the analysis of computer networks based on these theories / methods were then introduced, compared, and used to analyze simple reference topologies. The tools' heterogeneity posed the challenge of finding configurations that produce comparable results. The results show that, although the tools provided similar results for the tested networks, stochastic network calculus, specifically the DISCO SNC tool, could provide the tightest worst-case bounds. However, the limitation of the arbitrary scheduler available in the SNC MGF Toolbox contributed to the higher bounds produced by that tool. The results produced by DISCO DNCv2 were, unfortunately, incomparable with those produced by the other tools.

This paper was limited to the relatively theoretical application of stochastic network calculus. Further work could examine the bounds produced by the introduced tools in more complex settings and compare those with results from simulations in different simulators or real-world experiments.

References

[1] D. Ferrari, “Client requirements for real-time communication services,” *IEEE Communications Magazine*, vol. 28, pp. 65–72, 1990.

[2] R. L. Cruz, “A calculus for network delay. I. Network elements in isolation,” *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114–131, 1991.

[3] ——, “A calculus for network delay. II. Network analysis,” *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 132–141, 1991.

[4] Y. Jiang and Y. Liu, *Stochastic Network Calculus*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[5] M. Fidler and A. Rizk, “A guide to the stochastic network calculus,” *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 92–105, 2015.

[6] F. Baccelli, G. Cohen, G. Olsder, and J. Quadrat, “Synchronization and linearity - an algebra for discrete event systems,” *The Journal of the Operational Research Society*, vol. 45, 01 1994.

[7] A. Erlang, “The theory of probabilities and telephone conversations,” *Nyt Tidsskrift for Matematik B*, vol. 20, pp. 33–39, 01 1909.

[8] M. A. Beck and J. B. Schmitt, “The DISCO stochastic network calculator version 1.0 – when waiting comes to an end,” in *Proc. of the International Conference on Performance Evaluation Methodologies and Tools*, ser. ValueTools ’13, December 2013, pp. 282–285. [Online]. Available: https://dl.acm.org/citation.cfm?id=2631878

[9] P. Nikolaus, “Snc mgf toolbox.” [Online]. Available: https://github.com/paulnikolaus/snc-mgf-toolbox

[10] S. Bondorf and J. B. Schmitt, “The DiscoDNC v2 – a comprehensive tool for deterministic network calculus,” in *Proc. of the International Conference on Performance Evaluation Methodologies and Tools*, ser. ValueTools ’14, December 2014, pp. 44–49. [Online]. Available: https://dl.acm.org/citation.cfm?id=2747659

[11] A. Varga and R. Hornig, “An overview of the omnet++ simulation environment,” in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools ?08. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[12] M. A. Beck. [Online]. Available: https://disco.cs.uni-kl.de/discofiles/projects/SNC-Course/Script.pdf

An overview on Measurement Lab

Ben Julian Riegel, Simon Bauer, Johannes Zirngibl*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: ben.riegel@tum.de, bauersi@net.in.tum.de, zirngibl@net.in.tum.de

Abstract—
In times when more and more technologies become connected over the Internet, proper monitoring and maintenance of this network is more important than ever. Measurement Lab collects such data on Internet-scale and provides a public database for this data set.

This paper will show what Measurement Lab is, how it works and especially how to work with it. Our analysis shows that Transport Layer Security (TLS) can influence speed test results of weak clients with low computational power. The paper will also report that during the COVID-19 pandemic, the data set was spammed by single contributors, which highly impacted the results.

Index Terms—**measurement technology, internet speed tests, public data set, ndt, measurement-lab**

1. Introduction

Understanding and learning about the products you built, by monitoring the product and collection data about its behavior, is common practice. Just like this, network architects, researchers and consumers need accurate information on their Internet connection to properly maintain and monitor it.

The idea of Measurement Lab, short M-Lab, is to create a platform that satisfies consumers as well as researchers needs. Consumers receive information about how their specific connection performs, as well as researchers are provided with a publicly accessible database filled with information about worldwide broadband Internet performances collected by various open source services. Open source server software makes it possible for everyone to write own client sided software and offer it to the public to fulfill its own goals and simultaneously participate in the overall network of M-Lab. Also, this makes every service of M-Lab transparent and comprehensible [1].

M-Lab provides an infrastructure for the mentioned open source services to run. Therefore, they have servers spread all over the world. These servers collect all accruing data, which is generated by the services and put them together at one database. M-Lab also prepares data sets in different and useful views.

Since the platform was founded in 2009, it evolved and technologies became obsolete, the team behind M-Lab decided to completely rebuild the platform to so called M-Lab 2.0. With this update at the beginning of 2020, M-Lab retired services or completely rewrote them. All M-Lab 2.0 services now run on stock Linux kernel services, are dockerized (Docker: Software to isolate applications into containers) and are deployed on Googles container application management system Google Kubernetes Engine [2].

The remainder of this paper is to bring researchers closer to M-Lab. Therefore, Chapter 2 will present the technological background. Chapter 3 is based on this knowledge and presents an example of how to work with M-Lab's software, as it examines if TLS can influence the outcome of a speed test result. Chapter 4 will be dedicated to a wider view on the platform and the data. The paper will therefore show how to work with the raw data, as well as show how to use Big Query (a service by Google to manage and analyze large amounts of data) tables, M-Lab gives access to.

2. Background on M-Lab's Technology

Starting with a client who wants to know more details about his connection, M-Lab currently provides three services to choose from:

- Ndt (Network Diagnostic Tool):
  A test to find the maximum download and upload rates achievable.
- Neubot DASH:
  A test that emulates a video stream, to see how a connection performs under this circumstance.
- WeHe:
  A test how ones Internet connection handles traffic, collected from real world applications.

Since M-Lab is still changing a lot, this selection might differ over time. Moreover, the platform supports three core services. Core services are passively carried out all the time, when someone uses any service.

- Packet Header Service:
  Collects incoming packet headers for every incoming TCP connection.
- TCP Info:
  A service that collects data and creates statistics on all incoming TCP connections.
- Traceroute:
  A commonly known service that collects information about the network topology between server and client.

During the execution of one of the tests above, the M-Lab server saves all data which will be created throughout the execution to Google Cloud Storage.

The raw data then will be passed through a pipeline, where the data is summarized and additional information is added. In specific, every opened TCP connection throughout a test receives an Universally Unique Identifier (UUID), generated by concatenating the server host name, server boot time and the TCP socket cookie. This UUID can be used later to match query results with the actual raw data or join main service results with a core service. Geolocation information for the client's IP address is added as well. by the M-Lab annotation service[1]. The data then is accessible through Big Query. Important to mention is, that not yet every data set is supported by Big Query. To receive access to the Big Query project, you need to join the M-Lab discussion group on Google Groups.

Comparing the amount of entries in M-Labs data base for the Ndt service with other services, Ndt, by far, is the most used service on the platform. This is why I will focus on Ndt and its analysis in the following.

2.1. Ndt

Since Ndt7 is the latest version of the Network Diagnostic Tool protocol, provided by M-Lab, this paper will focus on this technology as older versions might not be supported anymore in the near future.

The main goal of this protocol is to flood a single TCP connection between a well-provisioned server and a client, to measure the maximum possible application layer throughput rates for up- and download. Ndt7 is based on HTTP WebSockets (chapter 2.2). If the congestion control algorithm TCP BBR (chapter 2.3) is available, it takes advantage of it. Otherwise, it will just use the systems default. [3]

2.2. WebSockets

A problem with the HTTP protocol is the large amount of overhead a real time connection generates because HTTP is not originally made for continuous communication. In general, the client must always generate a new request and receives the response afterwards. Multiple requests will be independent from each other. Because the execution of a Ndt7 test will generate a real time traffic between server and client, a lot of overhead in packet headers throughout the execution will be generated as well. Because Ndt7 is made to approximate the application level performance and HTTP operates on the application layer, using pure HTTP may highly impact the results. This is why websockets are used here. [4]

WebSockets work as an upgrade of a HTTP connection. To upgrade a connection, the client must specify the `Connection` and the `Upgrade` fields in the HTTP header during a request. An example request could look like this:

GET /index.html HTTP/1.1 \r\n
Host: www.hostname.com \r\n
Connection: upgrade \r\n
Upgrade: websocket \r\n
... \r\n

1. https://github.com/m-lab/annotation-service

The server will ignore the upgrade request with a standard `200 OK` response if it is not capable of upgrading. `101 Switching Protocols` will be the Status code for a confirmation. [4]

From now on the communication will adhere to the WebSocket specifications. Client and Server can now use a bidirectional communication channel, which maintains on one TCP session. In contrary to a HTTP header, a WebSocket header now only requires 8 Bytes. Therefore, less overhead will be generated throughout a real time connection. [4]

WebSockets differentiate between three main types of frames. Binary, textual and control frames. The payload of binary frames will be interpreted as pure bytes. Textual frames contain payload in UTF-8 encoding. Control frames are e.g. used to close a WebSocket channel. [4]

The WebSocket protocol supports two Uniform Resource Identifier Schemes. `WS://` indicates a standard communication channel, whereas `WSS://` indicates a TLS encrypted communication. [4]

2.3. TCP BBR

The TCP congestion control algorithm Bottleneck Bandwidth and Round-Trip Time (BBR) promises to maximize the TCP-level throughput and median RTT. It was developed by Google in 2016 and implemented in Linux v4.9. Taking advantage of TCP BBR as a service provider is easy, as there is no need for any action on the client side. It must only be deployed on the server side. [5]

A path from server to client consists of multiple hops which all store the incoming packages in a buffered queue. They process them (e.g. routing) and forward them. Thus, the RTT on this path starts increasing when the buffer of the slowest hop fills up faster than it drains. If the buffer is filled up completely, packet loss will appear. [6]

In difference to loss-based algorithms, BBR periodically monitors the RTT and the delivery rates. From this data it creates a model which includes the recent maximum available bandwidth, and the minimum recent RTT. This model then will be used to calculate how fast BBR will send the remaining data. This way, the throughput can be adjusted before packet loss appears, and the bandwidth will be better utilized while optimizing the median RTT and also generating less overhead in retransmission. [6]

Consequently, the bandwidth bottleneck of a BBR connection is the bit rate of the slowest hop in the path, which is exactly what we want to measure with Ndt7 [3]. Also, the kernel level data generated by BBR will be stored and attached to the Ndt7 raw data on Google Cloud storage.

2.4. Ndt7 Protocol Specification

Ndt7 differentiates between the upload and download tests as two completely independent tests. Therefore, the following HTTP paths are specified for GET requests: `/ndt/v7/download` and `/ndt/v7/upload` [3]

The client starts by requesting the desired test with the corresponding path, while requesting a WebSocket upgrade. If no error occurs, the server will reply with the `101 Switching Protocols` status code. [3]

When the WebSocket channel has been created successfully and the requested test was a download test, the server starts flooding the channel with binary frames, which must contain between 2^{10} and 2^{24} random bytes. If an upload test was requested, the client has to flood the WebSocket channel in this way. This frame size might be dynamically adjusted throughout the execution to better adapt to environmental requirements. [3]

A running time of up to 10 seconds for every execution is expected. During this interval, server or client are always permitted to provide textual frames, containing measured application level data in JSON format from its own side to support the counterpart with reliable speed measurements. E.g. these might be interesting if one would like to find out, how many bytes the counterparts application layer received of the acknowledged bytes on transportation layer.

These textual frames can be ignored completely from both sides as they only provide additional application-level information and are not mandatory to the protocol. If available, M-Lab servers attach these measurements under `Client Measurements` to the raw data. [3]

3. The influence of TLS on Ndt7

The specification itself claims that "Ndt7 should consume few resources" [3]. This gave the motivation why we wondered, whether TLS on weak clients with low CPU power, influences measurements, as it is the default choice for Ndt7 connections.

To test this, a stress test was set up, which would bring the protocol to its limits. A Raspberry Pi 2 Model B, running Raspbian Buster Lite release 2020-02-13, worked as a good representative of a weak client, according to the hardware specification[2]. It was directly plugged into a computer, running Ubuntu 20.04 LTS, using a cat 6 Ethernet cable. This computer provides way higher resources in terms of CPU power (i7-6700k) and network interface speed (1Gbit/s). The Raspberry Pi only offers a low-end CPU (ARM Cortex-A7) and a network interface with up to 100Mbit/s. The fast computer was running the official M-Lab server software[3] and BBR was enabled on the system. The Raspberry Pi was running a Ndt7 client software by Simone Basso[4].

The goal of this setup was to create a small network, where the results of a speed test would be given by the hardware of the Raspberry Pi, as it has the slowest hardware in every relevant point. If the computational power does not take any influences, we would expect the results to be capped by the network interface of the Raspberry Pi at around 100Mbit/s, no matter if the communication is encrypted or not.

The Client software was executed 10 times using the WS (no TLS) scheme and 10 times using the WSS (TLS) scheme. The client first executes a download test, followed by an upload test.

Figure 1 shows a boxplot of the measured TCP level throughput, using the data which was collected by the server throughout the tests. The median value

2. https://www.raspberrypi.org/products/raspberry-pi-2-model-b/
3. https://github.com/m-lab/ndt-server
4. https://github.com/bassosimone/ndt7-client-go-minimal

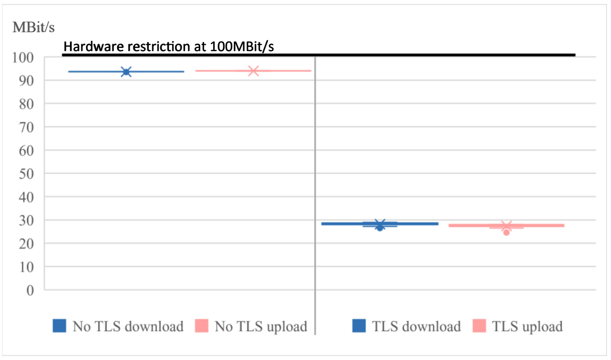


Figure 1: Box plot speed test results TLS vs. no TLS

(line between box borders), the mean value (cross), the 1st quartile (bottom border of each box) and the 3rd quartile (top border of each box) are represented in the Figure. Each plot includes 10 test executions to exclude measurement artifacts. We used the following method to calculate the results for Figure 1 from the raw data:

$$\text{Download} := \frac{AcknowledgedBytes_{max} * 8}{ElapsedTime_{max}} Mbit/s$$

$$\text{Upload} := \frac{ReceivedBytes_{max} * 8}{ElapsedTime_{max}} Mbit/s$$

As Figure 1 shows, for all results, median and mean value nearly line up. The boxes are represented as lines because the results only varied in the range of less than 1Mbit/s.

The median values of the non-encrypted tests are as expected around 100Mbit/s. The graph shows them at 93.6Mbit/s for the download and 93.9Mbit/s for the upload.

Interestingly, the encrypted tests revealed way lower results. The median is drawn at 28.5Mbit/s for the download and at 27.8Mbit/s for the upload.

As the encryption is the only thing that changed and encryption is a highly CPU consuming process, this highly indicates the low computational power of the Raspberry Pi to be the limiting factor in this connection. This would mean that speed test results can be influenced by how fast ones CPU is and does not necessarily show the maximum up- / download rate.

My suggestion would be to give weak clients the option to disable any kind of textual frames. So, no personal data would be leaked and they could safely execute non-encrypted tests, as this might be the only chance to receive results, which are not influenced by their CPU. Otherwise they might receive influenced results or reveal sensitive data.

4. Working with M-Lab

Switzerland was one of the first European countries, which considered taking down some of the most popular streaming websites during the COVID-19 pandemic because the internet congestion raised drastically. [7] To show how to access M-Labs data, this section will look at the up- and download speeds during the pandemic in Switzerland, to see if the mentioned increased internet congestion took a noticeable effect on peoples speed test results.

This is done in two approaches. The first one is about raw data from M-Labs Cloud Storage and processing it. The second one takes advantage of the prepared views on the data by using M-Labs Big Query tables. The time span from 01/01/2020 until 05/24/2020 was selected for both approaches, as this was the most recent entry in the data set at that time.

4.1. Approach 1: Using raw data

This approach works with unprocessed raw data, which was directly downloaded from the public Google Cloud Storage. Ndt7 is not yet supported by a Big Query table.

Ndt7 raw data is saved in JSON format. Parsing the data to a MySQL Database and adding geolocation information about the client's IP address by using the GeoLite2 Databases by Maxmind, revealed that around 77% of the tests in this data set came from the USA. Only 5826 of the approximately 7 million total samples could be traced back to Switzerland in the total data set. Even though all European M-Lab servers do provide Ndt7 support already, seemingly not many clients have implemented the newest version yet.

Because an average of only 40 Samples per day can be influenced easily by single contributors, this approach could not be further pursued, as the result will be unrepresentative. Still, this section could come up with useful information, which will be helpful for future work with M-Lab.

4.2. Approach 2: Using Big Query

To receive a more representative result than in chapter 4.1, more samples would be required. Therefore, we could also use legacy data from older Ndt versions, which are supported by Big Query.

The Big Query project by M-Lab provides two tables called `unified_downloads` and `unified_uploads`, which will be used in this section. They combine data from legacy Ndt versions. Geolocation information has already been added.

A SQL request asking for the median download and upload rates per day in Switzerland, showed a result, where the median values drastically dropped on 03/10/2020. To find out more, the SQL request was expanded by the amount of total test requests per day.

This revealed that the unexpected behavior of the median rates seem to correlate with the total amount of daily test requests (#reqests). #requests instantly raised from an average of 5000 to over 25000 on 03/10/2020, while up- and download rates decreased at the same time. This unfiltered data can be found in the data directory.

Further examinations showed that single IP addresses contributed multiple speed tests per day to the data set, which must not be surprising, as people can tests their internet speed multiple times per day. But besides some clients, which were already contributing hundreds of results, there were 3 conspicuous IP addresses, which had not contributed any data before 03/10/2020. They started spamming thousands of low-end results per day into the data set, beginning exactly on that date.

To filter them out, the next SQL request only took the maximum contribution per IP per day in consideration. This request is plotted in Figure 2. Mind that the y-axis on the right side correlates with the amount of total requests and the left y-axis with the up- and download median values in Mbit/s.

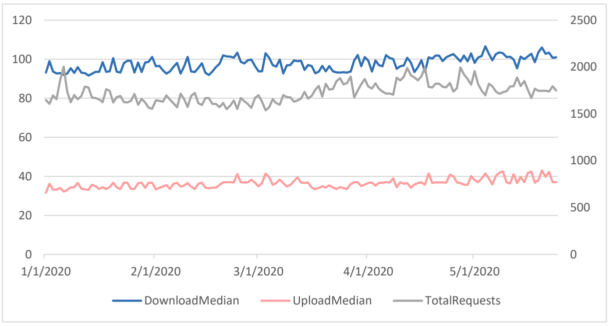


Figure 2: Throughput median in relation with #requests

According to this filtered data set, we can now see, that COVID-19 could not take very obvious influence on the median up- or download speed in Switzerland. The upload rates slightly fluctuate at around 97.6Mbit/s, while the upload values do so at around 36.7Mbit/s. No larger or unexpected outbursts are noticeable. In average, the download speed increased by 2.75Mbit/s from first half (01/01/2020 - 03/15/2020) to second half (03/16/2020 - 05/24/2020) of the time span. The upload did so by 1.69Mbit/s. More significant was the increase of #requests. This value has increased by an average of 145 for the second half.

In summary, we can not say that these decreased values at the beginning of the year necessarily have to do with the increased internet congestion during that time. The deviation is very small. The increased volume of #requests in the second half could be due to media coverage of the topic, so people wanted to test their internet speed, even though, it did not really decreased its median speed.

5. Conclusion and future work

After presenting the technological backround, the paper showed how to work with the open-souce software of the platform. It therefore revealed that a client can receive different speedtest results in the same enviroment, just by enabling the TLS encryption for the connection.

By presenting two possible approaches, the paper gave an idea on how to work with the M-Lab data set. The first one found out that for now, the platform lacks in sufficient Ndt7 sample data from europe. The second revealed a phenomena, where the median speedtest results in switzerland correlated with the amount of total requests made. This could be filtered out, but shows how the data can be influenced by single contributors.

In the future, it will be interesting to look at services like Neubot or Wehe and combine them with results from core tests, like Tranceroute.

Furthermore, it will be worth finding out more about the mentioned IP adresses, which were spamming the results from Chapter 4.2. Where are they located? This incidence might turn out to be related to COVID-19 in some way.

References

[1] “Open Internet Measurement,” https://www.measurementlab.net/about/, [Online; accessed 2020-05-28].

[2] S. Soltesz, “The 2.0 Platform Has Landed – Thank you!” https://www.measurementlab.net/blog/the-platform-has-landed/, 2020, [Online; accessed 2020-06-03].

[3] M-Lab, “ndt7 protocol specification,” https://github.com/m-lab/ndt-server/blob/master/spec/ndt7-protocol.md, [Online; accessed 2020-06-03].

[4] *The WebSocket Protocol*, https://tools.ietf.org/html/rfc6455, 2011.

[5] D. Scholz, B. Jaeger, L. Schwaighofer, D. Raumer, F. Geyer, and G. Carle, “Towards a Deeper Understanding of TCP BBR Congestion Control,” 2018.

[6] G. Huston, “Open Internet Measurement,” https://blog.apnic.net/2017/05/09/bbr-new-kid-tcp-block/, 2017, [Online; accessed 2020-06-03].

[7] “Netze wegen Corona-Krise überlastet: Müssen die Schweizer bald Netflix & Co. abschalten?” https://www.rtl.de/cms/schweizer-netze-in-corona-krise-ueberlastet-netflix-abschaltung-droht-wie-ist-es-in-deutschland-4506430.html, [Online; accessed 2020-12-08].

Using Self-Organizing Networks in 5G

Markus Schacherbauer, Anubhab Banerjee *
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: markus.schacherbauer@tum.de, anubhab.banerjee@tum.de

Abstract—**Over the years the number of mobile network users and the generated traffic by them has increased exponentially. To satisfy this growing demand, an efficient way of managing limited network resources is needed. An established way of doing so is using network automation. Currently in 4G, Self-Organizing Networks (SON) is the widely chosen approach for network automation. But as 5G brings new technologies and service requirements, SON has several drawbacks in such an environment as it was developed for 4G first. This paper addresses SON and several introduced advantages like reduced deployment effort and Automated Neighbor Relation (ANR) first. Afterwards, we explore some limiting factors like weak Self-Coordination that are hindering the deployment of SON as a sole network management automation (NMA) mechanism in 5G.**

Index Terms—**5G, network management, network automation, Self-Organizing Network**

1. Introduction

As described by Hämäläinen et al. in [1] SON was initially introduced with 3G but got a lot more attention in the last two decades as a key factor for deploying 4G. With a growing user base and demand for more traffic, MNOs (Mobile Network Operators) had to constantly upgrade their infrastructure further and use the existing one more efficiently. With no means of network automation at hand, all the planning for and configuration of a new base station was a human operator driven process. The same holds for ongoing optimization and error-solving in mobile networks. But as humans are very error-prone and slow regarding such tasks these areas were and still are very cost-intensive. To avoid high capital expenditures as well as operating expenses, network automation was needed.

SON especially aims to implement a kind of Plug-and-Play functionality for the deployment of new base stations as well as to introduce ways of automatically adapting control parameters of base stations to optimize e.g. Mobility Robustness. But as promised features of 5G as described by El Hattachi and Erfanian in [2] including greater throughput, ultra-high reliability, lower latency as well as higher mobility range and connectivity density require new technologies enabling these features, the bar for efficient network management automation will raise to a level where it remains questionable if SON can reach it.

In Section 2 we will first describe what the initial goals of SON were when it was developed. Afterwards, different architecture schemes for SON are presented as well as a black box model which is used to explain the workings of a SON function. In Section 3 the reader will find a selection of different advantages that SON usage promises, e.g. reduced deployment effort, energy-saving, Automated Neighbor Relation and more. In Section 4 we will discuss several drawbacks that SON usage in 5G has to face. The paper ends with a conclusion in Section 5 where also future developments of NMA are highlighted.

2. Description of SON

As shown by Lehser in [3] there are often named four main categories of SON use cases in general. These are:

- Planning
- Deployment
- Optimization
- Maintenance

While Planning and Deployment mainly focus on the initial deployment of automation in base stations, e.g. initial parameter configuration, Optimization and Maintenance do focus on the operational phase and often make use of SON functions. In the following section we will describe how a SON function works. The model can also be seen in Figure 1.

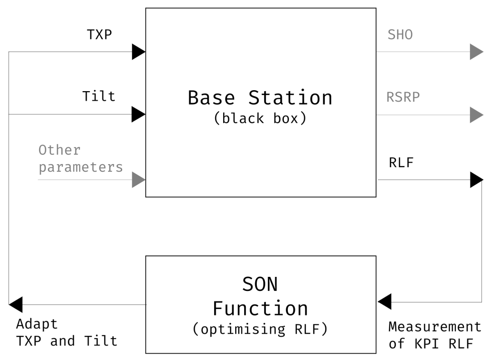


Figure 1: SON Function (optimizing RLF) measures Key Performance Indicator *Radio Link Failure* and adjusts the corresponding influencing control parameters *Antenna Tilt* and *Transmission Power*

2.1. SON Functions

When visualizing a base station one has to distinguish between two kinds of parameter sets. The first ones are the control parameters of the base station and the other ones are KPIs (Key Performance Indicators) that are somehow influenced by the control parameters. Control parameters may be but are not limited to:

- Transmission Power (TXP)
- Antenna Tilt or Remote Electrical Tilt (RET)

KPIs (Key Performance Indicators) may be but are not limited to:

- Radio Link Failures (RLF)
- Successful Handovers (SHO)
- Reference Signal Receive Power (RSRP)

RLF refers to some sort of break on the physical layer. SHO is a successful uninterrupted transfer of an ongoing connection from one cell to another. RSRP describes the received power of a special reference signal at an user equipment (UE), e.g. a smartphone. KPIs like RLF and SHO are measured by the base station itself while other KPIs, like e.g. RSRP can be measured by UEs and are reported back to the base station. One KPI gets always measured for fixed values of control parameters that are influencing it. The SON function which is responsible for this KPI regardless of where it resides — maybe in the base station itself — will adapt the control parameters (which are influencing the KPI) and trigger new measurements. From a pool of [Control parameters | KPI] vectors then the control parameters are chosen which optimize the KPI. So, for example a SON function trying to optimize RLF would conduct a variety of measurements of RLF while adapting TXP and Antenna Tilt as those two parameters have an impact on RLF. Out of all the [(TXP, Tilt) | RLF] vectors the (TXP, Tilt) configuration would be chosen for that RLF is minimal. SON functions often follow the structure of a basic Switch-Statement, following pre-defined rules for adjusting the control parameters of the base station. Those pre-defined rules are created by a human operator with a lot of expertise in the field of what the SON function tries to optimize and are not trivial to come up with. In general, another way of describing a SON function would be to call it a closed control loop.

2.2. SON Architectures

When looking at SON on a higher level, the question where to implement the actual SON functionality arises. In regards to Self-Configuration the functionality resides in the OAM (Operations, Administration, Maintenance) module of the MNO but regarding Self-Optimization and the black box model the SON function could be either inside the base station or it could also be somewhere else. In general, there are three possible answers to this question as described by Feng and Seidel in [4].

2.2.1. Centralized. In this approach all SON functionality resides in a dedicated module in the OAM system of the provider. That means that collected data and measurements first have to be passed to this module. On the one hand, it is easier to set up such a system as one only has to implement functionality at a rather high level in the architecture and at few places while on the other hand, this approach also comes with a drawback. As different providers have their different OAM systems, it becomes harder to optimize between them.

2.2.2. Distributed. This approach is very close to the black box model in Figure 1 when making the assumption that the SON function resides in the base station. While it becomes easier now to optimize between few neighboring base stations one has a higher effort to implement this architecture as there are many of them. Also, it gets harder to optimize with the number of base stations involved.

2.2.3. Hybrid. As both previous models suffered from major drawbacks, the solution is to combine the two approaches. So small optimizations are done on the level of the base stations by themselves but bigger optimization algorithms are run in the OAM system. The only drawback this model suffers from is the high effort to implement as especially interfaces have to be further extended.

3. SON Advantages

As seen in Section 2 SON does have a lot of different use cases ranging from the deployment of base stations to the subsequent maintenance as well as automated solving of problems that may occur. In the following some of those use cases are presented.

3.1. Reduced Deployment Effort

The deployment of a base station still mainly consists of manual efforts. Nevertheless, SON can support this deployment process and reduce the overall effort. When looking at the life cycle of the deployment of a base station SON can help with the following steps [3]:

- Authentication of the base station
- Installation of software, e.g. connecting to OAM and downloading configuration data
- Automated Transport and Radio Parameter Setup

3.2. Energy Saving

The basic idea of this use case is to adjust the network capacity to the needed load and not providing unused capacity and therefore wasting energy. The network capacity is reduced by switching off cells that are experiencing low traffic. To do this base stations have to hand over their current connections to neighboring or overlying cells first before being able to shut down. This deactivation function is triggered by the base station experiencing low traffic itself. The activation of a cell when the load on the network is increasing again however has to be performed by a neighboring cell. As shown by Roth-Mandutz and Mitschele-Thiel in [5] for example fingerprinting techniques can be used to identify the best fitting cell to be activated.

3.3. Automated Neighbor Relation

Base Stations take use of an intern NRT (Neighbor Relation Table) to perform certain actions, e.g. handovers. Such a table can be seen in Table 1 where entries are identified via the target cell id. Before the introduction of SON, NRTs of a base station were manually filled before the deployment by utilizing coverage predictions. But as those predictions were often error-prone and also

TABLE 1: Neighbour Relation Table (NRT) [4]

Neighbour	Local Cell ID	Target Cell ID	No Remove	No HO	No X2
1	L1	T1		x	
2	L1	T2	x		
3	L1	T3			x
4	L1	T4		x	x

network topologies tend to change over time a manual approach seems tedious. SON introduces an ANR module to automatically manage (delete and add) entries from the NRT in a way that in the end a base station can be deployed with an empty NRT without problems. The module consists of [4]:

- Neighbor Detection Function
- Neighbor Removal Function
- Neighbor Relation Table Management Function

The Neighbor Detection Function utilizes RRC (Radio Resource Control) signalling to detect new neighbors and decides whether to update the NRT or not by instructing the NRT Management Function to do so. After adding the new relation the NRT Management Function tells OAM about the change of the NRT and might get instructed to change some attributes (No Remove, No HO, No X2) or default values are used. The Neighbor Removal function is triggered whenever an entry in the NRT is used for a handover and starts a timer. When the entry is not used in a certain time frame again, it gets deleted. For further reading refer to [6] by Dahlén et al. and [4].

3.4. Optimization Algorithms in General

SON also introduced a huge variety of optimization algorithms [4]. These may be but are not limited to:

- Coverage Optimization
- Capacity Optimization
- Mobility Robust Optimization
- Mobility Load Balancing Optimization

Coverage and Capacity Optimization focus on maximizing the coverage (covered area of cell) and capacity of a cell. Mobility Robust Optimization attempts to detect and solve errors occurring due to too late or early handovers. Mobility Load Balancing Optimization handles the handing over from connections from cells facing high congestion to neighboring cells with free resources. In general, all these Optimization Algorithms follow the workings of a SON function as described in Section 2.1.

4. SON in 5G

While SON is still used in 5G there are several drawbacks that hinder the usage of SON the way it was introduced as the sole network management mechanism. SON advantages out of the Self-Configuration category mostly stay valid but primarily Self-Optimization use cases do not. Keshavamurthy and Ashraf state that even in 4G, Self-Optimization and Self-Healing functionality is not as widely deployed as it was initially planned [7]. In this section we will take a look at a selection of some of those drawbacks.

4.1. Reactive Character of SON

As already described in Section 2.1, SON functions themselves are working in a very reactive way. When using the same example with Radio Link Failures as in Section 2.1 those Radio Link Failures did already happen before SON attempts to combat those failures. SON only reacts to a problem after it already occurred. The simplified workflow basically can be reduced as described by Imran et al. in [8] to observing, diagnosing and reacting. As all those actions, especially observing, require a not to be neglected amount of time, the principle of SON functions automatically comes with an inherited delay. This was already an issue in 4G but now more then ever collides strongly with one of the goals of 5G, namely low latency. Therefore, in order to meet the requirements a much more preemptive SON would be needed.

4.2. No E2E (End-to-End) Network Visibility

As described by Mwanje, existing SON solutions primarily “considered automation for specific access network problems” [9]. But to guarantee the wanted quality of service, a complete overview of the performance of the system is strongly needed, especially when considering different network slices with different requirements. Another problem is that SON depends on the full availability of related data to the problem it should solve [7]. Data first has to be gathered through drive tests, OAM reports or customer complaints as seen in Figure 2. This approach also is not capable of predicting future behavior as it is not capable of generating a dynamic model of the system [8].

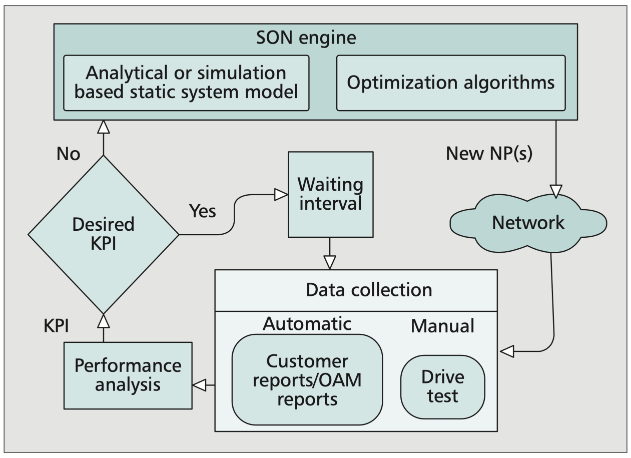


Figure 2: SON Engine: a data collection framework and a system model [8]

4.3. Missing Agreement on Common KPIs

As to the point of writing this paper, there is no unified agreement on specific key performance indicators (KPIs) across different mobile network operators to the best knowledge of the author. But especially with 5G multi-tenancy around the corner, a unified framework for performance evaluation is more important than ever. The usefulness of SON is very dependent on the chosen KPIs that SON functions are trying to optimize [8]. Shared use of infrastructure by different network operators (Multi-Tenancy) only seems possible when some common KPIs are established.

4.4. Weak Self-Coordination Functionality

When keeping in mind the model of a SON function as presented in Figure 1 it becomes clear that there are some SON functions trying to optimize different KPIs that do have common control parameters they are influenced by. This inevitably will lead to conflicts at some point. For example, imagine one SON function optimizing energy efficiency and therefore lowering the base station's transmission power. Another SON function however, may optimize the capacity of the cell and will raise the transmission power again. This will lead to a periodic oscillation of adapting the control parameter transmission power when not being managed by some instance in a higher layer. Unfortunately, even in 4G this still is only partially solved. A reason for that lies in the very foundation of the design of SON itself where SON functions have been developed rather independently of each other and only in hindsight coordination between functions has been added [8].

However, for 5G this Self-Coordination functionality has to be taken into account from the very beginning of development. With a trend towards network densification and network function virtualization (NFV) the amount of network components that have to be managed and coordinated will only grow further as described by Bhushan et al. in [10]. Multi-Tenancy on the one hand will introduce SON functions that only handle the performance of a specific slice but on the other hand also SON functions that perform optimization in between network slices. The RAN (Radio Access Network) Fronthaul Split will generate the additional need to also integrate new functionality to consider latency in the fronthaul [9]. So SON Self-Coordination will face serious challenges in 5G and is "an area of major concern" [7].

4.5. No Focus on Longtime Optimization

Current SON solutions mostly focus only on small-time-scale optimization but not on longtime optimization. This is problematic as there are not any more capacity gains to be expected on lower layers but rather in high network layers. Those higher layers often tend to work on a longer timescale and are important to be considered in order to adapt the network as well as possible to slow changes of user density and movement over time due to certain weekdays, months, seasons or even specific repeating events [8]. That way short time reactions to occurring problems could be avoided as a whole because they may not even occur at all in the first place.

5. Conclusion and Future Work

This paper first explained the reason for the invention of SON, which was the urgent need of mobile network operators to reduce their operating expenses. It then examined the different objectives of SON, namely Self-Configuration, Self-Optimization and Self-Healing and gave a detailed insight in an integral part of SON, the SON functions. Also, different SON architectures as well as use cases were discussed. We then proceeded to present different disadvantages that the deployment of SON would face in 5G and came to the result that while SON is still used in 5G there mainly only are Self-Configuration use cases that remain relevant. However, SON is not prepared to provide Self-Optimization and Self-Healing functionality in 5G. Future work outside the scope of this paper would be a discussion of key technologies that enable a shift from the reactive SON as presented here to a more proactive SON leveraging advancements in the fields of

- Machine Learning
- Big Data Analytics
- Knowledge Sharing

as seen in [9]. These technologies also mark an expected transition from Network Management Automation (NMA) towards Cognitive Network Management (CNM).

References

[1] S. Hämäläinen, H. Sanneck, and C. Sartori, *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Wiley, 2012.

[2] R. El Hattachi and J. Erfanian, "Ngmn 5g white paper," 2015.

[3] F. Lehser, "Next Generation Mobile Networks Use Cases related to Self Organising Network, Overall Description," 2008.

[4] S. Feng and E. Seidel, "Self-Organizing Networks (SON) in 3GPP Long Term Evolution," 2008.

[5] E. Roth-Mandutz and A. Mitschele-Thiel, "Lte energy saving son using fingerprinting for identification of cells to be activated," in *2013 Future Network Mobile Summit*, 2013, pp. 1–8.

[6] A. Dahlen, A. Johansson, F. Gunnarsson, J. Moe, T. Rimhagen, and H. Kallin, "Evaluations of lte automatic neighbor relations," in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, 2011, pp. 1–5.

[7] B. Keshavamurthy and M. Ashraf, "Conceptual design of proactive sons based on the big data framework for 5g cellular networks: A novel machine learning perspective facilitating a shift in the son paradigm," in *2016 International Conference System Modeling Advancement in Research Trends (SMART)*, 2016, pp. 298–304.

[8] A. Imran, A. Zoha, and A. Abu-Dayya, "Challenges in 5g: how to empower son with big data for enabling 5g," *IEEE Network*, vol. 28, no. 6, pp. 27–33, 2014.

[9] S. Mwanje, G. Decarreau, C. Mannweiler, M. Naseer-ul-Islam, and L. C. Schmelz, "Network management automation in 5g: Challenges and opportunities," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2016, pp. 1–6.

[10] N. Bhushan, J. Li, D. Malladi, R. Gilmore, D. Brenner, A. Damnjanovic, R. T. Sukhavasi, C. Patel, and S. Geirhofer, "Network densification: the dominant theme for wireless evolution into 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, 2014.

Atomic Broadcasts and Consensus: A Survey

Philipp Sedlmeier, Johannes Schleger*, Max Helm*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: philipp.sedlmeier@tum.de, schleger@net.in.tum.de, helm@net.in.tum.de

Abstract—
Atomic Broadcast and Consensus constitute important problems in distributed systems and have occupied computer scientists over the last four decades. As of late, they receive a new wave of attention. This paper provides an overview of the theoretical foundations of both topics. Furthermore, it presents recent approaches to the quest for the improvement of such protocols.

Index Terms—**consensus, atomic broadcast, fault tolerance**

1. Introduction

Today, one can find an abundance of distributed systems that rely on cooperating processes. For these processes, agreeing on data as the basis of their computations is fundamental. For example, the engine control and the flight surface control of an airplane's flight control system need to agree on whether to continue or abort a landing [1].

Malfunctions of a computer system may lead to disastrous outcomes; sticking to the airplane example, to a plane crash. This calls for fault-tolerant computers and computer networks. Fault-tolerance can, for example, be ensured by the use of process replication, in particular a synchronously replicated storage. This approach relies heavily on atomic broadcasts [2].

In this paper, we will provide a short comparison of recent concepts in these fields. In Section 2, we will investigate the topic's key terms and their relationship. We will present some related work in Section 3. Then, we will examine recent approaches in protocol development in Section 4. We will follow this with a discussion about if and how we can compare these protocols in Section 5.

2. Background

There exists an ample amount of literature about atomic broadcast and consensus [3]. However, there is a large divergence in the underlying definitions. Thus, we will provide the definitions of some essential terms in Section 2.1. Subsequently, we will give a short overview of theoretical results regarding the solvability of such problems in Section 2.2. Eventually, we will examine how the different concepts can be reduced to each other in Section 2.3.

2.1. Terms

Atomic Broadcast. The terms *Atomic Broadcast* and *Total Order Broadcast* are used interchangeably [3]. Informally, it requires all correct processes to "deliver the same messages in the same order" [4]. Formally, there are two major problem definitions. Following Cristian et al., an atomic broadcast protocol must fulfill the three properties *Atomicity*, *Order* and *Termination* [5]. However, all three properties depend on physical times, namely the points in time when each processor delivers an update.

Nowadays, definitions omitting such references are preferred [2]. Défago et al. emphasize the total order and identify the following four properties [3]:

- **Validity**: if a correct process broadcasts a message m, then it eventually delivers m.
- **Uniform Agreement**: if a process delivers m, all correct processes eventually deliver m.
- **Uniform Integrity**: for any message m, every process delivers m at most once, and only if m was previously broadcast by its sender.
- **Uniform Total Order**: if two processes p and q deliver messages m and n, then p delivers m before n if, and only if, q delivers m before n.

Nevertheless, they also point out that the agreement and total order properties are not necessarily specified uniformly and may also just apply to correct processes.

Consensus. The problem of *Consensus* is a problem of *agreement* between multiple processes on some value that one or more of those processes have proposed. The requirements for a consensus algorithm are summarized by Coulouris et al. [6]:

- **Agreement**: the decision value of all correct processes is the same.
- **Integrity**: if all correct processes proposed the same value, any correct process that has decided has chosen that value.
- **Termination**: Eventually, each correct process decides.

Integrity is also known as *validity* and sometimes defined differently; for instance, that the decision value initially must have been proposed by one of the processes [6].

Lamport et al. propose two variants of the consensus problem. In **interactive consistency**, each process computes a vector of values with an element for each process [7]. They require every correct process to compute the same vector, and every vector element corresponding to a correct process is that process's private value. The **Byzantine generals** problem involves one distinguished process that supplies a value the others are to agree upon [6], as is informally, but vividly illustrated by a commanding army general that sends messages to his lieutenants [8].

Permission. In the classical approach to consensus protocols, the communicating participants are already known beforehand. Such protocols are known as *permissioned*, in contrast to *permissionless* protocols, where participants can join or leave freely and neither their exact number nor their identity is known [9].

Failures. In the previous definitions, we already introduced the notion of a *correct* process. A correct process is any process that does not sustain process failures. These can be divided into four classes [3]:

- **Crash failure**: a process stops performing any activity.
- **Omission failure**: a process omits performing some actions, e.g. sending a message.
- **Timing failure**: a process violates timing assumptions of the system model[1].
- **Byzantine failure**: a process displays arbitrary or even malicious behavior.

These failure classes are nested in the above order, i.e. $CF \subset OF \subset TF \subset BF$ [5]. Indeed, most of the literature focuses only on crash and Byzantine failures.

2.2. Solvability

Consensus and atomic broadcast are easily solvable if the participating processes cannot fail [6]. Otherwise, the possibility of reaching consensus between processes is in question. We will shortly summarize the most influential results on this issue.

Fischer et al. showed that in an asynchronous system, every consensus algorithm has the possibility of nontermination, given the crash failure of only a single process [10][2]. Dolev et al. identified synchrony conditions and examined how they affect the number of faults that can be tolerated [11]. Dwork et al. introduced the concept of partial synchrony and determine the solvability of consensus for multiple partially synchronous models [12]. To bypass the problem altogether, Chandra and Toueg introduced unreliable failure detectors that can identify faulty processes and can be used to solve consensus, as long as the faulty processes are in the minority [4].

For any algorithm exist lower bounds on some properties. For example, if at most f processes sustain a crash failure, every algorithm reaching an agreement requires at least $f + 1$ rounds of information exchange [11]. In a system with n processes, of which f are faulty, consensus is solvable if and only if $n \geq 3f + 1$ (with Byzantine faults) and if and only if $n \geq 2f + 1$ (with non-Byzantine faults) [1], [13].

For a vivid description of the Byzantine case, we refer the avid reader to [8]. The basic intuition is that as long as $n \leq 3f$ holds, a node may be able to detect faulty behavior, but is not able to distinguish which node caused the faulty behavior and which node is a "victim" as well.

2.3. Problem Reduction

It has been shown that the previous concepts can be reduced to each other in many cases. We can see in [6] that a solution for any of the three variants of the consensus problem from Section 2.1, i.e. *Consensus*, *Interactive Consistency* and *Byzantine Generals*, can easily be used to construct a solution to one of the two other variants.

Chandra and Toueg proved consensus and atomic broadcast to be equivalent problems in asynchronous systems with crash failures. They also claim equivalence under arbitrary, i.e. Byzantine failures, but omit any proof [4]. Indeed, the relation between the two problems seems to be more complicated.

The first comprehensive study on this topic seems to be by Milosevic et al. They show that the equivalence of consensus and atomic broadcast does **not** hold in general, but the definition of *validity* determines whether they are equivalent, or one is harder than the other [14].

3. Related Work

Many theoretical results on consensus and related issues have been summarized by Fischer, for example in [1], [15]. The emergence of cryptocurrencies in the last decade, most notably Bitcoin, has attracted interest and progress in the development of the underlying consensus protocols. A survey of blockchain consensus protocols can, for example, be found by Xiao et al., who identify their core concepts and conduct a performance analysis [9].

Due to the scope of the topic, surveys on atomic broadcast protocols mostly concentrate on particular aspects. For example, Cason et al. characterize the latency variability of different atomic broadcast protocols [16]. The most extensive study so far seems to be one by Défago et al. that surveys - and classifies - around 60 algorithms [3]. Their classification is accompanied by a performance analysis concerning the message ordering strategies [17]. However, we did not find more recent studies that are nearly as extensive.

4. Available Protocols

We observe that many recently proposed protocols aim to improve particular aspects of already existing protocols. Consequently, we will investigate the issues that standard protocols suffer from, and present solutions that have been proposed recently.

In this survey, we will only cover permissioned protocols. Furthermore, all of these protocols assume a partially synchronous system [12]. We will examine crash-fault tolerant protocols (CFT) in Section 4.1 and those that tolerate Byzantine faults (BFT) in Section 4.2. We will present a new concept of fault tolerance in Section 4.3.

1. Naturally, timing failures can only occur in synchronous systems, as a system is considered to be *asynchronous* if we make no timing assumptions at all [4].

2. Due to the authors' initials, this result is informally known as *FLP impossibility*. Note, that it does not mean reaching consensus in such a system is not possible at all, there is just no deterministic solution.

4.1. Crash-Fault Tolerant Protocols

Paxos. The basis for most implementations of state machine replication until today is **Paxos**, first published in 1998 by Lamport [18]. Despite its prevalence, there are some substantial drawbacks to Paxos. To reach an agreement, more than half of the processes need to be running and communicating synchronously [18]. It is prone to failures that are common in some systems [19]. Furthermore, it depends on one process that acts as a leader. While Paxos is able to recover from a crash of this primary, this process is quite slow [19].

Raft. Another frequent critique of Paxos is that it is notoriously difficult to understand and not a good basis for practical systems. Therefore, Ongaro and Ousterhout developed **Raft** as an alternative, making use of problem decomposition and state space reduction [20]. Its new features include the concept of a strong leader that is authoritative for the distributed log entries. The leader election is performed through a heartbeat mechanism. That means, the current leader periodically sends heartbeat messages to its followers; if a follower does not receive such messages over a period of time, it begins an election for a new leader. To resolve or even prevent split votes in leader elections, Raft uses randomized timeouts. In contrast to Paxos, the leader election mechanism is a part of the consensus protocol itself [20].

However, while Raft's safety does not depend on timing assumptions, its availability does[3]. In particular, *broadcast time* $<$ *leader-election timeout* $<$ *mean time between failures* must hold. Indeed, Howard verifies Raft's efficiency in a well-understood network environment, where the parameters can be set accordingly [21]. She also states this is not the case in an internet-scale environment yet, but proposes modifications to that effect.

4.2. Byzantine-Fault Tolerant Protocols

Protocols that can tolerate Byzantine failures as well are widely considered to be badly scalable. Not only do they require more nodes than their CFT counterparts[4], node failures are also often assumed to be independent [22]. Yet, in order to achieve this, each node has to run with different operating systems and so forth. BFT protocols usually have higher time and message complexities as well. Nevertheless, research and development in such protocols have surged with the interest in permissioned blockchains [23].

Practical Byzantine Fault Tolerance (**PBFT**) by Castro and Liskov is the first practical implementation of a BFT consensus protocol [22]. The literature disagrees on whether PBFT is just inspired by Paxos or is its Byzantine version [9], [23]. Anyway, it is still "regarded as the 'baseline' for practical BFT implementations" [24]. However, Amir et al. showed the vulnerability of PBFT - and other leader-based protocols - to performance attacks by a small number of Byzantine nodes that can seriously impair the system's performance [25].

3. Otherwise, it would contradict the FLP impossibility result [10].
4. Namely at least $3f + 1$, as we have seen in Section 2.2

Scalability. Thai et al. proposed **HiBFT**, an extension of PBFT, that is supposed to be scalable up to hundreds of nodes [26]. Under the assumption that the system is organized in a hierarchical structure, multiple nodes are composed into logical groups. Then, not all nodes communicate with each other, but only the group leaders. Hence, HiBFT reduces the number of computationally expensive signature verifications and message complexity. Indeed, first results showed better performance in throughput and latency of HiBFT compared to PBFT, despite a sextupled number of nodes [26].

Performance. Zyzzyva, proposed by Kotla et al. [27], aims at reducing the replication overhead by omitting one communication phase by optimistic speculation. It executes client requests immediately, without running an agreement protocol first. This boosts the protocol's performance and reduces its message complexity - compared to PBFT - in gracious executions. Indeed, it is considered the "state of the art" concerning performance. However, Zyzzyva relies on the clients to resolve the cases where something went wrong; a client needs to detect whether it has received the same reply from all replicas [27]. Furthermore, safety violations of the protocol have recently been discovered [28].

Simplified View Change. Yin et al. presented **HotStuff** [28], which aims to achieve *optimistic responsiveness*, i.e. the designated leader needs to wait only for $n-f$ responses after global stabilization time (GST) [12] is reached. Hence, the protocol is designed to reach consensus fast, i.e. at the pace of the actual network delay. More importantly, it reduces the message complexity during view changes to linearity in the number of nodes even in the presence of leader failures. To achieve this, HotStuff adds a phase to the view change process and merges it into the regular protocol. This leads to a slightly higher latency, but also a higher throughput compared to other BFT protocols [28]. The modified version *Chained HotStuff* is essentially a pipelined version, where a quorum certificate can serve in different phases simultaneously. This version serves also as the basis of the LibraBFT consensus protocol, whose authors cite HotStuff's reduced communication costs as a main reason for using it [29].

Robustness. Clement et al. discovered that while "recently developed BFT state machine replication protocols are quite fast, they don't tolerate Byzantine faults very well" [30]. They state, that even single server or client failures can render such systems practically useless. Thus, they proposed **Aardvark**, a protocol specifically designed to be robust, i.e. that provides acceptable and predictable performance under all circumstances, including the occurrence of failures. To contain the effect of a Byzantine primary process, the system monitors its performance and changes the view if it is performing slowly. They found that Aardvark's performance is within 40% of the best performance of other state-of-the-art protocols on the same hardware during gracious executions, while it outperforms the same protocols by far under various attacks [30].

Leaderless BFT. Instead of improving BFT protocols by improving the leader-change mechanisms, another recent approach is the development of completely leaderless protocols. **AllConcur**, proposed by Poke et al., is a leaderless atomic broadcast protocol where the nodes "exchange messages concurrently through an overlay network, described by a digraph G" [31]. The algorithm makes use of a failure detector and an early termination mechanism. However, the algorithm's liveliness property only holds if the number of failures is bounded by the vertex connectivity of G and if the failure detector is complete and accurate. The authors claim that AllConcur's throughput beats that of a Paxos implementation by orders of magnitude; unfortunately, their comparison is not comprehensive and seems to cover only one specific case [31].

Crain et al. proposed **DBFT** that replaces a leader by a *weak coordinator* that does not impose its value on its fellow processes [32]. Thus, non-faulty processes can decide on a value without its help and a faulty coordinator cannot prevent the other processes from reaching consensus. Unfortunately, the authors do not provide a performance evaluation of the protocol, apart from reporting a quadratic message complexity. They claim, however, that DBFT is used by one of the fastest blockchains to date [33].

4.3. Cross-Fault Tolerant Protocols

Liu et al. argue that the overhead for multiple properties inherent to BFT is mostly due to the "assumption of a powerful adversary that can fully control not only the Byzantine faulty machines, but at the same time also the message delivery schedule across the entire network" [34]. However, they claim that this scenario is rather unrealistic in most cases. Hence, they propose the concept of **Cross-Fault tolerance**, **XFT** for short. XFT requires the same number of replicas as CFT, i.e. $2f + 1$, and provides all its reliability guarantees. Thus, it is strictly stronger than CFT. In addition, it provides safety and liveliness when Byzantine faults occur, as long as a majority of the replicas are correct and can communicate with each other synchronously.

In the same paper, they provide a XFT-SMR protocol, *XPaxos*. Its performance outperforms PBFT and Zyzzyva, while coming close to the performance of an optimized Paxos [34]. However, it doesn't scale with the number of faults and suffers from the same performance shortcomings in the case of failures as other leader-based protocols. Thus, Garg et al. recently proposed the multi-leader XFT consensus protocol **Elpis**. It introduces a concept of per-object ownership, where ownership of accessible objects is assigned to the nodes. Then, not a single leader is responsible for ordering all commands, but each node is responsible for ordering the commands that concern the objects it has been assigned to as owner. This ownership can also be changed dynamically. The authors claim that Elpis achieves a performance twice as high as XPaxos [35].

5. Comparing Protocols

As we have seen in Section 4, many protocols aim to improve or remedy particular shortcomings of already existing protocols. Even though all of these protocols solve the same problem - i.e. the problem of consensus or atomic broadcast - the underlying application determines which protocols can be used. For example, HiBFT can be used for a permissioned blockchain only if its replicas are - or can be - ordered in a hierarchical structure.

Singh et al. argue that comparing different BFT protocols and choosing an appropriate one is difficult because they are evaluated under different and benign conditions [24]. Therefore, they propose a simulation environment that provides identical and realistic conditions. Their comparison yields insight into which characteristics of a system environment favor or disadvantage a specific protocol. Nevertheless, they also conclude that there is no one-size-fits-all protocol and that such a protocol may be impossible to build at all [24]. A similar conclusion was drawn earlier by Défago et al. in their performance analysis of atomic broadcast protocols [17].

The discussion whether there is a protocol superior to the others is not restricted to BFT consensus alone. While Paxos and Raft are the two dominating algorithms for distributed consensus, it is an open discussion which is the better one, as Howard and Mortier point out [36].

Unfortunately, there are still very few comparisons of consensus or atomic broadcast protocols available. Those that are available focus on blockchain consensus and do not make use of a simulation environment [9], [37]. The finding from [24] that there is no single superior protocol available seems to have gained significantly more popularity. Among others, it inspired the development of **Hyperledger Fabric**, a blockchain platform with a modular consensus component, so that the implementation of consensus can be adjusted to the present use case [38].

We should also note that some problems are inherent to consensus protocols in general, regardless of their specific application. For example, a high message complexity is a burden on every protocol that solves BFT consensus. Thus, developments that reduce the cost of communication in distributed systems may benefit all consensus protocols similarly. To that end, Goren and Moses recently examined silent information transfer in the presence of failures [39].

6. Conclusion

As we have seen, there exists a vast and still growing number of consensus and atomic broadcast protocols. These protocols vary not only in design and performance but also in the underlying assumptions. Which protocol to use depends on the specific setting and is still an open discussion. Thus, modular and extensible projects may become even more relevant in the future.

References

[1] M. J. Fischer, "The Consensus Problem in Unreliable Distributed Systems (A Brief Survey)," in *Proceedings of the 1983 International FCT-Conference on Fundamentals of Computation Theory*, M. Karpinski, Ed. Springer, 1983.

[2] X. Défago, "Atomic Broadcast," in *Encyclopedia of Algorithms*, M.-Y. Kao, Ed. Boston: Springer, 2008.

[3] X. Défago, A. Schiper, and P. Urbán, "Total Order Broadcast and Multicast Algorithms: Taxonomy and Survey," *ACM Computing Surveys*, vol. 36, no. 4, Dec. 2004.

[4] T. D. Chandra and S. Toueg, "Unreliable Failure Detectors for Reliable Distributed Systems," *J. ACM*, vol. 43, no. 2, Mar. 1996.

[5] F. Cristian, H. Aghili, R. Strong, and D. Dolev, “Atomic Broadcast: From Simple Message Diffusion to Byzantine Agreement,” *Information and Computation*, vol. 118, no. 1, 1995.

[6] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. Boston: Addison-Wesley, 2012, ch. Consensus and related problems.

[7] M. Pease, R. Shostak, and L. Lamport, “Reaching Agreement in the Presence of Faults,” *J. ACM*, vol. 27, no. 2, Apr. 1980.

[8] L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, Jul. 1982.

[9] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, “A Survey of Distributed Consensus Protocols for Blockchain Networks,” *ArXiv*, vol. abs/1904.04098, 2019.

[10] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of Distributed Consensus with One Faulty Process,” *J. ACM*, vol. 32, no. 2, Apr. 1985.

[11] D. Dolev, C. Dwork, and L. Stockmeyer, “On the Minimal Synchronism Needed for Distributed Consensus,” *J. ACM*, vol. 34, no. 1, Jan. 1987.

[12] C. Dwork, N. Lynch, and L. Stockmeyer, “Consensus in the Presence of Partial Synchrony,” *J. ACM*, vol. 35, no. 2, Apr. 1988.

[13] L. Lamport, “Lower bounds for asynchronous consensus,” *Distributed Computing*, vol. 19, no. 2, Oct. 2006.

[14] Z. Milosevic, M. Hutle, and A. Schiper, “On the Reduction of Atomic Broadcast to Consensus with Byzantine Faults,” in *2011 IEEE 30th International Symposium on Reliable Distributed Systems*, 2011.

[15] M. J. Fischer, “A Theoretician’s View of Fault Tolerant Distributed Computing,” in *Fault-Tolerant Distributed Computing*, B. Simons and A. Spector, Eds. New York: Springer, 1990.

[16] D. Cason, P. J. Marandi, L. E. Buzato, and F. Pedone, “Chasing the Tail of Atomic Broadcast Protocols,” in *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*, 2015.

[17] X. Défago, A. Schiper, and P. Urbán, “Comparative Performance Analysis of Ordering Strategies in Atomic Broadcast Algorithms,” *IEICE Trans. on Information and Systems*, vol. E86-D, no. 12, Dec. 2003.

[18] L. Lamport, “The Part-Time Parliament,” *ACM Transactions on Computer Systems*, vol. 16, no. 2, May 1998.

[19] H. Howard, “Distributed consensus revised,” University of Cambridge, Tech. Rep. UCAM-CL-TR-935, Apr. 2019.

[20] D. Ongaro and J. Ousterhout, “In Search of an Understandable Consensus Algorithm,” in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia: USENIX Association, Jun. 2014.

[21] H. Howard, “ARC: Analysis of Raft Consensus,” University of Cambridge, Tech. Rep. UCAM-CL-TR-857, Jul. 2014.

[22] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. USENIX Association, 1999.

[23] C. Cachin and M. Vukolić, “Blockchain Consensus Protocols in the Wild,” in *31st International Symposium on Distributed Computing (DISC 2017)*, A. W. Richa, Ed., vol. 91. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[24] A. Singh, T. Das, P. Maniatis, P. Druschel, and T. Roscoe, “BFT Protocols under Fire,” in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 2008.

[25] Y. Amir, B. Coan, J. Kirsch, and J. Lane, “Prime: Byzantine Replication under Attack,” *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 4, 2011.

[26] Q. Thai, J.-C. Yim, T.-W. Yoo, H.-K. Yoo, J.-Y. Kwak, and S.-M. Kim, “Hierarchical Byzantine fault-tolerance protocol for permissioned blockchain systems,” *The Journal of Supercomputing*, vol. 75, Jun. 2019.

[27] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, “Zyzzyva: Speculative Byzantine Fault Tolerance,” *ACM Transactions on Computer Systems*, vol. 27, no. 4, Jan. 2010.

[28] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, “HotStuff: BFT Consensus in the Lens of Blockchain.” *arXiv: Distributed, Parallel, and Cluster Computing*, 2018.

[29] LibraBFT Team, “State Machine Replication in the Libra Blockchain,” May 2020.

[30] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, “Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults,” in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 2009.

[31] M. Poke, T. Hoefler, and C. W. Glass, “AllConcur: Leaderless Concurrent Atomic Broadcast,” in *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 2017.

[32] T. Crain, V. Gramoli, M. Larrea, and M. Raynal, “DBFT: Efficient Leaderless Byzantine Consensus and its Application to Blockchains,” *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, 2018.

[33] T. Crain, C. Natoli, and V. Gramoli, “Evaluating the Red Belly Blockchain,” *ArXiv*, vol. abs/1812.11747, 2018.

[34] S. Liu, P. Viotti, C. Cachin, V. Quéma, and M. Vukolić, “XFT: Practical Fault Tolerance beyond Crashes,” *ArXiv*, vol. abs/1502.05831, 2016.

[35] M. Garg, S. Peluso, B. Arun, and B. Ravindran, “Generalized Consensus for Practical Fault Tolerance,” in *Proceedings of the 20th International Middleware Conference*. ACM, 2019.

[36] H. Howard and R. Mortier, “Paxos vs Raft: Have We Reached Consensus on Distributed Consensus?” in *Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data*. New York: Association for Computing Machinery, 2020.

[37] S. Wan, M. Li, G. Liu, and C. Wang, “Recent advances in consensus protocols for blockchain: a survey,” *Wireless Networks*, Nov. 2019.

[38] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains,” in *Proceedings of the Thirteenth EuroSys Conference*. New York: Association for Computing Machinery, 2018.

[39] G. Goren and Y. Moses, “Silence,” *J. ACM*, vol. 67, no. 1, Jan. 2020.

Surveying P4 Compiler Development After 2016

Yue Wu, Henning Stubbe*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: yue02.wu@tum.de, stubbe@net.in.tum.de

***Abstract*—Software-defined networking (SDN) initially appeared as a new network management technology aimed at improving network performance. Since 2013, SDN associated with OpenFlow protocol (a communication protocol) has become an industry standard. However, SDN-related protocols need to specify their headers on the hardware device they operate, which limits the flexibility for targeting different devices and increases the complexity for future protocol expansion. To address this problem, the P4 language was introduced as a protocol-independent programming language for describing the process of network data packets and now has been widely used in many different devices, such as Application-specific integrated circuit (ASIC), Field-programmable gate array (FPGA), Network interface card (NIC), CPU etc. through the corresponding compiler. The purpose of this survey is to present the latest varieties of P4 compilers, including their respective characteristics and target equipment.**

***Index Terms*—P4 compiler, P4FPGA, P4LLVM, T4P4S, p4c-XDP**

1. Introduction

Nowadays, as the requirements for network performance increase, more network equipment is needed which leads to more and more cumbersome configuration of traditional equipment [1]. To prevent this trend, next generation networks are supposed to have the following characteristics: programmable customization on demand, centralized and unified management, dynamic traffic supervision and automated deployment [2]. That is why the concept of SDN was born. SDN is physically separated into a control plane and a forwarding plane [3]. The former provides the intelligent logic in network equipment, which controls how to manage data traffic, while the latter manages forwarding/manipulating/discarding network data traffic. This improvement will lead to a more efficient configuration process when modifying different network behavior, since the control plane is the only part to change.

However, the fact that only the control plane can be used for programming could still raise some problems. Under normal circumstances, data packets in the forwarding process are solidified by the forwarding chip of the device that usually does not support protocol expansion. In addition, the cost of developing new forwarding chips that support new protocols or extended protocol features is also very expensive. The need to design such hardware will lead to a series of problems, such as high update costs and long development time. Therefore, the new generation of SDN solutions should enable the forwarding plane to be programmable as well, so that the software can truly define the network and network equipment. P4 provides users with this function, which breaks the limitations of the hardware devices on the forwarding plane and allows programming to control the analysis and forwarding process of data packets. Therefore, the network and devices are "open" to users from top to bottom. Section 2 will provide a short introduction to the P4 language. After that, four different P4 compilers will be presented in Section 3. Section 4 will give an overview of the future research direction.

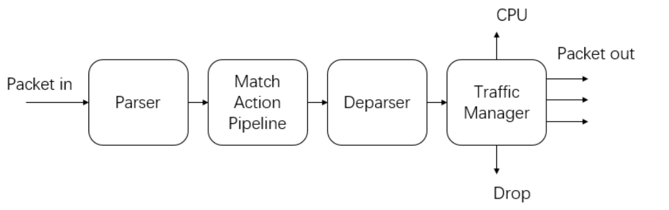


Figure 1: Example P4 Abstract Architecture [4]

2. P4 language

Before delving into the various P4 compilers, the P4 language will be introduced first.

The Programming Protocol-Independent Packet Processors (P4) is a Domain-Specific Language which was first proposed by Bosshart et al. in [3]. As a design goal, P4 is expected to achieve the following three design goals: 1) Protocol independence: Network equipment is not bound to any specific network protocol, and users can use P4 language to describe any network data plane protocol and packet processing behavior; 2) Target independence: Users do not need to care about the details of the underlying hardware to implement the programming description of the data packet processing method; 3) Reconfigurability: Users can change the program of packet parsing and processing at any time and configure the switch after compilation to truly implement on-site reconfiguration. In order to realize the above-mentioned goals, P4 language compilers are required to adopt a modular design, while the input and output of each module adopt standard configuration files. An abstract architecture of P4 is shown in Figure 1.

3. P4 Compilers

When P4 first appeared, it was still mainly oriented to the software control plane. In order to improve the performance of the programmable forwarding plane, a platform is needed which can help researchers to efficiently design on hardware. So far, hardware devices like ASIC, FPGA, NIC and CPU have been widely used, but been specified in their own programming language [5]. Therefore, P4 compilers show their importance in this case because they connect the P4 program with the underlying hardware that were initially unrelated. A typical P4 compiler has two main tasks: to generate the configuration at compile time to implement the data plane, and to generate the application programming interface (API) to populate tables [4].

3.1. General information and reference compiler

When the first version of P4 language $P4_{14}$ appeared, p4c-behavioral [6] was the standard P4 compiler, which used p4-hilr [7] to convert the source code to the P4 intermediate representation (IR). Typically IR is the data structure internally used by a compiler to represent source code. Later, $P4_{14}$ was found to have syntax and semantics problems [8]. In order to address these issues, a new version of P4 language $P4_{16}$ was released in 2016. Compared with the old version, in $P4_{16}$, a large number of language features have been transferred from the language to the libraries including counters, checksum units, meters, etc. As a result, the P4 language has been transformed into a more compact core language with libraries. p4c [9] is now the reference modular compiler for P4 that supports both $P4_{14}$ and $P4_{16}$. It can provide the target independent front-end compiler itself, and support different target specific backend compilers which will be introduced in the following subsections.

3.2. Target specific compiler - P4FPGA

FPGA is an ideal target platform for P4 due to its high degree of programmability [10]. However, the design of a compiler that converts P4 language into FPGA HDL code faces the following difficulties: 1) FPGA is mainly programmed through a low-level, non-portable code base; 2) Due to the differences between programs and different loading strategies adopted by different architectures, it is difficult to generate efficient hardware code implementation based on P4 source code; 3) Although the P4 language is not aware of the underlying hardware architecture, it relies on a series of "extern" syntax to import external valid functions, which makes code generation more complicated. To solve these problems, H. Wang et al. introduced the P4FPGA compiler in [4] which guarantees flexibility, efficiency and portability between the P4 program and FPGA device.

The proposed P4FPGA compiler reuses the reference P4 compiler p4c as its front-end to reduce engineering workload. As far as language version support is concerned, it can be used under $P4_{14}$ and $P4_{16}$ syntax, and can also be applied to different architecture configurations, which will be mentioned in 3.2.2. Figure 2 outlines the workflow of P4FPGA. Among them, code generation, P4FPGA runtime and optimization principles implemented as IR to IR transformers are the core parts.

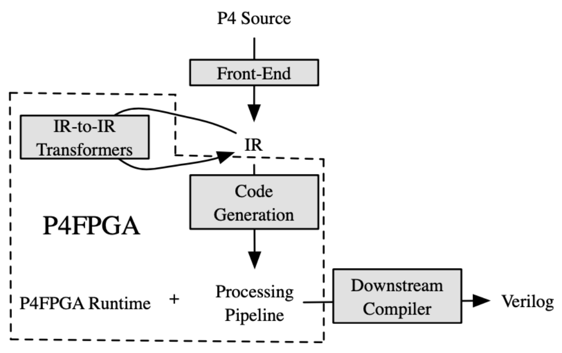


Figure 2: P4FPGA structure [4]

3.2.1. Code generation. In P4FPGA, the physical processing structure is generated as a block. These basic blocks are implemented in P4FPGA by using parameterized templates. During initialization, these templates are hardware modules used to implement the parser, match-action and deparser logic.

3.2.2. P4FPGA runtime. This is another important part of P4FPGA because it provides an efficient, flexible and scalable execution environment for the processing algorithms in P4. It defines a method that allows the generated code to access general-purpose functions through the untargeted abstraction, and provides an abstract architecture that can be implemented uniformly on different hardware platforms.

P4 programmers may write various network applications and put forward different requirements on the runtime. Therefore P4FPGA provides two architectures to support potential user scenarios: 1) Multi-port switching which is suitable for networking forward components like switches and routers and testing new network protocols; 2) Bump-in-the-wire which is suitable for network functions and network acceleration, but with only one input port and one output port.

3.2.3. Optimization principles. Finally, in order to ensure the high efficiency of generating code through P4FPGA, optimizations such as leveraging hardware parallelism in space and time to increase throughput, transforming sequential semantics to parallel semantics to reduce latency, using a resource-efficient component to implement match tables etc. are implemented in the context of the NetFPGA SUME platform in [4].

In summary, the P4FPGA includes a C++ based compiler along with a Bluespec-based [11] runtime system, as well as a p4c frontend and a custom backend. All source code is available at http://www.p4fpga.org.

3.3. Process optimizing compiler - P4LLVM

In addition to the perspective of a hardware-target backend compiler, there also exists compilers whose goals are optimization algorithms. P4 language with a better configuration framework can greatly shorten the packet processing time, thereby maximizing the use of network resources. P4LLVM was introduced by Dangeti et al.

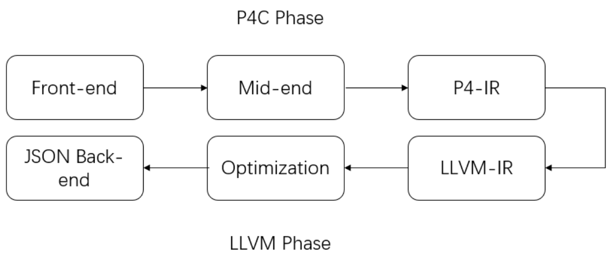


Figure 3: Flow of P4LLVM [12]

in [12] as a compiler based on the LLVM framework, and is proved to have a better performance than the standard p4c compiler.

LLVM is an abbreviation for Low Level Virtual Machine. It is a compiler framework designed to support transparent, life-long program analysis and transformation for arbitrary programs by providing high-level information to compiler transformations at compile-time, link-time, run-time and in the idle time between runs [13]. Just like p4c, the LLVM framework provides great convenience when plugging front-ends, back-ends and optimizations to different targets and accessing various machines. This advantage indicates that LLVM can be used in conjunction with P4, which is how P4LLVM invented.

P4LLVM only supports the $P4_{16}$ programs and reuses the p4c frontend module to check the lexical, syntactic and semantic correctness of the P4 code and mid-end module for preprocessing. After that, the intermediate representation of P4 (P4-IR) is converted to the intermediate representation of LLVM (LLVM-IR), then the IR is passed through various optimization sequences of LLVM and finally translated into JSON format (an open standard file format) to target a BMV2 switch (a software P4 switch).

In [12], the authors used the P4 code generated by Whippersnapper, which is a P4 benchmark suit designed to study the impact of the compiler on performance to demonstrate that P4LLVM has exceeded p4c with respect to percentage increase in average latency versus number of operations in the action block and number of tables.

At present, p4c only has implementations of dead state elimination, constant propagation, constant folding and expression simplification [12]. In contrast, the LLVM framework has been carefully designed and many other optimizations have been added, including all p4c characteristics. Therefore, the P4LLVM compiler will help target many common backends with minimal effort. Figure 3 describes the workflow on how to combine p4c and P4LLVM.

3.4. Multi-target compiler - T4P4S

Another aspect of designing a compiler is targeting different hardware with a single compiler because it is much more efficient than creating a separate compiler just for a specific device. Vörös et al. proposed a multi-target compiler T4P4S (Translator for P4 Switches) in [14], which achieves a good balance between complexity, portability and performance. In the design process of T4P4S, the following principals are regarded as the main features of T4P4S: 1) This compiler should be retargetable, which means it can be easily deployed on another hardware; 2) The compiler should present a universal performance on all supported hardware and software and behave comparably to the native methods; 3) When processing forwarding logic to suit the target, the compiler should generate high level programs that do not require additional modifications. In order to meet these three objectives, the compiler T4P4S is separated into two parts: a hardware-independent core and a NetHAL (Networking hardware abstraction layer) responsible for the hardware-related parts. T4P4S currently only supports the original $P4_{14}$ language; new version extensions that support the new $P4_{16}$ language are still under development. Figure 4 shows the workflow of T4P4S.

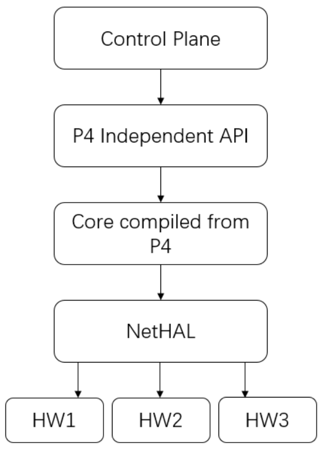


Figure 4: Architecture of T4P4S workflow [14]

3.5. Linux kernel targeted compiler - p4c-XDP

p4c-XDP is a Linux kernel target compiler, which can convert P4 programs into C code, then compile it to eBPF and then load it into the Linux kernel for packet filtering.

eBPF is an extended version of BPF and was reformed based on BPF by Alexei Starovoitov in 2013 [15]. BPF, known as Berkeley packet filter, was originally proposed by Steven McCanne et al. in [16]. Its purpose is to provide a method of filtering data packets and avoid useless copying of data packets from kernel space to user space. It initially consisted of a simple byte-code that is injected into the kernel from user space, where it is checked with a checker to avoid kernel crashes or security issues and attached to a socket and then runs on each received packet. In contrast, eBPF added new features to improve its performance, such as mapping and tail calls, and also rewrote the just-in-time compiler (a compiler can convert BPF instructions into native code). The new language is closer to the native machine language than before. Also, new attachment points will be created in the kernel.

An XDP program is a special case of the eBPF program and is used to process network packets. It is attached to the lowest level of the networking stack [17] and it is also a new fast path. XDP is used in conjunction with the Linux stack and the BPF is used to make packet processing faster.

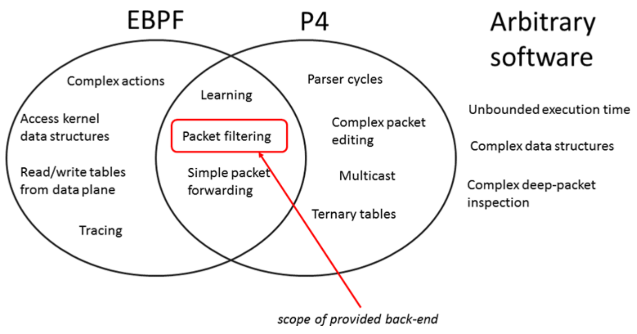


Figure 5: overlap between eBPF and P4 [18]

Although eBPF and P4 are two programming languages with different expression capabilities, there is overlap between the domain of network packet processing. Therefore, it is worth to develop a connection between P4 and eBPF [18].

Figure 5 presents the overlap between eBPF and P4.

3.6. Other compilers

Apart from the P4 compilers mentioned above, there are other P4 compilers with different functions, such as p4c-graphs, which can be used to generate visual representations of a P4 program [9]; p4test is a source-to-source P4 translator which can be used for testing, learning compiler internals and debugging [19]; p4c-ubfp can be used to generate eBPF code running in user-space [9]. However, due to insufficient research regarding these compilers, they cannot be introduced in detail in this survey, but they may become the motivation for future research.

4. Conclusion and future work

This survey outlines the latest research on P4 compilers from four different directions: The first kind is a hardware-specific compiler, which focuses on improving the efficiency of converting P4 programs into certain target hardware language like P4FPGA. The second type attempts to use existing mature compiler frameworks to optimize the performance of total packet processing and leads to an upgraded version of the standard P4 compiler. The third is to implement a multi-target compiler to reduce the effort spent on configuring with different hardware terminals like T4P4S. The last kind targets to the Linux kernel based on the standard p4c compiler.

In addition to the research effort on the P4 compiler, many researchers are also developing other extension features of P4. As mentioned at the beginning, SDN and P4 and other similar technologies will dominate the future network developing due to their flexibility and portability, which is a determining point compared to the traditional networking technology.

The P4 language and its compilers are still in the development stage. It is foreseeable that there will be more powerful P4 compilers in the future. These compilers can make the connection between the P4 program and the target hardware/software more robust and simple.

References

[1] H. Kim and N. Feamster, “Improving Network Management with Software Defined Networking,” *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[2] A. Gelberger, N. Yemini, and R. Giladi, “Performance Analysis of Software-Defined Networking (SDN),” in *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 2013, pp. 389–393.

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, “P4: Programming Protocol-Independent Packet Processors,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[4] H. Wang, R. Soulé, H. T. Dang, K. S. Lee, V. Shrivastav, N. Foster, and H. Weatherspoon, “P4FPGA: A Rapid Prototyping Framework for P4,” in *Proceedings of the Symposium on SDN Research*, 2017, pp. 122–135.

[5] J. S. da Silva, T. Stimpfling, T. Luinaud, B. Fradj, and B. Boughzala, “One for All, All for One: A Heterogeneous Data Plane for Flexible P4 Processing,” in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 440–441.

[6] p4language, “P4 compiler for the behavioral model,” https://github.com/p4lang/p4c-behavioral.html, 2017, accessed June 11, 2020.

[7] ——, “p4-hlir,” https://github.com/p4lang/p4-hlir.html, 2017, accessed June 11, 2020.

[8] T. P. L. Consortium, “P4$_{16}$ Language Specification,” https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html, 2017, accessed June 11, 2020.

[9] p4language, “P4$_{16}$ reference compiler,” https://github.com/p4lang/p4c.html, 2020, accessed June 11, 2020.

[10] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, “Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN,” *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 99–110, 2013.

[11] R. S. Nikhil and K. R. Czeck, “BSV by Example,” *CreateSpace, Dec*, 2010.

[12] T. K. Dangeti, R. Upadrasta *et al.*, “P4LLVM: An LLVM Based P4 Compiler,” in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 424–429.

[13] C. Lattner and V. Adve, “LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation,” in *International Symposium on Code Generation and Optimization, 2004. CGO 2004*. IEEE, 2004, pp. 75–86.

[14] P. Vörös, D. Horpácsi, R. Kitlei, D. Leskó, M. Tejfel, and S. Laki, “T4P4S: A Target-independent Compiler for Protocol-independent Packet Processors,” in *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2018, pp. 1–8.

[15] A. Starovoitov, “tracing filters with bpf,” https://lkml.org/lkml/2013/12/2/1066/, 2013, accessed June 3, 2020.

[16] S. McCanne and V. Jacobson, “The BSD Packet Filter: A New Architecture for User-level Packet Capture.” in *USENIX winter*, vol. 46, 1993.

[17] W. Tu, F. Ruffy, and M. Budiu, “Linux Network Programming with P4,” in *Linux Plumbers’ Conference 2018*, 2018.

[18] p4language, “ebpf backend,” https://github.com/p4lang/p4c/tree/master/backends/ebpf/, 2020, accessed June 3, 2020.

[19] O. N. Foundation, “PTF-based data plane tests for ONOS fabric.p4,” https://github.com/opennetworkinglab/fabric-p4test/, 2020, accessed June 3, 2020.

ISBN 978-3-937201-70-2
DOI 10.2313/NET-2020-11-1

ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)